EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

SINFONI Pipeline User Manual

VLT-MAN-ESO-19500-3600

Issue 1.0

Date 2005-10-19

		Name	Date	Signature
Prepared:	ESO SINFONI Pipeline Team		2005-10-19	
Approved:	P.Ballester, M. Peron			
Released:	P. Quinn			

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
1.0	19/10/2005	All	Public release

This page was intentionally left blank

Contents

1 Introduction **11**

1.1 Purpose . . . 11

1.2 Acknowledgements . . . 11

1.3 Scope . . . 11

1.4 Reference documents . . . 12

2 Overview **13**

3 SINFONI Instrument Description **14**

3.1 Instrument overview . . . 14

4 Quick start **16**

4.1 SINFONI pipeline recipes . . . 16

4.2 An introduction to Gasgano and EsoRex . . . 17

4.2.1 Using Gasgano . . . 17

4.2.2 Using EsoRex . . . 22

4.3 Example of data reduction using EsoRex . . . 24

5 Known problems **30**

6 Instrument Data Description **32**

6.1 Data features . . . 32

6.2 Raw frames . . . 33

7 Static Calibration Data **38**

7.1 Line reference table . . . 38

7.2 DRS setup table . . . 38

7.3 Reference bad pixel map . . . 38

7.4 First column reference table . . . 38

8 Data Reduction **39**

8.1 Data reduction overview . . . 39

8.2 Required input data 40
8.3 Reduction cascade 41

9 Pipeline Recipes Interfaces 44
9.1 si_rec_detlin 44
9.1.1 Input 44
9.1.2 Output 44
9.1.3 Quality control 45
9.1.4 Parameters 45
9.2 si_rec_mdark 45
9.2.1 Input 45
9.2.2 Output 46
9.2.3 Quality control 46
9.2.4 Parameters 46
9.3 si_rec_mflat 47
9.3.1 Input 47
9.3.2 Output 47
9.3.3 Quality control 47
9.3.4 Parameters 48
9.4 si_rec_distortion 49
9.4.1 Input 49
9.4.2 Output 50
9.4.3 Quality control 50
9.4.4 Parameters 51
9.5 si_rec_wavecal 53
9.5.1 Input 53
9.5.2 Output 54
9.5.3 Quality control 54
9.5.4 Parameters 55
9.6 si_rec_psf 56
9.6.1 Input 56

9.6.2 Output . . . 56
9.6.3 Quality control . . . 57
9.6.4 Parameters . . . 57
9.7 si_rec_stdstar . . . 58
9.7.1 Input . . . 58
9.7.2 Output . . . 58
9.7.3 Quality control . . . 58
9.7.4 Parameters . . . 59
9.8 si_rec_objnod . . . 59
9.8.1 Input . . . 60
9.8.2 Output . . . 60
9.8.3 Quality control . . . 60
9.8.4 Parameters . . . 60
9.9 si_utl_skymap . . . 61
9.9.1 Input . . . 61
9.9.2 Output . . . 62
9.9.3 Parameters . . . 62
9.10 si_utl_bp_mask_add . . . 62
9.10.1 Input . . . 62
9.10.2 Output . . . 62
9.10.3 Parameters . . . 63
9.11 si_utl_ima_arith . . . 63
9.11.1 Input . . . 63
9.11.2 Output . . . 63
9.11.3 Parameters . . . 63
9.12 si_utl_cube2ima . . . 63
9.12.1 Input . . . 63
9.12.2 Output . . . 63
9.12.3 Parameters . . . 64
9.13 si_utl_cube2spectrum . . . 64

9.13.1 Input . . . 64
9.13.2 Output . . . 64
9.13.3 Parameters . . . 64
9.14 si_utl_cube_arith . . . 65
9.14.1 Input . . . 65
9.14.2 Output . . . 65
9.14.3 Parameters . . . 65
9.15 si_utl_spectrum_divide_by_blackbody . . . 65
9.15.1 Input . . . 65
9.15.2 Output . . . 66
9.15.3 Parameters . . . 66
9.16 si_utl_spectrum_wavelength_shift . . . 66
9.16.1 Input . . . 66
9.16.2 Output . . . 66
9.16.3 Parameters . . . 66

10 Algorithms and recipe details 67

10.1 Algorithms . . . 67
10.1.1 Frame stacking . . . 67
10.1.2 Average with rejection . . . 67
10.1.3 Detector non linearity computation . . . 67
10.1.4 Nearest neighbours bad pixel cleaning . . . 68
10.1.5 Detector gain computation . . . 71
10.1.6 Read Out Noise computation . . . 71
10.1.7 Fixed Pattern Noise computation . . . 71
10.1.8 Line position determination . . . 71
10.1.9 Dispersion relation and wavelength map determination . . . 72
10.1.10 Line shift computation . . . 73
10.1.11 Dispersion relation adjustment . . . 73
10.1.12 Slitlet position computation . . . 74
10.1.13 Slitlet distances computation . . . 75

10.1.14 Cube construction: resampling . . . 75
10.1.15 Cube coaddition . . . 76
10.1.16 Estimation of the sky from object frames in case the input set is missing sky frames . . . 76
10.1.17 Efficiency computation . . . 77
10.1.18 Strehl computation . . . 78
10.1.19 Encircled energy computation . . . 78
10.1.20 Spectrum extraction . . . 78
10.1.21 Standard star position detection . . . 78
10.2 Recipes . . . 78
10.2.1 Detector linearity and non-linear bad pixel map determination: si_rec_detlin . . . 78
10.2.2 Master dark and bad pixel map determination: si_rec_mdark . . . 79
10.2.3 Master flat and threshold pixels (bad pixel map) determination: si_rec_mflat . . . 79
10.2.4 Optical distortion and slitlet distances determination: si_rec_distortions . . . 80
10.2.5 Wavelength solution determination: si_rec_wavecal . . . 81
10.2.6 Science observations: si_rec_objnod . . . 83
10.2.7 STD star data reduction: si_rec_stdstar . . . 83
10.2.8 PSF data reduction: si_rec_psf . . . 84

A Installation **85**
A.1 Supported platforms . . . 85
A.2 Building the SINFONI pipeline . . . 85
A.2.1 Requirements . . . 85
A.2.2 Compiling and installing the SINFONI pipeline . . . 86

B Abbreviations and acronyms **88**

1 Introduction

1.1 Purpose

The SINFONI pipeline is a subsystem of the *VLT Data Flow System* (DFS). Its target user is ESO *Data Flow Operations* (DFO) in the generation of master calibration data, in the reduction of scientific exposures, and in the data quality control. It should also serve as a quick look tool for *Paranal Science Operations* (PSO). Additionally, the SINFONI pipeline recipes are made public to the user community, to allow a more personalised processing of the data from the instrument. The purpose of this document is to describe a typical SINFONI data reduction sequence with the SINFONI pipeline.

This manual is a complete description of the data reduction recipes implemented by the the SINFONI pipeline, reflecting the status of the SINFONI pipeline as of Oct 19, 2005 (version 1.2.0).

1.2 Acknowledgements

The SINFONI pipeline is based on the SPIFFI Data Reduction Software developed by the Max-Planck-Institut für extraterrestrische Physik (MPE). We would like to thank the SPIFFI team for providing ESO with a complete and efficient data reduction software and for their help in documenting, testing, debugging the recipes and the pipeline during several commissioning and science verifications phases. We are particularly grateful to the MPE responsibles for the data reduction: Jurgen Schreiber, Matthew Horrobin and Roberto Abuter for their contributions and support.

Release 1.2.0benefited from the feedback provided by the SINFONI SV team and SINFONI instrument operations team. In particular we would like to thank Wolfgang Hummel and Juha Runanen for extensively testing and improving the pipeline and documentation, as well as Jochen Liske for proof reading the manual.

1.3 Scope

This document describes the SINFONI pipeline used at ESO-Garching and ESO-Paranal for the purpose of data assessment and data quality control.

Updated versions of the present document may be found on [1]. For general information about the current instrument pipelines status we remind the user of [2]. Quality control information are at [3].

Additional information on QFITS, the Common Pipeline Library (CPL) and ESOREX can be found respectively at [4], [5], [6]. The Gasgano tool is described in [14]. A description of the instrument is in [7]. The SINFONI instrument user manual is in [8] while results of Science Verifications (SV) are at [9].

1.4 Reference documents

[1] SINFONI Pipeline Users' Manual VLT-MAN-ESO-19500-3600
http://www.eso.org/projects/dfs/dfs-shared/web/vlt/vlt-instrument-pipelines.html
[2] Current pipeline status
http://www.eso.org/observing/dfo/quality/pipeline-status.html
[3] ESO-Data Flow Operation home page http://www.eso.org/observing/dfo/quality/
[4] QFITS home page http://www.eso.org/projects/aot/qfits/
[5] CPL home page http://www.eso.org/cpl
[6] ESOREX home page http://www.eso.org/cpl/esorex.html
[7] SINFONI home page http://www.eso.org/instruments/sinfoni/
[8] VLT SINFONI User Manual VLT-MAN-ESO-14700-3517
http://www.eso.org/instruments/sinfoni/usermanual.html
[9] SINFONI SV home page http://www.eso.org/science/vltsv/sinfonisv/
[10] VLT Data Flow System Specifications for Pipeline and Quality Control
VLT-SPE-ESO-19600-1233
[11] DFS Pipeline & Quality Control – User Manual VLT-MAN-ESO-19500-1619
[12] ESO DICB – Data Interface Control Document GEN-SPE-ESO-00000-0794
[13] Common Pipeline Library User Manual VLT-MAN-ESO-19500-2720
[14] Gasgano User's Manual VLT-PRO-ESO-19000-1932
[15] SINFONI Calibration Plan VLT-PLA-ESO-14700-2617

2 Overview

In collaboration with instrument consortia, the Data Flow Systems Department (DFS) of the Data Management and Operation Division is implementing data reduction pipelines for the most commonly used VLT/VLTI instrument modes. These data reduction pipelines have the following three main purposes:

Data quality control: pipelines are used to produce the quantitative information necessary to monitor instrument performance.

Master calibration product creation: pipelines are used to produce master calibration products (*e.g.*, combined bias frames, super-flats, wavelength dispersion solutions).

Science product creation: using pipeline-generated master calibration products, science products are produced for the supported instrument modes (*e.g.*, combined ISAAC jitter stacks; bias-corrected, flat-fielded FORS images, wavelength-calibrated UVES spectra). The accuracy of the science products is limited by the quality of the available master calibration products and by the algorithmic implementation of the pipelines themselves. In particular, adopted automatic reduction strategies may not be suitable or optimal for all scientific goals.

Instrument pipelines consist of a set of data processing modules that can be called from the command line, from the automatic data management tools available on Paranal or from Gasgano.

ESO offers two front-end applications for launching pipeline recipes, *Gasgano* [14] and *EsoRex*, both included in the pipeline distribution (see Appendix A, page 85). These applications can also be downloaded separately from `http://www.eso.org/gasgano` and `http://www.eso.org/cpl/esorex.html`. An illustrated introduction to Gasgano is provided in the "Quick Start" Section of this manual (see page 16).

The SINFONI instrument and the different types of SINFONI raw frames and auxilliary data are described in Sections 3, 6, and 7.

A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section 4. In section 5 we advice the user about known data reduction problems providing also possible solutions.

An overview of the data reduction, what are the input data, and the recipes involved in the calibration cascade is provided in section 8.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in section 9.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Section 10.

In Appendix A the installation of the SINFONI pipeline recipes is described and in Appendix B a list of used abbreviations and acronyms is given.

3 SINFONI Instrument Description

SINFONI has been developed by ESO and the Max-Planck-Institut für extraterrestrische Physik (MPE) in Garching.

The instrument has been made available to the community and started operations in Paranal on April 1^{st}, 2005.

In this chapter a brief description of the SINFONI instrument is given. A more complete documentation can be found in the SINFONI User Manual, downloadable from `http://www.eso.org/instruments/sinfoni/`

3.1 Instrument overview

SINFONI is a near-infrared (1.05 - 2.45 μm) integral field spectrograph (SPIFFI, developed by MPE) fed by an adaptive optics module (MACAO, developed by ESO, more details are given in [11]) .

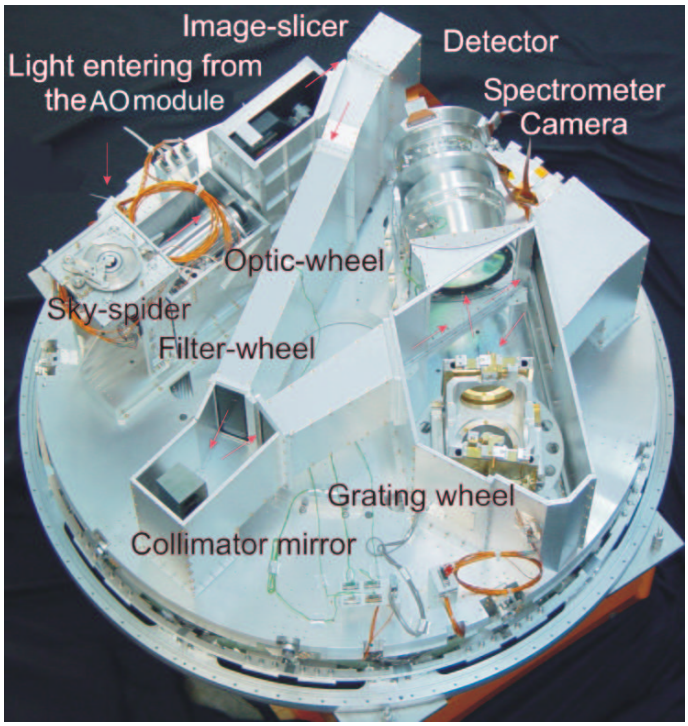


Figure 3.1.0: An inside view of SPIFFI: The cryostat cover and the reinforcing structure have been removed to provide a free view on the opto-mechanical components of SPIFFI. The light enters from the top, and passes the sky-spider. The pre-optics with a filter-wheel and interchangeable lenses provides three different image scales. The image slicer re-arranges the two-dimensional field into a pseudo-long slit, which is perpendicular to the base plate. Three diamond turned mirrors collimate the light onto the gratings. In total, four gratings are implemented on the grating drive. A multiple-lens system then focuses the spectra on a Rockwell HAWAII array. The diameter of the instrument is 1.3m.

The spectrograph operates with 4 gratings (J, H, K, H+K) providing a spectral resolution around 2000, 3000,

4000 in J, H, K, respectively, and 1500 in H+K - each wavelength band fitting fully on the 2048 pixels of the Hawaii 2RG (2kx2k) detector in the dispersion direction. The SINFONI field of view on the sky is sliced into 32 slices. Pre-optics allow to chose the width of the slices. The choices are 250mas, 100mas and 25mas, leading to field of views on the sky of 8"x8", 3"x3", or 0.8"x0.8" respectively. On raw frames each pixel images a rectangular region on the sky (125x250, 50x100, or 12.5x25 mas). Each SINFONI FOV image slice corresponds to a so called detector slitlet. Each one of the 32 slitlets is imaged onto 64 pixels of the detector. Thus one obtains 32x64 spectra of the imaged region on the sky.

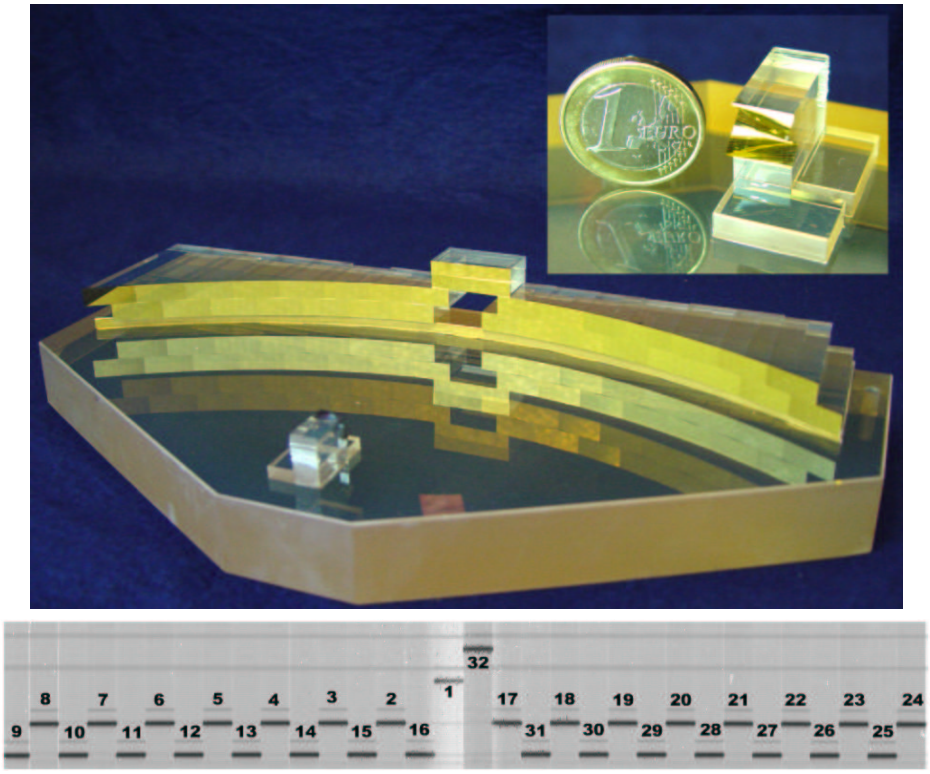


Figure 3.1.0: SPIFFI image slicer (top): The light enters through the hole in the big slicer. A stack of 32 small mirrors, the small slicer (also shown in the sub-panel), slices the image and redirects the light towards the 32 mirrors of the big slicer, which re-arranges the slitlets into a 31 cm long pseudo-slit. (bottom). The layout of the slitlets on a raw SPIFFI frame.

4 Quick start

This section describes the most immediate usage of the SINFONI pipeline recipes.

4.1 SINFONI pipeline recipes

The current SINFONI pipeline is based on a set of 8 stand-alone recipes involved in the data reduction cascade:

si_rec_detlin to evaluate the detector non linearity and generate a corresponding non linear pixel map.

si_rec_mdark to create a master dark and a hot-pixel map.

si_rec_mflat to create a master flat and a map of pixels which have intensities greater than a given threshold.

si_rec_distortion to compute the optical distortions and slitlets distances.

si_rec_wavecal for wavelength calibration.

si_rec_psf for PSF standard data reduction.

si_rec_stdstar for STD standard data reduction.

si_rec_objnod for science standard data reduction.

Additional 8 stand-alone recipes are also provided, that perform useful tasks:

si_utl_skymap to flag sky lines as bad pixels in a “sky map” (for Paranal operations, to prepare input data of the SINFONI Real Time Display).

si_utl_bp_mask_add to coadd bad pixel maps.

si_utl_ima_arith to do image arithmetics.

si_utl_cube2ima to collapse a cube to an image over a given wavelength range.

si_utl_cube2spectrum to extract a spectrum from a cube.

si_utl_cube_arith to perform cube arithmetics.

si_utl_spectrum_divide_by_blackbody to divide a spectrum by a black body spectrum.

si_utl_spectrum_wavelegth_shift to shift in wavelength a spectrum.

4.2 An introduction to Gasgano and EsoRex

Before being able to call pipeline recipes on a set of data, the data must be opportunely classified, and associated with the appropriate calibrations. The *Data Classification* consists of tasks such as: "What kind of data am I?", *e.g.*, BIAS, "to which group do I belong?", *e.g.*, to a particular Observation Block or template. *Data Association* is the process of selecting appropriate calibration data for the reduction of a set of raw science frames. Typically, a set of frames can be associated if they share a number of properties, such as instrument and detector configuration. As all the required information is stored in the FITS headers, data association is based on a set of keywords (called "association keywords") and is specific to each type of calibration.

The process of data classification and association is known as data organisation.

An instrument pipeline consists of a set of data processing modules that can be called from different host applications, either from the command line with *Esorex*, from the automatic data management tools available at Paranal, or from the graphical *Gasgano* tool.

Gasgano is a data management tool that simplifies the data organisation process, offering automatic data classification and making the data association easier (*even if automatic association of frames is not yet provided*). *Gasgano* determines the classification of a file by applying an instrument specific rule, while users must provide this information to the recipes when they are executed manually using *Esorex* from the command line. In addition, *Gasgano* allows the user to execute directly the pipeline recipes on a set of selected files.

4.2.1 Using Gasgano

To get familiar with the SINFONI pipeline recipes and their usage, it is advisable to begin with *Gasgano*, because it provides a complete graphic interface for data browsing, classification and association, and offers several other utilities such as easy access to recipes documentation and preferred data display tools.

Gasgano can be started from the system prompt in the following way:

```
gasgano &
```

The *Gasgano* main window will appear. On Figure 4.2.1 (next page), a view on a set of SINFONI IFU data is shown as an example. *Gasgano* can be pointed to the directories where the data to be handled are located using the navigation panels accessible via the *Add/Remove Files* entry of the *File* menu (shown on the upper left of the figure).

The data are hierarchically organised as preferred by the user. After each file name are shown the classification, the instrument setup id (which indicates the band), the instrument pre-optic (which indicates the camera setting), the template exposure number and the number of exposures in the template, and the value of the DPR.TYPE.

More information about a single frame can be obtained by clicking on its name: the corresponding FITS file header will be displayed on the bottom panel, where specific keywords can be opportunely filtered and searched. Images and tables may be easily displayed using the viewers specified in the appropriate *Preferences* fields.

Frames can be selected from the main window for being processed by the appropriate recipe: on Figure 4.2.2, the standard star frame and a sky frame, already produced master bad pixel map and master flat field frames, together with distortion and slitlet distance tables, and the necessary static calibration tables, are all selected

and sent to the *si_rec_stdstar* recipe. This will open a *Gasgano* recipe execution window (see Figure 4.2.3), having all the specified files listed in its *Input Frames* panel.

Help about the recipe may be obtained from the *Help* menu. Before launching the recipe, its configuration may be opportunely modified on the *Parameters* panel (on top). The window contents might be saved for later use by selecting the *Save Current Settings* entry from the *File* menu, as shown in figure.

At this point the recipe can be launched by pressing the *Execute* button. Messages from the running recipe will appear on the *Log Messages* panel at bottom, and in case of successful completion the products will be listed on the *Output Frames* panel, where they can be easily viewed and located back on the Gasgano main window.

Please refer to the *Gasgano User's Manual* [7] for a more complete description of the *Gasgano* interface.

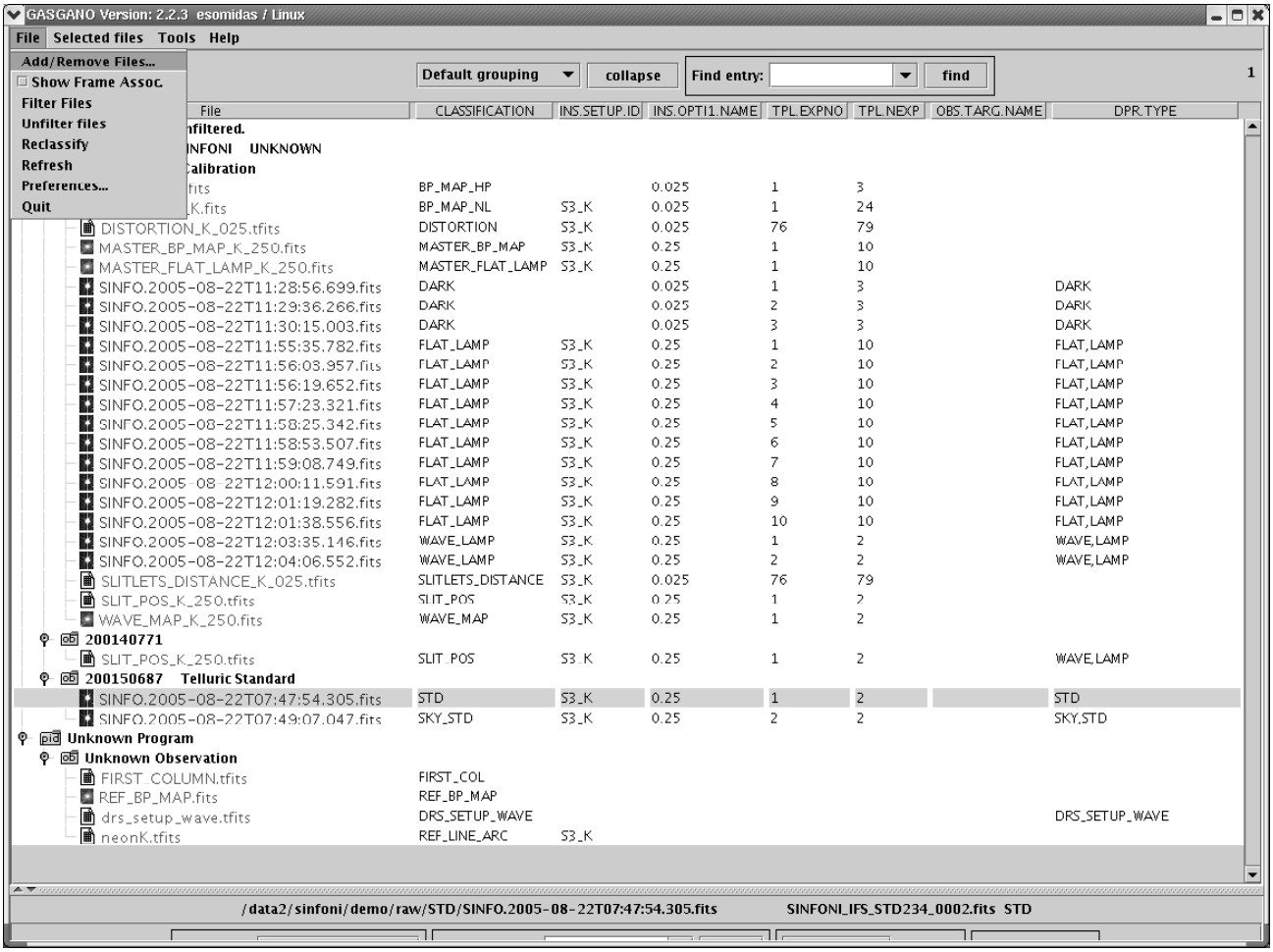


Figure 4.2.1: *The Gasgano main window.*

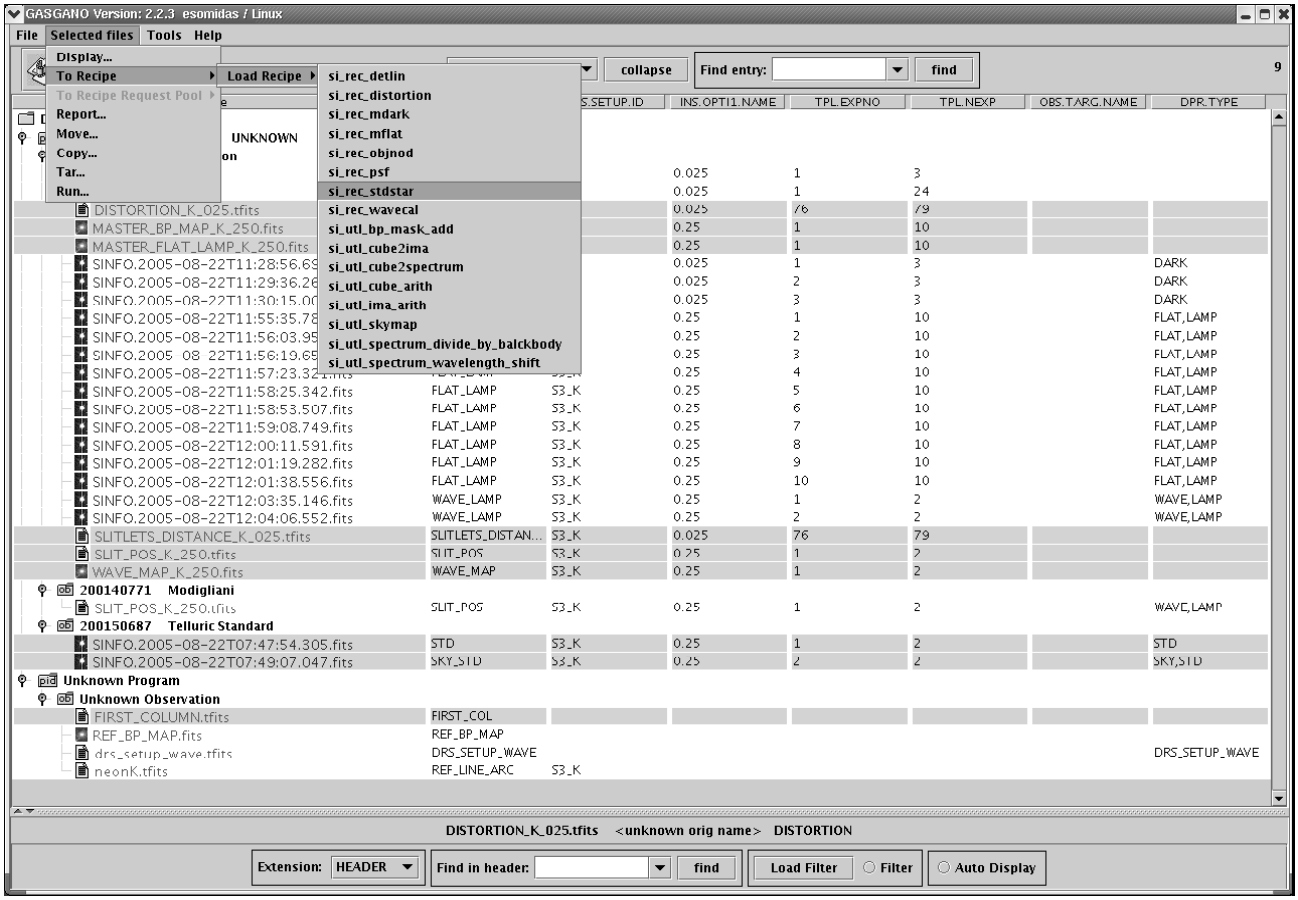


Figure 4.2.2: *Selecting files to be processed by a SINFONI pipeline recipe.*

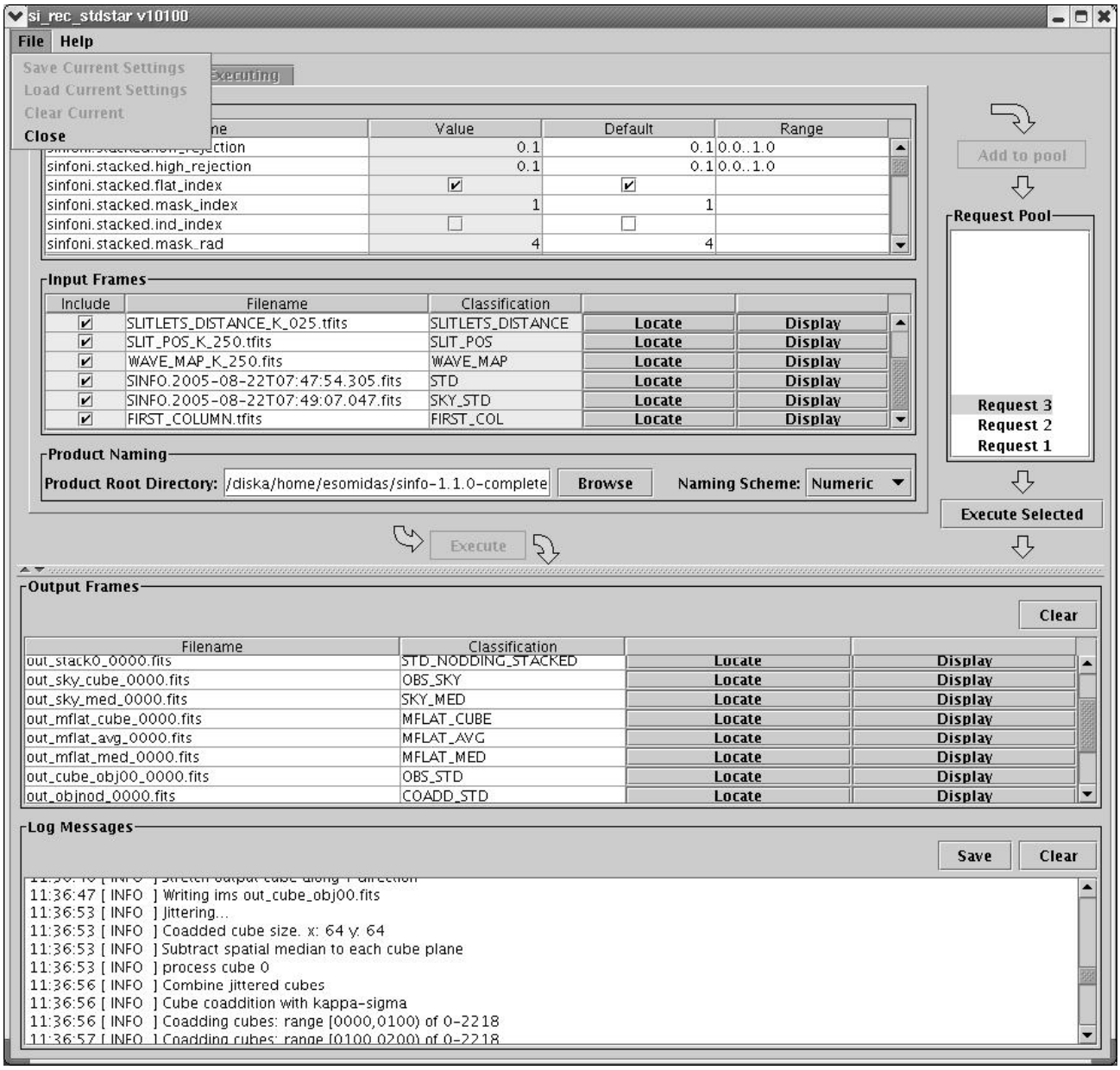


Figure 4.2.3: *The Gasgano recipe execution window.*

4.2.2 Using EsoRex

EsoRex is a command line utility for running pipeline recipes. It may be embedded by users into data reduction scripts for the automation of processing tasks. On the other side, *EsoRex* doesn't offer all the facilities available with *Gasgano*, and the user must classify and associate the data using the information contained in the FITS header keywords (see Section 6.2, page 33). The user should also take care of defining the input set-of-frames and the appropriate configuration parameters for each recipe run:

The set-of-frames: Each pipeline recipe is run on a set of input FITS data files. When using *EsoRex* the filenames must be listed together with their DO category [1] in an ASCII file, the *set-of-frames* (SOF), that is required when launching a recipe. [2]

Here is an example of SOF, valid for the *si_rec_wavecal* recipe [3]:

```
/file_path/SINFO.2004-08-14T10:20:56.497.fits  WAVE_LAMP
/file_path/SINFO.2004-08-14T10:22:44.285.fits  WAVE_LAMP
/file_path/xenon.tfits                         REF_LINE_ARC
/file_path/MASTER_BP_MAP_H_250.fits            MASTER_BP_MAP
/file_path/MASTER_LAMP_FLAT_H_250.fits         MASTER_FLAT_LAMP
/file_path/DISTORTION_H.fits                   DISTORTION
/file_path/drs_setup_wave.tfits                DRS_SETUP_WAVE
/file_path/SLIT_POS_H_250.tfits                SLIT_POS
```

It contains for each input frame the full path file name and its DO category. The pipeline recipe will access the listed files when required by the reduction algorithm.

Note that the SINFONI pipeline recipes do not verify in any way the correctness of the classification tags specified by the user in the SOF. In the above example, the recipe *si_rec_wavecal* will treat the frame `/file_path/SINFO.2004-08-14T10:20:56.497.fits` as a WAVE_LAMP, the frame `/file_path/MASTER_BP_MAP_H_250.fits` as a MASTER_BP_MAP, etc., even when they do not contain this type of data. The recipe will also assume that all frames are associated correctly, *i.e.*, that they all come from the same band and pre-optic, and that the appropriate calibration files have been specified.

The reason of this lack of control is that the SINFONI recipes are just the DRS component of the complete pipeline running on Paranal, where the task of data classification and association is carried out by separate applications. Moreover, using *Gasgano* as an interface to the pipeline recipes will always ensure a correct classification of all the data frames, assigning the appropriate DO category to each one of them (see Section 4.2.1, page 17).

A recipe handling an incorrect SOF may stop or display unclear error messages at best. In the worst cases, the recipe would apparently run without any problem, producing results that may look reasonable, but are actually flawed.

[1] The indicated *DO category* is a label assigned to any data type after it has been classified, which is then used to identify the frames listed in the *set-of-frames*

[2] The set-of-frames corresponds to the *Input Frames* panel of the *Gasgano* recipe execution window (see Figure 4.2.3, page 21).

[3] We list the file SLIT_POS_H_250.fits as an input file, as, for robustness, we suggest the user to set the parameter **slitpos_bootstrap_switch** to FALSE. A different setting would allow to reduce the data without including the SLIT_POS table

EsoRex syntax: The basic syntax to use ESOREX is the following:

esorex [esorex_options] recipe_name [recipe_options] set_of_frames

To get more information on how to customise ESOREX (see also [7]) run the command:

esorex --help

To generate a configuration file esorex.rc in the directory $HOME/.esorex run the command:

esorex --create-config

A list of all available recipes, each with a one-line description, can be obtained using the command:

esorex --recipes

All recipe parameters (aliases) and their default values can be displayed by the command

esorex --params recipe_name

To get a brief description of each parameter meaning execute the command:

esorex --help recipe_name

To get more details about the given recipe give the command at the shell prompt:

esorex --man-page recipe_name

Recipe configuration: Each pipeline recipe may be assigned an *EsoRex* configuration file, containing the default values of the parameters related to that recipe.[4] The configuration files are normally generated in the directory `$HOME/.esorex`, and have the same name as the recipe to which they are related, with the filename extension `.rc`. For instance, the recipe *si_rec_wavecal* has its *EsoRex* generated configuration file named `si_rec_wavecal.rc`, and is generated with the command:

esorex --create-config si_rec_wavecal

The definition of one parameter of a recipe may look like this:

```
# --stack-warpfix_kernel
# Warpfix kernel:  (tanh | sinc | sinc2 | lanczos | hamming | hann)
sinfoni.stacked.warpfix_kernel=tanh
```

In this example, the parameter `sinfoni.stacked.warpfix_kernel` is set to the value `tanh`. In the configuration file generated by *EsoRex*, one or more comment lines are added containing information about the possible values of the parameter, and an alias that could be used as a command line option.

The recipes provided by the SINFONI pipeline are designed to implement a cascade of macro data reduction steps, each controlled by its own parameters. For this reason and to prevent parameter name clashes we specify as parameter prefix not only the instrument name but also the name of the step they refer to. Shorter parameter aliases are made available for use on the command line.

The command

esorex --create-config recipe_name

generates a default configuration file **recipe_name.rc** in the directory **$HOME/.esorex**[5].

[4] The *EsoRex* recipe configuration file corresponds to the *Parameters* panel of the *Gasgano* recipe execution window (see Figure 4.2.3, page 21).

[5] If a number of recipe parameters are specified on the command line, the given values will be used in the created configuration file.

A recipe configuration file different from the default one can be specified on the command line:

esorex --recipe-config=my_alternative_recipe_config

Recipe parameters are provided in section 9 and their role is described in Section 10.

More than one configuration file may be maintained for the same recipe but, in order to be used, a configuration file not located under `$HOME/.esorex`, or having a name different from the recipe name, should be explicitly specified when launching a recipe.

Recipe execution: A recipe can be run by specifying its name to *EsoRex*, together with the name of a set-of-frames. For instance, the following command line would be used to run the recipe *si_rec_wavecal* for processing the files specified in the set-of-frames `si_rec_wavecal.sof`:

esorex si_rec_wavecal si_rec_wavecal.sof

The recipe parameters can be modifyed either by editing directly the used configuration file, or by specifying new parameter values on the command line using the command line options defined for this purpose. Such command line options should be inserted after the recipe name and before the SOF name, and they will supersede the system defaults and/or the configuration file settings. For instance, to set the *si_rec_wavecal* recipe *wcal-pixel_tol* parameter to `3.0`, the following should be typed:

esorex si_rec_wavecal --wcal-pixel_tol=3.0 si_rec_wavecal.sof

For more information on *EsoRex*, see `http://www.eso.org/cpl/esorex.html`.

4.3 Example of data reduction using EsoRex

A simple, typical data reduction procedure is described here.[6]

We suggest the user to organize his data per type, observed band and camera setting. Dark frames may be grouped per detector DIT, frames to compute distortion and frames to compute detector non linearities may be organized per observed band. The detector DIT is given by the value of the FITS keyword DET DIT[7]. The observed band is indicated by the value of the FITS keyword INS SETUP ID. The camera setting is indicated by the value of INS OPTI1 NAME. In the examples below we suppose the user has data acquired in band K and with the 100 mas pre-optic setting, and DIT=600. In the following examples `/path_raw/` indicates the full path to the source tree directory containing raw data.

Dark Frames: those frames are characterized by DPR.TYPE='DARK',

```
/path_raw/DARK/600/SINFO.2004-08-23T09:36:12.316.fits DARK
/path_raw/DARK/600/SINFO.2004-08-23T09:51:59.824.fits DARK
/path_raw/DARK/600/SINFO.2004-08-23T10:07:40.760.fits DARK
```

Detector linearity flat field frames: those frames are caracterized by DPR.TYPE='LINEARITY,LAMP'

[6] The procedure using *Gasgano* is conceptually identical.

[7] We omit here the prefix HIERARCH ESO

```
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:09:50.882.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:07.455.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:23.047.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:38.240.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:56.403.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:31.128.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:59.183.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:12:27.247.fits LINEARITY_LAMP
```

Fibre frames, flat frames and arc lamp frames to compute distortions: those frames are have DPR.TYPE respectively equal to 'DISTORTION,FIBRE,NS' 'DISTORTION,FLAT,NS', 'DISTORTION,WAVE,NS'

```
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:46:25.168.fits FIBRE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:46:53.132.fits FIBRE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:47:09.065.fits FIBRE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:47:31.008.fits FIBRE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:47:49.771.fits FIBRE_NS
.
.
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:52:35.116.fits FIBRE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:53:22.863.fits FLAT_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:53:45.397.fits FLAT_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:54:08.861.fits WAVE_NS
/path_raw/DISTORTION/K/SINFO.2005-03-14T11:55:10.220.fits WAVE_NS
```

Standard flat field frames: those frames are caracterized by DPR.TYPE='FLAT,LAMP'

```
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:27:43.232.fits FLAT_LAMP
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:28:05.846.fits FLAT_LAMP
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:28:18.820.fits FLAT_LAMP
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:28:32.593.fits FLAT_LAMP
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:28:45.566.fits FLAT_LAMP
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:29:08.165.fits FLAT_LAMP
```

Arc lamp frames: those frames are caracterized by DPR.TYPE='WAVE,LAMP'

```
/path_raw/WAVE/K/100/SINFO.2005-02-28T17:55:44.753.fits WAVE_LAMP
/path_raw/WAVE/K/100/SINFO.2005-02-28T18:00:15.875.fits WAVE_LAMP
```

science frames: those frames are caracterized by DPR.TYPE='OBJECT' or DPR.TYPE='SKY'

```
/path_raw/SCI/K/100/SINFO.2005-02-27T04:30:22.175.fits OBJECT_NODDING
/path_raw/SCI/K/100/SINFO.2005-02-27T04:30:54.620.fits SKY_NODDING
```

```
/path_raw/SCI/K/100/SINFO.2005-02-27T04:31:19.755.fits SKY_NODDING
/path_raw/SCI/K/100/SINFO.2005-02-27T04:31:46.369.fits OBJECT_NODDING
/path_raw/SCI/K/100/SINFO.2005-02-27T04:32:12.983.fits OBJECT_NODDING
/path_raw/SCI/K/100/SINFO.2005-02-27T04:32:46.418.fits SKY_NODDING
```

To have additional information on the instrument performance the user may want to reduce also telluric standard star frames: those frames are caracterized by DPR.TYPE='STD' or DPR.TYPE='SKY,STD'

```
/path_raw/STD/K/100/SINFO.2005-02-27T09:17:05.775.fits STD
/path_raw/STD/K/100/SINFO.2005-02-27T09:18:10.566.fits SKY_STD
```

or PSF standard star frames: those frames are caracterized by `DPR.TYPE='PSF-CALIBRATOR'` or `DPR.TYPE='SKY,PSF-CALIBRATOR'`

```
/path_raw/PSF/K/100/SINFO.2004-08-23T07:35:16.597.fits PSF_CALIBRATOR
/path_raw/PSF/K/100/SINFO.2004-08-23T07:36:55.413.fits SKY_PSF_CALIBRATOR
```

We describe below a typical data reduction sequence using EsoRex. In this section we assume that the user sets in the EsoRex configuration file ($HOME/.esorex/esorex.rc) the flag *suppress-prefix* to TRUE, so that the pipeline product file names have standard names, with extention .fits for images and .tfits for tables. We suggest to verify to have the flag *readonly* set to FALSE, if the user would like to run the same recipe several times with EsoRex having standard names for product files. This setting allows the pipeline to overwrite previously generated products [8]. In the following we indicate only those frames involved in the data reduction cascade, suggesting the user to rename them according to their PRO.CATG, INS.SETUP.ID and band, and to remove the other products after each recipe execution.

1. The user may start to generate a master dark. Raw dark frames may be put in an ASCII file, mdark_sof. This file will look like as follows:

   ```
   /path_raw/DARK/600/SINFO.2004-08-23T09:36:12.316.fits DARK
   /path_raw/DARK/600/SINFO.2004-08-23T09:51:59.824.fits DARK
   /path_raw/DARK/600/SINFO.2004-08-23T10:07:40.760.fits DARK
   ```

   Then the user can generate the master dark with the command

   esorex si_rec_mdark mdark_sof

   This command will generate two files: out_bp_noise.fits (PRO.CATG=BP_MAP_HP), a hot pixel map, and out_dark.fits (PRO.CATG=MASTER_DARK), a master dark. For convenience we indicate with `/path_cdb` the full path to a directory containing relevant data reduction products.

   mv out_bp_noise.fits /path_pro/BP_MAP_HP.fits

   mv out_dark.fits /path_pro/MASTER_DARK_600.fits

   rm -rf out*fits *.paf *.log

[8] By default installation in the EsoRex configuration file ($HOME/.esorex/esorex.rc) the flag *suppress-prefix* to FALSE and the flag *readonly* is set to FALSE, a possible combination, in which case pipeline product filenames will have a prefix out_, an increasing a four digit number, and extention .fits for images and .tfits for tables.

2. Then the user may generate a map of non linear pixels. A set of linearity raw flat field frames may be put in the ASCII file linearity_sof.

```
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:09:50.882.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:07.455.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:23.047.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:38.240.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:56.403.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:31.128.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:59.183.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:12:27.247.fits LINEARITY_LAMP
```

   The user can generate a non linearity bad pixel map (PRO.CATG=BP_MAP_NL) with the command:

   esorex si_rec_detlin linearity_sof

   This command will generate several files, including the non linearity bad pixel map, stored in the file “out_bp_lin.fits”.

   mv out_bp_lin.fits /path_pro/BP_MAP_NL_K.fits

   rm -rf out*fits *.paf *.log

3. Then the user may determine the optical distortions (PRO.CATG=DISTORTION) and the slitlets distances (PRO.CATG=SLITLETS_DISTANCE).

   The user will select all the files containing the string DISTORTION in their DPR.TYPE. Using those files and the line reference table of the corresponding band (K) and a drs setup table will generate an ASCII file distortion_sof:

```
/path_raw/SINFO.2005-03-14T11:28:19.007.fits FIBRE_NS
/path_raw/SINFO.2005-03-14T11:28:43.781.fits FIBRE_NS
/path_raw/SINFO.2005-03-14T11:28:59.723.fits FIBRE_NS
/path_raw/SINFO.2005-03-14T11:29:18.496.fits FIBRE_NS
.
.
.
/path_raw/SINFO.2005-03-14T11:53:22.863.fits FLAT_NS
/path_raw/SINFO.2005-03-14T11:53:45.397.fits FLAT_NS
/path_raw/SINFO.2005-03-14T11:54:08.861.fits WAVE_NS
/path_raw/SINFO.2005-03-14T11:55:10.220.fits WAVE_NS
/path_cdb/neonK.tfits REF_LINE_ARC
/path_cdb/drs_setup_wave.tfits    DRS_SETUP_WAVE
```

   Where with `/path_cdb` we have indicated the full path to the directory containing calibration data. The command

   esorex si_rec_distortion distortion_sof

Will generate several products. Between those also out_distortion.tfits, a table containing information on the polynomial distortion coefficients, and out_distances.tfits, a table containing information on the slitlet distances.

mv out_distortion.tfits /path_pro/DISTORTION_K.tfits

mv out_distances.tfits /path_pro/SLITLETS_DISTANCE_K.tfits

rm -rf out*fits *.paf *.log

4. Then one selects the raw flat fields and list them in an ASCII file mflat_sof together with some static calibrations and previously obtained products:

```
/path_raw/FLAT/SINFO.2005-02-28T16:27:43.232.fits FLAT_LAMP
/path_raw/FLAT/SINFO.2005-02-28T16:28:05.846.fits FLAT_LAMP
/path_raw/FLAT/SINFO.2005-02-28T16:28:18.820.fits FLAT_LAMP
/path_raw/FLAT/SINFO.2005-02-28T16:28:32.593.fits FLAT_LAMP
/path_raw/FLAT/SINFO.2005-02-28T16:28:45.566.fits FLAT_LAMP
/path_raw/FLAT/SINFO.2005-02-28T16:29:08.165.fits FLAT_LAMP
/path_cdb/REF_BP_MAP.fits REF_BP_MAP
/path_cdb/BP_MAP_NL_K.fits BP_MAP_NL
```

The command:

esorex si_rec_mflat mflat_sof

generates several frames, including the master flat field, stored in the file “out_flat.fits”, and the master bad pixel map, stored in the file “out_bpmap_sum.fits”.

mv out_flat.fits /path_pro/MASTER_FLAT_LAMP_K_100.fits

mv out_bpmap_sum.fits /path_pro/MASTER_BP_MAP_K_100.fits

rm -rf out*fits *.paf *.log

5. Then one lists raw arc lamp frames and additional static calibration frames and previously obtained master calibrations in an ASCII file wcal_sof:

```
/path_raw/WAVE/SINFO.2005-02-28T17:55:44.753.fits WAVE_LAMP
/path_raw/WAVE/SINFO.2005-02-28T18:00:15.875.fits WAVE_LAMP
/path_cdb/neonK.tfits REF_LINE_ARC
/path_pro/MASTER_FLAT_LAMP_K_100.fits MASTER_FLAT_LAMP
/path_pro/MASTER_BP_MAP_K_100.fits MASTER_BP_MAP
/path_pro/DISTORTION_K.tfits DISTORTION
/path_cdb/drs_setup_wave.tfits DRS_SETUP_WAVE
/path_cdb/SLIT_POS_K_100.tfits SLIT_POS
```

The command:

esorex si_rec_wavecal wcal_sof

generates several frames, including the master wavelength map, stored in the file “out_wavemap_ima.fits”, and the slitlet position table, stored in the file “out_slitpos.tfits”.

mv out_wavemap_ima.fits /path_pro/MASTER_WAVE_MAP_K_100.fits

mv out_slitpos.tfits /path_pro/SLIT_POS_K_100.tfits

rm -rf out*fits *.paf *.log

6. Finally the user will reduce the science data:

```
/path_raw/OBJNOD/SINFO.2005-02-27T06:09:19.358.fits OBJECT_NODDING
/path_raw/OBJNOD/SINFO.2005-02-27T06:10:10.996.fits SKY_NODDING
/path_raw/OBJNOD/SINFO.2005-02-27T06:11:33.949.fits OBJECT_NODDING
/path_raw/OBJNOD/SINFO.2005-02-27T06:13:08.204.fits SKY_NODDING
/path_raw/OBJNOD/SINFO.2005-02-27T06:14:40.068.fits OBJECT_NODDING
/path_raw/OBJNOD/SINFO.2005-02-27T06:16:11.122.fits SKY_NODDING
/path_pro/MASTER_BP_MAP_K_100.fits MASTER_BP_MAP
/path_pro/MASTER_FLAT_LAMP_K_100.fits MASTER_FLAT_LAMP
/path_pro/WAVE_MAP_K_100.fits WAVE_MAP
/path_pro/SLITLETS_DISTANCE_K.tfits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K_100.tfits SLIT_POS
/path_pro/DISTORTION_K.tfits DISTORTION
/path_cdb/FIRST_COLUMN.tfits FIRST_COL
```

7. If the user is interested to find information on the efficiency of the detector, it is necessary to reduce standard star calibrations. This can be done collecting appropriate data in a stdstar_sof file:

```
/path_raw/OBJNOD/SINFO.2005-02-27T06:09:19.358.fits STD
/path_raw/OBJNOD/SINFO.2005-02-27T06:10:10.996.fits SKY_STD
/path_pro/MASTER_BP_MAP_K_100.fits MASTER_BP_MAP
/path_pro/MASTER_FLAT_LAMP_K_100.fits MASTER_FLAT_LAMP
/path_pro/WAVE_MAP_K_100.fits WAVE_MAP
/path_pro/SLITLETS_DISTANCE_K.tfits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K.tfits SLIT_POS
/path_pro/DISTORTION_K.tfits DISTORTION
/path_cdb/FIRST_COLUMN.tfits FIRST_COL
```

8. If the user is interested to find information on the strehl, it is necessary to reduce PSF standard frames. This can be done collecting appropriate data in a psf_sof file:

```
/path_raw/OBJNOD/SINFO.2005-02-27T06:09:19.358.fits PSF_CALIBRATOR
/path_raw/OBJNOD/SINFO.2005-02-27T06:10:10.996.fits SKY_PSF_CALIBRATOR
/path_pro/MASTER_BP_MAP_K_100.fits MASTER_BP_MAP
/path_pro/MASTER_FLAT_LAMP_K_100.fits MASTER_FLAT_LAMP
/path_pro/WAVE_MAP_K_100.fits WAVE_MAP
/path_pro/SLITLETS_DISTANCE_K_100.tfits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K_100.tfits SLIT_POS
/path_pro/DISTORTION_K.tfits DISTORTION
/path_cdb/FIRST_COLUMN.tfits FIRST_COL
```

5 Known problems

We suggest the user to execute the data reduction recipes using parameter defaults and all the reference and master calibrations indicated in this manual. The following is a list of currently-known issues with SINFONI recipes, and workarounds, if available:

si_rec_wavecal This recipe sometimes fails to determine the positions of the slitlets. A possible reason is an improper data reduction parameter setting. For example the parameter **wcal-pixel_tol** has a default value of 3.0 to insure good accuracy. Values equal to 5 or 6 may insure higher robustness at the price of a minor accuracy.

The same is true for the parameters having aliases **wcal-min_diff**, **wcal-na_coeffs**, **wcal-nb_coeffs**, **wcal-pix_tol**, **wcal-y_box**, which in the current release, for accuracy, have default values as specified in the first column of the following table but in previous releases they had default values as specified in the other table rows, values which might offer greater robustness at a price of a minor accuracy.

band	**wcal-min_diff**	**wcal-na_coeffs**	**wcal-nb_coeffs**	**wcal-pix_tol**	**wcal-y_box**	comment
any	1.0	4	2	3.0	5.0	default setting
H	10.0	3	2	7.0	2.0	old setting
H+K	1.0	4	2	5.5	5.0	old setting
K	1.0	4	2	12.0	5.0	old setting
J	1.0	4	2	3.0	5.0	old setting

Another possible explanation of the failure is the usage of an improper bad pixel map. We suggest to use the REF_BP_MAP, provided with the present kit release, as input of the si_rec_mflat recipe which determines the MASTER_BP_MAP, which is input of the si_ref_wavecal recipe.

si_rec_objnod This recipe is much demanding in term of RAM. The amount of RAM required is approximatively given by the formula:

RAM=$[150 + 4 \times (2 \times sX \times sY \times sZ + 64 \times 64 \times sZ \times N_{frames})]$ MB,

where sX, sY, sZ indicate the size of the coadded cube and N_{frames} is the number of coadded cubes. sX and sY are a function of the minimum and maximum x and y offsets of the coadded cube components with respect to the first one:

$$sX = 2 \times floor(offx_{max} - offx_{min} + 0.5) + 64 \qquad sY = 2 \times floor(offy_{max} - offy_{min} + 0.5) + 64$$

The $floor(x)$ function computes the largest integral value not greater than x. We suggest the user to have at least 1GB of RAM and a swap area at least equal to the amount of RAM. In case, $sX = sY = 130$, $sZ = 2300$, $N_{frames} = 30$, the user needs at least 1.6 GB of RAM, the same amount of swap, dedicated to run the pipeline. We also suggest the user to check before starting the data reduction the values of keywords SEQ CUMOFFSETX and SEQ CUMOFFSETY. Sometimes those may be very different. This may be such that the size of the output coadded cube is large enough not to allow a full data reduction, due to the limited amount of available RAM.

In those cases, or if the single cube components correspond to observations of portions of the sky which do not overlap with each other, we suggest the user to set **objnod-jit_ind**=FALSE.

si_rec_objnod WCS coordinates in the coadded cube may be imprecise. WCS coordinates are based on the values of the FITS keywords RA and DEC. More accurate values are indeed given by INS TARG ALFA and INS TARG DELTA, but sometimes those are wrong. The WCS coordinates refer to the position of the first object of the list in the coadded space, which may not be at the center of the FOV.

si_rec_objnod Seldomly, this recipe may originate coadded cubes whose spectra along z axis may have spikes. This occurrence may be an indication that the kappa sigma clipping has removed some object point. In those cases we suggest to increase the value of kappa.

si_rec_objnod Some user may have observations taken with the telescope position angle not equal to 0. This occurrence can be revealed verifying that the value of the FITS keyword ADA POSANG is not 0. In those cases the pipeline may coadd several cubes components incorrectly.

6 Instrument Data Description

SINFONI data reduction often involves pair of frames: calibrations in which an “on” frame is acquired with a calibration lamp switched on and an “off” frame recorded with the same lamp switched off, or observations of science objects and of the sky background. Additionally, dark frames are taken as well as special fibre flat frames used to compute optical distortions. Due to a non uniform illumination of a small number of slitlets from the fibre link, several raw frames (usually 75) are necessary to compute the optical distortions. Each kind of raw frame is typically associated to a single SINFONI pipeline recipe, *i.e.*, the recipe assigned to the reduction of that specific frame type. In the pipeline environment this recipe would be launched automatically. The recipe to compute optical distortions takes as input several kinds of raw frames, including flats in which a few instrument slitlets are illuminated by a fibre, and standard pairs of flat field and arc lamp frames.

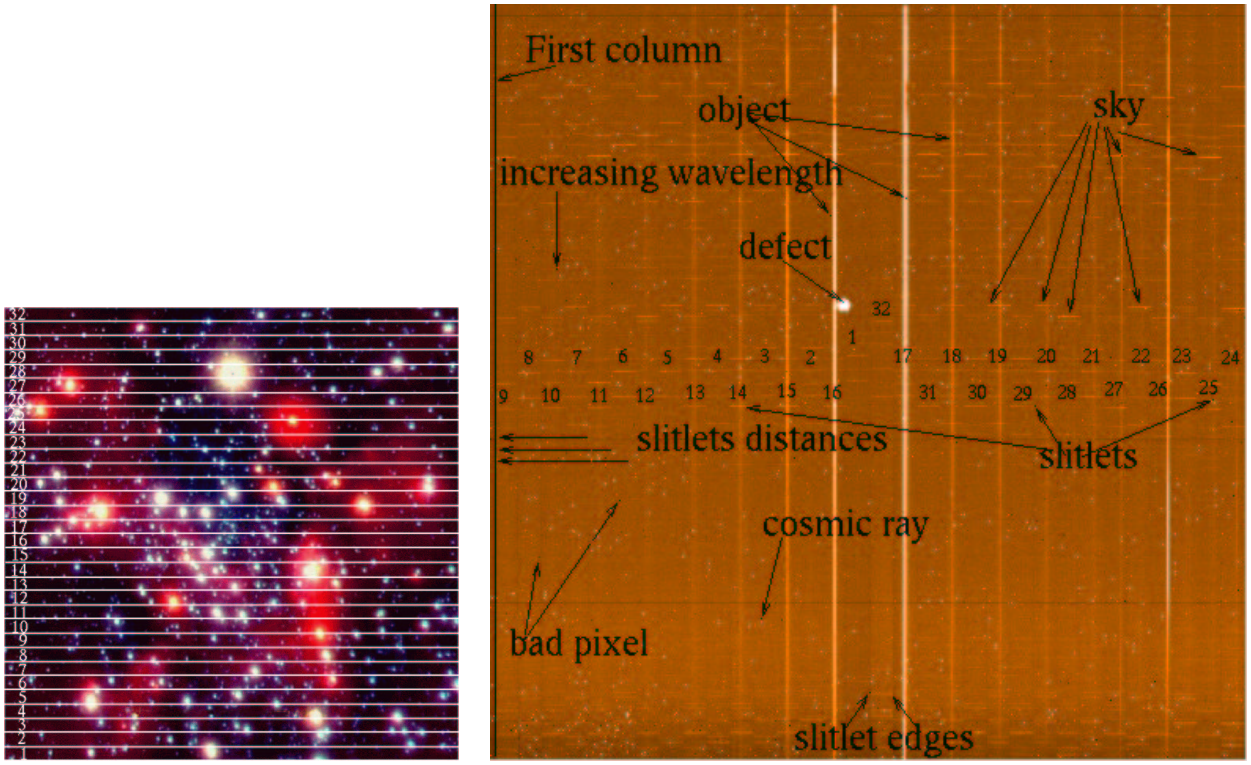
In the following sections after a brief description of how most of the raw data look like, all raw SINFONI data frames are listed, together with the keywords used for their classification and correct association. The indicated *DO category* is a label assigned to any data type after it has been classified, which is then used to identify the frames listed in the *Set of Frames* (see Section 4.2.2, page 22). We also provide snapshots of each of the main raw data on frames (“off” frames snapshots are not provided as they usually look like dark frames).

6.1 Data features

In figure 6.1.0 we have represented how a possible observation target, (a), is imaged on the detector, (b). The 32 input source’s slices generated by the two image slicers are imaged on the detector. These are also called slitlets and appear as vertical strips. Each slitlet has two edges, which can be seen by looking at the horizontal traces of sky emission lines or atmospheric absorption from the hydroxyl molecule OH. Those lines also show that the slitlets are arranged in a brick-wall pattern, to ease the detection of their edges, and that the first slitlet has a left edge not coincident with the left border of the detector. The detector column corresponding to this edge is called first column and it is taken as reference to measure the other 31 slitlets’ distances. This information, together with the actual position of each slitlet edge is necessary to reconstruct the input FOV image. The detector has a number of bad pixels and a circular defect near its center. The spatial information (both X and Y directions on the sky) is distributed along the horizontal direction alone (the X sky coordinate in each slitlet and the Y sky coordinate in the different slitlets) while the wavelength information is along the vertical direction, increasing downwards. The signal from a point-like object in the FOV with a finite PSF appears as a vertical stripe in each slitlet where its emission is relevant and it is superimposed on the sky background, more uniform along the spatial direction except in coincidence of emission lines visible as horizontal bright stripes. During an exposure the detector can be hit by cosmic rays.

In figure 6.1.0 we have the central plane of a coadded cube resulting from several target acquisitions observed at different telescope positions.

In figure 6.2.0 we have several examples of raw calibration and science frames.



(a) (b)

Figure 6.1.0: A possible target for observation (a) and a typical raw science frame layout (b).

6.2 Raw frames

We list in this section all raw frames with the relevant FITS keywords (omitting the prefix HIERARCH ESO) needed to classify them and associate to them the required calibration frames. Figure 6.2.0 shows the typical appearance of those frames.

- **Dark current:**

  DO category: `DARK`
  Processed by: `si_rec_mdark`

Classification keywords:	Association keywords:	Fig:

`DPR CATG = CALIB`
`DPR TYPE = DARK`
`DPR TECH = IMAGE`
	DET DIT	6.2.0 (a)

- **Flat Field:**

  DO category: `FLAT_LAMP`
  Processed by: `si_rec_mflat`

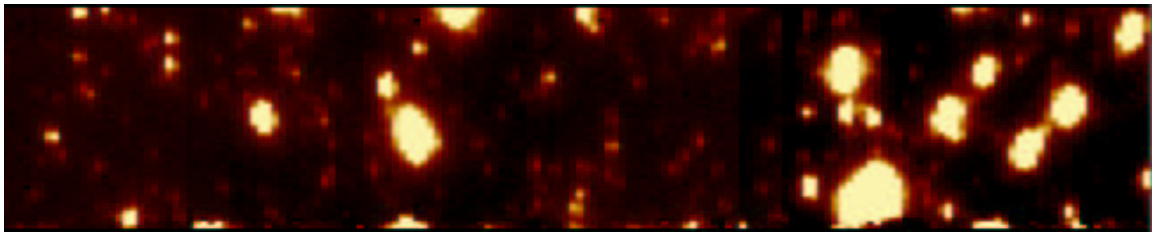


Figure 6.1.0: A coadded cube resulting from several “mosaiced” cubes.

Classification keywords:	Association keywords:	Fig:

`DPR CATG = CALIB`
`DPR TYPE = FLAT,LAMP`
`DPR TECH = IFU`
	`INS SETUP ID`
	`INS OPTI1 NAME`
		6.2.0 (b)

- **Detector linearity Flat Field:**

  DO category: `LINERITY_LAMP`
  Processed by: `si_rec_detlin`

Classification keywords:	Association keywords:	Fig:

`DPR CATG = CALIB`
`DPR TYPE = LINEARITY,LAMP`
`DPR TECH = IFU`
	INS SETUP ID	6.2.0 (b)

- **Flat field frames to compute distortions:**

  DO category: `FLAT_NS`
  Processed by: `si_rec_distortion`

Classification keywords:	Association keywords:	Fig:

`DPR CATG = CALIB`
`DPR TYPE = DISTORTION,FLAT,LAMP`
`DPR TECH = IFU`
	`INS SETUP ID`	6.2.0 (c)

- **Arc frames frames to compute distortions:**

  DO category: `WAVE_NS`
  Processed by: `si_rec_distortion`

Classification keywords:	Association keywords:	Fig:

`DPR CATG = CALIB`
`DPR TYPE = DISTORTION,WAVE,LAMP`
`DPR TECH = IFU`
	`INS SETUP ID`	6.2.0 (d)

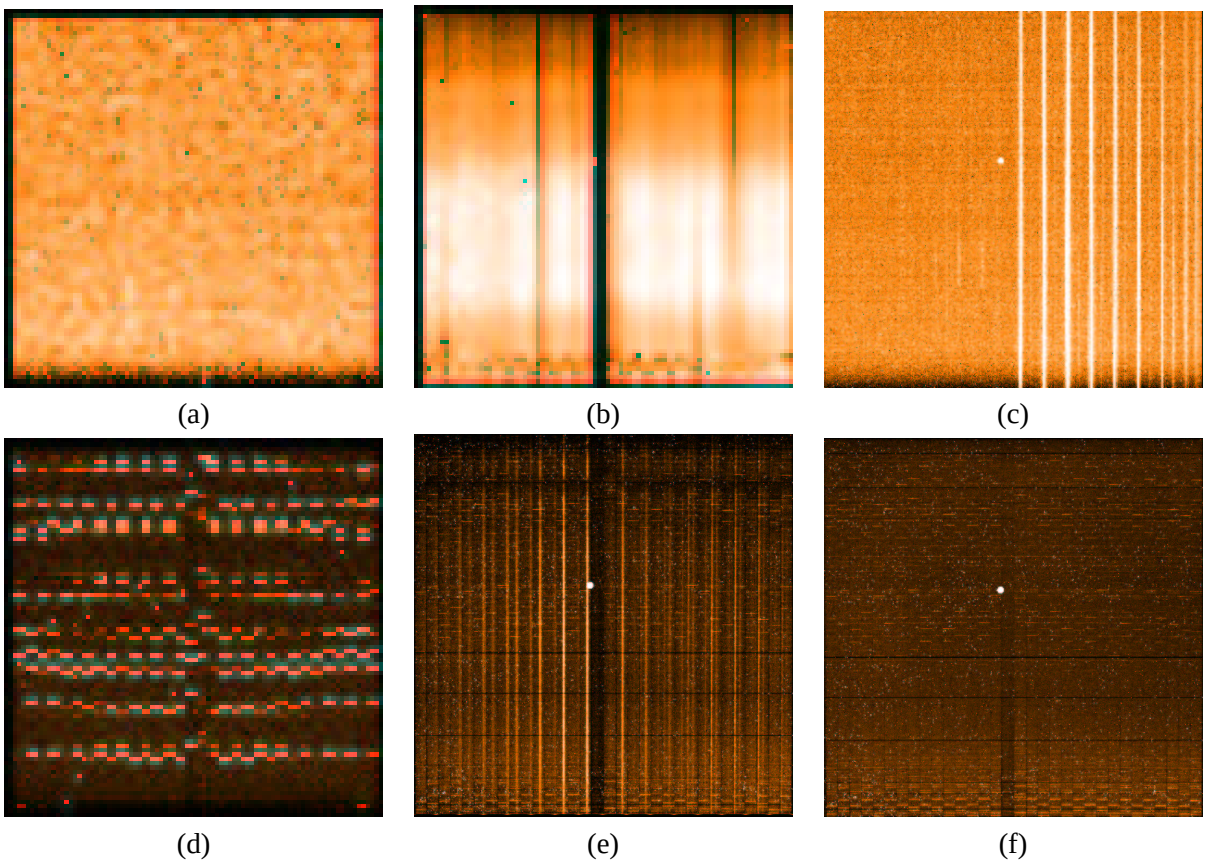


Figure 6.2.0: (a) a raw dark frame with DIT=1; (b) a flat field on in band H, 100mas; (c) a raw frame used to compute optical distortions; (d) an arc lamp on frame; (e) a science frame; (f) a sky frame.

- **Arc frames:**

  DO category: `WAVE_LAMP`
  Processed by: `si_rec_wavecal`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = CALIB`	`INS SETUP ID`	6.2.0 (d)
`DPR TYPE = WAVE,LAMP`	`INS OPTI1 NAME`	
`DPR TECH = IFU`		

- **STD PSF star frames:**

  DO category: `PSF_CALIBRATOR`
  Processed by: `si_rec_psf`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = CALIB`	`INS SETUP ID`	6.2.0 (e)

```
DPR TYPE = PSF-CALIBRATOR        INS OPTI1 NAME
DPR TECH = IFU
```

- **Sky frames taken with STD PSF star frames:**

  DO category: `SKY_PSF_CALIBRATOR`
  Processed by: `si_rec_psf`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = CALIB`	`INS SETUP ID`	6.2.0 (f)
`DPR TYPE = SKY,PSF-CALIBRATOR`	`INS OPTI1 NAME`	
`DPR TECH = IFU`		

- **STD star frames:**

  DO category: `STD`
  Processed by: `si_rec_stdstar`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = CALIB`	`INS SETUP ID`	6.2.0 (e)
`DPR TYPE = STD`	`INS OPTI1 NAME`	
`DPR TECH = IFU`		

- **Sky frames taken with STD star frames:**

  DO category: `SKY_STD`
  Processed by: `si_rec_stdstar`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = CALIB`	`INS SETUP ID`	6.2.0 (f)
`DPR TYPE = SKY,STD`	`INS OPTI1 NAME`	
`DPR TECH = IFU`		

- **Science frames:**

  DO category: `OBJECT_NODDING`
  Processed by: `si_rec_objnod`

Classification keywords:	Association keywords:	Fig:
`DPR CATG = SCIENCE`	`INS SETUP ID`	6.2.0 (e)
`DPR TYPE = OBJECT`	`INS OPTI1 NAME`	
`DPR TECH = IFU,NODDING`		

- **Sky frames taken with science frames:**

  DO category: `SKY_NODDING`
  Processed by: `si_rec_objnod`

Classification keywords:	Association keywords:	Fig:

`DPR CATG = SCIENCE`
`DPR TYPE = SKY`
`DPR TECH = IFU,NODDING`
	`INS SETUP ID`
	`INS OPTI1 NAME`
		6.2.0 (f)

7 Static Calibration Data

In the following section ancillary data required for SINFONI data reduction are listed. For each of them we indicate the corresponding value of the HIERARCH ESO PRO CATG, in short PRO.CATG, FITS keyword. This has to be used to identify the frames listed in the *Set of Frames* (see Section 4.2.2, page 22).

7.1 Line reference table

A reference list of arc lines is necessary to perform the wavelength calibration. Its PRO.CATG is REF_LINE_ARC. This frame is an input of the recipes si_rec_distortion and si_rec_wavecal.

7.2 DRS setup table

There are a few data reduction parameters which are band dependent and are supposed not to change. Those are collected in a table having PRO.CATG DRS_SETUP_WAVE. This frame is an input of the recipes si_rec_distortion and si_rec_wavecal.

band	W_START	W_DISP1	W_DISP2	W_HW	W_FWHM	W_MIN_AMP	W_LOW_POS	W_HI_POS
H	1.65	-2.00018796022e-4	9.3034524287e-10	7	2.83	5	750	1000
H+K	1.95	-5.001433e-4	4.567277e-9	8	3.00	2	1100	1260
K	2.20	-2.51669e-4	1.77136e-9	4	2.00	1	760	900
J	1.25	-1.500581e-4	5.8870678e-10	8	2.00	2	980	1175

This table set the values for data reduction parameters which in section 10 are respectively called **begin_wave**, **guess_disp1**, **guess_disp2**, **half_width**, **fwhm**, **min_amp**, **lo_pos**, **hi_pos**.

7.3 Reference bad pixel map

Detector defects are logged as bad pixels in a special mask having PRO.CATG REF_BP_MAP. This is an input of the recipe si_rec_mflat.

7.4 First column reference table

The data reduction requires as input also a frame indication the first column of the first slitlet. Its PRO.CATG is FIRST_COLUMN. It is an input of the recipes si_rec_wavecal, si_rec_objnod, si_rec_psf, si_rec_stdstar.

8 Data Reduction

In this section, after an overview of the main problems the data reduction needs to solve, we list the required data and the recipes which allow to solve them, giving the data reduction sequence necessary to reduce calibration and science data.

8.1 Data reduction overview

The principle of integral field spectroscopy is described by figure 8.1.0. A two-dimensional image of the sky is separated by an image slicer into several components. Those are then aligned on a slit and dispersed to separate its spectral information and then imaged on a detector. The main SINFONI data reduction problems to solve are the following.

- Correct for the detector signature: bad pixels, detector contribution to the measured signal, flat fielding (correct pixel to pixel gain variations and relative slitlet throughput differences), correct geometric distortions.
- Perform the wavelength calibration.
- Reconstruct the image FOV from the 32 image slices in a format which contains both the spatial and the spectral information.
- Devise proper calibrations and observations to be able to properly correct the emission from the sky, from the instrument and from the telescope which are very strong in the NIR. This requires to take sky frames together with the object frames in the night observations, daily calibrations with the flat lamp switched on and off, and possibly dark frames.

In the following description we also indicate in parenthesis for each frame the corresponding PRO.CATG. To locate the detector bad pixels one uses a bad pixel map. A master bad pixel map resulting from the combination of a set of (different) bad pixel maps is generated by the master flat recipe. First of all, as the detector is known to have construction defects, these will be indicated by a reference bad pixel map (REF_BP_MAP). Hot pixels will be determined on dark frames (BP_MAP_HP). Non-linear response pixels are instead indicated by a bad pixel map (BP_MAP_NL) obtained by evaluating the pixel response of a set of flat exposures of increasing intensity. Other bad pixels (BP_MAP_NO) are determined on a set of flat fields (on the master flat field).
A master flat field (MASTER_FLAT_LAMP) generated from a set of raw flat fields, is used to correct the different detector pixel sensitivities. It is known that the image sliced and projected on the detector is affected by distortions. The si_rec_distortion recipe computes the distortions (DISTORTION) and the slitlet distances (SLITLETS_DISTANCE). It uses a set of raw frames where only the first column of each slitlet is illuminated through fibres. In addition flats and arcs are taken within the north south template as required to reduce the data. A set of “on” and “off” arc lamp frames, a reference line table, a master flat field, the optical distortions map and a good guess of the slitlet positions are input of the wavecal recipe. This recipe determines a wavelength map and obtains a better computation of the left edge position of each slitlet.
Science (or PSF or telluric standard) frames are corrected for sky background, flat field, distortions, and calibrated in wavelength. The sorted slitlets are then stacked in a cube, taking into account the relative distances of the position of the edge of each slitlet to the one of the first slitlet, with a final image realignment to get sub

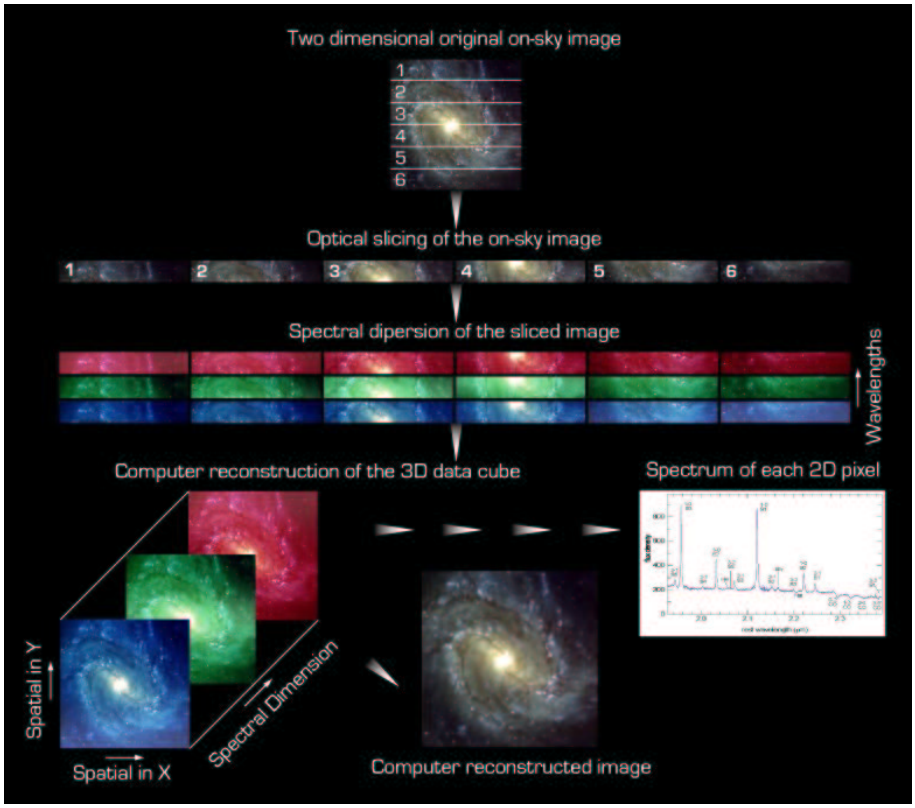


Figure 8.1.0: Integral Field Spectroscopy data reduction principle

pixel accuracy. The final product is thus a 3D data cube where the full spatial information is stored along the X and Y directions, and the wavelength information is stored along the Z direction. Each plane of the cube is a monochromatic reconstruction of the SINFONI FOV.

The north south test template traces each of the 32 slitlets by only one fibre exposure; therefore non linearities of the image scale within the 64 pixel of a single slitlets are currently not corrected and could cause minor slice to slit ripples in the reconstructed cube.

8.2 Required input data

To be able to reduce science data one needs to use raw, product data and pipeline recipes in a given sequence which provides all the input necessary to each pipeline recipe. We call this sequence a data reduction cascade. The SINFONI data reduction cascade involves the following input data:

- Reference files:
  - A reference bad pixel map, indicating known detector defects.
  - A list of arc lamp emission lines containing vacuum wavelengths and predicted intensities for wavelength calibration.

  - The DRS_SETUP_WAVE table as input of si_rec_distortion and si_rec_wavecal to specify parameter peculiar of the wavelength calibration algorithm.
  - A constant position assumed for the first column.

- Raw frames:
  - Linearity flat frames, to determine a map of non linear bad pixels.
  - Darks, to determine master darks.
  - Flat fields, to determine master flats.
  - Fibre frames, to trace the first column of each slitlet and, using also on/off lamp flats and arc lamp frames, to compute the optical distortions and slitlet distances.
  - Arc lamps, to perform the wavelength calibration.
  - Sky frames, to evaluate and subtract the strong and time-variable NIR sky emission.
  - Telluric STD star frames, to correct telluric absorption features.
  - PSF standards, to evaluate the strehl.
  - Science frames, to finally do science.

- Calibration data products.
  - Bad pixel maps, to correct for the detector defects.
  - Distortion coefficients, to correct for the optical distortions.
  - Master flats, to correct for different detector pixel efficiencies.
  - Master darks, to correct for the instrument bias if no off or sky frames are available.
  - Slitlet distances, to be able to properly reconstruct a cube.
  - Slitlets' left edge positions, to be able to properly reconstruct a cube.
  - Wavelength maps, to obtain a cube calibrated in wavelength.

Calibration data products can be generated from raw data using the pipeline recipes. Alternatively the user may use calibration products obtained from the ESO archive or from the ESO Data Flow Operation department. Master bad pixel maps, the bad pixel maps coming from the standard flats, the master flats, the slitlets position table, the wavelength map depend from the observed band and instrument's pre-optic. Bad pixel maps coming from the non linearity test, distortion tables, slitlet distances, reference line tables depend only on the observed band. The first column table, the reference bad pixel map, and master darks depend neither from the band nor the pre-optic. Science data requiring master dark need to have matching values of the FITS keyword HIERARCH ESO DET DIT.

8.3 Reduction cascade

The SINFONI data reduction follows the following sequence. A short description of the available recipes is given in section 4.1. In parenthesis we provide the value of the DO category corresponding to each frame.

- Run **si_rec_detlin** on a set of flats with increasing intensity (LINEARITY_LAMP) to determine the non linearity pixels bad pixel map (BP_MAP_NL).

- Run **si_rec_mdark** on a set of raw darks (DARK) to determine the master dark (MASTER_DARK) and the hot pixels bad pixel map (BP_MAP_HP). This map depends on DIT.
- Run **si_rec_mflat** on a set of standard flat fields (FLAT_LAMP), the BP_MAP_NL and the REF_BP_MAP to determine the master bad pixel map (MASTER_BP_MAP) and the master lamp flat (MASTER_FLAT_LAMP).
- Run **si_rec_distortion** on a set of fibre flats (FIBRE_NS), on/off arc lamps (WAVE_NS) and on/off lamp flats (FLAT_NS), using a reference line table (REF_LINE_ARC) to determine the optical distortions (DISTORTION) and the slitlet distances (SLITLET_DISTANCES). To set a few data reduction parameters which depends from the observed band and used instrument pre-optics the user has also to provide in input a DRS_SETUP_WAVE table frame.
- Run **si_rec_wavecal** on a set of arc lamp frames (FLAT_WAVE), a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, and a REF_LINE_ARC to determine the wavelength map (WAVE_MAP) and the slitlet edge position table (SLIT_POS). To set a few data reduction parameters which depends from the observed band and used instrument pre-optics the user has also to provide in input a DRS_SETUP_WAVE table. If the parameter **wcal-slitpos_bootstrap** has value set to FALSE, as we suggest for robustness, the user need to provide in input also an appropriate SLIT_POS table, for example the one we provide as part of dtata reduction kit.
- Run **si_rec_psf** on PSF standards and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS, a WAVE_MAP, and a FIRST_COL to reduce the PSF standard and get information on the instrument's strehl.
- Run **si_rec_stdstar** on a reference telluric standard (STD) and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS, a WAVE_MAP, and a FIRST_COL to reduce the telluric standards and get information on the instrument's response.
- Run **si_rec_objnod** on your scientific data (OBJECT_NODDING) and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS, a WAVE_MAP and a FIRST_COL to reduce science data.

The main data products involved in the data reduction cascade are indicated in the SINFONI association map shown in Figure 8.3.1. It summarise dependencies between raw data, calibration products and recipes involved in the correction of the instrument signature and reduction of science data. Examples of set of frames input for each recipe are provided in section 9.

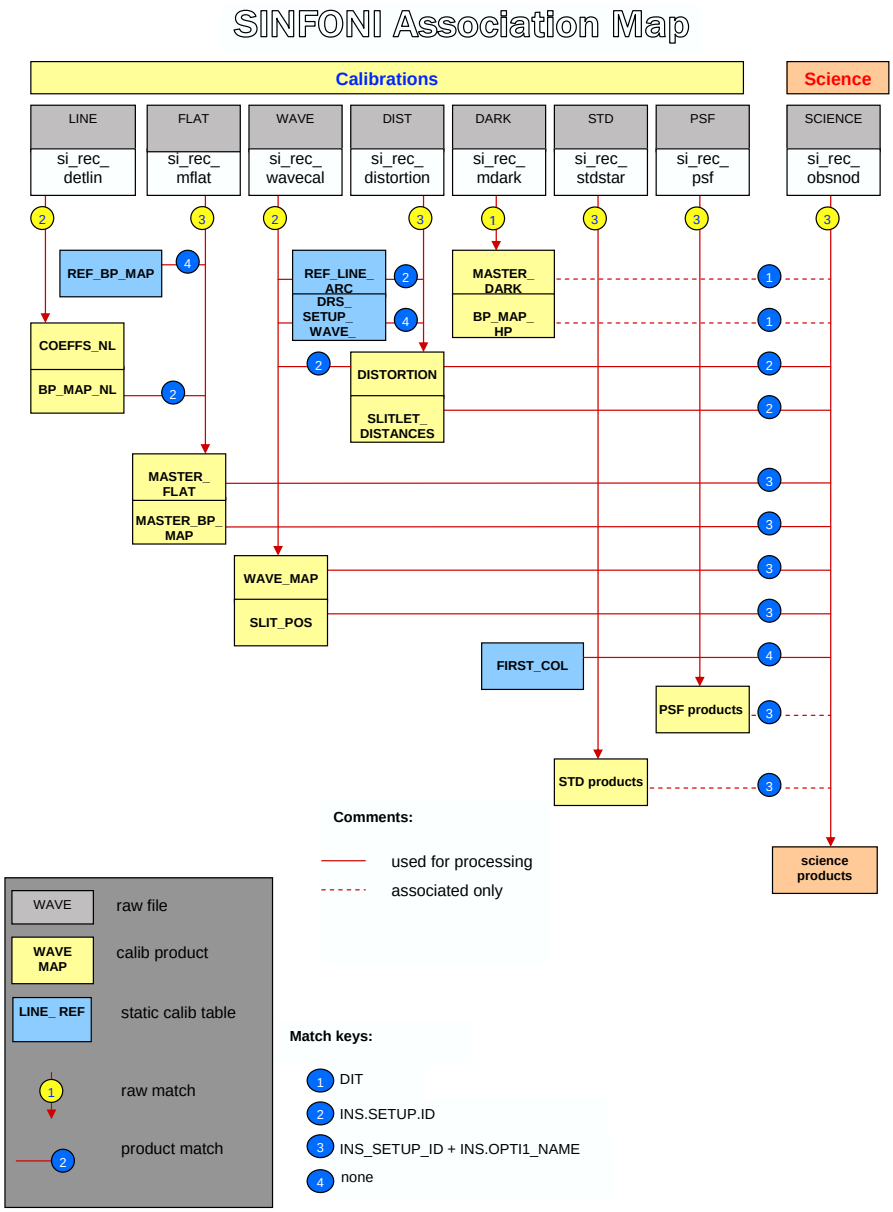


Figure 8.3.1: *SINFONI Association Map*

9 Pipeline Recipes Interfaces

In this section we provide for each recipe examples of the required input data (and their tags). In the following we assume that /path_file_raw/filename_raw.fits and /path_file_cdb/filename_cdb.tfits are existing FITS files (e.g. /data1/sinfoni/com2/SINFO.2004-08-16T02:54:04.353.fits and /cal/sinfo/ifu/cal/DISTORTION_K.tfits).

We also provide a list of the pipeline products for each recipe, indicating their default recipe name (eventually replaced by esorex to a given standard), the value of the FITS keyword HIERARCH ESO PRO CATG (in short PRO.CATG) and a short description. The relevant keywords are PRO.CATG, used to classify each frame, and to associate to each raw frame the proper calibration frame:

Association keyword	Information
HIERARCH ESO INS SETUP ID	band
HIERARCH ESO INS OPTI1 NAME	Pixel scale
HIERARCH ESO DET DIT	Integration time

For each recipe we also list in a table the input parameters (as they appear in the recipe configuration file), the corresponding aliases (the corresponding names to be eventually set on command line) and their default values. Also quality control parameters are listed. Those are stored in relevant pipeline products. More information on instrument quality control can be found on http://www.eso.org/qc

We distinguish between recipes involved in the data reduction cascade (having prefix si_rec) and user utilities (with prefix si_utl).

9.1 si_rec_detlin

The recipe si_rec_detlin computes the detector responsivity as a function of the pixel intensity and determines when it becomes non linear.

9.1.1 Input

```
/path_file_raw/SINFO.2004-08-16T02:54:04.353.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:53:37.089.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:52:23.028.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:51:59.774.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:50:38.991.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:50:11.797.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:49:04.887.fits LINEARITY_LAMP
/path_file_raw/SINFO.2004-08-16T02:48:36.792.fits LINEARITY_LAMP
```

9.1.2 Output

default recipe file name	PRO.CATG	short description

lin_det_info.tfits	LIN_DET_INFO	Table with coefficients of non linear fit to median of each flat image image
gain_info.tfits	GAIN_INFO	Table with detector's gain values
out_bplin_coeffsCube.fits	BP_COEFF	image with coefficients of non linear fit to pixel's intensity used to evaluate non linearity
out_bp_lin.fits	BP_MAP_NL	Non linear bad pixel map

9.1.3 Quality control

The pipeline computes the non linear coefficients, the number of non linear pixels per grating, the detector gain.

Detector non linearity The detector non linearity is computed as described in 10.1.3. The computed coefficients are QC.BP-MAP.LINi.MED (i=0,1,2).

A different method is applied to determine the same quantity as described in 10.1.3. The computed coefficients are QC.BP-MAP.LINi.MEAN (i=0,1,2).

Non linear bad pixels The pipeline computes the number of non linear bad pixels. Those are given by QC.BP-MAP.NBADPIX and are obtained with the method QC.BP-MAP.METHOD.

Detector gain The detector gain is computed as described in 10.1.5 and is given by the value of QC.GAIN.

9.1.4 Parameters

parameter	alias	default
sinfoni.bp_lin.order	bp_lin-order	2
sinfoni.bp_lin.thresh_sigma_factor	bp_lin-thresh_sigma_fct	10.0
sinfoni.bp_lin.nlin_threshold	bp_lin-nlin_threshold	0.5
sinfoni.bp_lin.low_rejection	bp_lin-lo_rej	10.0
sinfoni.bp_lin.high_rejection	bp_lin-hi_rej	10.0

9.2 si_rec_mdark

The recipe si_rec_mdark generates a master dark from a set of raw darks by stacking frames with rejection of outliers. It also generates a bad pixel map flagging the hot-current pixels.

9.2.1 Input

```
/path_file_raw/SINFO.2004-08-16T01:24:53.070.fits DARK
/path_file_raw/SINFO.2004-08-16T01:09:22.905.fits DARK
```

```
/path_file_raw/SINFO.2004-08-16T00:53:51.890.fits DARK
/path_file_raw/SINFO.2004-08-16T00:38:14.994.fits DARK
```

9.2.2 Output

default recipe file name	PRO.CATG	short description
out_bp_noise.fits	BP_MAP_HP	bad pixel map, method=”Noise”
out_dark.fits	MASTER_DARK	master dark

9.2.3 Quality control

Dark frames are processed to monitor the RON, Read Out Noise per DIT, the FPN, Fixed Patter Noise per DIT, the detector counts per DIT, the number of hot pixels per DIT.

RON The RON is computed on the whole detector chip and given as value of the QC.RON parameter. For quality control those values are monitored as a function of time and DIT. Two consecutive frames are subtracted from each other and the median standard deviation of a limited number of samples is taken and normalised to DET.NDIT=1. The RON is computed in two regions and is given by the values of QC.RON1 and QC.RON2.

Dark median counts The median and standard deviation of the counts in the master dark frame are monitored by DFO. Its value and standard deviation are given by the values of QC.DARKMED.AVE and QC.DARKMED.STDEV

Fixed Pattern Noise A histogram of the master dark is produced, and a fit is applied; the standard deviation (sigma) of the Gaussian is the FPN. This value, logged by parameter QC.DARKFPN, is monitored for different DITs. The FPN should scale linearly with the number of counts. For this reason the ratio FPN/counts is monitored for different DITs.

Number of hot pixels The number of pixels having an intensity greater than a threshold is monitored in the parameter QC.BP-MAP.NBADPIX

9.2.4 Parameters

parameter	alias	default
sinfoni.bp_noise.thresh_sigma_factor	bp_noise-thresh_sigma_fct	10.0
sinfoni.bp_noise.low_rejection	bp_noise-lo_rej	10.0
sinfoni.bp_noise.high_rejection	bp_noise-hi_rej	10.0
sinfoni.dark.low_rejection	dark-lo_rej	0.1
sinfoni.dark.high_rejection	dark-hi_rej	0.1
sinfoni.dark.qc_ron_xmin	dark-qc_ron_xmin	1

sinfoni.dark.qc_ron_xmax	dark-qc_ron_xmax	2048
sinfoni.dark.qc_ron_ymin	dark-qc_ron_ymin	1
sinfoni.dark.qc_ron_ymax	dark-qc_ron_ymax	2048
sinfoni.dark.qc_ron_hsize	dark-qc_ron_hsize	4
sinfoni.dark.qc_ron_nsamp	dark-qc_ron_nsamp	100
sinfoni.dark.qc_fpn_xmin	dark-qc_fpn_xmin	1
sinfoni.dark.qc_fpn_xmax	dark-qc_fpn_xmax	2047
sinfoni.dark.qc_fpn_ymin	dark-qc_fpn_ymin	1
sinfoni.dark.qc_fpn_ymax	dark-qc_fpn_ymax	2047
sinfoni.dark.qc_fpn_hsize	dark-qc_ron_hsize	2
sinfoni.dark.qc_fpn_nsamp	dark-qc_ron_nsamp	1000

9.3 si_rec_mflat

The recipe si_rec_mflat computes a master flat field frame and a bad pixel map indicating pixels whose intensity is beyond a given threshold.

9.3.1 Input

```
/path_file_raw/SINFO.2005-02-28T16:27:43.232.fits FLAT_LAMP
/path_file_raw/SINFO.2005-02-28T16:28:05.846.fits FLAT_LAMP
/path_file_raw/SINFO.2005-02-28T16:28:18.820.fits FLAT_LAMP
/path_file_raw/SINFO.2005-02-28T16:28:32.593.fits FLAT_LAMP
/path_file_cdb/REF_BP_MAP.fits REF_BP_MAP
/path_file_cdb/BP_MAP_NL_H_025.fits BP_MAP_NL
/path_file_cdb/BP_MAP_H_025.fits BP_MAP     **
/path_file_cdb/SLIT_POS_H_025.fits SLIT_POS     **
```

With ** we have listed an additional frames which are required only if **sinfoni.lamp_flats.interpol_index**==TRUE (usually not the case).

9.3.2 Output

default recipe file name	PRO.CATG	short description
out_flat.fits	MASTER_FLAT_LAMP	master flat field
out_bp_norm.fits	BP_MAP_NO	“Above threshold” bad pixel map
out_bpmap_sum.fits	MASTER_BP_MAP	master bad pixel map

9.3.3 Quality control

With this recipe one can monitor the lamp efficiency for each grating, the number of bad pixels for each grating, the number of counts in the lamp-off frames, the fixed patter noise in two given regions.

Lamp flux The flux of the halogen lamp as measured by the detector depends on the intrinsic brightness of the lamp but also on the spectroscopic setting and the alignment of optical elements. The lamp-off frame is subtracted from the lamp-on frame. The fluxes on this difference frame as computed by the recipe are monitored from the parameter QC.SPECFLAT.NCNTSAVG (the standard deviation of those values is given by QC.SPECFLAT.NCNTSSTD). Day-time spectral flats calibrations come in five pairs of lamp-on and lamp-off frames. Night time calibrations (attached calibrations) come as one pair.

For each lamp-on frame the corresponding lamp-off frame is subtracted and the median is calculated. The QC parameter QC.SPECFLAC.OFFFLUX is the average of these 5 values.

Bad pixels The number of bad pixels found on the master lamp flat are monitored from the value of QC.BPMAP.NBADPIX. The number of bad pixel in the master bad pixel map is given by the value of QC.MBPMAP.NBADPIX

Lamp-off flux It is important to monitor the light/heat contamination in the optical path. The flux measured in lamp-off frames is usually a few counts above the reset anomaly at the same DIT (the dark counts), since a broad band filter is used for the lamp-off frames, while the dark frames are taken with two excluding narrow band filters.The pipeline monitors HIERARCH ESO QC SPECFLAT OFFFLUX, that is the average of the (five) off-lamp medians.

Fixed Patter Noise The pipeline computes the Fixed Pattern noise, in a selected region on the array. This is a simple standard deviation of the flat product frame. The region is specified in the si_rec_mflat.rc file. Two regions are monitored by parameters QC.LFLAT.FPN1 and QC.LFLAT.FPN2. During the observing period P75 the first region was the central quarter [@512,@512:@1536,@1536]. In the same observing period the second region was an area on one single slitlet [@1350,@1000:@1390,@1200].

9.3.4 Parameters

parameter	alias	default
sinfoni.bp_norm.sigma_factor	bp_norm-s_factor	5.0
sinfoni.bp_norm.method_index	bp_norm-method_ind	1
sinfoni.bp_norm.factor	bp_norm-fct	10.0
sinfoni.bp_norm.iterations	bp_norm-it	8
sinfoni.bp_norm.low_rejection	bp_norm-lo_rej	0.1
sinfoni.bp_norm.high_rejection	bp_norm-hi_rej	0.1
sinfoni.bp_norm.llx	bp_norm-llx	270
sinfoni.bp_norm.lly	bp_norm-lly	1000
sinfoni.bp_norm.urx	bp_norm-urx	310
sinfoni.bp_norm.ury	bp_norm-ury	1200
sinfoni.bp_norm.threshold_index	bp_norm-thr_ind	TRUE
sinfoni.bp_norm.mean_factor	bp_norm-mean_fct	100.0
sinfoni.bp_norm.min_cut	bp_norm-min_cut	0.0
sinfoni.bp_norm.max_cut	bp_norm-max_cut	5e+04

sinfoni.lamp_flats.low_rejection	lamp_flats-lo_rej	0.1
sinfoni.lamp_flats.high_rejection	lamp_flats-hi_rej	0.1
sinfoni.lamp_flats.interpol_index	lamp_flats-interpol_index	FALSE
sinfoni.lamp_flats.max_rad	lamp_flats-max_rad	4
sinfoni.lamp_flats.bad_ind	lamp_flats-bad_ind	FALSE
sinfoni.lamp_flats.sigma_factor	lamp_flats-sigma_factor	5.0
sinfoni.lamp_flats.factor	lamp_flats-factor	3.0
sinfoni.lamp_flats.iterations	lamp_flats-iterations	8
sinfoni.lamp_flats.bad_low_rejection	lamp_flats-bad_lo_rej	10.0
sinfoni.lamp_flats.bad_high_rejection	lamp_flats-bad_hi_rej	10.0
sinfoni.lamp_flats.llx	lamp_flats-llx	1350
sinfoni.lamp_flats.lly	lamp_flats-lly	1000
sinfoni.lamp_flats.urx	lamp_flats-rrx	1390
sinfoni.lamp_flats.ury	lamp_flats-ury	1200
sinfoni.lamp_flats.thresh_ind	lamp_flats-tresh_ind	FALSE
sinfoni.lamp_flats.mean_factor	lamp_flats-mean_factor	10.0
sinfoni.lamp_flats.qc_fpn_xmin1	lamp_flats-qc_fpn_xmin1	512
sinfoni.lamp_flats.qc_fpn_xmax1	lamp_flats-qc_fpn_xmax1	1536
sinfoni.lamp_flats.qc_fpn_ymin1	lamp_flats-qc_fpn_ymin1	512
sinfoni.lamp_flats.qc_fpn_ymax1	lamp_flats-qc_fpn_ymax1	1536
sinfoni.lamp_flats.qc_fpn_xmin2	lamp_flats-qc_fpn_xmin2	1350
sinfoni.lamp_flats.qc_fpn_xmax2	lamp_flats-qc_fpn_xmax2	1390
sinfoni.lamp_flats.qc_fpn_ymin2	lamp_flats-qc_fpn_ymin2	1000
sinfoni.lamp_flats.qc_fpn_ymax2	lamp_flats-qc_fpn_ymax2	1200
sinfoni.lamp_flats.qc_thresh_min	lamp_flats-qc_thresh_min	0
sinfoni.lamp_flats.qc_thresh_max	lamp_flats-qc_thresh_max	49000

9.4 si_rec_distortion

The recipe si_rec_distortion is used to compute the optical distortion. It also computes the relative distances of the slitlets from the first one.

9.4.1 Input

```
/path_file_raw/SINFO.2004-08-16T13:17:28.050.fits FIBRE_NS
/path_file_raw/SINFO.2004-08-16T13:16:56.095.fits FIBRE_NS
/path_file_raw/SINFO.2004-08-16T13:16:34.352.fits FIBRE_NS
/path_file_raw/SINFO.2004-08-16T13:16:14.839.fits FIBRE_NS
......
/path_file_raw/SINFO.2004-08-16T13:15:55.156.fits FIBRE_NS
/path_file_raw/SINFO.2004-08-16T13:16:55.156.fits WAVE_NS
/path_file_raw/SINFO.2004-08-16T13:17:55.156.fits WAVE_NS
/path_file_raw/SINFO.2004-08-16T13:18:55.156.fits FLAT_NS
/path_file_raw/SINFO.2004-08-16T13:19:55.156.fits FLAT_NS
```

```
/path_file_cdb/neonK.tfits REF_LINE_ARC
/path_file_cdb/drs_setup_wave.tfits DRS_SETUP_WAVE
/path_file_cdb/BP_MAP_H_025.fits BP_MAP     **
/path_file_cdb/SLIT_POS_H_025.fits SLIT_POS     **
```

With ** we have listed an additional frames which are required only if **sinfoni.lamp_flats.interpol_index**==TRUE (usually not the case).

9.4.2 Output

default recipe file name	PRO.CATG	short description
out_flat.fits	MASTER_FLAT_LAMP	master flat field
out_bp_dist.fits	BP_MAP_DI	“Above threshold” bad pixel map
out_ns_stack_off.fits	FIBRE_NS_STACKED_OFF	fake off frame from FIBRE,NS
out_ns_stack_on.fits	FIBRE_NS_STACKED_ON	fake on frame from FIBRE,NS
out_ns_stack.fits	FIBRE_NS_STACKED	fake on-off
out_wcal_stack.fits	WAVE_LAMP_STACKED	stacked arc frame
out_slitlets_pos_predist.tfits	SLITLETS_POS_PREDIST	computed slitlets edge positions before distortion correction
out_distortion.tfits	DISTORTION	computed optical distortions
out_ns_stack_warp.fits	FIBRE_NS_STACKED_DIST	on-off frame corrected for distortions
out_ns.fits	MASTER_SLIT	on-off frame multiplied by BP map
out_distances.tfits	SLITLETS_DISTANCE	computed slitlet distances

9.4.3 Quality control

The recipe si_rec_distortion generates the DISTORTION product, that is a FITS table containing the coefficients of the optical distortion polynomial. Polynomial coefficients are monitored by QC.COEFFij, ij=00, 10, 01, 11, 20, 02, 21, 12.

Since the coefficients may be difficult, the optical distortion is applied to five points on the chip. These are the four quadrant centers and the center of the array: (512, 512), (512,1536), (1536,512), (1536,1536) and (1024,1024). The plot shows the location of the five points on the array (like the five symbol on a dice) and the applied shifts in x (along a row), enhanced by a given factor. The optical distortion will shift and shrink the initial dice figure. The QC parameters defined to monitor those shifts are QC.XSHIFT.CC, QC.XSHIFT.LL, QC.XSHIFT.LR, QC.XSHIFT.UR, QC.XSHIFT.UL.

The second product of the north south test recipe is the SLITLETS_DISTANCES product, the relative distance between the 32 slitlets, monitored by QC.SL.DISTi, =0,30.

Trending describes the variation of a QC1 parameter with time. The following QC1 parameters are determined and monitored:

The recipe si_rec_distortion determines the distortion in terms of polynomial coefficients

C0 + C1 X + C2 Y + C3 XY + C4 XX + C5 YY + C6 XXY + C7 XYY

The recipe also determines the relative pixel distance between consecutive slitlets, a value around 64.

Distortion coefficients : The coefficients C0 to C7 are monitored

Reference pixels shift : Distortion coefficients are applied to five points on the chip.

Slitlets Distances : the average distance between slitlets is monitored together with his uncertainty by QC parameters QC.SL.DISTAVG, QC.SL.DISTRMS.

9.4.4 Parameters

parameter	alias	default
sinfoni.lamp_flats.low_rejection	lamp_flats-lo_rej	0.1
sinfoni.lamp_flats.high_rejection	lamp_flats-hi_rej	0.1
sinfoni.lamp_flats.interpol_index	lamp_flats-interpol_index	FALSE
sinfoni.lamp_flats.max_rad	lamp_flats-max_rad	4
sinfoni.lamp_flats.bad_ind	lamp_flats-bad_ind	FALSE
sinfoni.lamp_flats.sigma_factor	lamp_flats-sigma_factor	5.0
sinfoni.lamp_flats.factor	lamp_flats-factor	3.0
sinfoni.lamp_flats.iterations	lamp_flats-iterations	8
sinfoni.lamp_flats.bad_low_rejection	lamp_flats-bad_lo_rej	10.0
sinfoni.lamp_flats.bad_high_rejection	lamp_flats-bad_hi_rej	10.0
sinfoni.lamp_flats.llx	lamp_flats-llx	1350
sinfoni.lamp_flats.lly	lamp_flats-lly	1000
sinfoni.lamp_flats.urx	lamp_flats-rrx	1390
sinfoni.lamp_flats.ury	lamp_flats-ury	1200
sinfoni.lamp_flats.thresh_ind	lamp_flats-tresh_ind	FALSE
sinfoni.lamp_flats.mean_factor	lamp_flats-mean_factor	10.0
sinfoni.lamp_flats.qc_fpn_xmin1	lamp_flats-qc_fpn_xmin1	512
sinfoni.lamp_flats.qc_fpn_xmax1	lamp_flats-qc_fpn_xmax1	1536
sinfoni.lamp_flats.qc_fpn_ymin1	lamp_flats-qc_fpn_ymin1	512
sinfoni.lamp_flats.qc_fpn_ymax1	lamp_flats-qc_fpn_ymax1	1536
sinfoni.lamp_flats.qc_fpn_xmin2	lamp_flats-qc_fpn_xmin2	1350
sinfoni.lamp_flats.qc_fpn_xmax2	lamp_flats-qc_fpn_xmax2	1390
sinfoni.lamp_flats.qc_fpn_ymin2	lamp_flats-qc_fpn_ymin2	1000
sinfoni.lamp_flats.qc_fpn_ymax2	lamp_flats-qc_fpn_ymax2	1200
sinfoni.lamp_flats.qc_thresh_min	lamp_flats-qc_thresh_min	0
sinfoni.lamp_flats.qc_thresh_max	lamp_flats-qc_thresh_max	49000
sinfoni.bp.method	bp-method	Normal
sinfoni.bp_dist.sigma_factor	bp_dist-s_factor	2.0
sinfoni.bp_dist.method_index	bp_dist-method_ind	1

sinfoni.bp_dist.factor	bp_dist-fct	999.0
sinfoni.bp_dist.iterations	bp_dist-it	8
sinfoni.bp_dist.low_rejection	bp_dist-lo_rej	0.1
sinfoni.bp_dist.high_rejection	bp_dist-hi_rej	0.1
sinfoni.bp_dist.llx	bp_dist-llx	1350
sinfoni.bp_dist.lly	bp_dist-lly	1000
sinfoni.bp_dist.urx	bp_dist-urx	1390
sinfoni.bp_dist.ury	bp_dist-ury	1200
sinfoni.bp_dist.threshold_index	bp_dist-thr_ind	TRUE
sinfoni.bp_dist.mean_factor	bp_dist-mean_fct	999.0
sinfoni.bp_dist.min_cut	bp_dist-min_cut	0.0
sinfoni.bp_dist.max_cut	bp_dist-max_cut	5e+04
sinfoni.stacked.low_rejection	stack-lo_rej	0.1
sinfoni.stacked.high_rejection	stack-hi_rej	0.1
sinfoni.stacked.flat_index	stack-flat_ind	TRUE
sinfoni.stacked.mask_index	stack-mask_ind	1
sinfoni.stacked.ind_index	stack-ind_ind	FALSE
sinfoni.stacked.mask_rad	stack-mask_rad	4
sinfoni.stacked.gauss_index	stack-gauss_ind	FALSE
sinfoni.stacked.kernel_half_width	stack-khw	2
sinfoni.stacked.warpfix_ind	stack-warpfix_ind	TRUE
sinfoni.stacked.warpfix_kernel	stack-warpfix_kernel	tanh
sinfoni.stack.qc_thresh_min	stack-qc_thresh_min	0
sinfoni.stack.qc_thresh_max	stack-qc_thresh_max	64000
sinfoni.distortion.calib_indicator	dist-calib_indicator	TRUE
sinfoni.distortion.min_diff_mean_med_col_int	dist-min_diff_mean_med_col_int	10.0
sinfoni.distortion.half_width	dist-hw	7
sinfoni.distortion.sigma	dist-sigma	2.0
sinfoni.distortion.fwhm	dist-fwhm	2.0
sinfoni.distortion.min_amplitude	dist-min_amplitude	5.0
sinfoni.distortion.max_residual	dist-max_residual	0.5
sinfoni.distortion.n_a_coefficients	dist-n_a_coeffs	4
sinfoni.distortion.n_b_coefficients	dist-n_b_coeffs	2
sinfoni.distortion.sigma_factor	dist-sigma_factor	1.5
sinfoni.distortion.write_coeffs_ind	dist-wcoeff_ind	TRUE
sinfoni.distortion.write_par_ind	dist-par_ind	TRUE
sinfoni.distortion.pixel_dist	dist-pixel-dist	12
sinfoni.distortion.pixel_tol	dist-pixel_tol	3.0
sinfoni.distortion.wave_map_ind	dist-wave_map_ind	FALSE
sinfoni.distortion.mag_factor	dist-mag_factor	8
sinfoni.distortion.slit_pos_indicator	dist-slit_pos_ind	TRUE
sinfoni.distortion.fit_boltz_indicator	dist-fit_boltz_ind	TRUE
sinfoni.distortion.fit_edge_indicator	dist-fit_edge_ind	FALSE
sinfoni.distortion.estimate_indicator	dist-estimate_ind	FALSE
sinfoni.distortion.box_length	dist-box_len	32

sinfoni.distortion.y_box	dist-y_box	5.0
sinfoni.distortion.diff_tol	dist-diff_toll	2.0
sinfoni.distortion.qc_thresh_min	dist-qc_thresh_min	0
sinfoni.distortion.qc_thresh_max	dist-qc_thresh_max	64000
sinfoni.distortion.lower_rejection	ns-lo_rejection	0.1
sinfoni.distortion.higher_rejection	ns-hi_rejection	0.1
sinfoni.distortion.mask_ind	ns-mask_ind	FALSE
sinfoni.distortion.gauss_ind	ns-gauss_ind	FALSE
sinfoni.distortion.kernel_half_width	ns-khw	2
sinfoni.distortion.ns_half_width	ns-hw	4
sinfoni.distortion.ns_fwhm	ns-fwhm	2.0
sinfoni.distortion.min_diff	ns-min_diff	1.0
sinfoni.distortion.dev_tol	ns-dev_tol	20.0
sinfoni.north_south_test.low_rejection	ns-lo_rej	0.1
sinfoni.north_south_test.high_rejection	ns-hi_rej	0.1
sinfoni.north_south_test.mask_ind	ns-mask_ind	FALSE
sinfoni.north_south_test.gauss_ind	ns-gauss_ind	FALSE
sinfoni.north_south_test.kernel_half_width	ns-khw	2
sinfoni.north_south_test.half_width	ns-hw	4
sinfoni.north_south_test.fwhm	ns-fwhm	2.0
sinfoni.north_south_test.min_diff	ns-min_diff	1.0
sinfoni.north_south_test.dev_tol	ns-dev_tol	20.0

9.5 si_rec_wavecal

The recipe si_rec_wavecal is used to determine the wavelength dispersion coefficients and construct a wavelength calibration map. It also determines the positions of the edges of each slitlet.

9.5.1 Input

```
/path_file_raw/SINFO.2004-08-15T11:26:49.348.fits WAVE_LAMP
/path_file_raw/SINFO.2004-08-15T11:26:19.304.fits WAVE_LAMP
/path_file_cdb/neonK.tfits REF_LINE_ARC
/path_file_cdb/MASTER_FLAT_LAMP_K_100.fits MASTER_FLAT_LAMP
/path_file_cdb/MASTER_BP_MAP_K_100.fits MASTER_BP_MAP
/path_file_cdb/DISTORTION_K.tfits DISTORTION
/path_file_cdb/drs_setup_wave.tfits DRS_SETUP_WAVE
/path_file_cdb/SLIT_POS_K_100.tfits SLIT_POS
```

The slitlet position table, classified as SLIT_POS, is a necessary input if the parameter **wcal-slitpos_bootstrap** is set to FALSE, which we suggest for robustness. As this is also a product of this step, we have provided a complete list of master calibrations for this frame as part of the data reduction kit. If the user would like to set the parameter **wcal-estimate_ind** to TRUE, an additional input table to have a guess of the slitlet positions must be provided, with the classification SLIT_POS_GUESS. The user may for example use for this table a

copy of the appropriate (band,pre-optics) provided SLIT_POS. If the parameter **wcal-calib_indicator** is set to FALSE, the user has to provide also the parabolic fit coefficients table, to be classified as WAVE_COEF_SLIT, for example the one which can be obtained running the recipe with **wcal-calib_indicator** set to TRUE (default).

9.5.2 Output

default recipe file name	PRO.CATG	short description
out_stack_mflat_dist.fits	MFLAT_STACKED_DIST	stacked FLAT,LAMP frames
out_wcal_stack.fits	WAVE_LAMP_STACKED	stacked WAVE,LAMP frames
out_wavemap_ima.fits	WAVE_MAP	wavelength map: wavelength=intensity
outCoeffsSlit.tfits	WAVE_COEF_SLIT	parabolic fit coefficients, for each pixel
out_fit_params.tfits	WAVE_FIT_PARAMS	parameters relative to the fit of lines: n_params: number of rows in the table, line: increasing value of line ID (0-8) fparN and dparN are fit params and their errors for: N meaning 0 Gaussian amplitude, 1 fwhm, 2 center, 3 background
out_slitpos.tfits	SLIT_POS	slitlets positions

9.5.3 Quality control

The recipe monitors the resolving power, the overall wavelength dispersion offset in μm and pixels, the slitlets position in pixels.

Dispersion solution The recipe returns several QC parameters. Among them the average and median coefficients of the wavelength solution (QC.COEFi.AVG, QC.COEFi.MED, i=0,3). The measured wavelength can be compared against the encoder value (INS.GRAT1.WLEN) and the nominal values (e.g. 2.2000 μm for the S3_K grating). During commissioning the grating have been set up in a way that the average and central wavelengths coincides with the nominal values.

The resolving power is monitored as the FWHM of the found arc lines (QC.FWHM.MED, QC.FWHM.AVG) divided by the central wavelength.

Offset DFO monitors the offset between the central wavelength as returned by the recipe and the grating encoder value (measured value minus encoder value). Assuming that the grating inizializes with the same encoder value, this parameter nominally measures the intrinsic errors of the pipeline recipe.

The offset between nominal central wavelength (as given by 1.250 J, 1.650 H, 2.200 K, 1.950 H+K) and the corresponding measured one as given by the wavecal recipe in pixel units (measured minus nominal) are monitored.

The offset between the encoder value and the nominal central wavelength, as given by 1.250 J, 1.650 H, 2.200 K, 1.950 H+K (encoder minus nominal) allows to monitor the position stability/variability of the grating.

Overall wavelength calibration error The value of QC.WAVE.POSERR indicates an estimate of the overall positioning error found during the wavelength calibration.

Maximum flux The maximum flux in an arc lamp frame, given by the value of QC.FRMON.MAXFLUX, gives an indication of the arc lamp aging.

9.5.4 Parameters

parameter	alias	default
sinfoni.stacked.low_rejection	stack-lo_rej	0.1
sinfoni.stacked.high_rejection	stack-hi_rej	0.1
sinfoni.stacked.flat_index	stack-flat_ind	TRUE
sinfoni.stacked.mask_index	stack-mask_ind	1
sinfoni.stacked.ind_index	stack-ind_ind	FALSE
sinfoni.stacked.mask_rad	stack-mask_rad	4
sinfoni.stacked.gauss_index	stack-gauss_ind	FALSE
sinfoni.stacked.kernel_half_width	stack-khw	2
sinfoni.stacked.warpfix_ind	stack-warpfix_ind	TRUE
sinfoni.stacked.warpfix_kernel	stack-warpfix_kernel	tanh
sinfoni.stack.qc_thresh_min	stack-qc_thresh_min	0
sinfoni.stack.qc_thresh_max	stack-qc_thresh_max	64000
sinfoni.wavecal.slitpos_boostrap_switch	wcal-slitpos_bootstrap	FALSE
sinfoni.wavecal.calib_indicator	wcal-calib_indicator	TRUE
sinfoni.wavecal.min_diff	wcal-min_diff	1
sinfoni.wavecal.half_width	wcal-hw	7
sinfoni.wavecal.sigma	wcal-sigma	2.0
sinfoni.wavecal.fwhm	wcal-fwhm	2.83
sinfoni.wavecal.min_amplitude	wcal-min_amplitude	5.0
sinfoni.wavecal.max_residual	wcal-max_residual	0.5
sinfoni.wavecal.n_a_coefficients	wcal-n_a_coeffs	4
sinfoni.wavecal.n_b_coefficients	wcal-n_b_coeffs	2
sinfoni.wavecal.sigma_factor	wcal-sigma_factor	1.5
sinfoni.wavecal.write_coeffs_ind	wcal-wcoeff_ind	TRUE
sinfoni.wavecal.write_par_ind	wcal-par_ind	TRUE
sinfoni.wavecal.pixel_dist	wcal-pixel_dist	12
sinfoni.wavecal.pixel_tol	wcal-pixel_tol	3.0
sinfoni.wavecal.wave_map_ind	wcal-wave_map_ind	FALSE
sinfoni.wavecal.mag_factor	wcal-mag_factor	8
sinfoni.wavecal.slit_pos_indicator	wcal-slit_pos_ind	TRUE
sinfoni.wavecal.fit_boltz_indicator	wcal-fit_boltz_ind	TRUE

sinfoni.wavecal.fit_edge_indicator	wcal-fit_edge_ind	FALSE
sinfoni.wavecal.estimate_indicator	wcal-estimate_ind	FALSE
sinfoni.wavecal.box_length	wcal-box_len	32
sinfoni.wavecal.y_box	wcal-y_box	5.0
sinfoni.wavecal.diff_tol	wcal-diff_toll	2.0
sinfoni.wavecal.qc_thresh_min	wcal-qc_thresh_min	0
sinfoni.wavecal.qc_thresh_max	wcal-qc_thresh_max	64000

9.6 si_rec_psf

The recipe si_rec_psf reduces PSF standard data. It subtracts the sky, corrects for the flat-field and resamples the data by constructing a wavelength calibrated cube in which each plane is a monochromatic image of the FOV. Finally it measures the PSF FWHM (x and y) and computes the instrument Strehl and the encircled energy.

9.6.1 Input

```
/path_file_raw/SINFO.2004-08-13T03:26:12.559.fits PSF_CALIBRATOR
/path_file_raw/SINFO.2004-08-13T03:25:30.323.fits SKY_PSF_CALIBRATOR
/path_file_cdb/MASTER_BP_MAP_K_250.fits MASTER_BP_MAP
/path_file_cdb/MASTER_FLAT_LAMP_K_250.fits MASTER_FLAT_LAMP
/path_file_cdb/WAVE_MAP_K_250.fits WAVE_MAP
/path_file_cdb/SLIT_POS_K_250.tfits SLIT_POS
/path_file_cdb/SLITLETS_DISTANCE_K.tfits SLITLETS_DISTANCE
/path_file_cdb/DISTORTION_K.tfits DISTORTION
/path_file_cdb/FIRST_COLUMN.tfits FIRST_COL
```

9.6.2 Output

default recipe file name	PRO.CATG	short description
sky00.fits	SKY_STACKED_DUMMY	stacked sky frame
out_sky_stack_dist.fits	SKY_STACKED_DIST	stacked sky frame distortion corrected
out_stack_mflat_dist.fits	MFLAT_STACKED_DIST	stacked master flat distortions corrected
out_stack0.fits	PSF_CALIBRATOR_STACKED	stacked PSF frames
out_sky_cube.fits	OBS_SKY	sky cube
out_sky_med.fits	SKY_MED	z-median SKY_CUBE
out_mflat_cube.fits	MFLAT_CUBE	Master flat cube
out_mflat_avg.fits	MFLAT_AVG	z-mean of MFLAT_CUBE
out_mflat_med.fits	MFLAT_MED	z-median of MFLAT_CUBE
cube_obj00.fits	OBS_PSF	PSF STD cube
out_objnod.fits	COADD_PSF	coadded cube
out_med_cube.fits	MED_COADD_PSF	z-median of COADD_PSF
out_bpmap_nodding.fits	MASK_COADD_PSF	COADD_PSF's mask

ao_performance.tfits	AO_PERFORMANCE	strehl information
encircled_energy.tfits	ENC_ENERGY	encircled energy information
out_psf.fits	MASTER_PSF	z-median of COADD_PSF

9.6.3 Quality control

Not yet implemented.

9.6.4 Parameters

parameter	alias	default
sinfoni.stacked.low_rejection	stack-lo_rej	0.1
sinfoni.stacked.high_rejection	stack-hi_rej	0.1
sinfoni.stacked.flat_index	stack-flat_ind	TRUE
sinfoni.stacked.mask_index	stack-mask_ind	1
sinfoni.stacked.ind_index	stack-ind_ind	FALSE
sinfoni.stacked.mask_rad	stack-mask_rad	4
sinfoni.stacked.gauss_index	stack-gauss_ind	FALSE
sinfoni.stacked.kernel_half_width	stack-khw	2
sinfoni.stacked.warpfix_ind	stack-warpfix_ind	TRUE
sinfoni.stacked.warpfix_kernel	stack-warpfix_kernel	tanh
sinfoni.stack.qc_thresh_min	stack-qc_thresh_min	0
sinfoni.stack.qc_thresh_max	stack-qc_thresh_max	64000
sinfoni.objnod.autojitter_method	objnod-aj_method	1
sinfoni.objnod.scales_sky	objnod-scales_sky	TRUE
sinfoni.objnod.ks_clip	objnod-ks_clip	TRUE
sinfoni.objnod.kappa	objnod-kappa	2.0
sinfoni.objnod.size_x	objnod-size_x	0
sinfoni.objnod.size_y	objnod-size_y	0
sinfoni.objnod.n_coeffs	objnod-no_coeffs	3
sinfoni.objnod.nord_south_index	objnod-ns_ind	TRUE
sinfoni.objnod.fine_tuning_method	objnod-fine_tune_mtd	P
sinfoni.objnod.order	objnod-order	2
sinfoni.objnod.lower_rejection	objnod-lo_rej	10.0
sinfoni.objnod.higher_rejection	objnod-hi_rej	10.0
sinfoni.objnod.tolerance	objnod-tol	2
sinfoni.objnod.jitter_index	objnod-jit_ind	TRUE
sinfoni.objnod.vllx	objnod-vllx	0
sinfoni.objnod.vlly	objnod-vlly	0
sinfoni.objnod.vurx	objnod-vurx	0
sinfoni.objnod.vury	objnod-vury	0

9.7 si_rec_stdstar

The recipe si_rec_stdstar reduces telluric standard star data. It subtracts the sky, corrects for the flat-field and resamples the data by constructing a wavelength calibrated cube in which each plane is a monochromatic image of the FOV. Finally it performs a simple extraction (sum of counts in each plane) and computes the instrument efficiency.

9.7.1 Input

```
/path_file_raw/SINFO.2004-08-14T08:13:06.745.fits STD
/path_file_raw/SINFO.2004-08-14T08:14:52.101.fits SKY_STD
/path_file_cdb/MASTER_BP_MAP_H_250.fits MASTER_BP_MAP
/path_file_cdb/MASTER_FLAT_LAMP_H_250.fits MASTER_FLAT_LAMP
/path_file_cdb/WAVE_MAP_H_250.fits WAVE_MAP
/path_file_cdb/SLIT_POS_H_250.tfits SLIT_POS
/path_file_cdb/SLITLETS_DISTANCE_H.tfits SLITLETS_DISTANCE
/path_file_cdb/DISTORTION_H.tfits DISTORTION
/path_file_cdb/FIRST_COLUMN.tfits FIRST_COL
```

9.7.2 Output

default recipe file name	PRO.CATG	short description
sky00.fits	SKY_STACKED_DUMMY	stacked sky frame
out_sky_stack_dist.fits	SKY_STACKED_DIST	stacked sky frame distortion corrected
out_stack_mflat_dist.fits	MFLAT_STACKED_DIST	stacked master flat frame distortion corrected
out_stack0.fits	STD_NODDING_STACKED	stacked PSF frames
out_sky_cube.fits	OBS_SKY	sky cube
out_sky_med.fits	SKY_MED	z-median of SKY_CUBE
out_mflat_cube.fits	MFLAT_CUBE	master flat cube
out_mflat_avg.fits	MFLAT_AVG	z-average of MFLAT_CUBE
out_mflat_med.fits	MFLAT_MED	z-median of MFLAT_CUBE
cube_obj00.fits	OBS_STD	STD STAR cube
out_objnod.fits	COADD_STD	coadded STD STAR cube
out_med_cube.fits	MED_COADD_STD	z-median of COADD_STD
out_bpmap_nodding.fits	MASK_COADD_STD	COADD_STD's mask
out_std_star_spectrum.tfits	STD_STAR_SPECTRA	extracted spectrum and efficiency information
starspectrum.fits	STD_STAR_SPECTRUM	extracted spectrum 1D image

9.7.3 Quality control

Not yet implemented.

9.7.4 Parameters

parameter	alias	default
sinfoni.stacked.low_rejection	stack-lo_rej	0.1
sinfoni.stacked.high_rejection	stack-hi_rej	0.1
sinfoni.stacked.flat_index	stack-flat_ind	TRUE
sinfoni.stacked.mask_index	stack-mask_ind	1
sinfoni.stacked.ind_index	stack-ind_ind	FALSE
sinfoni.stacked.mask_rad	stack-mask_rad	4
sinfoni.stacked.gauss_index	stack-gauss_ind	FALSE
sinfoni.stacked.kernel_half_width	stack-khw	2
sinfoni.stacked.warpfix_ind	stack-warpfix_ind	TRUE
sinfoni.stacked.warpfix_kernel	stack-warpfix_kernel	tanh
sinfoni.stack.qc_thresh_min	stack-qc_thresh_min	0
sinfoni.stack.qc_thresh_max	stack-qc_thresh_max	64000
sinfoni.objnod.autojitter_method	objnod-aj_method	1
sinfoni.objnod.scales_sky	objnod-scales_sky	TRUE
sinfoni.objnod.ks_clip	objnod-ks_clip	TRUE
sinfoni.objnod.kappa	objnod-kappa	2.0
sinfoni.objnod.size_x	objnod-size_x	0
sinfoni.objnod.size_y	objnod-size_y	0
sinfoni.objnod.n_coeffs	objnod-no_coeffs	3
sinfoni.objnod.nord_south_index	objnod-ns_ind	TRUE
sinfoni.objnod.fine_tuning_method	objnod-fine_tune_mtd	P
sinfoni.objnod.order	objnod-order	2
sinfoni.objnod.low_rejection	objnod-lo_rej	10.0
sinfoni.objnod.high_rejection	objnod-hi_rej	10.0
sinfoni.objnod.tolerance	objnod-tol	2
sinfoni.objnod.jitter_index	objnod-jit_ind	TRUE
sinfoni.objnod.kernel_type	objnod-kernel_typ	tanh
sinfoni.objnod.vllx	objnod-vllx	0
sinfoni.objnod.vlly	objnod-vlly	0
sinfoni.objnod.vurx	objnod-vurx	0
sinfoni.objnod.vury	objnod-vury	0
sinfoni.std_star.low_rejection	stack-lo_rej	0.1
sinfoni.std_star.high_rejection	stack-hi_rej	0.1
sinfoni.std_star.fwhm_factor	std_star-fwhm_fct	5.0
sinfoni.std_star.conversion_index	std_star-conv_ind	TRUE

9.8 si_rec_objnod

The recipe si_rec_objnod reduces science data. It subtracts the sky, corrects for the flat-field and resamples the data by constructing a wavelength calibrated cube in which each plane is a monochromatic image of the FOV. Finally it coadds all the observed target cube components into a mosaic.

9.8.1 Input

```
/path_file_raw/SINFO.2004-08-15T14:02:18.468.fits SKY_NODDING
/path_file_raw/SINFO.2004-08-15T14:01:54.844.fits OBJECT_NODDING
/path_file_raw/SINFO.2004-08-15T14:01:32.741.fits OBJECT_NODDING
/path_file_raw/SINFO.2004-08-15T14:01:09.827.fits SKY_NODDING
/path_file_raw/SINFO.2004-08-15T14:00:41.593.fits SKY_NODDING
/path_file_raw/SINFO.2004-08-15T14:00:19.119.fits OBJECT_NODDING
/path_file_cdb/MASTER_BP_MAP_K_250.fits MASTER_BP_MAP
/path_file_cdb/MASTER_FLAT_LAMP_K_250.fits MASTER_FLAT_LAMP
/path_file_cdb/WAVE_MAP_K_250.fits WAVE_MAP
/path_file_cdb/SLIT_POS_K_250.tfits SLIT_POS
/path_file_cdb/SLITLETS_DISTANCE_K.tfits SLITLETS_DISTANCE
/path_file_cdb/DISTORTION_K.tfits DISTORTION
/path_file_cdb/FIRST_COLUMN.tfits FIRST_COL
```

9.8.2 Output

default recipe file name	PRO.CATG	short description
sky00.fits	SKY_STACKED_DUMMY	stacked sky frame
out_sky_stack_dist.fits	SKY_STACKED_DIST	stacked sky frame distortion corrected
out_stack_mflat_dist.fits	MFLAT_STACKED_DIST	stacked master flat distortion corrected
out_stack0.fits	OBJECT_NODDING_STACKED	stacked PSF frames
out_sky_cube.fits	OBS_SKY	sky cube
out_sky_med.fits	SKY_MED	z-median of SKY_CUBE
out_mflat_cube.fits	MFLAT_CUBE	master flat cube
out_mflat_avg.fits	MFLAT_AVG	z-mean of MFLAT_CUBE
out_mflat_med.fits	MFLAT_MED	z-median of MFLAT_CUBE
out_mflat_cube.fits	OBS_OBJ	OBJ cube (intermediate product, there might be more than one of those)
out_objnod.fits	COADD_OBJ	coadded OBJ cube
out_med_cube.fits	MED_COADD_OBJ	z-median of COADD_OBJ
out_bpmap_nodding.fits	MASK_COADD_OBJ	COADD_OBJ's mask

9.8.3 Quality control

Not yet implemented.

9.8.4 Parameters

parameter	alias	default

sinfoni.stacked.low_rejection	stack-lo_rej	0.1
sinfoni.stacked.high_rejection	stack-hi_rej	0.1
sinfoni.stacked.flat_index	stack-flat_ind	TRUE
sinfoni.stacked.mask_index	stack-mask_ind	1
sinfoni.stacked.ind_index	stack-ind_ind	FALSE
sinfoni.stacked.mask_rad	stack-mask_rad	4
sinfoni.stacked.gauss_index	stack-gauss_ind	FALSE
sinfoni.stacked.kernel_half_width	stack-khw	2
sinfoni.stacked.warpfix_ind	stack-warpfix_ind	TRUE
sinfoni.stacked.warpfix_kernel	stack-warpfix_kernel	tanh
sinfoni.stack.qc_thresh_min	stack-qc_thresh_min	0
sinfoni.stack.qc_thresh_max	stack-qc_thresh_max	64000
sinfoni.objnod.autojitter_method	objnod-aj_method	1
sinfoni.objnod.scales_sky	objnod-scales_sky	TRUE
sinfoni.objnod.ks_clip	objnod-ks_clip	TRUE
sinfoni.objnod.kappa	objnod-kappa	2.0
sinfoni.objnod.n_coeffs	objnod-no_coeffs	3
sinfoni.objnod.nord_south_index	objnod-ns_ind	TRUE
sinfoni.objnod.fine_tuning_method	objnod-fine_tune_mtd	P
sinfoni.objnod.order	objnod-order	2
sinfoni.objnod.low_rejection	objnod-lo_rej	10.0
sinfoni.objnod.high_rejection	objnod-hi_rej	10.0
sinfoni.objnod.tolerance	objnod-tol	2
sinfoni.objnod.jitter_index	objnod-jit_ind	TRUE
sinfoni.objnod.kernel_type	objnod-kernel_typ	tanh
sinfoni.objnod.vllx	objnod-vllx	0
sinfoni.objnod.vlly	objnod-vlly	0
sinfoni.objnod.vurx	objnod-vurx	0
sinfoni.objnod.vury	objnod-vury	0

9.9 si_utl_skymap

This recipe, used to generate a special bad pixel map input of the SINFONI RTD is used to support Paranal operations. It flags sky lines as bad pixels.

9.9.1 Input

Input are sky frames with tag SKY

```
/path_file_raw/SINFO.2004-08-14T08:14:52.101.fits SKY
```

9.9.2 Output

The output image, a map of the sky lines, is called out_skymap.fits

9.9.3 Parameters

parameter	alias	default
si_rec_skymap.ysize	ysize	30
si_rec_skymap.xsize	xsize	1
si_rec_skymap.thresh	thresh	30.0

We list here for different pre-optics and bands the suggested parameter values:

grating	scale_out	DIT	scale_in	threshold	window
J	0.025	300	0.1	30.0	30
J	0.1	300	0.1	30.0	30
J	0.25	300	0.25	50.0	30
H	0.025	300	0.1	50.0	30
H	0.1	300	0.1	50.0	30
H	0.25	300	0.25	100.0	30
K	0.025	300	0.1	70.0	30
K	0.1	300	0.1	70.0	30
K	0.25	300	0.25	200.0	30
H+K	0.025	300	0.1	100.0	30
H+K	0.1	300	0.1	100.0	30
H+K	0.25	300	0.25	300.0	30

9.10 si_utl_bp_mask_add

This recipe performs bad pixel map coaddition.

9.10.1 Input

The input files are several (at least 2) bad pixel masks. Their tag should contain the string BP_MAP.

```
/path_file_cdb/BP_MAP_NL_K_100.fits BP_MAP_NL
/path_file_cdb/BP_MAP_NO_K_100.fits BP_MAP_NO
/path_file_cdb/BP_MAP_HP_K_100.fits BP_MAP_HP
```

9.10.2 Output

The output is an image resulting from the logical operator OR applied to all the masks.

9.10.3 Parameters

The recipe output filename for the product is bp_map_sum.fits

9.11 si_utl_ima_arith

This recipe performs image computation.

9.11.1 Input

The input files are 2 images and their associated tags should be IMA.

```
/path_file/ima1.fits IMA
/path_file/ima2.fits IMA
```

9.11.2 Output

The output is an image resulting from IMA op IMA where op indicates the operation to be performed. If a numerical value is specified at command line this is subtracted to the first input image.

9.11.3 Parameters

parameter	alias	default
si_utl_ima_arith.op	op	+
si_utl_ima_arith.value	value	9999.0

9.12 si_utl_cube2ima

This recipe performs cube to image conversion.

9.12.1 Input

The input file is a cube. Its tag should be CUBE.

```
/path_file/cube.fits CUBE
```

9.12.2 Output

The output is an image resulting from the average of the cube over a wavelength range which can be set by parameters sinfoni.si_utl_cube2ima.ws, sinfoni.si_utl_cube2ima.we having aliases ’ws’ ’we’. The recipe output

filename for the product is out_ima.fits

9.12.3 Parameters

parameter	alias	default
si_utl_cube2ima.ws	ws	0.9999
si_utl_cube2ima.we	we	2.999

9.13 si_utl_cube2spectrum

This recipe performs cube to 1D spectrum image conversion.

9.13.1 Input

The input files is a cube Its associated tag should be CUBE.

```
/path_file/cube.fits CUBE
```

9.13.2 Output

The output is an image resulting from the cube manipulated according to the value of the parameter op. over an aperture as specified by the parameter sinfoni.si_utl_cube2spectrum.aperture having alias 'op', 'ap'.

The recipe output filename for the product is out_spec.fits

9.13.3 Parameters

Possible operations are: average, clean_mean, median, sum. If the chosen operation is a clean mean one may define its lower and upper threshold cuts by setting parameters lo_rej and hi_rej. Possible apertures are: rectangle, circle. If the chosen aperture is a rectangle, its corner coordinates can be set with the parameters llx, lly, urx, ury (lower left x,y and upper right x,y). Those parameters follow the C-style convention: arrays start at 0 and end at size-1. If the chosen aperture is a circle, its position and size can be set by the parameters centerx, centery, radius.

parameter	alias	default
si_utl_cube2spectrum.op	op	average
si_utl_cube2spectrum.ap	ap	rectangle
si_utl_cube2spectrum.llx	llx	2
si_utl_cube2spectrum.lly	lly	2
si_utl_cube2spectrum.urx	urx	28
si_utl_cube2spectrum.ury	ury	28
si_utl_cube2spectrum.lo_rej	lo_rej	10

si_utl_cube2spectrum.hi_rej	hi_rej	10
si_utl_cube2spectrum.centerx	centerx	16
si_utl_cube2spectrum.centery	centery	16
si_utl_cube2spectrum.radius	radius	5

9.14 si_utl_cube_arith

This recipe performs cube arithmetics. If a parameter value is specified it is assumed that the input frame is a cube with tag CUBE. Else the input files are a cube and an image or a spectrum their associated tags should be respectively CUBE, IMA or SPECTRUM. The output is a cube (PRO.CATG=CUBE) resulting from the operation CUBE op IMA or CUBE op SPECTRUM or CUBE op value where op indicates the operation to be performed.

9.14.1 Input

```
/path_file/cube.fits      CUBE
/path_file/spectrum.fits  SPECTRUM
```

or

```
/path_file/cube.fits   CUBE
/path_file/image.fits  IMA
```

9.14.2 Output

The recipe output filename for the product is out_cube.fits

9.14.3 Parameters

parameter	alias	default
si_utl_cube_arith.op	op	/
si_utl_cube_arith.value	value	99999.0

9.15 si_utl_spectrum_divide_by_blackbody

This recipe divides a spectrum by a Black Body spectrum of given temperature.

9.15.1 Input

The input file is a spectrum. Its associated tag should be SPECTRUM.

`STD_STAR_SPECTRUM.fits SPECTRUM`

9.15.2 Output

The output is a spectrum resulting in the division of the input spectrum by a Black Body spectrum of given temperature (parameter 'temp').

The recipe output filename for the cube product is out_ima.fits

9.15.3 Parameters

parameter	alias	default
si_utl_spectrum_arith.temp	temp	1e+05

9.16 si_utl_spectrum_wavelength_shift

This recipe shift a spectrum in wavelength according to value of the input parameter 'method'.

9.16.1 Input

The input file is a spectrum. Its associated tag should be SPECTRUM.

`STD_STAR_SPECTRUM.fits SPECTRUM`

9.16.2 Output

The output is a spectrum resulting by a simple wavelength shift (parameter 'shift').

The recipe output filename for the cube product is out_ima.fits

9.16.3 Parameters

parameter	alias	default
si_utl_spectrum_arith.method	method	S
si_utl_spectrum_arith.shift	shift	0.1

The spectrum is shifted by shift to sub pixel accuracy resampling the intensity of the pixels using either a spline approximation (method=S) or a polynomial approximation (method=P).

10 Algorithms and recipe details

In this section we describe the main algorithms implemented in the SINFONI pipeline recipes. Relevant data reduction parameters are typed in **bold** face. For convenience we omit the common prefix **sinfoni.** for the recipe description as well as the step prefix name for the algorithm description.

10.1 Algorithms

10.1.1 Frame stacking

The input image frames are stacked to build a cube. Each plane of the cube corresponds to an input image.

10.1.2 Average with rejection

In this document we use often the terminology *clean mean*, *clean average*, or *average with rejection*. With those terms we mean that it is computed a mean of a list of values by avoiding outlier values like for example bad pixels, and therefore, this operation yelds a better SNR than a simple mean. In case the values to be averaged are pixels intensities of several images stacked in a cube, the mean along the z axis of the cube is computed at each x,y pixel after having rejected the intensity values which lie outside an interval [**low_rejection,high_rejection**].

10.1.3 Detector non linearity computation

- Method 1

  From the input set of raw flat frames, “off” and “on” frames are extracted and put in two sets. A pair of “on” and “off” frames with same DIT is selected from each set. Then, for each frame pair i, the ratio med_dit_i is computed as:

  $$med\_dit_i = \frac{median(frm_{on}(i)) - median(frm_{off}(i))}{DIT(i)}$$

  And the mean of all N med_dit_i values is computed as:

  $$mean = \frac{\sum_i^N med\_dit_i}{N}$$

  A parabolic fit of the product of $DIT(i) \times mean$, as a function of $med\_dit(i)$, is performed.

  The non linear coefficient of the fit is the detector non linearity parameter.

- Method 2

  The input flat field frames with in/decreasing intensity are stacked in a cube and for each pixel position the dependence of each plane pixel intensity with respect to the whole plane clean average intensity is determined. This curve is fit with a polynomial of order **order**. In the stacking process low and high intensity pixels may be rejected by properly setting the parameters **low_rejection** and **high_rejection**. A

clean mean of all polynomials is computed. The pixels whose intensity constant ($coef_0$) or slope ($coef_1$) coefficients exceed the mean by **thresh_sigma_factor** times the standard deviation of the pixel intensity are rejected. Pixels whose non quadratic coefficient exceeds **non_lin_threshold** are rejected.

Thus a map of the pixels which do not have a linear sensitivity is generated, together with a table containing the polynomial coefficients: the non-linear terms give an estimate of the detector non-linearity.

10.1.4 Nearest neighbours bad pixel cleaning

This method is applied in the recipes si_rec_mflat and si_rec_distortion to clean the bad pixels of a flat frame. The algorithm is controlled by the parameters **method_index** and **factor**, and, if **method_index** is 4, the bad pixel cleaning is repeated **iterations** times.

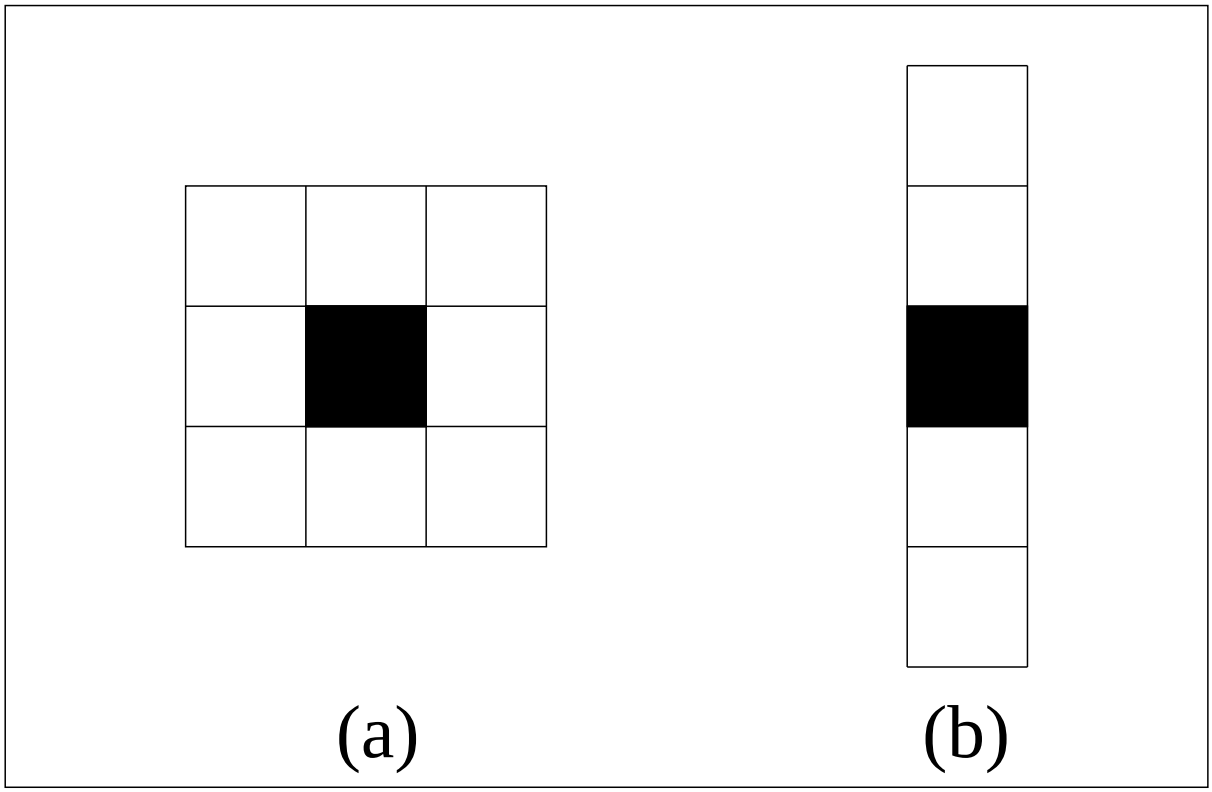


Figure 10.1.0: A bad pixel is indicated in black. If **method_index** is 1,2 or 4 this is corrected evaluating the eight nearest neighbours pixel intensities (a). If **method_index** is 3, the four closest pixels along the dispersion direction are considered (b).

- If **method_index** is 1, for each image pixel intensity, I_i, the eight nearest neighbours pixels intensities, I_j, are considered and the corresponding median, $median_i$, is computed excluding from this set those pixels whose intensity is NAN.

$$median_i = median(I_j)...I_j \neq NAN$$

– If the parameter **factor** is set to zero:

$$I_i = median_i$$

– If the parameter **factor** is negative:

$$I_i = \begin{cases} median_i & \text{if } \|I_i - median_i\| > -factor \\ I_i & \text{else} \end{cases}$$

– If the parameter **factor** is positive:

$$I_i = \begin{cases} median_i & \text{if } \|I_i - median_i\| > factor \times \sqrt{median_i} \\ I_i & \text{else} \end{cases}$$

- If **method_index** is 2, the absolute distances $dist_i$ of the eight nearest neighbours pixels j are computed for each bad pixel i as:

$$dist_i = \frac{\sqrt{\sum_{j=1}^{8}(I_i - I_j)^2 * w_j}}{\sum_{j=1}^{8} w_j}$$

where

$$w_j = \begin{cases} 0 & \text{if } I_j = NAN \\ 1 & \text{else} \end{cases}$$

The median distance and its standard deviation are then computed as:

$$median\_distance = median(dist_i)$$

$$stdev = \sqrt{(\sum_i dist_i^2) - (\sum_i dist_i)^2}$$

– If the parameter **factor** is zero:

$$I_i = dist_i$$

– If the parameter **factor** is negative:

$$I_i = \begin{cases} dist_i & \text{if } \|median\_distance - dist_i\| > -factor \times stdev \\ I_i & \text{else} \end{cases}$$

– if the parameter **factor** is positive:

$$I_i = \begin{cases} dist_i & \text{if } \|median\_distance - dist_i\| > factor \times stdev\sqrt{dist_i} \\ I_i & \text{else} \end{cases}$$

- If **method_index** is 3, the intensities of the four closest pixels in spectral direction are considered and their mean intensity is computed for each bad pixel i as:

$$mean_i = \frac{\sqrt{\sum_{j=1}^{4}(I_i - I_j) * w_j}}{\sum_{j=1}^{4} w_j}$$

where

$$w_j = \begin{cases} 0 & \text{if } I_j = NAN \\ 1 & \text{else} \end{cases}$$

  - If the parameter **factor** is zero:

$$I_i = mean_i$$

  - If the parameter **factor** is negative:

$$I_i = \begin{cases} mean_i & \text{if } \|I_i - mean_i\| > -factor \\ I_i & \text{else} \end{cases}$$

  - If the parameter **factor** is positive:

$$I_i = \begin{cases} mean_i & \text{if } \|I_i - mean_i\| > factor \times \sqrt{mean_i} \\ I_i & \text{else} \end{cases}$$

- If **method_index** is 4, the local clean (**low_rejection, high_rejection**) standard deviation $clean\_stdev$ in a box (**llx, lly, urx, ury**) is computed. Then the difference of the pixel and the median of the nearest neighbours is computed by using the eight closest pixels of every pixel.

$$median_i = median(I_j) ... I_j \neq NAN, j = 1...8$$

  - if the parameter **factor** is zero:

$$I_i = median_i$$

  - if the parameter **factor** is negative:

$$I_i = \begin{cases} median_i & \text{if } \|I_i - median_i\| > -factor \times clean\_stdev \\ I_i & \text{else} \end{cases}$$

  - if the parameter **factor** is positive:

$$I_i = \begin{cases} median_i & \text{if } \|I_i - median_i\| > factor \times \sqrt{\|median_i\|} \\ I_i & \text{else} \end{cases}$$

The previous operations are repeated for (**iterations**) times to be able to consider small clusters of bad pixels.

10.1.5 Detector gain computation

The gain is computed as part of the recipe si_rec_detlin.

Pairs of consecutive off-flats (be those frm_off1 and frm_off2) and pairs of consecutive on-flats (be those frm_{on1} and frm_{on2}) are selected from the input data frames and their difference is computed ($dif\_frm_{on} = frm_{on2} - frm_{on1}$ and $dif\_frm_{off} = frm_{off2} - frm_{off1}$). Then the mean of each frame ($\overline{frm_{on1}}$, $\overline{frm_{on2}}$, $\overline{frm_{off1}}$, $\overline{frm_{off2}}$) and the standard deviation of the difference $stdev(dif\_frm_{on})$ and $stdev(dif\_frm_{off})$ are computed.

Finally the gain is given by:

$$gain = \frac{(\overline{frm_{on1}} + \overline{frm_{on2}}) - (\overline{frm_{off1}} + \overline{frm_{off2}})}{(stdev(dif\_frm_{on})^2 - stdev(dif\_frm_{off})^2)}$$

10.1.6 Read Out Noise computation

All possible consecutive pairs having the same DIT are extracted from the input set of frames. For each pair the second frame is subtracted from the first one. Then the noise is computed in a region defined by parameters **xmin, xmax, ymin, ymax**, over **nsamp** samples each of size **hsize** taken randomly in the given region. The noise multiplied by $\sqrt{NDIT}$, where NDIT the number of DIT repetitions, gives the read out noise.

10.1.7 Fixed Pattern Noise computation

The FPN is computed in the same way as the RON but it is applied to a master dark/flat. The factor $\sqrt{NDIT}$ does not apply.

10.1.8 Line position determination

The locations of the arc lamp lines are determined on an arc lamp frame, input of the recipe si_rec_wavecal or si_rec_distortion.

- For each detector column, an initial guess wavelength is assigned to each detector row pixel by using the pixel row value and the input initial guess value for the starting wavelength, (**begin_wave**), the linear and quadratic terms of the dispersion (**guess_disp1** and **guess_disp2**) at the given instrument setting. Then the detector rows corresponding to each entry of the reference line list are identified. Line positions and intensities given by the reference line table are assigned to those columns. A delta-shaped spectrum is obtained in this way. This spectrum is convolved with a Gaussian profile of given **sigma**. A pixel shift is obtained by correlating this artificial line spectrum with the arc lamp frame spectrum. The correlation takes into account only those lines whose intensity is greater than a threshold set by parameter **min_diff**. Thus one obtains accurate values for the position and the wavelength assigned to each image position.
- To determine the dispersion relation (see 10.1.9) more accurately the recipe selects and uses only line identifications whose distance from the nearest neighbour is greater than **half_width** pixels.

- The recipe finally determines several lists of line positions:
  - a line list of row indeces for the line positions;
  - a clean list of well separated line positions;
  - a list of wavelengths corresponding to the raw positions;
  - a list of wavelengths corresponding to the clean raw positions.

  Those lists allow to associate each emission line in each spectrum with an approximate pixel position and the exact wavelength.

10.1.9 Dispersion relation and wavelength map determination

- A Levenberg-Marquardt fit of a Gaussian profile of given **fwhm** is performed considering the raw positions which lie in regions defined by boxes of half-size **half_width**, each centered at the line positions determined by the algorithm described in 10.1.8. The fit is restricted to the lines whose amplitude is greater than **min_amplitude** with respect to the background. This fit allows to determine each line position with sub pixel accuracy.
- Detector defects may lead to erroneous line detections. To prevent these misidentifications the fit is limited to lines which lie no more than **pixel_tolerance** pixels from the corresponding position, obtained assuming a linear dispersion **guess_disp1** model. Possible outliers are flagged by setting the line-position parameter of the fit to zero.
- For each image column, a polynomial fit of degree **n_a_coefficients** and coefficients $acoef_k$ is performed, so as to determine the dispersion relation between the listed wavelength values $\lambda(j)$ and the Gauss-fitted positions $y\_pos(j)$ for each image column using the singular value decomposition method.

$$\lambda(j) = \sum_{k=0}^{\mathbf{n\_a\_coefficients}} acoef_k * y\_pos(j)^k$$

  Data points which lie beyond **max_residuals** pixels from the corresponding fit value are rejected. In the fit each data point is weighted by a factor proportional to the product of the guess dispersion value **guess_disp1** and the error associated with each position point from the previous Gaussian fit.
- Then the positions of the slitlet edges are determined. Offset coefficients $acoef_o$ resulting from the polynomial fit for adjacent columns are compared. When they differ by more than **pixel_dist*guess_disp1**, it is assumed that this is due to the crossing of a slitlet's edge. Here **pixel_dist** indicates the minimal distance in pixels between adjacent slitlets measured along the wavelength direction. This provides a first approximation of the slitlets' edges.
- A clean average of the coefficients is determined by rejecting the extreme 10% low and high values and performing a kappa-sigma clipping, where the constant kappa is set by **sigma_factor**.
- Then a singular value decomposition fit of the polynomial coefficients is performed across the spatial extent of each slitlet using a polynomial of low degree **n_b_coefficients** and the smoothed polynomial coefficients $\overline{acoef_{i,x}}$ are computed anew.

- Finally a wavelength map is built by associating with each image pixel a wavelength corresponding to the value resulting from the polynomial fit of degree **n_a_coefficients**. The intensity $I_{map}(x, y)$ at each image point x, y is given by

$$I_{map}(x, y) = \lambda(x, y)$$

where:

$$\lambda(x, y) = \sum_{i=0}^{n\_a\_coefficients} \overline{acoef_{i,x}} \times (y - off)^i$$

and

$$off = \frac{(ny - 1)}{2}$$

where ny is the number of raws of the image.

10.1.10 Line shift computation

The recipe searches for the five brightest lines in each column of the input arc lamp. The shift between each line position on the frame and the one resulting from the polynomial dispersion relation of degree **n_a_coefficients** obtained in 10.1.9 is computed. Finally the overall wavelength calibration error is computed as the clean average of the shifts obtained after removing 10% of the outliers. Then the wavelength positioning error at different wavelengths is computed in a similar way. This value is monitored for quality control.

10.1.11 Dispersion relation adjustment

If a dispersion relation and a wavelength map are already available one may still want to adjust them to properly match the given arc lamp frame.

The positions of the slitlets' edges are initially determined by checking that the values of the zero order coefficient $acoef_0$ for adjacent image columns differ by more than **guess_disp1** $\times$ **pixel_dist**.

An artificial spectrum is generated by convolving each entry of the reference line list with a Gaussian of position and intensity as given by the reference line table and a sigma equal to $(\mathbf{mag\_factor} - 1)^2$.

Bad and negative pixels are filtered out from the input image and a low-pass filter of half width **mag_factor** is applied. The resulting cleaned and smoothed image is convolved with the artificial spectrum to compute their shift, the maximum of the correlation value and the position at which this maximum is reached.

A new value of the zero order dispersion coefficients $acoef_0$ is determined. A clean mean is performed on the dispersion coefficients $acoef_k$ and new values for the clean dispersion coefficients $\overline{acoef_k}$ are determined with a single value decomposition fit over each slitlet spatial domain.

Finally a new wavelength map is generated using the clean dispersion coefficients $\overline{acoef_k}$ as described at the end of section 10.1.9.

10.1.12 Slitlet position computation

In order to be able to reconstruct the original observed image out of the raw image the positions of the edges of each slitlet must be determined accurately. The 32 slitlets must be stacked one on top of the other by using accurate reference positions, otherwise the resulting image rows would be shifted with respect to the others yelding weird images.

To determine the slitlets positions an exposure of an arc lamp is used. The slitlets edges are clearly visible if looking at a bright, stand-alone emission line (for example see Figure 6.2.0 (d)). Such an emission line can be used to determine the absolute positions of the left and the right edge of each slitlet using a non-linear least square fit of an appropriate fitting function.

Two methods are possible to determine the slitlet positions. They differs only for the choice of the fitting function.

Initially the brightest lines are searched in the image column domain corresponding to the first slitlet. From this list the lines which have a bright neighbour within **y_box** pixels are filtered out. Then a first estimate of the slitlet edges is performed: a slitlet edge is reached when the first (offset) dispersion coefficient $coef_0$ has a variation greater than **y_box**. Those values are checked to remove misidentifications possibly due to bad pixels.

Then the image intensity values of the maxima reached in each slitlet are determined by searching in rectangular boxes centered at each slitlet positions *edge(j)* each of size **y_box** along the spectral direction, and size *edge(j)-edge(j-1)* + 2 $\times$ **box_length** along the spatial direction.

In case the user sets the parameter **wcal-estimate_ind** value to TRUE the search of the image intensity maxima is limited to the interval [**lo_pos**,**hi_pos**] along the spectral direction.

The minimum value of the maxima is determined to have an initial value of the backround intensity at which a slitlet edge detection should be triggered. The list of maxima is divided in two sets and on each data set a least square fit is performed with an appropriate fitting function.

The Edge method uses in the non-linear fit a linear step function that means a linear function between two positions with two constant backgrounds to the left and to the right of the slitlet positions. Free parameters of the fit are the two positions of the left and right background values. The mean of both fitted positions is used as resulting left or right slitlet edge position.

The Boltzmann method uses as fitting function a sigmoidal Boltzmann function, which describes the transition between two values:

$$y = \frac{A_1 - A_2}{1 + \exp\frac{x - x_0}{dx}} + A_2$$

where A_1 is the left background value, A_2 is the right background value, dx is the width of the transition region and x_0 is the center of the transition region (turning point of the function). Free parameters of the fit are both the background values, the width dx and the center x_0, which is then used as left or right slitlet edge position.

10.1.13 Slitlet distances computation

The slitlet distances are computed on a frame which is the result of an average with rejection performed on several (usually 75) fibre frames. Each frame has the first column of a few slitlets illuminated by the light coming from a fiber.

For each raw of the stacked frame the following operations are performed:

- The pixel positions whose intensity is greater than three times the mean of the image are determined on the stacked, distortion corrected, fibre frame. This generates for each fibre of the input frame a set of columns where the intensity is greater than the mean of the frame.
- The intensity maximum of each column is accurately determined by comparing image pixel intensities in a spatial range of size **half_width** centered at each column position of the found set. Possible wrong column detections, due to bad pixels, can be flagged by verifying that the found position has a value which differs from the one of the previous slitlet by **estimated_dist** with a tolerance **dev_tol**.
- A least square fit with a Gaussian function of FWHM **fwhm** is performed to precisely locate the fibres and thus determine the slitlet positions.

In this way one determines for each image raw the slitlets' positions and can build their relative difference. Making a mean of those relative distances along the spectral direction, excluding istances for which the relative distance differs from the **estimated_distance** by more than **dev_tol**, one finally finds 31 values of the relative slitlet distances.

10.1.14 Cube construction: resampling

The algorithm described below is executed by recipes si_rec_psf, si_rec_stdstar, si_rec_objnod. Given a source image and a corresponding wavelength calibration file, an image is produced in which elements in a given row are associated with a single wavelength. In this way the wavelength shifts between adjacent elements in the rows of the input image are corrected. The output image in the wavelength domain is larger than the input image. Due to the brick wall pattern of the raw frames, some pixels in the first and last few rows have undefined values that are flagged by setting them to NAN. The distribution of these undefined values varies from column to column. The input image is resampled at discrete wavelength intervals using a polynomial interpolation of degree **n_coeffs**. Different values of the wavelength sampling size and the central wavelength are used for each observed band. Thus, each row has a defined wavelength for each observed band.

Since each frame row is now associated with a defined wavelength, each row is used to construct an image which has a defined wavelength. This is done by stacking 32 slitlets on each other, of which each consists of 64 spatial pixels (called also spaxels). Due to the fact that the slitlets length is not exactly 64 pixels and the distance between slitlets is not exactly 64 pixels the edge positions of the slitlets must be known to sub-pixel accuracy (see 10.1.12). Furthermore, the slitlets must be sorted in the correct sequence (see Figure 3.1.0 (bottom) or Figure 6.1.0 (b)) to get the correct sequence of the rows in the final images.

The centers of each slitlet on the resampled image are determined by averaging the edge positions of the slitlets computed as described in 10.1.12 to get the center positions. Then the centers of the slitlets on the raw image are adjusted on the centers of the corresponding raws of the stacked data cube images. Since only integer pixels

can be used a sub-pixel error is made at centering, which is stored and used to shift the rows in the reconstructed images of the data cube to the correct sub-pixel position using a user definable method (**objnod-fine_tune_mtd**). As the edges of the left-most or right-most slitlets may be too near to or fall outside the image margins, then the fitting of the edges may fail. Then the slitlets distances determined as described in 10.1.13 must be used to accurately align each slitlet in the final reconstructed image plane.

10.1.15 Cube coaddition

Each cube component is vignetted chopping out from each cube plane left as many pixels as specified by parameters **vllx, vlly, vurx, vury** (defaulted to 0, 0, 0, 0). As consequence of this the actual corner coordinates of each cube plane being coadded are:

$$llx = 1 + vllx \qquad lly = 1 + vlly \qquad urx = 64 - vurx \qquad ury = 64 - vury$$

If **sky_scales** == TRUE, before cube coaddition, the spatial median of each cube plane is subtracted from each contributing cube plane to remove sky background residuals possibly not corrected in the previous data reduction steps (e.g. during the frame stacking), for example due to sky variations with time.

Each target object offset is determined by reading the HIERARCH ESO CUMOFFSETX/Y FITS keywords. The exposure time corresponding to each target acquisition is read from the FITS header. Then the recipe computes the minimum size of the cross section along the z-axis of a parallelepipedus which cover the full observed field of view and determines the offsets to apply to each cube component to properly merge them in the coadded cube.

In this manual the 3D frame obtained after cube coaddition is often called cube for simplicity.

Then each contributing cube plane is shifted to locate it in its proper position in the coadded cube. Each pixel intensity of the coadded cube has an intensity given by the weighed mean of the intensities of each corresponding overlapping pixel from the contributing cubes, the weight being given by the exposure time of each target frame.

If **ks_clip** == TRUE, in the cube coaddition step a kappa-sigma clipping of the contributing overlapping pixels intensities is performed using a user defined value of **kappa**.

10.1.16 Estimation of the sky from object frames in case the input set is missing sky frames

Since it is possible that there is no time for the acquisition of sky frames when executing a science OB, it is necessary to provide a proper algorithm capable of providing a good estimate of the sky.

The SINFONI pipeline supports three methods, each corresponding to different values of the parameter **aj_method**. If **aj_method** is 0 no sky is estimated and thus no sky is subtracted. If **aj_method** is 1, then for each object observation the algorithm uses the object exposure with the closest MJD-OBS as an approximation of a sky exposure. If **aj_method** is 2 the sky is given by a median of all the contributing objects.

Provided that the target object positions in the acquired frames are separated by at least three times the FWHM of the object, the data reduction scheme implemented for the case **aj_method**=1 should provide an accurate sky estimation. If this is not the case the user may set **aj_method** to 2 or try to subtract the sky before data reduction and then reduce the sky-subtracted data as science object frames choosing aj_method=0.

If **aj_method** is 1 each contributing cube (and the coadded cube) will show regions with negative intensity.

If **aj_method** is 1, **ks_clip** == TRUE, and each contributing frame fills most of the FOV (for example because the chosen camera scale is 25 mas) the user may have to use a value of **kappa** greater than the default, which is set to 2, to prevent that some object point is clipped, which could result possibly in local spikes in the object spectrum.

10.1.17 Efficiency computation

The estimation of the efficiency, i.e. the throughput of the atmosphere+telescope+instrument, is described in this section.

1. The flux on the detector is converted to units of ergs/s/cm^2/Å for a star with 0^{th} magnitude. The quanta are photons.

   - The following operations are applied to the wavelength-calibrated, extracted spectra. The signal is integrated over an area of five FWHM of the input source PSF to ensure that all the light is included.
   - This spectrum is rescaled by
     (a) dividing by the DIT,
     (b) dividing by the surface area, A, of M1 in cm^2,
     (c) multiplying by the Gain (2.42),
     (d) multiplying this by the flux ratio of the observed star and a 0th magnitude star, i.e. $10^{\frac{Mag}{2.5}}$,
     (e) dividing by the dispersion (Å per pixel),
     (f) multiplying by the energy of the photon ($\frac{hc}{\lambda}$).

     This corresponds to multiplying the spectrum by the following factor:

$$\frac{gain \times 10^{\frac{Mag}{2.5}}}{DIT \times A \times Dispersion} \times \frac{hc}{\lambda}$$

     where the following numerical values for the physical constants are used:
     h = 6.62618e-34 Js (Planck constant)
     c = 2.998e8 m/s (Speed of Light)
     k = 1.3807e-23 J/K (Boltzmann constant)

2. The recipe computes the theoretical spectrum of a 0^{th} magnitude star in units of ergs/s/cm^2/Å

   The following method is inaccurate for spectral types that are not A0V.

   - The ratio between the flux from a blackbody at wavelength λ and the flux from the blackbody with T=10,000 K at wavelength λ_c is computed.
   - This curve is scaled so that the value at the central flux is f_0, were f_0 is for a 0th magnitude star in J, H, K and "HK":

band	λ_c (μm)	f_0 (ergs/s/cm^2/Å)
J	1.225	3.11e-10
H	1.675	1.15e-10
K	2.175	4.10e-11
HK	1.950	8e-11

A more accurate method is described in the following and needs to be implemented.

- The spectral type of the star is determined looking into a catalogue.
- From this information is obtained the star energy distribution.
- This curve is scaled so that the value at the central flux is f_0.

3. The result of step 1 is divided by the one from step 2. This is the efficiency.

10.1.18 Strehl computation

The central maximum in units of the total flux of a PSF is a measure of the image quality and AO performance. The Strehl ratio is the observed ratio in units of theoretically possible (diffraction limited) ratio. It is a number usually between $\simeq 0.01$ (seeing limited) and 0.6-0.8 depending on the performance and the ambient conditions.

10.1.19 Encircled energy computation

The encircled energy is the signal integrated in a given area centered on the position where the object reaches its maximum. For this computation it is critical to estimate the flux of the background.

10.1.20 Spectrum extraction

The standard star spectrum is extracted by summing the object signal in each plane of the cube. The residual sky background contribution is given by the constant term of a 2D Gaussian fit of the image obtained by averaging the cube along z, in a range of **factor** times the mean FWHM (FWHM=0.5*(FWHM_X+FWHM_Y)) of the observed standard star.

10.1.21 Standard star position detection

The recipe determines the position of the maximum in the image obtained by collapsing the cube along the z-axis. Then the STD star object (which is assumed to be the only object in the FOV) is more accurately located using a centroid algorithm assuming a 2D Gaussian shape approximation for the object PSF, which also allows to estimate the STD star FWHM along the X and Y directions.

10.2 Recipes

10.2.1 Detector linearity and non-linear bad pixel map determination: si_rec_detlin

The si_rec_distortion recipe computes the detector non-linearity coefficient as described in 10.1.3 (using methods 1 and 2). The detector gain is also computed as described in 10.1.5. In this way several pieces of information are generated: a table with information on detector non-linearity (obtained with method 1), a table with information on the gain, a cube with the coefficients of a polynomial fit to each pixel intensity which provides pixel-by-pixel information on the detector non-linearity, a bad pixel map with non-linear pixels (the last two products are obtained as described using method 2 as described in 10.1.3).

10.2.2 Master dark and bad pixel map determination: si_rec_mdark

A set of input raw dark frames is stacked in a cube. An average with rejection (parameters **bp_noise.low_rejection** and **bp_noise.high_rejection**), yelds a *mean* and a standard deviation, *stdev*. Pixels which deviate from the *mean* more than a user defined factor (**dark.threshold_sigma_factor**) times the *stdev* are flagged as bad pixels. This results in a bad (hot) pixel map which flags pixels with a high dark current.

A set of input raw dark frames is sorted according to DIT thus generating corresponding groups. Then an average with rejection (controlled by parameters **dark.low_rejection** and **dark.high_rejection**) is computed within each group of frames. This results in a master dark frame for each DIT.

On each possible pair of consecutive raw frames the read-out noise is determined in a region defined by the parameters **dark.qc_ron_xmin, dark.qc_ron_xmax, dark.qc_ron_ymin, dark.qc_ron_ymax**, and using **dark.qc_ron_nsamp** random samples each of size **dark.qc_ron_hsize** as described in 10.1.6. On the master dark the fixed pattern noise is determined in two regions defined by the parameters **dark.qc_fpn_xmin, dark.qc_fpn_xmax, dark.qc_fpn_ymin, dark.qc_fpn_ymax**, using **dark.fpn_ron_nsamp** random samples each of size **dark.fpn_ron_hsize** as described in 10.1.7.

10.2.3 Master flat and threshold pixels (bad pixel map) determination: si_rec_mflat

- The input flat field frames are stacked. An average with rejection (parameters **lamp_flats.low_rejection** and **lamp_flats.high_rejection**) is computed to remove dynamic bad pixels (either cosmic rays or transient bad pixels). The mean lamp-off frame is subtracted from the mean lamp-on frame.

- If **lamp_flats.bad_ind**==TRUE the intensity tilt of each column is removed (the fit of the pixel intensity is subtracted from the pixel intensity) considering in this operation only pixels whose intensity differs from the linear fit value by no more than **lamp_flats.sigma_factor** times the sigma of the pixel intensity.

- To find the strong intensity deviations of bad pixels a threshold value must be found. For this reason, on a rectangular region defined by parameters **lamp_flats.llx**, **lamp_flats.lly**, **lamp_flats.urx**, **lamp_flats.ury**, the recipe computes a *clean_mean* of the intensity (**lamp_flats.bad_low_rejection** and **lamp_flats.bad_high_rejection**) and its clean standard deviation *clean_stdev* (to have an estimate of the noise variations in the flat field). The threshold value is given by the product *clean_stdev****lamp_flats.factor**.

  If **lamp_flats.thresh_index**==TRUE the image corrected for the intensity tilt is further filtered, indicating as bad pixels the ones which lie outside the intensity range [*clean_mean*-**lamp_flats.mean_factor****clean_stdev*]. Else no filter is applied. This results in a reference image. A median filter with a radius equal to *clean_stdev****lamp_flats.factor** is applied for **lamp_flats.iterations** iterations to remove small clusters of bad pixels. Finally pixels which have different values are promoted to bad pixels by comparing the median filtered image with the reference image.

- A master flat field is determined. If **lamp_flats.interpol_ind**==TRUE, using the input bad pixel mask, and the slitlets position information, bad pixels are interpolated over a given radius **lamp_flats.max_rad**.

  Then the intensity is normalised to that of the central pixel.

  For QC purposes the fixed pattern noise is monitored on the resulting master flat field over two rectangular regions defined by parameters **lamp_flats.qc_fpn_xmin1**, **lamp_flats.qc_fpn_xmax1**, **lamp_flats.qc_fpn_ymin1**, **lamp_flats.qc_fpn_ymax1**, and **lamp_flats.qc_fpn_xmin2**,

lamp_flats.qc_fpn_xmax2, **lamp_flats.qc_fpn_ymin2**, **lamp_flats.qc_fpn_ymax2**., and the recipe computes the number of pixels which lie outside a given range (**lamp_flats.qc_thresh_min** and **lamp_flats.qc_thresh_max**).

A bad pixel mask is then determined on the master flat field by flagging all pixels which deviate too much from a threshold.

The input flat field frames (in our case only the master flat) are stacked. An average with rejection (parameters **bp_norm.low_rejection** and **bp_norm.high_rejection**) is computed. Pixels whose intensity lie outside the range [**bp_norm.min_cut**, **bp_norm.max_cut**] are flagged as bad.

- Then the intensity tilt of each column is removed (the fit of the pixel intensity is subtracted from the pixel intensity) considering in this operation only pixels whose intensity differs from the linear fit value by no more than **bp_norm.sigma_factor** times the sigma of the pixel intensity.
- The standard deviation and the mean within a defined (**bp_norm.llx, bp_norm.lly, bp_norm.urx, bp_norm.ury**) rectangular zone is determined in a way that the extreme low (**bp_norm.low_cut**) and the extreme high (**bp_norm.high_cut**) values are chopped off.
- A bad pixel map of all the pixels which deviate from a certain threshold level is determined on the master flat field. The input frames are stacked in a cube and an average with rejection (**bp_norm.low_rejection, bp_norm.high_rejection**) is taken to remove cosmics. The pixels which lie outside an intensity range set by parameters **bp_norm.min_cut** and **bp_norm.low_cut** are removed from the resulting median image.
- If **bp_norm.threshold_index**==TRUE, pixels deviating more than (**bp_norm.mean_factor*bp_norm.sigma**) from the clean mean are declared bad. Then a further pixel cleaning is performed involving a pixel's nearest neighbours using one of several methods defined by corresponding values of the parameter **bp_norm.method_index** and depending on the values of **bp_norm.factor** and possibly **bp_norm.iterations**, as described in 10.1.4.
- Then the resulting image is compared with the input image having the column intensity tilt removed and each changed pixel is indicated as bad. Finally, from this final image, a bad pixel mask is produced in which each good pixel is marked with 1 and each bad pixel with 0.

10.2.4 Optical distortion and slitlet distances determination: si_rec_distortions

- Normal flat frames are combined to determine a master flat and a bad pixel map. To perform this operation some parameters have peculiar default values: **bp_dist.factor=999.0**, **bp_dist.mean_factor=999.0**. These values have been chosen to ensure robustness.
- A set of fibre lamp-on frames are stacked (see 10.2.5 recipe for more details) with different rejection thresholds to determine a fake fibres-on and a fake fibres-off frame. Since the distortion coefficients are not yet known, no distortion correction is applied (internally the recipe assumes **stacked.warpfix_ind** == FALSE). The fake fibres-off frame is subtracted from the fake fibres-on frame.
- Undistorted arc lamp frames are used to determine a wavelength calibration solution.
- The distortion map is computed on the fake fibres-on - fibres-off frame. This step involves the following operations.

- The stacked fibres-on - fibres-off and ThAr frames are multiplied by the bad pixel map.
- On the ThAr corrected frame the arc lamp line positions are determined (parameters **distortion.begin_wave**, **distortion.guess_disp1**, **distortion.guess_disp2**, **distortion.min_diff_col_tilt_int**, **distortion.half_width**, **distortion.sigma**) as described in 10.1.8.
- A wavelength calibration is performed (the result depends on the following parameters: **distortion.guess_disp1**, **distortion.half_width**, **distortion.min_amplitude**, **distortion.max_residual**, **distortion.fwhm**, **distortion.n_a_coefficients**, **distortion.n_b_coefficients**, **distortion.sigma_factor**, **distortion.pixel_dist**, **distortion.pixel_tol**) as described in 10.1.9.
- A possible line shift is computed as described in 10.1.10. This result depends also on: **distortion.n_a_coefficients**.
- The slitlet positions are computed as a function of **distortion.box_length**, **distortion.y_box**, **distortion.diff_tol** as described in 10.1.12.
- The slitlet distances are computed with a north-south test as described in 10.1.13. In this operation relevant user definable parameters are **north_south_test.n_slits**, **distortion.ns_half_width**, **distortion.ns_fwhm**, **distortion.min_diff**, **distortion.dev_tol**, **distortion.lo_pos**, **distortion.hi_pos**).
- Those are averaged to get the position of the first slitlet.
- Finally the distortions are determined (assuming as offset the position found for the first slitlet).

- Finally the fake fibres-on - fibres-off frame is corrected for distortions and the slitlet distances are determined again with a north south test on the frame corrected for distortions.

  The north south test involves the following operations:

  - The input frames are stacked in a cube and an average with rejection is computed (parameters **north_south_test.low_rejection**, **north_south_test.high_rejection**).
  - If **north_south_test.gauss_ind** == TRUE the resulting image is convolved with a Gaussian of FWHM **north_south_test.kernel_half_width**.
  - If **north_south_test.mask_ind** == TRUE the resulting image is multiplied by the input bad pixels map. This image is saved as a product.
  - Then the slitlet distances are computed and saved as a table product. The result depends on the following user definable parameters: **north_south_test.half_width**, **north_south_test.fwhm** , **north_south_test.min_diff**, **north_south_test.dev_tol**.
  - Each of the 32 continuum slit spectra at the same spatial position is fit by a Gaussian along the spatial direction. This determines for each detector's raw the position of each slitlet. Those are averaged along the row. Then the distance of the left edge of each slitlet from a reference one is computed and thus 31 relative distances are found.

10.2.5 Wavelength solution determination: si_rec_wavecal

Frame stacking This macro step is common to several recipes: si_rec_distortion, si_rec_wavecal, si_rec_objnod, si_rec_psf, si_rec_stdstar.

- The input arc lamp frames (which can be "on" or "off" frames, or dark calibrations, or objects or sky frames) are stacked.

- The “off” (or sky) frame is subtracted from the “on” (or object) frame.
- If **stacked.flat_index**==TRUE the result is flat-fielded using the input master flat field.
- Static bad pixels are indicated (**stacked.mask_ind=1**) or not (**stacked.mask_ind=0**). If **stacked.mask_ind=1**:
  - If **stacked.ind_index** is FALSE bad pixels are interpolated over a given range defined by **stacked.mask_rad** and using the information on the slitlet edge positions contained in the SLIT_POS frame.
  - If **stacked.ind_index** is TRUE bad pixels are indicated by multiplying the frame by the bad pixel map.
- If **stacked.warpfix_ind**==TRUE the distortion is corrected using a given kernel (**stacked.warpfix_kernel**).
- Finally some quality control is performed on the resulting stacked frame by monitoring the number of saturated pixels (**stacked.qc_thresh_max**) and the maximum flux.

Wavelength calibration This macro data reduction step determines the dispersion relation and the slitlet position table. If an input slitlet position table is not present one can determine it from scratch by setting the parameter **wavecal.slitpos_bootstrap_switch** to TRUE. For the sake of robustness, it is suggested to use **wavecal.slitpos_bootstrap_switch** == FALSE and provide an input slitlets positions table, for example the corresponding one (band, preoptics) provided in the calibration data available with this release, and then let the recipe refine it. Reference slitlet position tables are provided together with the SINFONI pipeline.

- The input arc lamp-on and lamp-off frames are loaded.
- If **wavecal.calib_indicator** == TRUE and **wavecal.wave_map_ind** == TRUE the input arc lamp reference line list is loaded.
- If one needs to do the wavelength calibration (**wavecal.calib_indicator** == TRUE and **wavecal.wave_map_ind** == FALSE), the recipe finds the emission lines present in each arc lamp spectrum as described in 10.1.8
- Then the dispersion relation is determined as described in 10.1.9.
- The overall line shift between the solution found from the fit as described in 10.1.10 and the position of the lines in the frame is monitored for quality control.
- Then the recipe logs additional QC parameters: the number of found lines, the number of saturated pixels, the flux maximum of the frame difference “on” - “off”.
- The wavelength map is saved. This is an image in which each pixel is associated with a wavelength value in microns.
- If **wavecal.write_coeff_ind**==TRUE the polynomial fit coefficients are saved in a FITS table.
- If **wavecal.write_par_ind**==TRUE the Gaussian fit parameters are saved in a table. The number of detected lines, the median and the average FWHM are monitored for quality control.

- If a wavelength map is available (**wavecal.wave_map_ind**==TRUE) and a dispersion relation does not need to be found (**wavecal.calib_ind**==0) one may want to correct what is already available to match the actual arc lamp frame, by creating a shifted dispersion relation. Then the overall line shift between the solution found from the fit and the position of the lines in the frame is computed.

- Finally if **wavecal.calib_indicator**==TRUE and **wavecal.estimate_indicator**==TRUE the slitlet positions are fit using a parameter list input FITS table. The slitlet position determination may follow either the Boltzmann or the estimate methods as described in 10.1.12.

10.2.6 Science observations: si_rec_objnod

- The input frames are selected defining obj-sky pairs. In case a sky frame is missing in the set of frames a sky frame is estimated as described in 10.1.16. Each pair is stacked.

- Each stacked frame is resampled using the information from the input calibration image (WAVE_MAP) doing a polynomial interpolation with **objnod.n_coeffs** parameters to generate an output 3D frame as described in 10.1.14. This operation determines the dispersion, the minimum, the maximum and the central wavelength as well as the central pixel of the output calibrated image.

- If **objnod.north_south_ind** == FALSE, a 3D frame is reconstructed from each stacked frame by using the information contained in the slitlet position table SLIT_POS.

- The 3D frame is refined using different methods (**objnod.fine_tuning_method**). Possible values are P (polynomial of order **objnod.order**), S (Spline). The 3D frame is stretched in Y to make it appear as a parallepipedus with a square cross-section. We usually call this a “3D cube”.

- If **objnod.jitter_index**==TRUE, finally all the cubes corresponding to the different object jittered positions are merged into a big data cube by averaging the overlap regions weighted by integration times and using a given kernel type **objnod.kernel_type** as described in 10.1.15 possibly scaling the sky background (if **objnod.scale_sky** == TRUE) and doing a kappa sigma clipping (if **objnod.kappa_sigma** == TRUE) of intensity outliers.

10.2.7 STD star data reduction: si_rec_stdstar

The initial data reduction is the same as the one of a normal science frame (si_rec_objnod). Then:

- The position of the source maximum, its centroid and FWHM, are determined from the average image of the coadded cube. Those parameters are used to automatically set the optimal extraction parameters in a square region whose size depends on the average of the FWHM in the X and Y directions and a user definable scale parameter **std_star.factor**, eventually reduced in case the star position falls outside the actual z-cross section plane. The output is in the form of an image and a table. If **std_star.conversion_index**==TRUE a conversion magnitude to counts/seconds is performed.

- The instrument efficiency is determined as described in 10.1.17.

- The extracted star spectrum and the efficiency are saved in a table.

10.2.8 PSF data reduction: si_rec_psf

The initial data reduction is the same as the one of a normal science frame. Then the main PSF standard parameters, the instrument strehl and the encircled energy are determined as described in 10.1.18 and 10.1.19.

A Installation

This chapter gives generic instructions on how to obtain, build and install the SINFONI pipeline. Even if this chapter is kept as up-to-date as much as possible, it may not be fully applicable to a particular release. This might especially happen for patch releases. One is therefore advised to read the installation instructions delivered with the SINFONI pipeline distribution kit. These release-specific instructions can be found in the file `README` located in the top-level directory of the unpacked SINFONI pipeline source tree. The supported platforms are listed in Section A.1. It is recommended reading through Section A.2.2 before starting the installation.

A bundled version of the SINFONI pipeline with all the required tools and an installer script is available from `http://www.eso.org/pipelines/`, for users who are not familiar with the installation of software packages.

A.1 Supported platforms

The utilisation of the GNU build tools should allow to build and install the SINFONI pipeline on a variety of UNIX platforms, but it has only been verified on the VLT target platforms:

- Linux (glibc 2.1 or later),
- Sun Solaris 2.8 or later,

using the GNU C compiler (version 3.2 or newer).

A.2 Building the SINFONI pipeline

This section shows how to obtain, build and install the SINFONI pipeline from the official source distribution.

A.2.1 Requirements

To compile and install the SINFONI pipeline one needs:

- the GNU C compiler (version 3.2 or later),
- the GNU `gzip` data compression program,
- a version of the `tar` file-archiving program, and,
- the GNU `make` utility.

An installation of the Common Pipeline library (CPL) must also be available on the system. Currently the CPL version 2.1.1 or newer is required. The CPL distribution can be obtained from `http://www.eso.org/cpl`.

Please note that CPL itself depends on an existing qfits installation. The qfits sources are available from the CPL download page or directly from the qfits homepage at `http://www.eso.org/projects/aot/qfits`. In conjunction with CPL 2.1.1 qfits 5.3.1 must be used.

In order to run the SINFONI pipeline recipes a front-end application is also required. Currently there are two such applications available, a command-line tool called *EsoRex* and the Java based data file organizer, *Gasgano*, which provides an intuitive graphical user interface (see Section 4.2, page 17). At least one of them must be installed. The *EsoRex* and *Gasgano* packages are available at `http://www.eso.org/cpl/esorex.html` and `http://www.eso.org/gasgano` respectively.

For installation instructions of any of the additional packages mentioned before please refer to the documentation of these packages.

A.2.2 Compiling and installing the SINFONI pipeline

The SINFONI pipeline distribution kit 1.0 contains:

sinfoni-manual-1.0.pdf	The SINFONI pipeline manual
install_pipeline	Install script
qfits-5.3.1.tar.gz	QFITS 5.3.1
cpl-2.1.1.tar.gz	CPL 2.1.1
esorex-3.5.1.tar.gz	esorex 3.5.1
gasgano-2.2.3-Linux.tar.gz	GASGANO 2.2.3 for Linux
gasgano-2.2.3-SunOS.tar.gz	GASGANO 2.2.3 for SunOS
eclipse-sinfo.tar.gz	eclipse library modified to support SINFONI data reduction
sinfoni-1.2.0.tar.gz	SINFONI 1.2.0
sinfoni-calib-1.2.0.tar.gz	SINFONI calibration files 1.2.0

Here is a description of the installation procedure:

1. Change directory to where you want to retrieve the SINFONI pipeline recipes 1.2.0 package. It can be any directory of your choice but not:

   ```
   $HOME/gasgano
   $HOME/.esorex
   ```

2. Download from the ESO ftp server, `http://www.eso.org/pipelines/`, the latest release of the SINFONI pipeline distribution.

3. Verify the checksum value of the tar file with the cksum command.

4. Unpack using the following command:

   ```
   tar -xvf sinfo-kit-1.2.tar
   ```

   Note that the size of the installed software (including *Gasgano*) together with the static calibration data is about 27Mb.

5. Install: after moving to the top installation directory,

   ```
   cd sinfo-kit-1.2
   ```

   it is possible to perform a simple installation using the available installer script (*recommended*):

   ```
   ./install_pipeline
   ```

   (beware: the execution may take a few minutes on Linux and several minutes on SunOS).

   Note that this release still needs to link to the eclipse library. At the end of the installation the user in addition to follow what reported by the installation script, needs to source an file ($HOME/..eclipse_bash.rc or $HOME/..eclipse_bash.rc, depending from the user shell) to set a few environment variables used by a few low level eclipse library based modules.

   By default the script will install the SINFONI recipes, *Gasgano*, *EsoRex*, all the necessary libraries, and the static calibration tables, into a directory tree rooted at `$HOME`. A different path may be specified as soon as the script is run.

   The only exception to all this is the *Gasgano* tool, that will always be installed under the directory `$HOME/gasgano`. Note that the installer will move an existing `$HOME/gasgano` directory to `$HOME/gasgano.old` before the new *Gasgano* version is installed.

   Important: the installation script would ensure that any existing *Gasgano* and *EsoRex* setup would be inherited into the newly installed configuration files (avoiding in this way any conflict with other installed instrument pipelines).

Alternatively, it is possible to perform a manual installation (*experienced users only*): the `README` file located in the top installation directory contains more detailed information about a step-by-step installation.

B Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CPL	Common Pipeline Library
DFO	Data Flow Operations department
DFS	Data Flow System department
DMD	Data Management and Operations Division
DRS	Data Reduction System
ESO	European Southern Observatory
ESOREX	ESO-Recipe Execution tool
FITS	Flexible Image Transport System
FOV	Field Of View
FPN	Fixed Patter Noise
GUI	Graphical User Interface
OB	Observation Block
PSO	Paranal Science Operations
QC	Quality Control
RON	Read Out Noise
SINFONI	Spectrograph for INtegral Field Observations in the Near Infrared
SOF	Set Of Frames
SPIFFI	SPectrograph for Infrared Faint Field Imaging
UT	Unit Telescope
VLT	Very Large Telescope