<table>
<thead>
<tr>
<th>Issue/Rev.</th>
<th>Date</th>
<th>Section/Parag. affected</th>
<th>Reason/Initiation/Documents/Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>19/10/2005</td>
<td>All</td>
<td>Public release</td>
</tr>
<tr>
<td>2.0</td>
<td>29/06/2006</td>
<td>sec 6,10</td>
<td>added information on table products content</td>
</tr>
<tr>
<td>3.0</td>
<td>17/01/2007</td>
<td>all</td>
<td>Update, match new release, more figures to clarify algorithm and recipe description</td>
</tr>
<tr>
<td>4.0</td>
<td>15/05/2007</td>
<td>sec 3,11</td>
<td>Update, more accurate sky subtraction, WCS, fixed some bugs on Strehl computation</td>
</tr>
<tr>
<td>5.0</td>
<td>19/10/2007</td>
<td>sec 3</td>
<td>Update</td>
</tr>
<tr>
<td>6.0</td>
<td>02/04/2008</td>
<td>sec 3,5,10,11</td>
<td>Update</td>
</tr>
<tr>
<td>7.0</td>
<td>15/07/2008</td>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>8.0</td>
<td>20/10/2008</td>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>9.0</td>
<td>22/04/2009</td>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>10.0</td>
<td>30/07/2009</td>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>10.1</td>
<td>15/10/2009</td>
<td></td>
<td>Update</td>
</tr>
<tr>
<td>10.2</td>
<td>16/11/2009</td>
<td>sec 8, 10.8, 11.1</td>
<td>added correction due atmospheric refraction</td>
</tr>
<tr>
<td>10.3</td>
<td>15/12/2009</td>
<td>sec 10.17</td>
<td>flux catalog for efficiency; atmospheric refraction</td>
</tr>
<tr>
<td>11.0</td>
<td>21/06/2010</td>
<td></td>
<td>sinfoni public release 2.2.1</td>
</tr>
<tr>
<td>13.0</td>
<td>12/08/2010</td>
<td></td>
<td>sinfoni public release 2.2.3</td>
</tr>
<tr>
<td>14.0</td>
<td>07/08/2011</td>
<td></td>
<td>sinfoni public release 2.2.5</td>
</tr>
<tr>
<td>15.0</td>
<td>12/10/2011</td>
<td></td>
<td>sinfoni public release 2.2.8</td>
</tr>
<tr>
<td>16.0</td>
<td>02/11/2011</td>
<td></td>
<td>sinfoni public release 2.2.9</td>
</tr>
<tr>
<td>17.0</td>
<td>10/05/2012</td>
<td></td>
<td>sinfoni public release 2.2.9</td>
</tr>
<tr>
<td>18.0</td>
<td>21/04/2013</td>
<td></td>
<td>sinfoni public release 2.3.3</td>
</tr>
<tr>
<td>19.0</td>
<td>15/11/2014</td>
<td></td>
<td>sinfoni public release 2.4.0</td>
</tr>
<tr>
<td>19.1</td>
<td>09/04/2014</td>
<td></td>
<td>sinfoni public release 2.5.2</td>
</tr>
<tr>
<td>19.2</td>
<td>24/02/2015</td>
<td></td>
<td>sinfoni public release 2.6.0</td>
</tr>
<tr>
<td>19.3</td>
<td>20/07/2015</td>
<td></td>
<td>sinfoni public release 2.6.5</td>
</tr>
<tr>
<td>19.4</td>
<td>14/09/2015</td>
<td></td>
<td>sinfoni public release 2.6.8</td>
</tr>
<tr>
<td>19.5</td>
<td>26/10/2015</td>
<td></td>
<td>sinfoni public release 2.7.0</td>
</tr>
<tr>
<td>19.6</td>
<td>03/03/2016</td>
<td></td>
<td>sinfoni public release 2.9.0</td>
</tr>
<tr>
<td>3.3.2</td>
<td>20/05/2021</td>
<td></td>
<td>sinfoni public release 3.3.2</td>
</tr>
</tbody>
</table>
This page was intentionally left blank
Contents

1 Introduction 13
  1.1 Purpose ................................................. 13
  1.2 Acknowledgements ........................................ 13
  1.3 Scope .................................................. 13
  1.4 Reference documents ....................................... 13

2 Overview 15

3 Recent improvements 16
  3.1 What’s new in release 3.3.2 ............................... 16

4 SINFONI Instrument Description 17
  4.1 Instrument overview ....................................... 17

5 Quick start 19
  5.1 SINFONI pipeline recipes ................................. 19
  5.2 An introduction to Reflex, Gasgano and EsoRex ............ 20
    5.2.1 Using Reflex ........................................ 20
    5.2.2 Example of data reduction using the Reflex-based SINFONI workflow .... 21
    5.2.3 Using Gasgano ........................................ 26
    5.2.4 Using EsoRex ......................................... 30
  5.3 Example of data reduction using EsoRex ..................... 32

6 Known problems 41

7 Instrument Data Description 44
  7.1 Data features ........................................... 44
  7.2 Raw frames ............................................. 45

8 Static Calibration Data 51
  8.1 Line reference table .................................... 51
  8.2 DRS setup table ........................................ 51
8.3 Reference bad pixel map .......................................................... 51
8.4 Reference table with flux of STD star for efficiency computation .......... 51
8.5 Reference tables with polynomial coefficients for correction due atmospheric refraction .... 51

9 Data Reduction 52
9.1 Data reduction overview .......................................................... 52
9.2 Required input data ................................................................. 53
9.3 Reduction cascade ................................................................. 54

10 Pipeline Recipes Interfaces 57
10.1 sinfo_rec_detlin ................................................................. 57
10.1.1 Input ................................................................. 58
10.1.2 Output ................................................................. 58
10.1.3 Quality control ............................................................. 58
10.1.4 Parameters ................................................................. 59
10.2 sinfo_rec_lingain .............................................................. 60
10.2.1 Input ................................................................. 60
10.2.2 Output ................................................................. 60
10.2.3 Quality control ............................................................. 62
10.2.4 Parameters ................................................................. 62
10.3 sinfo_rec_mdark .............................................................. 63
10.3.1 Input ................................................................. 63
10.3.2 Output ................................................................. 63
10.3.3 Quality control ............................................................. 63
10.3.4 Parameters ................................................................. 65
10.4 sinfo_mdark_detmon .......................................................... 65
10.4.1 Input ................................................................. 65
10.4.2 Output ................................................................. 66
10.4.3 Quality control ............................................................. 66
10.4.4 Parameters ................................................................. 66
10.5 sinfo_rec_mflat .............................................................. 66
10.5.1 Input ............................................. 66
10.5.2 Output ......................................... 67
10.5.3 Quality control ................................. 67
10.5.4 Parameters ................................. 68

10.6 sinfo_rec_distortion ..................................... 69
10.6.1 Input ............................................. 69
10.6.2 Output ......................................... 70
10.6.3 Quality control ................................. 71
10.6.4 Parameters ................................. 71

10.7 sinfo_rec_wavecal ..................................... 74
10.7.1 Input ............................................. 74
10.7.2 Output ......................................... 76
10.7.3 Quality control ................................. 77
10.7.4 Parameters ................................. 78

10.8 sinfo_rec_jitter ..................................... 79
10.8.1 Input ............................................. 79
10.8.2 Output ......................................... 80
10.8.3 Quality control ................................. 82
10.8.4 Parameters ................................. 82

10.9 sinfo_rec_pupil ..................................... 84
10.9.1 Input ............................................. 84
10.9.2 Output ......................................... 84
10.9.3 Quality control ................................. 86
10.9.4 Parameters ................................. 86

10.10 sinfo_utl_bp_mask_add .................................. 86
10.10.1 Input ............................................. 86
10.10.2 Output ......................................... 86
10.10.3 Parameters ................................. 86

10.11 sinfo_utl_ima_arith .................................. 86
10.11.1 Input ............................................. 86
10.11.2 Output .................................................. 87
10.11.3 Parameters .............................................. 87

10.12 sinfo_utl_cube_create .................................... 87
10.12.1 Input ................................................... 87
10.12.2 Output .................................................. 87
10.12.3 Parameters .............................................. 87

10.13 sinfo_utl_cube2ima ...................................... 88
10.13.1 Input ................................................... 88
10.13.2 Output .................................................. 88
10.13.3 Parameters .............................................. 88

10.14 sinfo_utl_cube2spectrum .................................. 88
10.14.1 Input ................................................... 88
10.14.2 Output .................................................. 89
10.14.3 Parameters .............................................. 89

10.15 sinfo_utl_cube_arith ...................................... 89
10.15.1 Input ................................................... 89
10.15.2 Output .................................................. 90
10.15.3 Parameters .............................................. 90

10.16 sinfo_utl_cube_combine ................................... 90
10.16.1 Input ................................................... 90
10.16.2 Output .................................................. 91
10.16.3 Parameters .............................................. 91

10.17 sinfo_utl_skycor .......................................... 92
10.17.1 Input ................................................... 92
10.17.2 Output .................................................. 92
10.17.3 Parameters .............................................. 92

10.18 sinfo_utl_skymap .......................................... 93
10.18.1 Input ................................................... 93
10.18.2 Output .................................................. 93
10.18.3 Parameters .............................................. 93
10.19 sinfo_utl_spectrum_divide_by_blackbody ........................................ 94
  10.19.1 Input ................................................................. 94
  10.19.2 Output ............................................................... 94
  10.19.3 Parameters ........................................................... 94

10.20 sinfo_utl_spectrum_wavelength_shift ............................................ 94
  10.20.1 Input ................................................................. 94
  10.20.2 Output ............................................................... 95
  10.20.3 Parameters ........................................................... 95

11 Algorithms and recipe details ......................................................... 96
  11.1 Algorithms .................................................................. 96
    11.1.1 sinfo_rec_mdark_detmon ............................................. 96
    11.1.2 Read Out Noise determination (common algorithm across pipelines) .... 96
    11.1.3 Histogram computation ................................................ 96
    11.1.4 PTC and MED Algorithms for Gain computation ...................... 96
    11.1.5 Detector linearity computation (IR): irplib_detmon_nir_linear ........ 98
    11.1.6 Iteratives clean mean and sigma computation ........................ 99
    11.1.7 Frame stacking ....................................................... 100
    11.1.8 Average with rejection ............................................... 100
    11.1.9 Detector non linearity computation ................................... 100
    11.1.10 Nearest neighbours bad pixel cleaning ................................ 101
    11.1.11 Detector gain computation ......................................... 103
    11.1.12 Read Out Noise computation ....................................... 104
    11.1.13 Fixed Pattern Noise computation .................................... 104
    11.1.14 Line position determination ........................................ 104
    11.1.15 Dispersion relation and wavelength map determination ............. 105
    11.1.16 Line shift computation ............................................. 106
    11.1.17 Dispersion relation adjustment ..................................... 106
    11.1.18 Slitlet position computation ....................................... 107
    11.1.19 Slitlet distances computation ..................................... 108
    11.1.20 Optical distortions computation .................................... 108
11.1.21 Cube construction: resampling ......................................................... 109
11.1.22 Cube coaddition ................................................................. 109
11.1.23 Mosaic cube size determination ........................................ 111
11.1.24 Cubes alignment ................................................................. 111
11.1.25 Estimation of the sky from object frames in case the input set is missing sky frames ... 111
11.1.26 Sky residual correction ......................................................... 112
11.1.27 Efficiency computation ......................................................... 116
11.1.28 Response computation ......................................................... 118

11.2 Instrument response determination .............................................. 118
11.2.1 Flux calibration ................................................................. 119
11.2.2 Strehl computation ............................................................... 120
11.2.3 Standard star position detection ........................................ 120
11.2.4 Atmospheric refraction algorithm ........................................ 121
11.2.5 Encircled energy computation .............................................. 122
11.2.6 Spectrum extraction ............................................................. 122

11.3 Recipes .............................................................................. 122
11.3.1 Detector linearity and non-linear bad pixel map determination: sinfo_rec_detlin .... 122
11.3.2 Detector linearity and gain computation: sinfo_rec_lingain ......................... 123
11.3.3 Master dark and bad pixel map determination: sinfo_rec_mdark .................. 123
11.3.4 Master dark determination: sinfo_rec_mdark_detmon .......................... 123
11.3.5 Master flat and threshold pixels (bad pixel map) determination: sinfo_rec_mflat ... 123
11.3.6 Optical distortion and slitlet distances determination: sinfo_rec_distortions ......... 125
11.3.7 Wavelength solution determination: sinfo_rec_wavecal .......................... 126
11.3.8 Science observations: sinfo_rec_jitter ........................................ 127
11.3.9 STD star data reduction: sinfo_rec_jitter ...................................... 128
11.3.10 PSF data reduction: sinfo_rec_jitter ......................................... 128
11.3.11 PUPIL data reduction: sinfo_rec_pupil ..................................... 129

A Workflow technicalities .............................................................. 130
A.1 Data requirements and optional inputs ..................................... 130
A.2 Overall layout ................................................................. 130
A.3 OCA rules ................................................................. 131
A.4 Workflow fine tuning and hints ................................. 132

B Installation ................................................................. 134

B.1 Supported platforms ............................................... 134
B.2 Building the SINFONI pipeline ................................. 134
  B.2.1 Requirements ................................................... 134
  B.2.2 Compiling and installing the SINFONI pipeline .... 135

C Abbreviations and acronyms ........................................ 137
1 Introduction

1.1 Purpose

The SINFONI pipeline is a subsystem of the VLT Data Flow System (DFS). Its target user is ESO Data Products Department (DPD) 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 May 20, 2021 (version 3.3.2).

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 1.2.0 benefited 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. Very useful was the collaboration with Richard Davies from MPE which lead to an improvement of the sky subtraction quality since release 1.6.0. From May 2009 till end of March 2009 Andrea Modigliani passed the responsibility for this project to Konstantin Mirny. Then Andrea continued to be responsible of this project.

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 [17]. For general information about the current instrument pipelines status we remind the user of [8]. Quality control information are at [7].

Additional information on the Common Pipeline Library (CPL) and ESOREX can be found respectively at [13], [15]. The Gasgano front end is described in [16]. A description of the instrument is in [9]. The SINFONI instrument user manual is in [11]. while results of Science Verifications (SV) are at [10]. A clear and compact description of the SINFONI pipeline is in [1].

1.4 Reference documents


2 Overview

In collaboration with instrument consortia, the Pipeline Systems Department (PSD) of the Software Development 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 [16] and EsoRex [15], both included in the pipeline distribution (see Appendix B, page 134). These applications can also be downloaded separately from www.eso.org/gasgano and www.eso.org/cpl/esorex.html. An illustrated introduction to Gasgano is provided in the "Quick Start" section of this manual (see page 19).

The SINFONI instrument and the different types of SINFONI raw frames and auxiliary data are described in Sections 4, 7, and 8.

A brief introduction to the usage of the available reduction recipes using Gasgano or EsoRex is presented in Section 5. In section 6 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 9.

More details on what are inputs, products, quality control measured quantities, and controlling parameters of each recipe is given in section 10.

More detailed descriptions of the data reduction algorithms used by the individual pipeline recipes can be found in Section 11.

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

3.1 What’s new in release 3.3.2

The following improvements have been made to the SINFONI pipeline:

- Fixed a bug in passing parameter values to the reflex python interactive workflows.
- Updated third party library dependencies (e.g. CPL) in conjunction with the annual coordinated public release.
4 SINFONI Instrument Description

SINFONI has been developed by ESO and the Max-Planck-Institut für extraterrestrische Physik (MPE) in Garching in collaboration with NOVA. The instrument has been made available to the community and started operations in Paranal on April 1st, 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 [www.eso.org/sci/facilities/paranal/instruments/sinfoni](http://www.eso.org/sci/facilities/paranal/instruments/sinfoni).

4.1 Instrument overview

SINFONI is composed of two subsystems: the Multi-Application Curvature Adaptive Optics unit (MACAO, developed by ESO), which allows diffraction and seeing limited observations; and the SPectrograph for Integral Faint Field Imaging, a near-infrared (1.05 - 2.45 µm) integral field spectrograph, (SPIFFI, developed by MPE). It is described with more details in [21], [2], [20].

![Figure 4.1.1: 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.](image)

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 250, 10 and 25 mas/pixel, 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 4.1.2: 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.
5 Quick start

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

5.1 SINFONI pipeline recipes

The current SINFONI pipeline is based on a set of 6 stand-alone recipes involved in the data reduction cascade:

- `sinfo_rec_detlin` to evaluate the detector linearity and generate a corresponding non linear pixel map.
- `sinfo_rec_mdark` to create a master dark and a hot-pixel map.
- `sinfo_rec_mflat` to create a master flat and a map of pixels which have intensities greater than a given threshold.
- `sinfo_rec_distortion` to compute the optical distortions and slitlets distances.
- `sinfo_rec_wavecal` for wavelength calibration.
- `sinfo_rec_jitter` for PSF, telluric standard and other science data reduction.

Other 3 stand-alone recipes are also provided, used for quality control:

- `sinfo_rec_pupil` for PUPIL data reduction (for Paranal operations, to monitor telescope pupil centering on the SINFONI FOV)
- `sinfo_rec_lingain` for detector characterization: it computes detector’s linearity and gain and generates a corresponding non linear pixel map, sharing implementation across pipelines.
- `sinfo_rec_dark_detmon` Master dark and detector signal non uniformity map generation (recipe used for quality control)

Additional 13 stand-alone recipes are also provided, that perform useful tasks:

- `sinfo_utl_bp_mask_add` to coadd bad pixel maps.
- `sinfo_utl_cube2ima` to collapse a cube to an image over a given wavelength range.
- `sinfo_utl_cube2spectrum` to extract a spectrum from a cube.
- `sinfo_utl_cube_arith` to perform cube arithmetics.
- `sinfo_utl_cube_combine` to combine and coadd cubes or mosaics.
- `sinfo_utl_cube_create` to create a cube out of any input raw on/off frame pair.
- `sinfo_utl ima arith` to do image arithmetics.
**sinfo_utl_ima_line_corr** to correct detector bad rows generated by the data processing hardcoded at the detector level.

**sinfo_utl_skycor** to correct object spectra from sky residuals.

**sinfo_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).

**sinfo_utl_spectrum_divide_by_blackbody** to divide a spectrum by a black body spectrum.

**sinfo_utl_spectrum_wavelegth_shift** to shift in wavelength a spectrum.

### 5.2 An introduction to Reflex, Gasgano and EsoRex

Before being able to call pipeline recipes to process a set of data, the data must be correctly 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 observing 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. Since all the required information is stored in the FITS headers, data association is based on a set of header keywords (called "association keywords") and the process 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, namely:

- **Reflex** is a graphical tool that helps the user to execute data reduction workflows which contain several recipes. This dramatically decreases the time the user needs to run a whole reduction chain, from calibration and raw data down to the final products. **Reflex** takes care of grouping the different data sets, associating the calibration frames and managing the interdependencies between recipes in the calibration cascade. **Reflex is the recommended software tool for reducing your data.**

- **Gasgano** is an alternative data management tool that simplifies the data organization process. In addition, **Gasgano** allows the user to execute directly the pipeline recipes on a set of selected files.

- **EsoRex** is a command line tool used to run the pipeline recipes. **EsoRex** commands can be easily scripted.

- The Paranal observatory implements automatic data management tools that trigger the execution of pipeline recipes. This aspect is not covered in this manual.

#### 5.2.1 Using Reflex

**Reflex** is the recommended tool to reduce complete data sets that include all the calibration frames. It is an advanced tool, and yet easy to use, that is geared towards maximum scientific return. It is based on the workflow engine **Kepler** [19].

This manual does not cover the installation of **Reflex**. Please refer to [18] for the installation procedure which also contains a detailed description of the **Reflex** application. What follows is a very brief summary of it.
Once installed, Reflex can be executed with the command:

```
user@host# reflex &
```

![Figure 5.2.1: Fresh Reflex canvas.](image)

Reflex main concepts are workflows and actors. Workflows are canvasses which show the interdependence of the pipeline recipes, allowing the user to easily obtain an overview of the reduction steps. Workflows have the advantage of requiring a small learning curve in order to get the pipeline running.

Actors are the entities which actually perform some kind of operation. In Reflex, to each main actors correspond the pipeline recipes themselves, which perform the data reduction steps, but there are other actors such as the DataOrganizer, or the FitsRouter that are useful to manage the data files. Each actor can be configured by right-clicking on it and selecting Configure Actor as shown in Figure 5.2.2. In the case of the recipe actors, the recipe parameters are part of the actor and make up the second group of parameters.

In addition to those elements, the workflow contains variables that contain the most important settings, such as the directories where data is located and will be saved.

To start using Reflex with this pipeline, please refer to [18].

### 5.2.2 Example of data reduction using the Reflex-based SINFONI workflow

For the user who is keen on starting reductions without being distracted by detailed documentation, we describe the steps to be performed to reduce the science data provided in the SINFONI demo data set supplied with the Reflex 2.9 release. By following these steps, the user should have enough information to attempt a reduction of his/her own data without any further reading:
Figure 5.2.2: Parameters of a recipe actor. The first group of parameters affect the execution of the pipeline recipe and are common to all recipe actors. The second group of parameters are specific to the pipeline recipe to be called and they are identical to those that can be configured in EsoRex (see 5.2.4).
Figure 5.2.3: The empty Reflex canvas.

Figure 5.2.4: SINFONI workflow general layout.
Figure 5.2.5: The “Select Datasets” pop-up window.
1. Start the Reflex application:

```
reflex &
```

The empty Reflex canvas as shown in Figure 5.2.1 will appear.

2. Now open the SINFONI workflow by clicking on File -> Open File, selecting first sinfo-3.3.2 and then the file sinfo.xml in the file browser. You will be presented with the workflow canvas shown in Figure 5.2.4. Note that the workflow will appear as a canvas in a new window.

3. To aid in the visual tracking of the reduction cascade, it is advisable to use component (or actor) highlighting. Click on Tools -> Animate at Runtime, enter the number of milliseconds representing the animation interval (100 ms is recommended), and click OK.

4. Under “Setup Directories” in the workflow canvas there are seven parameters that specify important directories (green dots). Setting the value of ROOT_DATA_DIR is the only necessary modification if you want to process data other than the demo data\(^1\), since the value of this parameter specifies the working directory within which the other directories are organised. Double-click on the parameter ROOT_DATA_DIR and a pop-up window will appear allowing you to modify the directory string, which you may either edit directly, or use the Browse button to select the directory from a file browser. When you have finished, click OK to save your changes.

5. Click the button to start the workflow

6. The workflow will highlight the Data Organiser actor which has recursively scanned the raw data directory (specified by the parameter RAWDATA_DIR under “Setup Directories” in the workflow canvas) and constructs the DataSets. Note that the calibration and reference data must be present either in RAWDATA_DIR or in CALIB_DATA_DIR, otherwise DataSets may be incomplete and cannot be processed. However, if the same reference file was downloaded twice in different places this creates a problem as Reflex cannot decide which one to use.

7. The Data Set Chooser actor will be highlighted next and will display a “Select Datasets” window (see Figure 5.2.5) that lists the DataSets along with the values of a selection of useful header keywords\(^2\). The first column consists of a set of tick boxes which allow the user to select the DataSets to be processed, and by default all complete DataSets are selected.

8. Click the Continue button and watch the progress of the workflow by following the red highlighting of the actors. A window will show which DataSet is currently being processed.

---

\(^1\)If you used the install script install_reflex, then the value of the parameter ROOT_DATA_DIR will already be set correctly to the directory where the demo data was downloaded.

\(^2\)The keywords listed can be changed by right-clicking on the DataOrganiser Actor, selecting Configure Actor, and then changing the list of keywords in the second line of the pop-up window. Make sure that the Lazy Mode is not active and then click on Commit to save the change.
9. When the reduction of the current DataSet finishes, a pop-up window will appear showing the directory where the final products have been saved.

10. The workflow will continue with the remaining DataSets following the same steps described above.

11. After the workflow has finished, all the products from all the DataSets can be found in a directory under `END_PRODUCTS_DIR` with the name with the workflow start timestamp. Further subdirectories will be found with the name of each DataSet.

5.2.3 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 5.2.6 (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 5.2.7, 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 *sinfo_rec_stdstar* recipe. This will open a *Gasgano* recipe execution window (see Figure 5.2.8), 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* [16] for a more complete description of the *Gasgano* interface.

Please refer to the *Gasgano User’s Manual* [16] for a more complete description of the *Gasgano* interface.
Figure 5.2.6: The Gasgano main window.
Figure 5.2.7: Selecting files to be processed by a SINFONI pipeline recipe.
Figure 5.2.8: *The Gasgano recipe execution window.*
5.2.4 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 7.2, page 45). 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\(^3\) in an ASCII file, the set-of-frames (SOF), that is required when launching a recipe.\(^4\)

Here is an example of SOF, valid for the sinfo_rec_wavecal recipe:\(^5\)

```
/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.fits 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.fits DRS_SETUP_WAVE
/file_path/SLIT_POS_H_250.fits 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 sinfo_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 5.2.3, page 26).

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.

\(^3\)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.

\(^4\)The set-of-frames corresponds to the Input Frames panel of the Gasgano recipe execution window (see Figure 5.2.8, page 29).

\(^5\)We list the file SLIT_POS_H_250.fits as an input file, as, for robustness, we suggest the user to set the parameter slit-pos_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 [15]) 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.\(^6\) 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 sinfo_rec_wavecal has its EsoRex generated configuration file named sinfo_rec_wavecal.rc, and is generated with the command:

```
esorex -create-config sinfo_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.\(^7\)

---

\(^6\)The EsoRex recipe configuration file corresponds to the Parameters panel of the Gasgano recipe execution window (see Figure 5.2.8, page 29).

\(^7\)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 10 and their role is described in Section 11.

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 `sinfo_rec_wavecal` for processing the files specified in the set-of-frames `sinfo_rec_wavecal.sof`:

```
esorex sinfo_rec_wavecal sinfo_rec_wavecal.sof
```

The recipe parameters can be modified 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 `sinfo_rec_wavecal` recipe `wcal-pixel_tol` parameter to 3.0, the following should be typed:

```
esorex sinfo_rec_wavecal --wcal-pixel_tol=3.0 sinfo_rec_wavecal.sof
```

For more information on `EsoRex`, see [www.eso.org/cpl/esorex.html](http://www.eso.org/cpl/esorex.html).

### 5.3 Example of data reduction using EsoRex

A simple, typical data reduction procedure is described here.\(^8\)

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`. 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 characterized by DPR.TYPE='LINEARITY,LAMP'

\(^8\)The procedure using *Gasgano* is conceptually identical.

\(^9\)We omit here the prefix `HIERARCH ESO`
In case the user would like to use these frames with the new recipe `sinfo_rec_lingain` the tag to be used must also indicate if the lamp used to illuminate the detector was switched “on” or “off”, for example as in the following example:

```
/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.2005-02-26T20:10:56.403.fits LINEARITY_LAMP
```

Fibre frames, flat frames and arc lamp frames to compute distortions: those frames have DPR.TYPE respectively equal to 'DISTORTION,FIBRE,NS' 'DISTORTION,FLAT,NS' (usually 2 frames, one with calibration flat lamp switched on, one with the lamp switched off), 'DISTORTION,WAVE,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
```

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

```
/path_raw/FLAT/K/100/SINFO.2005-02-28T16:27:43.232.fits FLAT_LAMP
```
Arc lamp frames: those frames are characterized by DPR.TYPE='WAVE,LAMP'

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

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

or PSF standard star frames: those frames are characterized by DPR.TYPE='PSF-CALIBRATOR' or DPR.TYPE='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 extension .fits for images and 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. 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.

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 extension .fits for images and tables.
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 sinfo_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
```

It is possible to generate a master dark and a detector signal non uniformity (DSNU) map with the command
```
esorex sinfo_rec_mdark_detmon mdark_sof
```

This command will generate three files: sinfo_rec_mdark_detmon_master_dit_1.fits, a master dark frame (PRO.CATG=MASTER_DARK), sinfo_rec_mdark_detmon_dsnu_map_1.fits, a DSNU image map (PRO.CATG=DSNU_MAP), and sinfo_rec_mdark_detmon_dsnu_table_1.fits, a DSNU table (PRO.CATG=DSNU_TABLE).

This recipe is being used for quality control and being tested for the reduction chain. We recommend the user not to use it in the usual reduction chain with this release.
```
rm -rf out*fits *.paf *.log
```

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.2005-02-26T20:10:38.240.fits LINEARITY_LAMP
... 
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:56.403.fits LINEARITY_LAMP
/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:59.183.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 sinfo_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”.

\texttt{mv out_bp_lin.fits /path_pro/BP_MAP_NL_K.fits}
\texttt{rm -rf out*fits *.paf *.log}

3. Then the user may generate a map of non linear pixels and compute the detector’s gain by using the additional recipe \texttt{sinfo_rec_lingain}. A set of linearity raw flat field frames may be put in the ASCII file lingain.sof. As indicated below, despite those frame are the same input as in the \texttt{sinfo_rec_detlin} recipe, the tags should indicate also if the lamp used to illuminate the frame is “off” or “on” (the Halogen lamp status can be determined by the value of the FITS keyword INS1.LAMP5.ST). The user must provide at least 4*(order+1) input frames: by default at least 16 frames.

\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:09:50.882.fits LINEARITY_LAMP_OFF}
\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:07.455.fits LINEARITY_LAMP_ON}
\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:23.047.fits LINEARITY_LAMP_ON}
\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:10:38.240.fits LINEARITY_LAMP_OFF}
\texttt{...}
\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:11:59.183.fits LINEARITY_LAMP_ON}
\texttt{/path_raw/LINEARITY/K/SINFO.2005-02-26T20:12:27.247.fits LINEARITY_LAMP_OFF}

The user can generate a non linearity bad pixel map (PRO.CATG=BP_MAP_NL) with the command:
\texttt{esorex sinfo_rec_lingain lingain.sof}

This command will generate several files: two tables, one (sinfo_rec_lingain.linearity_table.fits) including the non linearity information (PRO.CATG=DET_LIN_INFO), the other (sinfo_rec_lingain.gain_table.fits) including the detector’s gain information (PRO.CATG=GAIN_INFO), a 3D data product (sinfo_rec_lingain.coeffs_cube.fits) with the information on the pix-to-pix polynomial coefficients used to evaluate the detector’s linearity (PRO.CATG=COEFFS_CUBE), and (sinfo_rec_lingain.bpm.fits) a non linear pixel map (PRO.CATG=BP_NL_MAP).

This recipe is being used for quality control and being tested for the reduction chain. We recommend the user not to use it in the usual reduction chain with this release.

\texttt{rm -rf out*fits *.paf *.log}

4. 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 \texttt{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:

\texttt{/path_raw/SINFO.2005-03-14T11:28:19.007.fits FIBRE_NS}
\texttt{/path_raw/SINFO.2005-03-14T11:28:43.781.fits FIBRE_NS}
\texttt{/path_raw/SINFO.2005-03-14T11:28:59.723.fits FIBRE_NS}
\texttt{/path_raw/SINFO.2005-03-14T11:29:18.496.fits FIBRE_NS}
To successfully run the distortion recipe are needed: several (usually 75) FIBRE_NS frames, needed to cover all slitlets and generate an uniformly illuminated synthetic fibre flat; two FLAT_NS frames, one with calibration flat lamp switched on, one with the lamp switched off; two WAVE_NS frames, one with calibration flat lamp switched on, one with the lamp switched off; and the two static frames, the reference line list of the appropriate band, with tag REF_LINE_ARC, and the static parameters table with tag DRS_SETUP_WAVE as indicated above.

Where with /path_cdb we have indicated the full path to the directory containing calibration data. REF_LINE_ARC is the file TAG (PRO.CATG) assigned to the frame table containing the reference line list to be used to perform the wavelength calibration and DRS_SETUP_WAVE is an additional reference table storing instrument setting dependent parameters, which are expected to be stable, used during the wavelength calibration (see also Sec. 8). The command

```
esorex sinfo_rec_distortion distortion_sof
```

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

```
mv out_distortion.fits /path_pro/DISTORTION_K.fits
mv out_distances.fits /path_pro/SLITLETS_DISTANCE_K.fits
rm -rf out*fits *.paf *.log
```

5. 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.2005-02-28T16:28:05.846.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_pro/BP_MAP_NL_K.fits BP_MAP_NL
```

The frame REF_BP_MAP is an image indicating reference positions of known detector bad pixels (see also Sec 8).

The command:
esorex sinfo_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

6. 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.fits 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.fits DISTORTION
/path_cdb/drs_setup_wave.fits DRS_SETUP_WAVE
/path_cdb/SLIT_POS_K_100.fits SLIT_POS

The command:

esorex sinfo_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.fits”.

mv out_wavemap_ima.fits /path_pro/MASTER_WAVE_MAP_K_100.fits
mv out_slitpos.fits /path_pro/SLIT_POS_K_100.fits
rm -rf out*fits *.paf *.log

7. 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.fits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K_100.fits SLIT_POS
/path_pro/DISTORTION_K.fits DISTORTION
The command:

```
esorex sinfo_rec_jitter sci_sof
```

generates several frames, including a reconstructed cube of the PSF standard, stored in the file “out_cube_obj##.fits” (##=00,01,..NN), its median image, stored in the file “out_objnod_med_cube.fits”, the associated bad pixel map, stored in the file “out_objnod_bpmap.fits”, a table with Strehl values as a function of wavelength, stored in the file “out_ao_performance.fits”, a table with encircled energy values, stored in the file “out_encircled_energy.fits”. A table with the extracted average object spectrum, stored in the file “out_std_star_spectrum.fits”. We remind the user to read section 11.1.26 for important information on how to obtain the best quality of sky subtracted object spectra.

8. 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/STD/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.fits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K.fits SLIT_POS
/path_pro/DISTORTION_K.fits DISTORTION
```

The command:

```
esorex sinfo_rec_jitter stdstar_sof
```

generates several frames, including a reconstructed cube of the telluric standard, stored in the file “out_cube_obj##.fits” (##=00,01,..NN), its median image, stored in the file “out_objnod_med_cube.fits”, the associated bad pixel map, stored in the file “out_objnod_bpmap.fits”, a table with Strehl values as a function of wavelength, stored in the file “out_ao_performance.fits”, a table with encircled energy values, stored in the file “out_encircled_energy.fits”. A table with the extracted average object spectrum, stored in the file “out_std_star_spectrum.fits”.

9. 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/PSF/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.fits SLITLETS_DISTANCE
/path_pro/SLIT_POS_K_100.fits SLIT_POS
/path_pro/DISTORTION_K_100.fits DISTORTION
```

The command:
esorex sinfo_rec_jitter psf_sof

generates several frames, including a reconstructed cube of each target observation, stored in the file “out_cube_obj##.fits” (##=00,01,...NN), (##=00-01-NN), a mosaic cube coadding each single cube observation, stored in the file “out_objnod.fits” its median image, stored in the file “out_objnod_med_cube.fits”, the associated bad pixel map, stored in the file “out_objnod_bpmap.fits”, and for point-like target observations (for objects whose PSF can well approximated by a 2D Gaussian), a table with Strehl values as a function of wavelength, stored in the file “out_ao_performance.fits”, a table with encircled energy values, stored in the file “out_encircled_energy.fits”. A table with the extracted average object spectrum, stored in the file “out_std_star_spectrum.fits”.

6 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:

A user reported once an installation problem due to the fact FFTW and WCSLIB libraries are not present in the package. Those libraries are actually not required by the SINFONI pipeline, and as currently releases adopted by the pipelines ESO delivers to public (3.1.2 for FFTW and 4.4.4 for WCSLIB) and not very recent and have installation portability problems on some platforms, the SINFONI pipeline maintainer decided not to include them. But if any user would run into the mentioned problem, a possible solution is to include those library versions in the kit and start again a clean installation (after removing the previous ones).

Some user has reported that occasionally recipes may fail with error message “Error code -1”, that is not very clear. That message is conventional, and the last message of the error stack. We invide the user to check the full error stack: usually the real source of the problem is shown by some other error messages on the error stack.

Bad lines present on long exposure frames. This problem, visible for example on dark frames, is introduced by the data processing hardcoded at detector level. The four pixels surrounding the detector are not illuminated and are used by the hardcoded processing to estimate the “bias” level for each frame. The mean of each line is then subtracted from the frame prior to write the data onto the instrument workstation. When one of this hot pixel is located among the non illuminated “edge” pixels, the mean value of the corresponding line is then over-estimated, which creates some dark stripes in the raw frames when it is subtracted. This defect can be corrected using the utility sinfo_utl_ima_line_corr, or setting to “TRUE” the relevant recipes common parameter $lc_sw$.

sinfo_rec_detlin LINEARITY, LAMP data taken in the period 24/11/2006-28/02/2007 have been taken with increased lamp intensities. This means that in particular frames taken with high values of DET.DIT (for example DET.DIT=45) have several pixels in the non linear detector's range. This affects particularly the J band. As a consequence if the user uses recipe parameter defaults may obtain non linear pixel maps (ESO.PRO.CATG=BP_MAP_NL) collecting significantly more pixels than usual, and this would propagate in the MASTER_BP_MAP. Those pixels consequently would not be taken in consideration in the calibration, despite the level of the signal in the arc lamp frames is in the linear regime, this possibly leading to a lower quality WAVE_MAP and wavelength calibration solution. To solve this problem the user may:

• Not use raw frames at higher DET.DIT to build the non linear bad pixel map.
• Use all the raw frames and increase the parameter $bp_lin-thresh_sigma_fct$ from the default 10 to 15 or more.

sinfo_rec_distortion Occasionally this recipe may fail indicating that too less slitlets have been found. The user may try to decrease the parameter $bp_dist.min_cut$, or increase the parameter $smooth_rad$ or the parameter $arcs_min_arclen_factor$.
sinfo_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 \texttt{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. Another parameter to eventually change to improve robustness is \texttt{wcal-pixel_dist}. It has a default value of 12. One may try to increase it a bit, for example to 15. The same is true for the parameters having aliases \texttt{wcal-min_diff}, \texttt{wcal-na_coeff}, \texttt{wcal-nb_coeff}, \texttt{wcal-pix_tol}, \texttt{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.


<table>
<thead>
<tr>
<th>band</th>
<th>wcal-min_diff</th>
<th>wcal-na_coeff</th>
<th>wcal-nb_coeff</th>
<th>wcal-pix_tol</th>
<th>wcal-y_box</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>any</td>
<td>1.0</td>
<td>4</td>
<td>2</td>
<td>3.0</td>
<td>5.0</td>
<td>default setting</td>
</tr>
<tr>
<td>H</td>
<td>10.0</td>
<td>3</td>
<td>2</td>
<td>7.0</td>
<td>2.0</td>
<td>old setting</td>
</tr>
<tr>
<td>H+K</td>
<td>1.0</td>
<td>4</td>
<td>2</td>
<td>5.5</td>
<td>5.0</td>
<td>old setting</td>
</tr>
<tr>
<td>K</td>
<td>1.0</td>
<td>4</td>
<td>2</td>
<td>12.0</td>
<td>5.0</td>
<td>old setting</td>
</tr>
<tr>
<td>J</td>
<td>1.0</td>
<td>4</td>
<td>2</td>
<td>3.0</td>
<td>5.0</td>
<td>old setting</td>
</tr>
</tbody>
</table>

Another possible explanation of the failure is the usage of an improper bad pixel map. We suggest to use the \texttt{REF_BP_MAP}, provided with the present kit release, as input of the \texttt{sinfo_rec_mflat} recipe which determines the \texttt{MASTER_BP_MAP}, which is input of the \texttt{sinfo_ref_wavecal} recipe.

sinfo_rec_jitter  Sky subtraction quality. Despite we provide an improved routine to correct for possible sky line residuals, we suggest the user to use it only when (at least) a sky frame is available. Otherwise (if the user uses such correction in absence of a sky frame, case in which the next in time object frame is used as “dummy” sky), there is the risk that object lines cancel out after subtraction. The user should also choose proper values of the critical parameters \texttt{skycor-sky_bkg_filter_width}, \texttt{skycor-line_hw}, \texttt{skycor-rot_cor}, \texttt{skycor-fit_obj_noise}.

This release allows also to indicate the area where is supposed to be imaged the object, through the parameters \texttt{skycor-llx}, \texttt{skycor-lly}, \texttt{skycor-urx}, \texttt{skycor-ury}, which should allow a more accurate determination of the sky lines residuals and the corresponding spectrum shift with respect to the one of the object. The thermal background emission subtraction can be eventually switched off with the parameter \texttt{skycor-sub_thr_bkg_from_obj}. If the user has estimated the sky-object spectral shift independently, or would like to overwrite the value computed by the pipeline, can set this quantity (in pixel units) with the parameter \texttt{skycor-pshift}.

A check of the results provided by the product table(s) with PRO.CATG QC\_SPECTRA is always recommended to compare the extracted spectra before and after the correction.

sinfo_rec_jitter  WCS coordinates in the coadded cube may be inaccurate. We invite the user to check the accuracy of the reference catalog used, it may be that the VLT-SINFONI combo is more accurate; eventually correct object coordinate to take into account of its possible proper motion.

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

sinfo_rec_jitter  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. In those cases we suggest the user to use \texttt{sinfo_utl_cube_combine} and provide proper offsets. See also section 10.16.
sinfo_rec_jitter This recipe may fail to perform a 2D Gaussian fit to the observed target. In this case the std_star_spectra and std_star_spectrum pipeline products will not be generated. The user may try to reduce the value of the parameter std_star_fwhm_factor to get also those products.

sinfo_rec_jitter Has performance limits. The fastest execution may be obtained setting product-density to 0.

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

\[ \text{RAM} = [150 + 4 \times (2 \times sX \times sY \times sZ + 64 \times 64 \times sZ \times N_{frames}) \times 10^{-6}] \text{ MB}, \]

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:

\[ sX = 2 \times \text{floor}(offx_{\text{max}} - offx_{\text{min}} + 0.5) + 64 \quad sY = 2 \times \text{floor}(offy_{\text{max}} - offy_{\text{min}} + 0.5) + 64 \]

The \( \text{floor}(x) \) 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.

EsoRex may report at the end of a recipe execution the following wrong information:

\[ \text{[ INFO ] esorex: (no ESO PRO CATG keyword found...)} \]

The user should ignore it, the SINFONI pipeline products have the ESO.PRO.CATG keyword properly set.
7 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. Fibre flats are taken with the calibration fibre moved along the y-axis perpendicular to the image slices. Due to a non uniform illumination of a small number of slitlets from the fibre link, several raw frames (usually 75) are necessary to build a synthetic all-slitlets fibre fed flat needed 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 executed 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 5.2.4, page 30). 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).

7.1 Data features

In figure 7.1.1 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 7.1.2 we have the central plane of a coadded cube resulting from several target acquisitions observed at different telescope positions.

In figure 7.2.1 we have several examples of raw calibration and science frames.
7.2 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 7.2.1 shows the typical appearance of those frames.

- **Dark current:**
  
  DO category: DARK
  
  Processed by: sinfo_rec_mdark

  Classification keywords:  
  DPR CATG = CALIB  
  DPR TYPE = DARK  
  DPR TECH = IMAGE

  Association keywords:  
  DET DIT

  Fig:  
  7.2.1 (a)

- **Flat Field:**
  
  DO category: FLAT_LAMP
  
  Processed by: sinfo_rec_mflat

Figure 7.1.1: A possible target for observation (a) and a typical raw science frame layout (b).
Figure 7.1.2: A coadded cube resulting from several “mosaiced” cubes.

- **Detector linearity Flat Field:**
  - **DO category:** LINEARITY_LAMP
  - **Processed by:** sinfo_rec_detlin

- **Detector-monitoring linearity Flat Field:**
  - **DO category:** LINEARITY_LAMP_ON/OFF
  - **Processed by:** sinfo_rec_lingain

- **Flat field frames to compute distortions:**
  - **DO category:** FLAT_NS
  - **Processed by:** sinfo_rec_distortion
Figure 7.2.1: (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 to compute distortions:**
  
  **DO category:** WAVE_NS  
  **Processed by:** sinfo_rec_distortion  

  Classification keywords:  
  DPR CATG = CALIB  
  DPR TYPE = DISTORTION,WAVE,LAMP  
  DPR TECH = IFU  

  Association keywords:  
  INS SETUP ID  

  Fig:  
  7.2.1 (d)

- **Arc frames:**
  
  **DO category:** WAVE_LAMP  
  **Processed by:** sinfo_rec_wavecal  

  Classification keywords:  
  DPR CATG = CALIB  

  Association keywords:  
  INS SETUP ID  

  Fig:  
  7.2.1 (d)
• **STD PSF star frames:**

**DO category:** PSF_CALIBRATOR  
**Processed by:** sinfo_rec_psf

<table>
<thead>
<tr>
<th>Classification keywords</th>
<th>Association keywords</th>
<th>Fig:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR CATG = CALIB</td>
<td>INS SETUP ID</td>
<td>7.2.1(e)</td>
</tr>
<tr>
<td>DPR TYPE = PSF-CALIBRATOR</td>
<td>INS OPTI1 NAME</td>
<td></td>
</tr>
<tr>
<td>DPR TECH = IFU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**DO category:** SKY_PSF_CALIBRATOR  
**Processed by:** sinfo_rec_jitter

<table>
<thead>
<tr>
<th>Classification keywords</th>
<th>Association keywords</th>
<th>Fig:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR CATG = CALIB</td>
<td>INS SETUP ID</td>
<td>7.2.1(f)</td>
</tr>
<tr>
<td>DPR TYPE = SKY,PSF-CALIBRATOR</td>
<td>INS OPTI1 NAME</td>
<td></td>
</tr>
<tr>
<td>DPR TECH = IFU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

• **STD star frames:**

**DO category:** STD  
**Processed by:** sinfo_rec_jitter

<table>
<thead>
<tr>
<th>Classification keywords</th>
<th>Association keywords</th>
<th>Fig:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR CATG = CALIB</td>
<td>INS SETUP ID</td>
<td>7.2.1(e)</td>
</tr>
<tr>
<td>DPR TYPE = STD</td>
<td>INS OPTI1 NAME</td>
<td></td>
</tr>
<tr>
<td>DPR TECH = IFU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

**DO category:** SKY_STD  
**Processed by:** sinfo_rec_jitter

<table>
<thead>
<tr>
<th>Classification keywords</th>
<th>Association keywords</th>
<th>Fig:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR CATG = CALIB</td>
<td>INS SETUP ID</td>
<td>7.2.1(f)</td>
</tr>
<tr>
<td>DPR TYPE = SKY,STD</td>
<td>INS OPTI1 NAME</td>
<td></td>
</tr>
<tr>
<td>DPR TECH = IFU</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• **Science frames:**

  **DO category:** OBJECT_NODDING  
  **Processed by:** sinfo_rec_jitter

  Classification keywords:  
  DPR CATG = SCIENCE  
  DPR TYPE = OBJECT  
  DPR TECH = IFU,NODDING

  Association keywords:  
  INS SETUP ID  
  INS OPT1 NAME

  **Fig:**  
  7.2.1 (e)

• **Sky frames taken with science frames:**

  **DO category:** SKY_NODDING  
  **Processed by:** sinfo_rec_jitter

  Classification keywords:  
  DPR CATG = SCIENCE  
  DPR TYPE = SKY  
  DPR TECH = IFU,NODDING

  Association keywords:  
  INS SETUP ID  
  INS OPT1 NAME

  **Fig:**  
  7.2.1 (f)

---

Figure 7.2.2: On-lamp raw data example for (a) PUPIL,IMAGE and (b) PRE,IMAGE data.

• **PUPIL,IMAGE frames:**

  **DO category:** PUPIL_IMAGE  
  **Processed by:** sinfo_rec_pupil

  Classification keywords:  
  DPR CATG = CALIB  
  DPR TYPE = PUPIL,IMAGE  
  DPR TECH = IFU,NODDING

  Association keywords:  
  INS SETUP ID  
  INS OPT1 NAME

  **Fig:**  
  7.2.2 (a)
• **PRE,IMAGE frames:**

  **DO category:** PRE_IMAGE
  **Processed by:** sinfo_rec_jitter

<table>
<thead>
<tr>
<th>Classification keywords:</th>
<th>Association keywords:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DPR CATG = SCIENCE</td>
<td>INS SETUP ID</td>
</tr>
<tr>
<td>DPR TYPE = OBJECT</td>
<td>INS OPTI1 NAME</td>
</tr>
<tr>
<td>DPR TECH = PRE,IMAGE</td>
<td></td>
</tr>
</tbody>
</table>

  **Fig:** 7.2.2 (b)
8 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 5.2.4, page 30).

8.1 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 recipe sinfo_rec_distortion and sinfo_rec_wavecal.

8.2 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 sinfo_rec_distortion and sinfo_rec_wavecal.

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

This table set the values for data reduction parameters which in section 11 are respectively called begin_wave, guess_disp1, guess_disp2, half_width, fwhm, min_AMP, lo_pos, hi_pos. To change the values of such data reduction parameter, the user must edit this table.

8.3 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 sinfo_rec_mflat.

8.4 Reference table with flux of STD star for efficiency computation

To compute the efficiency, the user must provide a FITS table containing at least the two columns “WAVELENGTH” (in Angstrom units) and “FLUX” (in units of ergs cm$^{-2}$ s$^{-1}$ Å$^{-1}$) describing the analyzed standard star flux calibrated spectrum. This table has to be tagged as FLUX_STD_CATALOG.

8.5 Reference tables with polynomial coefficients for correction due atmospheric refraction

The coefficients of the 2D polynomial fit for the correction due atmospheric refraction (see section 11.2.4 for more details), are saved into calibration tables (for example ATM_REF_CORR_025_H.poly.fits, for the 25mas H band setting and similar tables for the other settings. In the 250mas setting this correction is not offered as this spatial setting is more complex to calibrate and less affected by atmospheric refraction).
9 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.

9.1 Data reduction overview

The principle of integral field spectroscopy is described by figure 9.1.1. 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 sinfo_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. From release 1.9.8 the user may also perform an OH lines based wavelength calibration on sky frames (see 10.7.1). 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 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.

9.2 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 sinfo_rec_distortion and sinfo_rec_wavecal to specify parameters peculiar of the wavelength calibration algorithm.

• 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 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.

9.3 Reduction cascade

The SINFONI data reduction follows the following sequence. A short description of the available recipes is given in section 5.1. In parenthesis we provide the value of the DO category corresponding to each frame.

• Run sinfo_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 `sinfo_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 `sinfo_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 `sinfo_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 `sinfo_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 data reduction kit.

• Run `sinfo_rec_jitter` on PSF standards and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS and a WAVE_MAP to reduce the PSF standard and get information on the instrument’s Strehl.

• Run `sinfo_rec_jitter` on a reference telluric standard (STD) and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS and a WAVE_MAP to reduce the telluric standards and get information on the instrument’s response.

• Run `sinfo_rec_jitter` on your scientific data (OBJECT_NODDING) and a MASTER_BP_MAP, a MASTER_FLAT_LAMP, a DISTORTION, a SLITLET_DISTANCES, a SLIT_POS and a WAVE_MAP to reduce science data.

The main data products involved in the data reduction cascade are indicated in the SINFONI association map shown in Figure 9.3.1. 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 10.
Figure 9.3.1: SINFONI Association Map
10 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.fits are existing FITS files (e.g. /data1/sinfoni/com2/SINFO.2004-08-16T02:54:04.353.fits and /cal/sinfo/ifu/cal/DISTORTION_K.fits).

We also provide a list of the pipeline products for each recipe, indicating their default recipe name (eventually set by esorex to a given standard), the value of the FITS keyword HIERARCH ESO PRO CATG (in short PRO.CATG) and a short description. To improve performance and allow the user to generate different level of information, some recipes may have a variable number of products according to the value of the global parameter product-density. If this value is 0 the minimum number of products needed for the data reduction chain is generated. If this value is 1, the recipe sinfo_rec_jitter generates also some wavelength calibrated cubes needed to be used for eventual sky residual correction. If this value is 2 (default), additional products used for quality control are generated. If this value is 3, all possible products are generated, for debugging purposes. Where relevant we indicate the minimum value of product-density which generates the corresponding product in a column labeled as “pd”.

The relevant keywords are PRO.CATG, used to classify each frame, and to associate to each raw frame the proper calibration frame:

<table>
<thead>
<tr>
<th>Association keyword</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>HIERARCH ESO INS SETUP ID</td>
<td>band</td>
</tr>
<tr>
<td>HIERARCH ESO INS OPTI1 NAME</td>
<td>Pixel scale</td>
</tr>
<tr>
<td>HIERARCH ESO DET DIT</td>
<td>Integration time</td>
</tr>
<tr>
<td>HIERARCH ESO INS1 LAMP5 ST</td>
<td>Indicates the Halogen lamp status (“on”/”off”)</td>
</tr>
</tbody>
</table>

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 www.eso.org/qc.

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

The recipes share the following parameters:

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfonyi.general.overwrite_parameters</td>
<td>gen-overpar</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfonyi.general.lc_sw</td>
<td>lc_sw</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfonyi.general.lc_kappa</td>
<td>lc_kappa</td>
<td>18</td>
</tr>
<tr>
<td>sinfonyi.general.lc_filt_rad</td>
<td>lc_filt_rad</td>
<td>3</td>
</tr>
</tbody>
</table>

10.1 sinfo_rec_detlin

The recipe sinfo_rec_detlin computes the detector responsivity as a function of the pixel intensity and determines when it becomes non linear.
10.1.1 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

10.1.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>lin_det_info.fits</td>
<td>LIN_DET_INFO</td>
<td>Table with coefficients of non linear fit to median of each flat image image</td>
<td>2</td>
</tr>
<tr>
<td>gain_info.fits</td>
<td>GAIN_INFO</td>
<td>Table with detector’s gain values</td>
<td>2</td>
</tr>
<tr>
<td>out_bplin_coeffsCube.fits</td>
<td>BP_COEFF</td>
<td>cube. Each cube’s plane logs coefficients of non linear fit to pixel’s intensity used to evaluate non linearity</td>
<td>0</td>
</tr>
<tr>
<td>out_bp_lin.fits</td>
<td>BP_MAP_NL</td>
<td>Non linear (bad) pixels map image</td>
<td>0</td>
</tr>
</tbody>
</table>

The LIN_DET_INFO table for each on-off linearity flat pair monitors the following values:

- med: median on the whole frame pair difference
- avg: average on the whole frame pair difference
- med_dit: median divided detector’s DIT
- avg_dit: average divided detector’s DIT
- dit: detector’s DIT
- adl: values of the product med_dit*dit

The GAIN_INFO table for each on-off linearity flat pair monitor the following values:

- adu: intensity in adu
- gain: corresponding gain

10.1.3 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 11.1.9. The computed coefficients are QC.BP-MAP.LIN\(i\).MED \((i=0,1,2)\) (see Figure 10.1.1).

A different method is applied to determine the same quantity as described in 11.1.9. The computed coefficients are QC.BP-MAP.LIN\(i\).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 11.1.11 and is given by the value of QC.GAIN. It is expressed in e\(^{-}\)/ADU units and measures the intensity recorded by the detector for each incident photon. Its inverse is usually called conversion factor.

![Figure 10.1.1: Detector linearity quality control](image)

Figure 10.1.1: Detector linearity quality control: (a) the non-linearity function (non-linearity detector response) and a linear relation as a reference; (b) trending plot of detector linearity first order coefficient; (c) trending plot of detector linearity second order coefficient.

### 10.1.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoni.bp_lin.order</td>
<td>bp_lin-order</td>
<td>2</td>
</tr>
<tr>
<td>sinfoni.bp_lin.thresh_sigma_factor</td>
<td>bp_lin-thresh_sigma_fct</td>
<td>10.0</td>
</tr>
<tr>
<td>sinfoni.bp_lin.nlin_threshold</td>
<td>bp_lin-nlin_threshold</td>
<td>0.5</td>
</tr>
<tr>
<td>sinfoni.bp_lin.low_rejection</td>
<td>bp_lin-lo_rej</td>
<td>10.0</td>
</tr>
<tr>
<td>sinfoni.bp_lin.high_rejection</td>
<td>bp_lin-hi_rej</td>
<td>10.0</td>
</tr>
</tbody>
</table>
10.2 sinfo_rec_lingain

The recipe sinfo_rec_lingain reduces linearity and gain of the detector in the infrared domain. The linearity reduction part computes the detector responsivity as a function of the pixel intensity and determines when it becomes non linear. The gain reduction part computes a gain value for each flux level. Both reduction steps are merged in this recipe as both accept the same input interface.

10.2.1 Input

This recipe expects a set of pairs of LINEARITY_LAMP_ON frames and a set of pairs of LINEARITY_LAMP_OFF frames with different exposure time values.

If no modification is done to default behaviour, there must be two frames tagged LINEARITY_LAMP_ON and two frames tagged LINEARITY_LAMP_OFF for each distinct EXPTIME value found among input frames.

The number of distinct EXPTIME values among input frames must be at least order + 1. This means by default this recipe should receive at least 16 frames.

```
/path_raw/SINFO.2005-09-02T16:44:07.314.fits LINEARITY_LAMP_OFF
/path_raw/SINFO.2005-09-02T16:44:27.248.fits LINEARITY_LAMP_ON
/path_raw/SINFO.2005-09-02T16:44:45.351.fits LINEARITY_LAMP_ON
/path_raw/SINFO.2005-09-02T16:45:57.794.fits LINEARITY_LAMP_OFF
...
/path_raw/SINFO.2005-09-02T17:05:46.234.fits LINEARITY_LAMP_OFF
/path_raw/SINFO.2005-09-02T17:06:37.943.fits LINEARITY_LAMP_ON
/path_raw/SINFO.2005-09-02T17:07:20.400.fits LINEARITY_LAMP_ON
/path_raw/SINFO.2005-09-02T17:08:54.097.fits LINEARITY_LAMP_OFF
```

10.2.2 Output

The following table lists all possible products of this recipe. Please note that not all of them are always created. Creation of some of these products depends on the value(s) of some input parameter(s).

If no modification is done to default behaviour, this recipe produces only the first four products in the list.

Further details about conditions for product creation can be found along this section.

For each input setting, the following outputs are created:

<table>
<thead>
<tr>
<th>Default recipe filename</th>
<th>PRO.CATG</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_rec_lingain_linearity_table.fits</td>
<td>DET_LIN_INFO</td>
<td>Linearity Table</td>
</tr>
<tr>
<td>sinfo_rec_lingain_gain_table.fits</td>
<td>GAIN_INFO</td>
<td>Gain Table</td>
</tr>
<tr>
<td>sinfo_rec_lingain_coeffs_cube.fits</td>
<td>COEFFS_CUBE</td>
<td>Coefficients cube</td>
</tr>
<tr>
<td>sinfo_rec_lingain.bpm.fits</td>
<td>BP_MAP_NL</td>
<td>Bad pixel map</td>
</tr>
<tr>
<td>sinfo_rec_lingain_diff_flat_i.fits</td>
<td>DIFF_FLAT</td>
<td>Difference images</td>
</tr>
<tr>
<td>sinfo_rec_lingain_autocorr_i.fits</td>
<td>AUTOCORR</td>
<td>Autocorrelation images</td>
</tr>
</tbody>
</table>
Both Linearity and Gain Tables are created in all possible execution cases.

**Content of the Linearity Table**

For each distinct EXPTIME value, this table logs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIT</td>
<td>Detector Integration Time in s</td>
</tr>
<tr>
<td>MED</td>
<td>Median intensity of [ (on_{1} - off_{1}) + (on_{2} - off_{2}) ]/2</td>
</tr>
<tr>
<td>MEAN</td>
<td>Mean intensity of [ (on_{1} - off_{1}) + (on_{2} - off_{2}) ]/2</td>
</tr>
<tr>
<td>MED_DIT</td>
<td>Ratio MED / DIT</td>
</tr>
<tr>
<td>MEAN_DIT</td>
<td>Ratio MEAN / DIT</td>
</tr>
<tr>
<td>ADL</td>
<td>Column DIT multiplied by mean value of column MED_DIT</td>
</tr>
</tbody>
</table>

**Content of the Gain Table**

For each distinct EXPTIME value, this table logs:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIT</td>
<td>Detector Integration Time in s</td>
</tr>
<tr>
<td>MEAN_ON1</td>
<td>Mean intensity of (&lt;on_{1}&gt;))</td>
</tr>
<tr>
<td>MEAN_ON2</td>
<td>Mean intensity of (&lt;on_{2}&gt;))</td>
</tr>
<tr>
<td>MEAN_OFF1</td>
<td>Mean intensity of (&lt;off_{1}&gt;))</td>
</tr>
<tr>
<td>MEAN_OFF2</td>
<td>Mean intensity of (&lt;off_{2}&gt;))</td>
</tr>
<tr>
<td>SIG_ON_DIF</td>
<td>Standard deviation in [on_{1} - on_{2}] (\sigma_{ondif})</td>
</tr>
<tr>
<td>SIG_OFF_DIF</td>
<td>Standard deviation in [off_{1} - off_{2}] (\sigma_{offdif})</td>
</tr>
<tr>
<td>GAIN</td>
<td>(\frac{(&lt;on_{1})+&lt;on_{2})-(&lt;off_{1})+&lt;off_{2})}{(\sigma_{ondif}^{2} - \sigma_{offdif}^{2})})</td>
</tr>
<tr>
<td>AUTOCORR</td>
<td>Autocorrelation factor</td>
</tr>
<tr>
<td>GAIN_CORR</td>
<td>GAIN / AUTOCORR</td>
</tr>
<tr>
<td>ADU</td>
<td>((&lt;on_{1})+&lt;on_{2}) - (&lt;off_{1})+&lt;off_{2})/2)</td>
</tr>
<tr>
<td>X_FIT</td>
<td>X-fit coordinate</td>
</tr>
<tr>
<td>X_FIT_CORR</td>
<td>X-fit coordinate corrected by the autocorrelation function</td>
</tr>
<tr>
<td>Y_FIT</td>
<td>Y-fit coordinate</td>
</tr>
</tbody>
</table>

The Coefficients Cube is created only if **pix2pix** is set to TRUE (default behaviour for this recipe). It contains as many planes as **order** + 1, each pixel in plane \(i\) indicating the value of the coefficient or order \(i\) of the fitting computed for that position.

The Bad Pixel map depends on information contained in the Coefficients Cube, so it is created together with it. Its content may vary depending on **bpmbin**.

If **bpmbin** is set to FALSE (default behaviour for this recipe), each pixel has an integer value, power of 2 or combination of powers of 2. Power \(i\) of 2 means that pixel deviates from the median polynomial coefficient of order \(i\).
If `bpmbin` is set to TRUE, then the Bad Pixel Map will be a binary one, containing only 0 for correct pixels and 1 for all possible bad ones.

The Difference and Autocorrelation images are only created if `intermediate` is set to TRUE (default behaviour for this recipe is FALSE). They represent intermediate steps of the Autocorrelation factor computation.

### 10.2.3 Quality control

#### Lamp flux

QC LAMP FLUX in ADU/s.

#### Detector gain

The median of the computed gain values and relative stdev QC GAIN MED, QC GAIN STD, in $e^{-}$/ADU and its inverse QC CONAD in ADU/$e^{-}$, and the autocorrelation factor QC AUTCORFA.

The median calculated via polynomial fit is also stored as QC GAIN FIT.

#### Readnoise

QC READNOISE in $e^{-}$.

And from the linearity part of the recipe:

#### Lamp flux RMS

It is monitored the RMS of the lamp flux (QC LAMP STD).

#### Non linear coefficients

Are monitored the polynomial fit coefficients (QC LIN COEFi, i=0-2) and the corresponding errors (QC LIN COEFi ERR).

#### Effective non linearity

An additional QC parameter is QC LIN EFF, the effective non-linearity correction at a reference user defined level `ref_level`: $Q'(ref_level)$.

#### Number of non polynomial pixels

It is monitored the number of pixels deviating from the polynomial fit (QC NUM BPM).

### 10.2.4 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Alias</th>
<th>Default, Other</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sinfo.sinfo_rec_lingain.method</code></td>
<td>method</td>
<td><code>PTC, MED</code></td>
<td>Method applied for Gain Computation</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.order</code></td>
<td>order</td>
<td>3</td>
<td>Order of the polynomial fitting applied in the Linearity part of the recipe</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.kappa</code></td>
<td>kappa</td>
<td>9</td>
<td>Multiplier used for Kappa-sigma clipping</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.niter</code></td>
<td>niter</td>
<td>25</td>
<td>Number of iterations kappa-sigma clipping is applied</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.llx</code></td>
<td>llx</td>
<td>1</td>
<td>llx of the area to compute</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.lly</code></td>
<td>lly</td>
<td>1</td>
<td>lly of the area to compute</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.urx</code></td>
<td>urx</td>
<td><code>nx of 1. input</code></td>
<td>urx of the area to compute</td>
</tr>
<tr>
<td><code>sinfo.sinfo_rec_lingain.ury</code></td>
<td>ury</td>
<td><code>ny of 1. input</code></td>
<td>ury of the area to compute</td>
</tr>
</tbody>
</table>
sinfo.sinfo_rec_lingain.ref_level  ref_level  10000  Reference level on which linearity efficiency is computed
sinfo.sinfo_rec_lingain.intermediate  intermediate  FALSE, TRUE  Activates saving of intermediate products
sinfo.sinfo_rec_lingain.autocorr  autocorr  FALSE, TRUE  Activates use of Autocorrelation Factor
sinfo.sinfo_rec_lingain.m  m  26  Shift in the X-axis for Autocorrelation
sinfo.sinfo_rec_lingain.n  n  26  Shift in the Y-axis for Autocorrelation
sinfo.sinfo_rec_lingain.collapse  collapse  FALSE, TRUE  Activates collapsing of OFF frames
sinfo.sinfo_rec_lingain.rescale  rescale  FALSE, TRUE  Activates rescaling of images
sinfo.sinfo_rec_lingain.pix2pix  pix2pix  TRUE, FALSE  Activates pixel-to-pixel linearity computation
sinfo.sinfo_rec_lingain.bpmbin  bpmbin  TRUE, FALSE  Activates binary bad pixel map
sinfo.sinfo_rec_lingain.filter  filter  -1  Level beyond which frames are considered saturated and excluded.
sinfo.sinfo_rec_lingain.exts  exts  0  Extension on which recipe is computed

NOTES

Please note that if ref_level is not in the interval where Flux(DIT) is monotonely growing (that is, if it is a saturation value) the recipe will fail.

Please note that to compute on all extensions the user must set it to -1 (default is 0 - no extension case).

10.3  sinfo_rec_mdark

The recipe sinfo_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.

10.3.1  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

10.3.2  Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_hp_noise.fits</td>
<td>BP_MAP_HP</td>
<td>bad pixel map image, method=“Noise”</td>
</tr>
<tr>
<td>out_dark.fits</td>
<td>MASTER_DARK</td>
<td>master dark image</td>
</tr>
</tbody>
</table>

10.3.3  Quality control

Dark frames are processed to monitor the RON, Read Out Noise per DIT, the FPN, Fixed Pattern 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. Usually the dark template includes three consecutive frames. In this case the two RON values are obtained using the same detector region as QC.RON but QC.RON1 is computed on the frame2-frame1 subtracted frame, while QC.RON2 is computed on the frame3-frame2 subtracted frame. If the user provides N input frames there will N-1 QC.RONi parameters, obtained in a similar way by the N-1 frames obtained by subtracting consecutive frame pairs.

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

Figure 10.3.1: Dark frame quality control plot: **Upper left**: the mean (collapsed) column (red) and the column at X=2048/4 of the product frame. **Upper middle**: two rows at Y=2048/4 (red) and Y=2048*(3/4) (blue) of the product frame. **Upper right**: a parabolic function (blue) is fit to the central part (-3 to 3 counts) of the log of the raw2-raw1 difference frame (red). Histogram of the raw3-raw2 difference frame (green). **Lower left**: mean column (collapse), column x=500 of the product and of the first of the three raw input files. A smaller y-range is shown to show the difference between fixed-pattern (=pixel-to-pixel) noise and the statistical noise. **Lower middle**: the Y=500 row of the product minus the Y=500 row of a reference product. Y=600 row of the product minus the Y=600 row of a reference product. X=500 column of the master minus the X=500 column of a reference product. The reference product is renewed usually once a period or after an intervention. **Lower right**: histogram of the difference frame (second raw minus first raw) (red) and Gaussian fit between -3 and 3 counts (blue). Histogram of the raw3-raw2 difference frame (green).
**Fixed Pattern Noise**  The CPL function `cpl_flux_get_noise_window()` is used to compute the FPN. This function computes the noise in a frame with Monte Carlo approach. 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

### 10.3.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sinfoni.bp_noise.thresh_sigma_factor</code></td>
<td><code>bp_noise-thresh_sigma_factor</code></td>
<td>10.0</td>
</tr>
<tr>
<td><code>sinfoni.bp_noise.low_rejection</code></td>
<td><code>bp_noise-lo_rej</code></td>
<td>10.0</td>
</tr>
<tr>
<td><code>sinfoni.bp_noise.high_rejection</code></td>
<td><code>bp_noise-hi_rej</code></td>
<td>10.0</td>
</tr>
<tr>
<td><code>sinfoni.dark.low_rejection</code></td>
<td><code>dark-lo_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.dark.high_rejection</code></td>
<td><code>dark-hi_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_xmin</code></td>
<td><code>dark-qc_ron_xmin</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_xmax</code></td>
<td><code>dark-qc_ron_xmax</code></td>
<td>2048</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_ymin</code></td>
<td><code>dark-qc_ron_ymin</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_ymax</code></td>
<td><code>dark-qc_ron_ymax</code></td>
<td>2048</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_hsize</code></td>
<td><code>dark-qc_ron_hsize</code></td>
<td>4</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_ron_nsamp</code></td>
<td><code>dark-qc_ron_nsamp</code></td>
<td>100</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_xmin</code></td>
<td><code>dark-qc_fpn_xmin</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_xmax</code></td>
<td><code>dark-qc_fpn_xmax</code></td>
<td>2048</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_ymin</code></td>
<td><code>dark-qc_fpn_ymin</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_ymax</code></td>
<td><code>dark-qc_fpn_ymax</code></td>
<td>2048</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_hsize</code></td>
<td><code>dark-qc_fpn_hsize</code></td>
<td>2</td>
</tr>
<tr>
<td><code>sinfoni.dark.qc_fpn_nsamp</code></td>
<td><code>dark-qc_fpn_nsamp</code></td>
<td>1000</td>
</tr>
</tbody>
</table>

### 10.4 `sinfo_mdark_detmon`

The recipe `detmon_dark` computes the master dark, the Read Out Noise (RON), the Detector Signal Non Uniformity (DSNU).

The input files (at least one) must be tagged as DARK. Files with different DIT are processed separately.

### 10.4.1 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
```
Here RAW is an alias to be replaced by the instrument specific tag designed to reduce the corresponding data.

### 10.4.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_rec_mdark_detmon_master_dark.fits</td>
<td>MASTER_DARK</td>
<td>master dark frame</td>
</tr>
<tr>
<td>sinfo_rec_mdark_detmon_dsnu_map.fits</td>
<td>DSNU_MAP</td>
<td>DSNU image map</td>
</tr>
<tr>
<td>sinfo_rec_mdark_detmon_dsnu_table.fits</td>
<td>DSNU_TABLE</td>
<td>DSNU table</td>
</tr>
</tbody>
</table>

### 10.4.3 Quality control

QC DARK, the dark level in ADU.

QC DARK STDEV, the dark level standard deviation in ADU.

### 10.4.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo.sinfo_rec_mdark_detmon.dark.ron.method</td>
<td>ron.method</td>
<td>SIMPLE</td>
</tr>
<tr>
<td>sinfo.sinfo_rec_mdark_detmon.dark.dsnu.method</td>
<td>dsnu.method</td>
<td>STDEV</td>
</tr>
<tr>
<td>sinfo.sinfo_rec_mdark_detmon.dark.opt_nir</td>
<td>opt_nir</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfo.sinfo_rec_mdark_detmon.dark.exts</td>
<td>exts</td>
<td>0</td>
</tr>
</tbody>
</table>

### 10.5 sinfo_rec_mflat

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

#### 10.5.1 Input

/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_HP_H.fits BP_MAP_HP *
/path_file_cdb/BP_MAP_H_025.fits BP_MAP **
/path_file_cdb/SLIT_POS_H_025.fits SLIT_POS **

With * we have indicated a bad pixel map obtained with a list of dark frames. We suggest the user to use
darks taken with a low DIT. With ** we have listed an additional frames which are required only if `sinfoni.lamp_flats.interpol_index`==TRUE (usually not the case).

### 10.5.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_flat.fits</td>
<td>MASTER_FLAT_LAMP</td>
<td>master flat field image</td>
<td>0</td>
</tr>
<tr>
<td>out_bp_norm.fits</td>
<td>BP_MAP_NO</td>
<td>“Above threshold” bad pixel map image</td>
<td>2</td>
</tr>
<tr>
<td>out_bpmap_sum.fits</td>
<td>MASTER_BP_MAP</td>
<td>master bad pixel map image</td>
<td>0</td>
</tr>
</tbody>
</table>

The MASTER_BP_MAP results from the logic OR coaddition of the BP_MAP_NO image and any other bad pixel map.

### 10.5.3 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 pattern 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.BP-MAP.NBADPIX. The number of bad pixel in the master bad pixel map is given by the value of QC.MBP-MAP.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 Pattern 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 sinfo_rec_mflat.rc file. Two regions are monitored by parameters QC.LFLAT.FPN1 and QC.LFLAT.FPN2. During the observing period P75
Figure 10.5.1: Detector flat field quality control: (a) Trending plot of lamp average flux; (b) Trending plot of lamp fixed pattern noise in the first reference region.

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].

10.5.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoni.bp_norm.sigma_factor</td>
<td>bp_norm-s_factor</td>
<td>5.0</td>
</tr>
<tr>
<td>sinfoni.bp_norm.method_index</td>
<td>bp_norm-method_ind</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.bp_norm.factor</td>
<td>bp_norm-fct</td>
<td>10.0</td>
</tr>
<tr>
<td>sinfoni.bp_norm.iterations</td>
<td>bp_norm-it</td>
<td>8</td>
</tr>
<tr>
<td>sinfoni.bp_norm.low_rejection</td>
<td>bp_norm-lo_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.bp_norm.high_rejection</td>
<td>bp_norm-hi_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.bp_norm.llx</td>
<td>bp_norm-lxl</td>
<td>270</td>
</tr>
<tr>
<td>sinfoni.bp_norm.ury</td>
<td>bp_norm-ury</td>
<td>1200</td>
</tr>
<tr>
<td>sinfoni.bp_norm.threshold_index</td>
<td>bp_norm-thr_ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.bp_norm.mean_factor</td>
<td>bp_norm-mean_fct</td>
<td>100.0</td>
</tr>
<tr>
<td>sinfoni.bp_norm.min_cut</td>
<td>bp_norm-min_cut</td>
<td>0.0</td>
</tr>
<tr>
<td>sinfoni.bp_norm.max_cut</td>
<td>bp_norm-max_cut</td>
<td>5e+04</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.low_rejection</td>
<td>lamp_flats-lo_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.high_rejection</td>
<td>lamp_flats-hi_rej</td>
<td>0.1</td>
</tr>
</tbody>
</table>
10.6  \texttt{sinfo\_rec\_distortion}

The recipe \texttt{sinfo\_rec\_distortion} is used to compute the optical distortion. It also computes the relative distances of the slitlets from the first one.

10.6.1 Input

```
/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.2004-08-16T13:15:55.156.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.fits REF_LINE_ARC
```
With ** we have listed an additional frames which are required only if `sinfoni.lamp_flats.interpol_index`==TRUE (usually not the case).

To successfully run the distortion recipe are needed: several (usually 75) FIBRE_NS frames, needed to cover all slitlets and generate an uniformly illuminated synthetic fibre flat; two FLAT_NS frames, one with calibration flat lamp switched on, one with the lamp switched off; two WAVE_LAMP frames, one with calibration flat lamp switched on, one with the lamp switched off; and the two static frames, the reference line list of the appropriate band, with tag `REF_LINE_ARC`, and the static parameters table with tag `DRS_SETUP_WAVE` as indicated above.

### 10.6.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_flat.fits</td>
<td>MASTER_FLAT_LAMP</td>
<td>master flat field image</td>
<td>2</td>
</tr>
<tr>
<td>out_bp_dist.fits</td>
<td>BP_MAP_DI</td>
<td>“Above threshold” bad pixel map image</td>
<td>2</td>
</tr>
<tr>
<td>out_ns_stack_off.fits</td>
<td>FIBRE_NS_STACKED_OFF</td>
<td>fake off image from FIBRE,NS</td>
<td>2</td>
</tr>
<tr>
<td>out_ns_stack_on.fits</td>
<td>FIBRE_NS_STACKED_ON</td>
<td>fake on image from FIBRE,NS</td>
<td>2</td>
</tr>
<tr>
<td>out_wcal_stack.fits</td>
<td>WAVE_LAMP STACKED</td>
<td>fake on-off image</td>
<td>2</td>
</tr>
<tr>
<td>out_slitlets_pos_predist.fits</td>
<td>SLITLETS_POS_PREDIST</td>
<td>table with computed slitlets edge positions before distortion correction</td>
<td>2</td>
</tr>
<tr>
<td>out_distortion.fits</td>
<td>DISTORTION</td>
<td>table computed optical distortions</td>
<td>0</td>
</tr>
<tr>
<td>out_ns_stack_warp.fits</td>
<td>FIBRE_NS_STACKED_DIST</td>
<td>on-off image corrected for distortions</td>
<td>2</td>
</tr>
<tr>
<td>out_ns.fits</td>
<td>MASTER_SLIT</td>
<td>on-off image multiplied by BP map</td>
<td>2</td>
</tr>
<tr>
<td>out_distances.fits</td>
<td>SLITLETS_DISTANCE</td>
<td>table with computed slitlet distances</td>
<td>0</td>
</tr>
</tbody>
</table>

The `SLITLETS_POS_PREDIST` table for each slitlet measure the following values:

- **degx**: x degree
- **degy**: y degree
- **coeff**: polynomial distortion coefficient

The `DISTORTION` table stores the computed distortion coefficients:

- **pos1**: left slitlet edge x position
- **pos2**: right slitlet edge x position

The `SLITLET_DISTANCE` table stores the computed slitlets distances from the left edge of the first slitlet:

- **slitlet_distance**: computed slitlet distances from the left edge of the first slitlet
10.6.3 Quality control

The recipe sinfo_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 to monitor, 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.DIST, =0,30.

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

The recipe sinfo_rec_distortion determines the distortion in terms of polynomial coefficients

\[ C_0 + C_1 X + C_2 Y + C_3 XY + C_4 XX + C_5 YY + C_6 XXY + C_7 XYY + C_8 XXX + C_9 YYY \]

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

### Distortion coefficients
- The coefficients C0 to C9 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 its uncertainty by QC parameters QC.SL.DISTAVG, QC.SL.DISTRMS.

#### 10.6.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoni.lamp_flats.low_rejection</td>
<td>lamp_flats-lo_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.high_rejection</td>
<td>lamp_flats-hi_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.interpol_index</td>
<td>lamp_flats-interpol_index</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.max_rad</td>
<td>lamp_flats-max_rad</td>
<td>4</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.bad_ind</td>
<td>lamp_flats-bad_ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.sigma_factor</td>
<td>lamp_flats-sigma_factor</td>
<td>5.0</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.factor</td>
<td>lamp_flats-factor</td>
<td>3.0</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.iterations</td>
<td>lamp_flats-iterations</td>
<td>8</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.bad_low_rejection</td>
<td>lamp_flats-bad_lo_rej</td>
<td>10.0</td>
</tr>
<tr>
<td>sinfoni.lamp_flats.bad_high_rejection</td>
<td>lamp_flats-bad_hi_rej</td>
<td>10.0</td>
</tr>
<tr>
<td>Variable</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------</td>
<td></td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.llx</code></td>
<td><code>lamp_flats-llx</code></td>
<td>1350</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.lly</code></td>
<td><code>lamp_flats-lly</code></td>
<td>1000</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.urx</code></td>
<td><code>lamp_flats-rrx</code></td>
<td>1390</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.ury</code></td>
<td><code>lamp_flats-ury</code></td>
<td>1200</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.thresh_ind</code></td>
<td><code>lamp_flats-tresh_ind</code></td>
<td>FALSE</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.mean_factor</code></td>
<td><code>lamp_flats-mean_factor</code></td>
<td>10.0</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_xmin1</code></td>
<td><code>lamp_flats-qc_fpn_xmin1</code></td>
<td>512</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_xmax1</code></td>
<td><code>lamp_flats-qc_fpn_xmax1</code></td>
<td>1536</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_ymin1</code></td>
<td><code>lamp_flats-qc_fpn_ymin1</code></td>
<td>512</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_xmax2</code></td>
<td><code>lamp_flats-qc_fpn_xmax2</code></td>
<td>1350</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_ymin2</code></td>
<td><code>lamp_flats-qc_fpn_ymin2</code></td>
<td>1390</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_fpn_ymax2</code></td>
<td><code>lamp_flats-qc_fpn_ymax2</code></td>
<td>1200</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_thresh_min</code></td>
<td><code>lamp_flats-qc_thresh_min</code></td>
<td>0</td>
</tr>
<tr>
<td><code>sinfoni.lamp_flats.qc_thresh_max</code></td>
<td><code>lamp_flats-qc_thresh_max</code></td>
<td>49000</td>
</tr>
<tr>
<td><code>sinfoni.bp.method</code></td>
<td><code>bp-method</code></td>
<td>Normal</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.sigma_factor</code></td>
<td><code>bp_dist-s_factor</code></td>
<td>2.0</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.method_index</code></td>
<td><code>bp_dist-method_ind</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.factor</code></td>
<td><code>bp_dist-fct</code></td>
<td>999.0</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.iterations</code></td>
<td><code>bp_dist-it</code></td>
<td>8</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.low_rejection</code></td>
<td><code>bp_dist-lo_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.high_rejection</code></td>
<td><code>bp_dist-hi_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.llx</code></td>
<td><code>bp_dist-llx</code></td>
<td>1350</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.lly</code></td>
<td><code>bp_dist-lly</code></td>
<td>1000</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.urx</code></td>
<td><code>bp_dist-urx</code></td>
<td>1390</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.ury</code></td>
<td><code>bp_dist-ury</code></td>
<td>1200</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.threshold_index</code></td>
<td><code>bp_dist-thr_ind</code></td>
<td>TRUE</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.mean_factor</code></td>
<td><code>bp_dist-mean_fct</code></td>
<td>999.0</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.min_cut</code></td>
<td><code>bp_dist-min_cut</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.bp_dist.max_cut</code></td>
<td><code>bp_dist-max_cut</code></td>
<td>5e+04</td>
</tr>
<tr>
<td><code>sinfoni.stacked.low_rejection</code></td>
<td><code>stack-lo_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.stacked.high_rejection</code></td>
<td><code>stack-hi_rej</code></td>
<td>0.1</td>
</tr>
<tr>
<td><code>sinfoni.stacked.flat_index</code></td>
<td><code>stack-flat_ind</code></td>
<td>TRUE</td>
</tr>
<tr>
<td><code>sinfoni.stacked.mask_index</code></td>
<td><code>stack-mask_ind</code></td>
<td>1</td>
</tr>
<tr>
<td><code>sinfoni.stacked.ind_index</code></td>
<td><code>stack-ind_ind</code></td>
<td>FALSE</td>
</tr>
<tr>
<td><code>sinfoni.stacked.mask_rad</code></td>
<td><code>stack-mask_rad</code></td>
<td>4</td>
</tr>
<tr>
<td><code>sinfoni.stacked.gauss_index</code></td>
<td><code>stack-gauss_ind</code></td>
<td>FALSE</td>
</tr>
<tr>
<td><code>sinfoni.stacked.kernel_half_width</code></td>
<td><code>stack-khw</code></td>
<td>2</td>
</tr>
<tr>
<td><code>sinfoni.stacked.warpfix_ind</code></td>
<td><code>stack-warpfix_ind</code></td>
<td>TRUE</td>
</tr>
<tr>
<td><code>sinfoni.stacked.warpfix_kernel</code></td>
<td><code>stack-warpfix_kernel</code></td>
<td>tanh</td>
</tr>
<tr>
<td><code>sinfoni.stack.qc_thresh_min</code></td>
<td><code>stack-qc_thresh_min</code></td>
<td>0</td>
</tr>
<tr>
<td><code>sinfoni.stack.qc_thresh_max</code></td>
<td><code>stack-qc_thresh_max</code></td>
<td>64000</td>
</tr>
<tr>
<td><code>sinfoni.distortion.calib_indicator</code></td>
<td><code>dist-calib_indicator</code></td>
<td>TRUE</td>
</tr>
<tr>
<td><code>sinfoni.distortion.min_diff_mean_med_col_int</code></td>
<td><code>dist-min_diff_mean_med_col_int</code></td>
<td>10.0</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td></td>
</tr>
<tr>
<td>---------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.half_width</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.sigma</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.fwhm</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.min_amplitude</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.max_residual</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.n_a_coefficients</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.n_b_coefficients</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.sigma_factor</td>
<td>1.5</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.write coeffs_ind</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.write_par_ind</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.pixel_dist</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.pixel_tol</td>
<td>3.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.wave_map_ind</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.mag_factor</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.slit_pos_indicator</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.fit_boltz_indicator</td>
<td>TRUE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.fit_edge_indicator</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.estimate_indicator</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.box_length</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.y_box</td>
<td>5.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.diff_tol</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.qc_thresh_min</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.qc_thresh_max</td>
<td>64000</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.lower_rejection</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.higher_rejection</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.mask_ind</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.gauss_ind</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.kernel_half_width</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.ns_half_width</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.ns_fwhm</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.min_diff</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.dev_tol</td>
<td>20.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.arcs_thresh_factor</td>
<td>0.33333</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.arcs_min_arclen_factor</td>
<td>1.19</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.arcs_window_size</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>sinfoni.distortion.smooth_rad</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.low_rejection</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.high_rejection</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.mask_ind</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.gauss_ind</td>
<td>FALSE</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.kernel_half_width</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.fwhm</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.min_diff</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>sinfoni.north_south_test.dev_tol</td>
<td>20.0</td>
<td></td>
</tr>
</tbody>
</table>
Figure 10.6.1: Optical distortions quality control: (a) trending plot of upper left X shift; (b) trending plot of slitlet #17 position.

Please note that the default of the parameter `bp_norm-min_cut`, in case of J band data taken after 31/1/2018, is set to 0.05 instead of 0.1, unless the user sets explicitly a different value. This is to cope with the strong vignetting of the central slice.

### 10.7 sinfo_rec_wavecal

The recipe `sinfo_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.

#### 10.7.1 Input

```
/path_file_raw/SINFO.2004-08-15T11:26:49.348.fits WAVE_LAMP
/path_file_cdb/neonK.fits 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.fits DISTORTION
/path_file_cdb/drs_setup_wave.fits DRS_SETUP_WAVE
/path_file_cdb/SLIT_POS_K_100.fits 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).

Figure 10.7.1: Wavelength calibration based on arcs and sky lines: (a) Object cube planes calibrated using a wave map generated using arc lamp frames; (b) Object cube planes calibrated using a wave map generated using sky frames; The more uniform color palette of image (b) is an indication of higher accuracy in the SINFONI FOV. Images refers to H band observations.

Since release 1.9.8 the user may also perform an OH based wavelength calibration. In this case in place of the on/off ac lamp frames, one needs to input a sky frame (tagged as SKY_NODDING) and an dark frame (tagged as MASTER_DARK). The WAVE_MAP that one may obtain in this way may grant a better science data reduction accuracy if used in input of the siunfo_rec_jitter recipe.

```
/path_file_raw/SINFO.2008-10-01T03:26:49.348.fits SKY_NODDING
/path_file_cdb/MASTER_DARK.fits MASTER_DARK
/path_file_cdb/neonK.fits 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
```
Notes:

the input table tagged as DRS_SETUP_WAVE, has has different values of W_LOW_POS and W_HI_POS for H band, the appropriate ones for a SKY frame in H band, as in the case we have performed tests. We provide this table in the file drs_setup_wave_oh.fits part of the demo distribution. The user willing to use sky lines to perform the wavelength calibration on other bands, need to change the W_LOW_POS and W_HI_POS values for the column corresponding to the band of interest, so that the specified Y range includes at least one (or a few) sky line(s) with good signal to noise ratio.

For accuracy one needs to have input sky frames with sufficiently high signal to noise ratio. This is usually the case for 250 mas observations, and often in case of 100 mas observations. In case of 25 mas observations the signal to noise ratio is usually too low to use the frame for wavelength calibration, unless one may acquire deep sky exposures (that on the other side are affected by sky lines temporal variations).

While for H and J band an OH based wavelength calibration may work, in K band it may be less robust/accurate beyond 2.3 um, due to absence of OH lines. We leave the decision about using sky frames as input of this recipe to the user.

10.7.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_stack_mflat_dist.fits</td>
<td>MFLAT_STACKED_DIST</td>
<td>stacked FLAT,LAMP image</td>
<td>2</td>
</tr>
<tr>
<td>out_wcal_stack.fits</td>
<td>WAVE_LAMP_STACKED</td>
<td>stacked WAVE,LAMP image</td>
<td>2</td>
</tr>
<tr>
<td>out_wavemap_ima.fits</td>
<td>WAVE_MAP</td>
<td>wavelength map image: wavelength=intensity</td>
<td>0</td>
</tr>
<tr>
<td>outCoeffsSlit.fits</td>
<td>WAVE_COEF_SLIT</td>
<td>table with parabolic fit coefficients, for each pixel</td>
<td>2</td>
</tr>
<tr>
<td>out_fit_params.fits</td>
<td>WAVE_FIT_PARAMS</td>
<td>table with parameters relative to the fit of lines</td>
<td>2</td>
</tr>
<tr>
<td>out_slitpos.fits</td>
<td>SLIT_POS</td>
<td>slitlets positions</td>
<td>0</td>
</tr>
<tr>
<td>out_resampled_arclamp.fits</td>
<td>RESAMPLED_WAVE</td>
<td>stacked, resampled WAVE,LAMP image</td>
<td>2</td>
</tr>
</tbody>
</table>

The WAVE_COEF_SLIT table stores the computed polynomial coefficients:

- coeff0: coefficient polynomial term order 0
- coeff1: coefficient polynomial term order 1
- coeff2: coefficient polynomial term order 2
- coeff3: coefficient polynomial term order 3

The WAVE_FIT_PARAMS table stores the computed parameters of lines fit:

- n_params: number of rows in table
- column: detector’s column number
• line: increasing value of line ID
• fpar0: line Gaussian amplitude
• dpar0: line Gaussian amplitude’s error
• fpar1: line FWHM
• dpar1: line FWHM’s error
• fpar2: line center
• dpar2: line center’s error
• fpar3: line background
• dpar3: line background’s error

The SLIT_POS table stores the computed slitlets left and right edges positions:

• pos1: left slitlet edge position
• pos2: right slitlet edge position

10.7.3 Quality control

The recipe monitors the resolving power, the overall wavelength dispersion offset in $\mu$m 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 $\mu$m 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 DPD 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.

![Figure 10.7.2: Wavelength calibration quality control: (a) trending plot of overall positioning error; (b) trending plot of median line’s FWHM.](image)

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.

Resampled arc lamp frame  This frame is used for quality control to check the wavelength calibration accuracy (after frame resampling).

10.7.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoni.stacked.low_rejection</td>
<td>stack-lo_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.stacked.high_rejection</td>
<td>stack-hi_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.stacked.flat_index</td>
<td>stack-flat_ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.stacked.mask_index</td>
<td>stack-mask_ind</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.stacked.ind_index</td>
<td>stack-ind_ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.stacked.mask_rad</td>
<td>stack-mask_rad</td>
<td>4</td>
</tr>
<tr>
<td>sinfoni.stacked.gauss_index</td>
<td>stack-gauss_ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.stacked.kernel_half_width</td>
<td>stack-khw</td>
<td>2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>----------------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>sinfoni.stacked warpedfix_ind</td>
<td>stack warpedfix_ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.stacked warpedfix_kernel</td>
<td>stack warpedfix kernel</td>
<td>tanh</td>
</tr>
<tr>
<td>sinfoni.stack qc thresh min</td>
<td>stack qc thresh min</td>
<td>0</td>
</tr>
<tr>
<td>sinfoni.stack qc thresh max</td>
<td>stack qc thresh max</td>
<td>64000</td>
</tr>
<tr>
<td>sinfoni.wavecal slitpos bootstrap switch</td>
<td>wcal slitpos bootstrap switch</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.wavecal calib indicator</td>
<td>wcal calib indicator</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.wavecal min diff</td>
<td>wcal min diff</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.wavecal half width</td>
<td>wcal half width</td>
<td>7</td>
</tr>
<tr>
<td>sinfoni.wavecal sigma</td>
<td>wcal sigma</td>
<td>2.0</td>
</tr>
<tr>
<td>sinfoni.wavecal fwhm</td>
<td>wcal fwhm</td>
<td>2.83</td>
</tr>
<tr>
<td>sinfoni.wavecal min amplitude</td>
<td>wcal min amplitude</td>
<td>5.0</td>
</tr>
<tr>
<td>sinfoni.wavecal max residual</td>
<td>wcal max residual</td>
<td>0.5</td>
</tr>
<tr>
<td>sinfoni.wavecal n a coefficients</td>
<td>wcal n a coeff</td>
<td>4</td>
</tr>
<tr>
<td>sinfoni.wavecal n b coefficients</td>
<td>wcal n b coeff</td>
<td>2</td>
</tr>
<tr>
<td>sinfoni.wavecal sigma factor</td>
<td>wcal sigma factor</td>
<td>1.5</td>
</tr>
<tr>
<td>sinfoni.wavecal write coeffs ind</td>
<td>wcal write coeffs ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.wavecal write par ind</td>
<td>wcal write par ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.wavecal pixel dist</td>
<td>wcal pixel dist</td>
<td>12</td>
</tr>
<tr>
<td>sinfoni.wavecal pixel tol</td>
<td>wcal pixel tol</td>
<td>3.0</td>
</tr>
<tr>
<td>sinfoni.wavecal wave map ind</td>
<td>wcal wave map ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.wavecal mag factor</td>
<td>wcal mag factor</td>
<td>8</td>
</tr>
<tr>
<td>sinfoni.wavecal slit pos indicator</td>
<td>wcal slit pos indicator</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.wavecal fit boltz indicator</td>
<td>wcal fit boltz indicator</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.wavecal fit edge indicator</td>
<td>wcal fit edge indicator</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.wavecal estimate indicator</td>
<td>wcal estimate indicator</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.wavecal box length</td>
<td>wcal box len</td>
<td>32</td>
</tr>
<tr>
<td>sinfoni.wavecal y box</td>
<td>wcal y box</td>
<td>5.0</td>
</tr>
<tr>
<td>sinfoni.wavecal diff tol</td>
<td>wcal diff tol</td>
<td>2.0</td>
</tr>
<tr>
<td>sinfoni.wavecal qc thresh min</td>
<td>wcal qc thresh min</td>
<td>0</td>
</tr>
<tr>
<td>sinfoni.wavecal qc thresh max</td>
<td>wcal qc thresh max</td>
<td>64000</td>
</tr>
</tbody>
</table>

### 10.8 sinf-rec-jitter

The recipe sinf-rec-jitter reduces PSF standard, telluric standard or other scientific targets 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. A cube is generated for each target object. Finally a mosaic of all the cubes is reconstructed. Finally it measures the PSF FWHM (x and y) and computes the instrument Strehl and the encircled energy. Moreover for target frames whose point spread function may be approximated by a 2D Gaussian, it extract the object average spectrum and computes the instrument+detector+telescope efficiency.

#### 10.8.1 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
```
The input frames can be either PSF calibrator standards (indicated in the set of frame by PSF_CALIBRATOR, SKY_PSF_CALIBRATOR), flux standars (indicated in the set of frame by STD, SKY_STD), or other objects observations (indicated in the set of frame by OBJ, SKY_OBJ). The inputs tagged as FLUX_STD_CATALOG and ATMEXT_TABLE are required either to compute efficiency or response. The input tagged as EFFICIENCY_WINDOWS is also required to determine the efficiency.

The inputs tagged as RESP_FIT_POINTS_CATALOG, RESPONSE_WINDOWS, TELL_MOD_CATALOG, QUALITY AREAS, FIT AREAS, HIGH ABS REGIONS, are also required to determine the response.

The FLUX_STD_CATALOG, is the catalog of flux std stars. The EXTCOEFF_TABLE, is the atmospheric extinction correction table. The EFFICIENCY WINDOWS, sets the regions used to compute QC on efficiency. The RESP_FIT_POINTS_CATALOG, is the catalog indicating points to fit the response. The RESPONSE_WINDOWS, are the regions used to compute QC on the response. The TEL_MOD_CATG, is the Catalog of Telluric models; the QUALITY AREAS, are the areas where the quality of the Telluric fitting is calculated*. The FIT AREAS, are the areas used for Telluric model fitting. The HIGH ABS REGIONS, are the High absorption regions*.

### 10.8.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PROC_CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
<tbody>
<tr>
<td>sky00.fits</td>
<td>SKY_STACKED_DUMMY</td>
<td>stacked sky image</td>
<td>2</td>
</tr>
<tr>
<td>out_sky_stack_dist0.fits</td>
<td>SKY_STACKED_DIST</td>
<td>stacked sky image distortion corrected</td>
<td>2</td>
</tr>
<tr>
<td>out_stack_mflat_dist.fits</td>
<td>MFLAT_STACKED_DIST</td>
<td>stacked master flat image</td>
<td>2</td>
</tr>
<tr>
<td>out_stackN.fits</td>
<td>X_CALIBRATOR_STACKED</td>
<td>stacked X image, N=0,1,...</td>
<td>0</td>
</tr>
<tr>
<td>out_resampled_obj0.fits</td>
<td>RESAMPLED_OBJ</td>
<td>stacked, resampled object image</td>
<td>2</td>
</tr>
<tr>
<td>out_resampled_sky0.fits</td>
<td>RESAMPLED_SKY</td>
<td>stacked, resampled sky image</td>
<td>2</td>
</tr>
<tr>
<td>out_resampled_flat0.fits</td>
<td>RESAMPLED_FLAT_LAMP</td>
<td>stacked, resampled flat image, distortions corrected</td>
<td>2</td>
</tr>
<tr>
<td>out_sky_cube00.fits</td>
<td>OBS_SKY</td>
<td>sky cube</td>
<td>1</td>
</tr>
<tr>
<td>File Name</td>
<td>Description</td>
<td>Density</td>
<td></td>
</tr>
<tr>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>out_sky_med00.fits</td>
<td>SKY_MED z-median image</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>out_mflat_cube.fits</td>
<td>MFLAT_CUBE Master flat cube</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>out_mflat_avg.fits</td>
<td>MFLAT_AVG MFLAT_CUBE z-mean image</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>out_mflat_med.fits</td>
<td>MFLAT_MED MFLAT_CUBE z-median image</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>out_cube_objNN.fits</td>
<td>OBS_X X cube, NN=00.01,...</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_cube_obj_corNN.fits</td>
<td>OBS_X X cube, sky residual corrected, NN=00.01,...</td>
<td>1/3</td>
<td></td>
</tr>
<tr>
<td>out_int_objNN.fits</td>
<td>QC_SPECTRA QC info on sky residual corrected spectra</td>
<td>1/3</td>
<td></td>
</tr>
<tr>
<td>out_objnod.fits</td>
<td>COADD_X coadded cube</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_objnod_med_cube.fits</td>
<td>MED_COADD_X COADD_X z-median image</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_objnod_bmap.fits</td>
<td>MASK_COADD_X COADD_X's mask cube</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_ao_performance.fits</td>
<td>AO_PERFORMANCE table with strehl information</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_encircled_energy.fits</td>
<td>ENC_ENERGY table with encircled energy information</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_psf.fits</td>
<td>MASTER_PSF z-median of COADD_X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_std_star_spectrum.fits</td>
<td>STD_STAR_SPECTRA table with extracted spectrum, and efficiency information for COADD_X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_starspectrum.fits</td>
<td>STD_STAR_SPECTRUM collapsed image spectrum COADD_X</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_response.fits</td>
<td>RESPONSE computed response</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_spectra_fluxcal.fits</td>
<td>STD_STAR_SPECTRA_FLUXCAL table with flux calibrated spectra</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_cube_fluxcal.fits</td>
<td>COADD_STD_FLUXCAL flux calibrated coadded cube</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>out_efficiency.fits</td>
<td>EFFICIENCY table with computed efficiency</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Here X=PSF or STD or OBJ if in input are PSF or telluric standards or other science frames. Product indicated with * are produced only if product-density is set to 1 or 3 and objnod-sky_cor is set to TRUE.

The last five frames are generated only in case of input PSF or STD frames.

The AO_PERFORMANCE table stores information about the computed Strehl:

- wavelength: wavelength at which Strehl is computed
- strehl: Strehl value
- strehl_error: Strehl error

The STD_STAR_SPECTRA table stores information about the extracted spectrum:

- wavelength: wavelength [um]
- counts_tot: total counts integrated in a given window [ADU]
- counts_bkg: total counts - background counts in the same window [ADU]
- bkg_tot: background counts in the same window [ADU]

The ENCIRCLED_ENERGY table stores information about the encircled energy:

- r_pix: radii in pix units
- r_mas: radii in milli arcsecs
- r_dif: radii in utints of diffraction limit radii
• abs_energy: absolute encircled energy, the flux integrated in a circle of radius r_pix centered on the maximum position (star position) of the image obtained by median collapsing the star cube.

• rel_energy: relative encircled energy, equal to abs_energy divided by the maximum value of the image obtained by median collapsing the star cube.

10.8.3 Quality control

QC.STREHLi log the instrument’s Strehl at wavelength QC.LAMBDAi. It also logged the median Strehl (QC.STREHL.MED) and the mean Strehl (QC.STREHL.AVG), and the error associated to the median Strehl (QC.STREHL.MEDERR). The values of QC.PUPILI.SHIFTX, QC.PUPILI.SHIFTY, indicate the position of the centroid of the pupil image. QC.ENC.CORE give the encircled energy in the PSF’s core. QC.FWMX, QC.FWHMY, QC.CONVFCT, QC.CHECKi, QC.SHIFTX, QC.SHIFTY. The values of QC.FWHM.LLX and QC.FWHM.LLY indicate the position of the lower left X and Y position of the rectangular box where a 2D Gaussian fit to the object PSF is calculated to extract the object spectrum and the QC.FWHM.HBX and QC.FWHM.HBY are the corresponding half box sizes. The parameters QC.FWHM.MAJ, QC.FWHM.MIN, QC.THETA indicate the size of the major and minor ellipsoid axes and their angle with respect to the X-Y standard axes in the 2D ellipsoid fit to the image PSF. For the quality control are also computed the heliocentric and barycentric radial velocity corrections, QC.VRAD.HELCOR and QC.VRAD.BARYCOR. Those corrections are not applied to the 3D cube.

10.8.4 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoni.product.density</td>
<td>product-density</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.stacked.low_rejection</td>
<td>stack-lo_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.stacked.high_rejection</td>
<td>stack-hi_rej</td>
<td>0.1</td>
</tr>
<tr>
<td>sinfoni.stacked.flat_index</td>
<td>stack-flat_ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.stacked.mask_index</td>
<td>stack-mask_ind</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.stacked.ind_index</td>
<td>stack-ind_ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.stacked.mask_rad</td>
<td>stack-mask_rad</td>
<td>4</td>
</tr>
<tr>
<td>sinfoni.stacked.gauss_index</td>
<td>stack-gauss_ind</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.stacked.kernel_half_width</td>
<td>stack-khw</td>
<td>2</td>
</tr>
<tr>
<td>sinfoni.stacked.warpfix_ind</td>
<td>stack-warpfix_ind</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.stacked.warpfix_kernel</td>
<td>stack-warpfix_kernel</td>
<td>tanh</td>
</tr>
<tr>
<td>sinfoni.stack.qc_thresh_min</td>
<td>stack-qc_thresh_min</td>
<td>0</td>
</tr>
<tr>
<td>sinfoni.stack.qc_thresh_max</td>
<td>stack-qc_thresh_max</td>
<td>64000</td>
</tr>
<tr>
<td>sinfoni.objnod.autojitter_method</td>
<td>objnod-aj_method</td>
<td>1</td>
</tr>
<tr>
<td>sinfoni.objnod.scales_sky</td>
<td>objnod-scales_sky</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfoni.objnod.pk</td>
<td>objnod-ks_clip</td>
<td>TRUE</td>
</tr>
<tr>
<td>sinfoni.objnod.kappa</td>
<td>objnod-kappa</td>
<td>2.0</td>
</tr>
<tr>
<td>sinfoni.objnod.size_x</td>
<td>objnod-size_x</td>
<td>0</td>
</tr>
<tr>
<td>sinfoni.objnod.size_y</td>
<td>objnod-size_y</td>
<td>0</td>
</tr>
<tr>
<td>sinfoni.objnod.n_coeffs</td>
<td>objnod-no_coeffs</td>
<td>3</td>
</tr>
</tbody>
</table>
Note the parameter objnod.fcol should have the value of the detector’s first slice’s first column. This value is computed by the recipe sinfo_rec_distortion and displayed in the recipe log as (for example)

```
[ INFO ] sinfo_rec_distortion: Firstcol =13.252930
```

Note that to obtain better sky subtraction quality the user should set both the parameters product-density to 3 and the parameter objnod-sky_cor to TRUE. The correction of sky residuals should be performed having (at least) a sky frame available. In case a sky frame is missing the pipeline uses the next in time object frame as
a “dummy” sky frame and activating the the sky residual correction there is the risk that object lines cancel out after subtraction.

The user should also choose proper values of the critical parameters `skycor-sky_bkg_filter_width`, `skycor-line_hw`, `skycor-rot_cor`, `skycor-fit_obj_noise`. A check of the results provided by the product table(s) with PRO.CATG QC_SPECTRA is always recommended to compare the extracted spectra before and after the correction.

This release allows also to indicate the area where the object lies, through the parameters `skycor-llx`, `skycor-lly`, `skycor-urx`, `skycor-ury`. This to allow a more accurate determination of the sky lines residuals and the corresponding spectrum shift with respect to the one of the object in case of low signal to noise target observations. The thermal background emission subtraction can be eventually switched off with the parameter `skycor-sub_thr_bkg_from_obj`. Having and independent sky-object spectral shift estimation (in pixel units), or willing to overwrite the value computed by the pipeline, the user can set this quantity with the parameter `skycor-pshift`.

This release has the default of the parameter `product-density` set to 2 to make sure that at Paranal the data reduction is fast enough.

Setting the parameter `scale_sky` to TRUE allows to correct the spatial median from each plane of the reconstructed cube before combining the cubes one with the other.

If the sky subtraction was correct during each cube creation there is no need to further correct each cube by removing a median. That is why the default of the parameter `scale_sky` is set to FALSE. The pipeline keeps this parameter available for those cases where the user may still want to correct the spatial median.

### 10.9 sinfo_rec_pupil

This recipe performs the same data reduction as the `sinfo_rec_jitter` on PUPIL, IMAGE frames. Please refer to 10.8 for parameters description. We describe here only the main differences.

#### 10.9.1 Input

```
/path_file_raw/SINFO.2004-08-13T03:26:12.559.fits PUPIL_LAMP
/path_file_raw/SINFO.2004-08-13T03:25:30.323.fits PUPIL_LAMP
/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.fits SLIT_POS
/path_file_cdb/SLITLETS_DISTANCE_K.fits SLITLETS_DISTANCE
/path_file_cdb/DISTORTION_K.fits DISTORTION
```

#### 10.9.2 Output

<table>
<thead>
<tr>
<th>default recipe file name</th>
<th>PRO.CATG</th>
<th>short description</th>
<th>pd</th>
</tr>
</thead>
</table>
### Table of Outputs

<table>
<thead>
<tr>
<th>File Name</th>
<th>Description</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>sky00.fits</td>
<td>stacked sky image</td>
<td>2</td>
</tr>
<tr>
<td>out_sky_stack_dist0.fits</td>
<td>stacked sky image distortion corrected</td>
<td>2</td>
</tr>
<tr>
<td>out_stack_mflat_dist.fits</td>
<td>stacked master flat image</td>
<td>2</td>
</tr>
<tr>
<td>out_stackN.fits</td>
<td>stacked X image, N=0,1,...</td>
<td>0</td>
</tr>
<tr>
<td>out_resampled_sky0.fits</td>
<td>stacked, resampled object image</td>
<td>2</td>
</tr>
<tr>
<td>out_resampled_skyN.fits</td>
<td>stacked, resampled sky image</td>
<td>2</td>
</tr>
<tr>
<td>out_resampled_mflat0.fits</td>
<td>stacked, resampled flat image</td>
<td>2</td>
</tr>
<tr>
<td>out_sky_cube00.fits</td>
<td>distortions corrected</td>
<td>1</td>
</tr>
<tr>
<td>out_sky_med00.fits</td>
<td>SKY_CUBE z-median image</td>
<td>2</td>
</tr>
<tr>
<td>out_mflat_cube.fits</td>
<td>Master flat cube</td>
<td>2</td>
</tr>
<tr>
<td>out_mflat_avg.fits</td>
<td>MFLAT_CUBE z-mean image</td>
<td>2</td>
</tr>
<tr>
<td>out_mflat_med.fits</td>
<td>MFLAT_CUBE z-median image</td>
<td>2</td>
</tr>
<tr>
<td>cube_objN.fits</td>
<td>X STD cube, NN=00,01,...</td>
<td>0</td>
</tr>
<tr>
<td>out_objnod.fits</td>
<td>coadded cube</td>
<td>0</td>
</tr>
<tr>
<td>out_objnod_med_cube.fits</td>
<td>COADD_X z-median image</td>
<td>0</td>
</tr>
<tr>
<td>out_objnod_bpmmap.fits</td>
<td>COADD_X’s mask cube</td>
<td>0</td>
</tr>
<tr>
<td>out_AO_performance.fits</td>
<td>table with strehl information</td>
<td>0</td>
</tr>
<tr>
<td>out_encircled_energy.fits</td>
<td>table with encircled energy information</td>
<td>0</td>
</tr>
<tr>
<td>out_psf.fits</td>
<td>z-median of COADD_X</td>
<td>0</td>
</tr>
<tr>
<td>out_std_star_spectrum.fits</td>
<td>table with extracted spectrum,</td>
<td>0</td>
</tr>
<tr>
<td>out_starspectrum.fits</td>
<td>and efficiency information for COADD_X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>collapsed image spectrum COADD_X</td>
<td>0</td>
</tr>
</tbody>
</table>

Here X=PSF or STD or OBJ if in input are PSF or telluric standards or other science frames.

Figure 10.9.1: Telescope pupil centering quality control: (a) pupil image (median of reconstructed cube); (b) trending plot of X centroid shift; (c) trending plot of Y centroid shift.
10.9.3 Quality control

The values of QC.PUPILi.SHIFTX, QC.PUPILi.SHIFTY, indicate the position of the centroid of the pupil image.

10.9.4 Parameters

This recipe has the same parameters as the `sinfo_rec_jitter` recipe.

10.10 `sinfo_utl_bp_mask_add`

This recipe performs bad pixel map coaddition.

10.10.1 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
```

10.10.2 Output

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

10.10.3 Parameters

The recipe output filename for the product is bp_map_sum.fits

10.11 `sinfo_utl_ima_arith`

This recipe performs image computation.

10.11.1 Input

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

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

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

10.11.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_ima_arith.op</td>
<td>op</td>
<td>+</td>
</tr>
<tr>
<td>sinfo_utl_ima_arith.value</td>
<td>value</td>
<td>9999.0</td>
</tr>
</tbody>
</table>

10.12 sinfo_utl_cube_create

This recipe, after subtracting the off from the on frame, applying bad pixel and distortion correction, uses the information contained in the FIRST_COL, SLITLETS_DISTANCE, SLIT_POS and WAVE_MAP to re-sample the data in a cube format with the wavelength information along the cube’s Z axis and the spatial information on the X and Y axis.

10.12.1 Input

Input are an ON and and OFF frame and the usual master calibration frames input of the jitter recipe:

```
/path_file_raw/SINFO.2004-08-23T09:36:12.316.fits RAW_ON
/path_file_raw/SINFO.2004-08-23T09:51:59.824.fits RAW_OFF
/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/FIRST_COLUMN.fits FIRST_COL
/path_file_cdb/WAVE_MAP_H_250.fits WAVE_MAP
/path_file_cdb/SLITLETS_DISTANCE_H_025.fits SLITLETS_DISTANCE
/path_file_cdb/SLIT_POS_H_250.fits SLIT_POS
/path_file_cdb/DISTORTION_H_025.fits DISTORTION
```

10.12.2 Output

The output is a cube with PRO.CATG set to CUBE.

10.12.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_ima_arith.op</td>
<td>op</td>
<td>+</td>
</tr>
<tr>
<td>sinfo_utl_ima_arith.value</td>
<td>value</td>
<td>9999.0</td>
</tr>
</tbody>
</table>
This recipe performs cube to image conversion, that means it collapse the cube along the Z axis.

### 10.13 sinfo_utl_cube2ima

This recipe performs cube to image conversion, that means it collapse the cube along the Z axis.

### 10.13.1 Input

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

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

### 10.13.2 Output

The output is an image resulting from the average of the cube over a wavelength range which can be set by parameters sinfoi.sinfo_utl_cube2ima.ws, sinfoi.sinfo_utl_cube2ima.we having aliases 'ws' and 'we'. The recipe output filename for the product is out_ima.fits

### 10.13.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_cube2ima.ws</td>
<td>ws</td>
<td>0.9999</td>
</tr>
<tr>
<td>sinfo_utl_cube2ima.we</td>
<td>we</td>
<td>2.999</td>
</tr>
</tbody>
</table>

### 10.14 sinfo_utl_cube2spectrum

This recipe performs cube to 1D spectrum image conversion.

#### 10.14.1 Input

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

```
/path_file/cube.fits CUBE
```
10.14.2 Output

The output is an 1D (spectrum) image resulting by collapsing the cube flux data using the operation defined by the parameter sinfoi.sinfo_utl_cube2spectrum.op performed on each cubes’ plane over an aperture as specified by the parameter sinfoi.sinfo_utl_cube2spectrum.aperture. These parameters have respectively alias values of 'op', 'ap'.

The recipe output filename for the product is out_spec.fits

10.14.3 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.

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.op</td>
<td>op</td>
<td>average</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.ap</td>
<td>ap</td>
<td>rectangle</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.llx</td>
<td>llx</td>
<td>2</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.lly</td>
<td>lly</td>
<td>2</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.urx</td>
<td>urx</td>
<td>28</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.ury</td>
<td>ury</td>
<td>28</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.lo_rej</td>
<td>lo_rej</td>
<td>10</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.hi_rej</td>
<td>hi_rej</td>
<td>10</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.centerx</td>
<td>centerx</td>
<td>16</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.centery</td>
<td>centery</td>
<td>16</td>
</tr>
<tr>
<td>sinfoi.sinfo_utl_cube2spectrum.radius</td>
<td>radius</td>
<td>5</td>
</tr>
</tbody>
</table>

10.15 sinfoi.sinfo_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 cube and a spectrum; their associated tags should be respectively CUBE, IMA or CUBE, 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.

10.15.1 Input

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

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

### 10.15.2 Output

The recipe output filename for the product is out_cube.fits If a parameter value is specified its the value is subtracted from each cube’s pixel value; else if the inputs are a CUBE and an IMAGE, each image’s pixel value is subtracted from the corresponding pixel value of each of the input cube's image planes; else if the inputs are a CUBE and a SPECTRUM, each spectrum pixel is subtracted as a constant value from the corresponding wavelength cube’s image plane.

### 10.15.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_cube_arith.op</td>
<td>op</td>
<td>/</td>
</tr>
<tr>
<td>sinfo_utl_cube_arith.value</td>
<td>value</td>
<td>99999.0</td>
</tr>
</tbody>
</table>

### 10.16 sinfo_utl_cube_combine

#### 10.16.1 Input

The user needs to provide in an input ASCII file the file names of all the cube component to be coadded and their tags (OBS_OBJ); then to set the recipe input parameter `name_i` to the filename of another ASCII file specifying the corresponding x and y offsets. The default recipe output filename for the cube product is out_cube.fits and can be set by the parameter `name_o`. The name of the corresponding mask is by default out_maskcube.fits For example if the following list indicates the input cubes (for clarity we indicate also the values of the corresponding offsets, which do not need to appear in the input set of frame)

OBS_OBJ1.fits  OBS_OBJ  0.0  0.0
OBS_OBJ2.fits  OBS_OBJ  1.0  -1.0
OBS_OBJ3.fits  OBS_OBJ  2.0  -2.0

One needs to set the recipe parameter `name_i` to the filename of a file with the following content:

0.0 0.0
1.0 -1.0
2.0 -2.0

The user should read sections 11.1.23 and 11.1.24 to understand how the size of the coadded cube is determined and how to set proper offset values.
In case a user has observations taken with different templates, each template may have a different reference pointing (usually aligned with the first object frame of the template). The user will first build the cube components and their coadded mosaic corresponding to each template observation using the recipe sinfo_rec_jitter.

From release 2.0.0 the sinfo_utl_cube_combine utility has been extended to support also coaddition of cube mosaics. The user will have to determine the proper offset to be applied at each cube mosaic addendum with respect to the first mosaic of the list (taken at reference position X,Y = 0,0) in order to proper coadd the mosaics. In this case the user will assign to each mosaic component the tag COADD_OBJ.

Alternatively, to properly combine cube components coming from different templates one needs to compute proper relative offsets in pixel units of each cube with respect to a reference. If the user has for example two data sets, observed with two different templates, one will be used as reference, and the corresponding CUMOFFX/Y keywords will be taken as offset values to be applied to the cube components of that template (we assume here those values are properly set). For the frames of the second template, the offsets will be given from the values of the CUMOFFX/Y keywords to which one needs to add the offset in pix units between the reference frames observed in the second template and the one of the previous template. This offset can be computed using the INS.TARG.ALPHA|DELTA keywords of the reference frames of the two templates and the pixel scale.

For example if one need to combine the following frames:

OBS_OBJ1_set1.fits OBS_OBJ
...
OBS_OBJN_set1.fits OBS_OBJ
OBS_OBJ1_set2.fits OBS_OBJ
...
OBSOBJM_set2.fits OBS_OBJ

Will set the offsets as:

coffx_set1_c1 coffy_set1_c1
coffx_set1_c2 coffy_set1_c2
coffx_set1_c3 coffy_set1_c3
coffx_set2_c1 coffy_set2_c1
coffx_set2_c2 coffy_set2_c2
coffx_set3_c3 coffy_set2_c3

where usually coffx_set1_c1=coffy_set1_c1=0, coffx_set1_cn,coffy_set1_cn, n=1...N are given from the CUMOFFX/Y keywords of the first set and coffx_set1_cm, coffy_set1_cm, m=1...M are given from the CUMOFFX/Y keywords of the second set plus the offset in pixel between the first frame of the two data sets.

### 10.16.2 Output

The recipe generates the corresponding coadded cube.

### 10.16.3 Parameters
10.17 sinfo_utl_skycor

This recipe allow the user to remove possible residuals of sky lines in the object spectrum. The data reduction performed is described in section 11.1.26.

10.17.1 Input

Input are an object cube (OBS_OBJ, OBS_STD, or OBS_PSF) and a sky cube (OBS_SKY).

10.17.2 Output

Output are a sky residual corrected object frame (OBS_OBJ, OBS_STD, or OBS_PSF) and a quality control table (QC_SPECTRA) containing the following columns:

- WAVE wavelength in Å
- INT_OBJ_ORG original object spectrum
- SKY_INT_ORG original sky spectrum
- BKG_INT_FIT fit of thermal background sky emission
- SKY_INT_SMO smoothed sky spectrum
- COR_FCT_VIB corrective factor to remove contribute from sky lines from vibrational transitions
- COR_FCT_ALL corrective factor to remove contribute from sky lines from vibrational and rotational transitions
- INT_OBJ_COR corrected object spectrum
- INT_SKY_COR corrected sky spectrum

10.17.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_cube_combine.name_i</td>
<td>name_i</td>
<td>offset.list</td>
</tr>
<tr>
<td>sinfo_utl_cube_combine.name_o</td>
<td>name_o</td>
<td>out_coadd_cube.fits</td>
</tr>
<tr>
<td>sinfo_utl_cube_combine.ks_clip</td>
<td>ks_clip</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfo_utl_cube_combine.scale_sky</td>
<td>scale_sky</td>
<td>FALSE</td>
</tr>
<tr>
<td>sinfo_utl_cube_combine.xsize</td>
<td>xsize</td>
<td>80</td>
</tr>
<tr>
<td>sinfo_utl_cube_combine.ysize</td>
<td>ysize</td>
<td>80</td>
</tr>
</tbody>
</table>
### 10.18 sinfo_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.

#### 10.18.1 Input

Input are sky frames with tag SKY

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

#### 10.18.2 Output

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

#### 10.18.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_rec_skymap.ysize</td>
<td>ysize</td>
<td>30</td>
</tr>
<tr>
<td>sinfo_rec_skymap.xsize</td>
<td>xsize</td>
<td>1</td>
</tr>
<tr>
<td>sinfo_rec_skymap.thresh</td>
<td>thresh</td>
<td>30.0</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>grating</th>
<th>scale_out</th>
<th>DIT</th>
<th>scale_in</th>
<th>threshold</th>
<th>window</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>0.025</td>
<td>300</td>
<td>0.1</td>
<td>30.0</td>
<td>30</td>
</tr>
<tr>
<td>J</td>
<td>0.1</td>
<td>300</td>
<td>0.1</td>
<td>30.0</td>
<td>30</td>
</tr>
<tr>
<td>J</td>
<td>0.25</td>
<td>300</td>
<td>0.25</td>
<td>50.0</td>
<td>30</td>
</tr>
<tr>
<td>H</td>
<td>0.025</td>
<td>300</td>
<td>0.1</td>
<td>50.0</td>
<td>30</td>
</tr>
<tr>
<td>H</td>
<td>0.1</td>
<td>300</td>
<td>0.1</td>
<td>50.0</td>
<td>30</td>
</tr>
<tr>
<td>H</td>
<td>0.25</td>
<td>300</td>
<td>0.25</td>
<td>100.0</td>
<td>30</td>
</tr>
<tr>
<td>K</td>
<td>0.025</td>
<td>300</td>
<td>0.1</td>
<td>70.0</td>
<td>30</td>
</tr>
<tr>
<td>K</td>
<td>0.1</td>
<td>300</td>
<td>0.1</td>
<td>70.0</td>
<td>30</td>
</tr>
</tbody>
</table>
### 10.19 sinfo_utl_spectrum_divide_by_blackbody

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

#### 10.19.1 Input

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

```
STD_STAR_SPECTRUM.fits SPECTRUM
```

#### 10.19.2 Output

The output is a spectrum resulting in the division of the input spectrum signal at a given wavelength by the corresponding flux intensity according a Black Body model at the same wavelength and temperature specified by the parameter 'temp'.

The recipe output filename for the cube product is out ima.fits

#### 10.19.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_spectrum_arith.temp</td>
<td>temp</td>
<td>1e+05</td>
</tr>
</tbody>
</table>

### 10.20 sinfo_utl_spectrum_wavelength_shift

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

#### 10.20.1 Input

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

```
STD_STAR_SPECTRUM.fits SPECTRUM
```
10.20.2 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

10.20.3 Parameters

<table>
<thead>
<tr>
<th>parameter</th>
<th>alias</th>
<th>default</th>
</tr>
</thead>
<tbody>
<tr>
<td>sinfo_utl_spectrum_arith.method</td>
<td>method</td>
<td>S</td>
</tr>
<tr>
<td>sinfo_utl_spectrum_arith.shift</td>
<td>shift</td>
<td>0.1</td>
</tr>
</tbody>
</table>

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).
11 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.

11.1 Algorithms

11.1.1 sinfo_rec_mdark_detmon

A master dark is calculated and the clean mean dark level is computed. The master dark is the clean mean of three or more dark frames obtained with the same setup (same DIT, NDIT). This is a pipeline product.

11.1.2 Read Out Noise determination (common algorithm across pipelines)

The RON may be measured as the rms on defined areas dark.llx dark.lly, dark.urx dark.ury, of the difference between two dark frames, divided by $\sqrt{2}$. The RON must be determined independently for each quadrant and port. The RON and its rms are logged as QC parameters: QC.DARK.RON QC.DARK.RONRMS.

Other pipelines (SINFONI, ISAAC, NACO, VISIR) computes the RON by taking all possible consecutive DARK pairs having the same DIT. For each pair the second frame is subtracted to the first one. Then the noise is computed in a region defined by the parameters ron.llx, ron.lly, ron.urx, ron.ury, over nsamp samples each of size hsize, taken randomly in this region. The noise multiplied by $\sqrt{\text{NDIT}}$, where the NDIT gives the number of DIT repetitions, gives the read out noise.

11.1.3 Histogram computation

This algorithm build an histogram of the input frame intensity values. Input parameters are the minimum and maximum allowed frame intensity values, histo_min, histo_max in order to remove the contribution of threshold pixels, and the number of histogram sampling points histo_npoints. The interval $[\text{histo_min, histo_max}]$ is divided in equal steps of size $\frac{\text{histo_max} - \text{histo_min}}{\text{histo_npoints}}$, and an histogram of the frame intensity values is build. The histogram maximum and its corresponding bin is determined (this is the frame intensity clean mean). Then a Gaussian is fit to the histogram. The Gaussian sigma gives the frame intensity clean sigma. To make the fit more robust one may fit the logarithm of the intensity as a function of the square of the histogram frequency (abscissa). This is a linear function and the corresponding fit should be quite robust and reliable.

11.1.4 PTC and MED Algorithms for Gain computation

To measure the detector’s gain, a single algorithm is implemented common to both optical and infrared wavelength domains. In optical and infrared domain, the input is different. In the following description what we indicate as off frame is in the near infrared an off-lamp frame, but in the optical domain is a bias frame.

Input frames for NIR are a set of pairs of on/off flats, each set with a different DIT.
If we indicate with $on_{dif}$ the on frames difference:

$$on_{dif} = on_2 - on_1$$ (1)

If we indicate with $of_{dif}$ the off frames difference:

$$of_{dif} = of_2 - of_1$$ (2)

If we indicate with $mon_i$ the on frame clean mean (obtained either with the algorithm 11.1.3 or with the algorithm 11.1.6) relative to the integration time (EPTIME or DIT) $i$:

$$mon_i = \text{mean}(on_i)$$ (3)

And with $mof_i$ the similar off frames clean mean:

$$mof_i = \text{mean}(of_i)$$ (4)

One may find the gain by fitting the following linear (one may even fit a parabola and verify that the coefficient of the quadratic term is small enough, this being a QC parameter) relation:

$$(mon_1 + mon_2) - (mof_1 + mof_2) = \text{gain}(e^-/ADU)(\sigma_{ondif}^2 - \sigma_{ofdif}^2)\alpha$$ (5)

where $\alpha$ is the autocorrelation function computed as:

$$\alpha = \sum_{m,n} R_{m,n}$$ (6)

where:

$$R_{m,n} = \frac{\sum_{i,j} V_{i,j} * V_{i+m,j+n}}{\sum_{i,j} V_{i,j}^2}$$ (7)

where $V_{i,j}$ is the intensity of the pixel $(i, j)$.

The values of $mon_1$, $mof_2$, $mof_1$, $mof_2$, $\sigma_{ondif}^1$, $\sigma_{ondif}^2$ from equation 5 together with the values of the frame exposure time (DIT or EXPTIME) for each frame set should be logged in the corresponding columns of the product table having PRO.CATG set to GAIN_INFO.

If the parameter method is set to PTC (Photon Transfer Curve method) the gain is obtained with a linear fit of the values in eq. 5. Else, if the method is set to MED, the gain is obtained computing a simple median of the gain values that can be obtained with equation 5.

$$conad = \text{gain}^{-1}$$ (8)
The gain and its error are computed and logged as QC parameters (QC GAIN, QC GAIN MSE, in e⁻/ADU and QC CONAD in ADU/e⁻, QC AUTCORFA).

In eq. 5 it is critical to compute a clean mean and a clean standard deviation. Two methods are possible:

1. Build a histogram and compute the histogram centre and sigma (11.1.3).
2. Determine the mean and standard deviation repeating niter a clean mean computation with kappa-sigma cleaning of bad pixels (11.1.6).

11.1.5 Detector linearity computation (IR): irplib_detmon_nir_linear

The input interface is the same to irplib_detmon_gain, that is, for each exposure time value two pairs of frames (a pair ON and a pair OFF) are provided. The input frames DPR keyword should be (we recommend the following, but each instrument pipeline may decide to adopt also different values):

<table>
<thead>
<tr>
<th>DPR CATG</th>
<th>DPR TECH</th>
<th>DPR TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALIB</td>
<td>IMAGE</td>
<td>LINEARITY,FLAT,DETCHEF</td>
</tr>
</tbody>
</table>

For each exposure time value, two difference frames are computed: \( \text{dif}_1 = \text{on}_1 - \text{off}_1 \) and \( \text{dif}_2 = \text{on}_2 - \text{off}_2 \). These two difference frames are then averaged pixel by pixel. Then the mean and the median values of the difference frame in a rectangle defined by \( llx, lly, urx \) and \( ury \), are computed and written in the output table (PRO.CATG=LIN_INFO), in the column MEAN and MED, together with the corresponding pair integration time (DIT or EXPTIME), and the corresponding ratio, MEAN_DIT and MEAN_DIT.

Each average frame corresponding to a given DIT value is stored in a different plane of cube. Then for each x,y pixel intensity \( I(x,y,t) \) is computed a polynomial fit of order \( \text{order} \) in direction \( t \).

\[
Q(x, y, t) = \sum_{1,\text{order}} c_i(x, y)t^i
\]  
(9)

\[
Q'(x, y, t) = \frac{Q(x, y, t)}{c_1(x, y)t}
\]  
(10)

The corresponding coefficients \( c_i \) are stored in a cube of size \( \text{order} \). This is a recipe product (LIN_COEF). From each cube plane are computed the corresponding medians and standard deviations. Those values are used to identify bad pixels (threshold pixels, trap columns, dead pixels, non linear pixels, noisy pixels). Bad pixels are flagged and stored in a corresponding bad pixel map. The median and standard deviation computation is repeated to remove the contribution of bad pixels. Finally the median and standard deviation of each coefficient plane are monitored as quality control parameters (QC LIN COEFi, QC LIN COEFi ERR).

\[
Q(t) = \sum_{1,\text{order}} c_i t^i
\]  
(11)

\[
Q'(t) = \frac{Q(t)}{c_1 t}
\]  
(12)
Alternatively the detector linearity may be monitored on the frame obtained by integrating the signal over a user defined (llx, lly, urx, ury) detector region.

For each input frame on-off pair the difference frame is computed: \( \text{dif} = \text{on} - \text{of} \). Each frame difference is stored on an imagelist. Then the clean mean/median and standard deviation are computed on a given frame region (llx, lly, urx, ury). Those values are computed for each frame difference and stored in a vector. Then computed a polynomial fit of order order of vector values is computed:

\[
P(t) = \sum_{i=0}^{\text{order}} a_i t^i \quad (13)
\]

\[
P'(t) = \frac{P(t)}{a_1} \quad (14)
\]

The corresponding coefficients \( a_i \) are monitored as quality control parameters (QC LIN COEFi, QC LIN COEFi ERR).

Note that \( Q'(t) \) (or \( P'(t) \)) is independent from the lamp flux.

The correction that can be applied in a later step of the cascade would be:

\[
I_{\text{corrected}} = \frac{I_{\text{original}}}{Q'(t)} \quad (15)
\]

A map of deviating pixels is also generated. Those pixels are the one which deviate by more than kappa sigma by the fit, where sigma is the standard deviation of the points from the fit.

Indeed several bad pixel masks can be obtained: A map of threshold pixels (pixels with intensities beyond a given threshold). A map of dead pixels (pixels for which the \( c_0 \) - or the corresponding pixel intensity - is below a minimum threshold threshold_min). A map of trap pixels (pixels for which \( c_0 \) deviates more than the kappa*sigma by the median of \( c_0 \)). A map on non linear pixels (pixels which deviates more than kappa*sigma from the linear behaviour-or pixels whose \( c_1 \) deviates more than kappa*sigma from the median of \( c_1(x, y) \)). A map of non quadratic pixels (pixels which deviates more than kappa*sigma from the quadratic behaviour-or pixels whose \( c_2 \) deviates more than kappa*sigma from the median of \( c_2(x, y) \)).

An additional QC parameter is QC LIN EFF, the effective non-linearity correction at a reference user defined level ref_level: \( Q'(\text{ref_level}) \).

### Iteratives clean mean and sigma computation

This algorithm computes a clean mean and standard deviation through iterative kappa-sigma clipping of outliers. At each iteration the value of the clean mean and sigma is updated. Optionally the user may set the minimum and maximum allowed intensity values (threshold_min, threshold_max). Initially the input frame intensity values outside the interval [threshold_min, threshold_max] are flagged to remove their contribute from the following operations. Then an initial value of the intensity mean and stdev are computed. A loop is repeated niter iterations, and in each iteration are clipped intensity values which are outside the interval [mean-kappa*stdev,mean+kappa*stdev]. The final values of mean and stdev are the clean frame intensity mean and sigma.
11.1.7 Frame stacking

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

11.1.8 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 yields 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].

11.1.9 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:

\[
med_{dit_i} = \frac{\text{median}(frm_{on}(i)) - \text{median}(frm_{off}(i))}{DIT(i)}
\]

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.
11.1.10 Nearest neighbours bad pixel cleaning

This method is applied in the recipes sinfo_rec_mflat and sinfo_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 11.1.1: A bad pixel is indicated in black. If method_index is 1,2 or 4 this is corrected by 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, \( \text{median}_i \), is computed excluding from this set those pixels whose intensity is NAN.

\[
\text{median}_i = \text{median}(I_j)...I_j \neq \text{NAN}
\]

- If the parameter factor is set to zero:

\[
I_i = \text{median}_i
\]

- If the parameter factor is negative:
\[ I_i = \begin{cases} 
\text{median}_i & \text{if } ||I_i - \text{median}_i|| > \text{-factor} \\
I_i & \text{else} 
\end{cases} \]

- If the parameter \textbf{factor} is positive:

\[ I_i = \begin{cases} 
\text{median}_i & \text{if } ||I_i - \text{median}_i|| > \text{factor} \times \sqrt{\text{median}_i} \\
I_i & \text{else} 
\end{cases} \]

- If \textbf{method_index} is 2, the absolute distances \( dist_i \) of the eight nearest neighbours pixels \( j \) are computed for each bad pixel \( i \) as:

\[ dist_i = \sqrt{\frac{\sum_{j=1}^{8} (I_i - I_j)^2 \times w_j}{\sum_{j=1}^{8} w_j}} \]

where

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

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

\[ \text{median}_{distance} = \text{median}(dist_i) \]
\[ \text{stdev} = \sqrt{\left(\sum_i dist_i^2\right) - \left(\sum_i dist_i\right)^2} \]

- If the parameter \textbf{factor} is zero:

\[ I_i = dist_i \]

- If the parameter \textbf{factor} is negative:

\[ I_i = \begin{cases} 
dist_i & \text{if } ||\text{median}_{distance} - dist_i|| < \text{-factor} \times \text{stdev} \\
I_i & \text{else} 
\end{cases} \]

- If the parameter \textbf{factor} is positive:

\[ I_i = \begin{cases} 
dist_i & \text{if } ||\text{median}_{distance} - dist_i|| > \text{factor} \times \text{stdev} \times dist_i \\
I_i & \text{else} 
\end{cases} \]

- If \textbf{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:

\[ mean_i = \sqrt{\frac{\sum_{j=1}^{4} (I_i - I_j) \times w_j}{\sum_{j=1}^{4} w_j}} \]

where

\[ w_j = \begin{cases} 
0 & \text{if } I_j = \text{NAN} \\
1 & \text{else} 
\end{cases} \]
− If the parameter factor is zero:

\[ I_i = \text{mean}_i \]

− If the parameter factor is negative:

\[ I_i = \begin{cases} 
\text{mean}_i & \text{if } ||I_i - \text{mean}_i|| > -\text{factor} \\
I_i & \text{else}
\end{cases} \]

− If the parameter factor is positive:

\[ I_i = \begin{cases} 
\text{mean}_i & \text{if } ||I_i - \text{mean}_i|| > \text{factor} \times \sqrt{\text{mean}_i} \\
I_i & \text{else}
\end{cases} \]

If method_index is 4, the local clean (low_rejection, high_rejection) standard deviation clean_stdev in a box (lx, 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.

\[ \text{median}_i = \text{median}(I_j)...I_j \neq \text{NAN}, j = 1...8 \]

− if the parameter factor is zero:

\[ I_i = \text{median}_i \]

− if the parameter factor is negative:

\[ I_i = \begin{cases} 
\text{median}_i & \text{if } ||I_i - \text{median}_i|| > -\text{factor} \times \text{clean_stdev} \\
I_i & \text{else}
\end{cases} \]

− if the parameter factor is positive:

\[ I_i = \begin{cases} 
\text{median}_i & \text{if } ||I_i - \text{median}_i|| > \text{factor} \times \sqrt{||\text{median}_i||} \\
I_i & \text{else}
\end{cases} \]

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

### 11.1.11 Detector gain computation

The gain is computed as part of the recipe sinfo_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 (\( \text{dif}_\text{frm}_\text{on} = \text{frm}_\text{on2} - \text{frm}_\text{on1} \) and \( \text{dif}_\text{frm}_\text{off} = \text{frm}_\text{off2} - \text{frm}_\text{off1} \)). Then the mean of each frame (\( \text{frm}_\text{on1}, \text{frm}_\text{on2}, \text{frm}_\text{off1}, \text{frm}_\text{off2} \)) and the standard deviation of the difference \( \text{stdev}(\text{dif}_\text{frm}_\text{on}) \) and \( \text{stdev}(\text{dif}_\text{frm}_\text{off}) \) are computed.

Finally the gain is given by:

\[
\text{gain} = \frac{(\text{frm}_\text{on1} + \text{frm}_\text{on2}) - (\text{frm}_\text{off1} + \text{frm}_\text{off2})}{(\text{stdev}(\text{dif}_\text{frm}_\text{on})^2 - \text{stdev}(\text{dif}_\text{frm}_\text{off})^2)}
\]
11.1.12 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 $x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}$, over $n_{\text{samp}}$ samples each of size $h_{\text{size}}$ taken randomly in the given region. The noise multiplied by $\sqrt{NDIT}$, where $NDIT$ the number of DIT repetitions, gives the read out noise.

11.1.13 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.

11.1.14 Line position determination

The locations of the arc lamp lines are determined on an arc lamp frame, input of the recipe sinfo_rec_wavecal or sinfo_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, $(\text{begin\_wave})$, the linear and quadratic terms of the dispersion $(\text{guess\_disp1}$ and $\text{guess\_disp2}$) at the given instrument setting (see Figure 11.1.2 (a)). 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 $(\text{sigma})$ (see Figure 11.1.2 (a)). 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 $(\text{min\_diff})$. Thus one obtains approximate values (with accuracy of the order of a pixel) for the position and the wavelength assigned to each image position.

- To determine the dispersion relation (see 11.1.15) more accurately the recipe selects and uses only line identifications whose distance from the nearest neighbour is greater than $\text{half\_width}$ 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.
11.1.15 Dispersion relation and wavelength map determination

- A Levenberg-Marquardt fit of a Gaussian profile of given \texttt{fwhm} is performed considering the raw positions which lie in regions defined by boxes of half-size \texttt{half_width}, each centered at the line positions determined by the algorithm described in 11.1.14. The fit is restricted to the lines whose intensity is greater than \texttt{min_amplitude} with respect to the background. This fit allows to determine each line position with sub pixel accuracy (see Figure 11.1.2 (b)).

- Detector defects may lead to erroneous line detections. To prevent these misidentifications the fit is limited to lines which lie no more than \texttt{pixel_tolerance} pixels from the corresponding position, obtained assuming a linear dispersion \texttt{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 \texttt{n_a_coefficients} and coefficients \texttt{acoef} is performed, so as to determine the dispersion relation between the listed wavelength values $\lambda(j)$ and the Gauss-fitted positions $y_{\text{pos}}(j)$ for each image column using the singular value decomposition method.

\[
\lambda(j) = \sum_{k=0}^{\text{n_a_coefficients}} acoef_k \cdot y_{\text{pos}}(j)^k
\]

Data points which lie beyond \texttt{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 \texttt{guessDisp1} and the error associated with each position point from the previous Gaussian fit.
• Then the positions of the slitlet edges are determined. Offset coefficients \( a_{cof} \) resulting from the polynomial fit for adjacent columns are compared. When they differ by more than \( \text{pixel\_dist} \times \text{guess\_disp1} \), it is assumed that this is due to the crossing of a slitlet’s edge. Here \( \text{pixel\_dist} \) 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 \( \text{sigma\_factor} \).

• 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 \( a_{cof}i,x \) 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:

\[
\lambda(x,y) = \sum_{i=0}^{n_a\_coefficients} a_{cof}i,x \times (y - \text{off})^i
\]

and

\[
\text{off} = \frac{(ny - 1)}{2}
\]

where \( ny \) is the number of rows of the image.

11.1.16 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 11.1.15 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.

11.1.17 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 \( a_{cof}0 \) for adjacent image columns differ by more than \( \text{guess\_disp1} \times \text{pixel\_dist} \).
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 \((\text{mag}\_\text{factor} - 1)^2\).

Bad and negative pixels are filtered out from the input image and a low-pass filter of half width \text{mag}\_\text{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 \(\text{acoef}_0\) is determined. A clean mean is performed on the dispersion coefficients \(\text{acoef}_k\) and new values for the clean dispersion coefficients \(\overline{\text{acoef}}_k\) 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 \(\overline{\text{acoef}}_k\) as described at the end of section 11.1.15.

### 11.1.18 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 yielding 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 7.2.1 (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 \(\text{y}\_\text{box}\) 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 \(\text{coe}\_f_0\) has a variation greater than \(\text{y}\_\text{box}\). 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 \(\text{edge}(j)\) each of size \(\text{y}\_\text{box}\) along the spectral direction, and size \(\text{edge}(j) - \text{edge}(j-1) + 2 \times \text{box}\_\text{length}\) along the spatial direction.

In case the user sets the parameter \text{wcal-estimate_ind} value to TRUE the search of the image intensity maxima is limited to the interval \([\text{lo}\_\text{pos}, \text{hi}\_\text{pos}]\) along the spectral direction.

The minimum value of the maxima reached in each slitlet are determined by searching in rectangular boxes centered at each slitlet positions \(\text{edge}(j)\) each of size \(\text{y}\_\text{box}\) along the spectral direction, and size \(\text{edge}(j) - \text{edge}(j-1) + 2 \times \text{box}\_\text{length}\) along the spatial direction.

In case the user sets the parameter \text{wcal-estimate_ind} value to TRUE the search of the image intensity maxima is limited to the interval \([\text{lo}\_\text{pos}, \text{hi}\_\text{pos}]\) 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.

11.1.19 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 \(\text{halfwidth}\) 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 \(\text{estimated}_\text{dist}\) with a tolerance \(\text{dev}_\text{tol}\).
- A least square fit with a Gaussian function of FWHM \(\text{fwhm}\) 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 distances for which the relative distance differs from the \(\text{estimated}_\text{distance}\) by more than \(\text{dev}_\text{tol}\), one finally finds 31 values of the relative slitlet distances.

11.1.20 Optical distortions computation

The distortion function, which characterises the curvature of the spectral traces of the 32 slices, is calculated from a series of continuous light fibre spectra taken with the fibre moved perpendicularly to the slices. For every position of the fibre a raw FITS data file is recorded. With an initial data reduction step the single fibre spectra are co-added to an all-fibers synthetic frame and the offset of the first slitlet is calculated as a reference position. The distortion is computed in three further steps (see Figure. 11.1.3):

- Detecting the fibres and tracing the curved fibre spectra.
• Constructing two numerical grids on the distorted and the undistorted fibre positions.

• Solving a 2D polynomial fit to transform positions from raw to undistorted coordinates.

Finally, the recipe determines the position of the edges of the slitlets by fitting the edges of the brightest arc line of each slitlet with a linear step function or a Boltzmann function.

11.1.21 Cube construction: resampling

The algorithm described below is executed by recipe sinfo_rec_jitter. 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 (see Figure. 11.1.4).

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 11.1.18). Furthermore, the slitlets must be sorted in the correct sequence (see Figure 4.1.2 (bottom) or Figure 7.1.1 (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 11.1.18 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 11.1.19 must be used to accurately align each slitlet in the final reconstructed image plane.

11.1.22 Cube coaddition

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

\[
llx = 1 + vllx \\
lly = 1 + vly \\
wrx = 64 - vurx \\
ury = 64 - vury
\]
Figure 11.1.3: This image describes the concept of the distortion computation algorithm. The spectra of the fibres are traced, two grids on the distorted and undistorted space are built, then a 2D polynomial transformation is performed. This figure displays only two of the 32 slitlets.

Figure 11.1.4: Cube reconstruction: raw data are resampled using a wavelength map to remove the brick-wall pattern. The slitlets are then stacked into a cube taking slitlet distances and edge positions into account. Each plane of the cube is a monochromatic image of the instrument FOV.

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`. 
11.1.23 Mosaic cube size determination

The size of the cube mosaic is determined as follows. The recipe make reference to the values of the FITS keywords SEQ.CUMOFFSETX/Y (later for brevity CUMX/Y), or to the values set by the user if the utility sinfo_utl_cube_combine is used. The minimum and maximum values of CUMX/Y are determined. The mosaic center position is determined as:

\[
\begin{align*}
\text{offx}_c &= (\text{offx}_{\text{min}} + \text{offx}_{\text{max}})/2 \\
\text{offy}_c &= (\text{offy}_{\text{min}} + \text{offy}_{\text{max}})/2 \\
\text{sizex} &= 2 \times \text{floor}(\text{offx}_{\text{max}} - \text{offx}_{\text{min}} + 0.5) + \text{sizex}_{\text{min}} \\
\text{sizey} &= 2 \times \text{floor}(\text{offy}_{\text{max}} - \text{offy}_{\text{min}} + 0.5) + \text{sizey}_{\text{min}}
\end{align*}
\]

where sizex_{min} and sizey_{min} the minimum X and Y sizes of the input cubes (eventually mosaic) components.

11.1.24 Cubes alignment

To properly align and coadd object cubes observed with different telescope pointings the recipe make reference to the values of the FITS keywords CUMX/Y, or to the values set by the user if the utility sinfo_utl_cube_combine is used. The recipe uses as reference the CUMX/Y values of the first object cube of the list. Then are computed the relative differences:

\[
\begin{align*}
\text{difx}_i &= \text{cumx}_i - \text{cumx}_{\text{ref}} \\
\text{difx}_i &= \text{cumy}_i - \text{cumy}_{\text{ref}}
\end{align*}
\]

Then to each cubes is assigned the following offset:

\[
\begin{align*}
\text{offx}_i &= \begin{cases} 
+2 \times \text{difx}_i & \text{if } \text{mjd}_\text{obs} > 53825 \text{ (1st April, 2006)} \\
-2 \times \text{difx}_i & \text{if } 53421.58210082 < \text{mjd}_\text{obs} \leq 53825 \\
-2 \times \text{difx}_i & \text{else}
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{offy}_i &= \begin{cases} 
+2 \times \text{dify}_i & \text{if } \text{mjd}_\text{obs} > 53825 \text{ (1st April, 2006)} \\
-2 \times \text{dify}_i & \text{if } 53421.58210082 < \text{mjd}_\text{obs} \leq 53825 \\
-2 \times \text{dify}_i & \text{else}
\end{cases}
\end{align*}
\]

The utility sinfo_utl_cube_combine uses user defined offsets following the same conventions. This means that assuming the coordinate values in the FITS header are not correct only of a small quantity, the user may apply the corresponding correction by applying the correct change to the values set in the FITS header.

11.1.25 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.

11.1.26 Sky residual correction

More details on the algorithm to remove residual OH emission from near infrared spectra may be found in [3].

In this release the pipeline can correct for improper sky subtraction; in older releases this caused residuals in on-off sky-subtracted frames. The sky emission is removed from the object observation by subtracting the sky frame from the object frame. This assumes that the two frames have a stable spectral format. However, it has been found that in on-off, object-sky sequences the instrument setting can occasionally have instabilities up to a significant fraction of a pixel. The residual sky features (resembling a P-Cygni profile) can be significantly reduced by using an improved data reduction procedure developed in collaboration with Richard Davies from MPE (Figure 11.1.5, right panel and 11.1.6).

Near infrared airglow emission originates in OH radicals which are created by reactions between ozone and hydrogen high in the atmosphere. Removing the emission lines which result from the subsequent radiative cascade is a crucial part of processing near infrared (1-2.5 µm) spectra.

This problem is relevant as it has been proven that the strongest OH lines, which lie in the H band, have fluxes of the order of 400 phs⁻¹ m⁻² arcsec⁻². This constrasts strongly with the background continuum measurable between lines of only 590 phs⁻¹ m⁻² arcsec⁻² µm⁻¹. This means that even at a moderate spectral resolution R ≈ 3000, like the one of SINFONI, the background level on an OH line can be more than three orders of magnitude higher than that between them.

Due to temporal changes in the absolute flux in OH sky lines exposure times are usually limited to 2-3 minutes. This sets a lower limit to the statistical photon noise. An additional source of problems is the peculiar change of absolute flux in OH lines. Finally the effects of instrumental flexures, which are typical of instrument rotation in a Cassegrain or Nasmyth telescope configuration, may lead to P-Cygni type residuals.

While long-slit spectrographs, using a slit length much longer than the object of interest may allow to prevent such problems after performing an appropriate fit of the residual sky background, in integral field spectrographs like SINFONI, the tipically very limited FOV, impose constraints on the observations and to devise appropriate data reduction techniques.

We describe here an algorithm, first proposed by Richard Davies [18], and implemented in the SINFONI pipeline with a few modifications, introduced for robustness purposes.
In a nutshell the algorithm allows one to find a scaling as function of wavelength that can be applied to a spectrum from a sky cube in order to match it optimally to the sky background in an object cube. This scaling function is then applied separately to the spectrum at each wavelength position of the sky cube, creating a modified sky cube. This modified cube is then subtracted from the object cube.

The scaling function may be found by taking into account that the main contributes to the OH lines is coming from vibrational and rotational transitions, with the first kind contributing the most. Transitions between vibrational bands lie within well defined wavelength limits, and it is only small the amount of overlap between different vibrational transitions. Thus, to a first approximation one may divide the spectrum into sections corresponding to specific vibrational transitions and treat these separately.

The sky residual correction algorithm follows the following steps:

- The wavelength ranges to which correspond the different vibrational and rotational transitions which originate the OH emission lines are defined. The full wavelength range over which to perform the operations involved in the sky lines residual correction can be defined setting parameters `skycor-mask_ws` and `skycor-mask_we`.

- The noise associated to the object frame is calculated.
  - It is determined the histogram of the object intensity values.
  - The histogram distribution is selected to the region nearby the histogram’s peak.
  - The histogram distribution center and its width are found: the center is initially the abscissa value at which the distribution reach its maximum. The width is half of the distance between the center and the abscissa at which the histogram has a minimum. if the parameter `skycor-fit_obj_noise` is set to TRUE those values are refined with a Gaussian fit.

- Object intensities more than two times the object’s noise above the background are flagged.
  - From the object cube and from the flagged object cube are extracted points having wavelengths values within the merged range corresponding to the OH vibrational transitions previously defined.
  - For each cube’s spectra at a given point x,y, are computed the number of object pixels with finite values and the number of flagged pixels and their ratio. If the number of object pixels with finite values is greater than half of the spectral pixels, the given point x,y is considered a good one and the corresponding pixel in a ratio image is set equal to the value of the ratio previously computed.
  - If the number of good pixels is less than one it is issued an error.

- Sky pixels are identified, flagged, and selected as good sky pixels when they make up at least a given fraction (this can be defined by the user modifying the value of the parameter `skycor-min_frac` which is set by default to (80%) of the spectrum.
  - It is computed the ratio of good image pixels divided by the total number of the image pixels.
  - If this ratio is less than a given threshold ratio `skycor-min_frac` (80%) a new selection of object good pixels is performed to increase this number to the threshold ratio.

- The average object and sky spectra are calculated. For each plane in the cube (both object and sky) an average is computed by excluding pixels which are more than three sigmas above the median. This
defines the object and sky intensity. The parameters \texttt{skycor-llx}, \texttt{skycor-lly}, \texttt{skycor-urx}, \texttt{skycor-ury}, allow the user to specify the position where the object is supposed to be in each cube plane. This may allow a more accurate determination of the sky lines residuals and the corresponding spectrum shift with respect to the one of the object.

- The thermal contribution to the sky background is estimated and subtracted from the sky spectrum.

If \texttt{skycor-sub_thr_bkg_from_obj} is set to TRUE the sky background is subtracted from the object spectrum.

If \texttt{stack-sub_raw_sky} is set to TRUE (sky frame subtraction from the object, as by default), \texttt{skycor-sub_thr_bkg_from_obj} should be set to FALSE.

Viceversa if \texttt{stack-sub_raw_sky} is set to FALSE, \texttt{skycor-sub_thr_bkg_from_obj} should be set to TRUE.

To improve accuracy, as the sky spectrum contains many emission lines, we first smooth this spectrum with a running box and then fit a Boltzmann function (see Fig. 11.1.5, left panel).

- Initially the sky spectrum is smoothed on a window of size (\texttt{skycor-sky_bkg_filter_width}).
- Then the smoothed spectrum is fit by a Boltzmann temperature distribution:

\[
fac(\lambda, T) = \frac{1}{\lambda^b} \exp \left( \frac{1}{\sqrt[5]{\lambda T}} \right) - 1
\]

where \(c = 14387.7\), \(\lambda\) is the wavelength in \(\mu\text{m}\) and \(T\) is the temperature in K. Are calculated the minimum and maximum values of this distribution, \(fac_{\text{min}}\) and \(fac_{\text{max}}\).

- For robustness are selected sky spectrum points with intensity greater than -2. This selection is done to prevent contributes from negative region of the sky spectrum, possibly occurring in the case the sky frame is approximated by the closest object frame in the observation sequence, as in the case in which no sky frame is observed in the same template (or no sky frame is provided in the set of frames). Then are furtherly selected sky points whose intensity is different from the median but less than ten sigmas.

- A Levenberg-Marquardt fit of the Boltzmann distribution:

\[
B = a_0 \times a_1 \times fac(\lambda, T)
\]

is performed to this selected sky spectrum. The fit is repeated \texttt{skycor-niter} times to get three-digit accurate values of the Boltzmann temperature and the constants \(a_0\) and \(a_1\).

- If the fit fails (this may happen sometimes in case of a very flat sky spectrum), a simple uniform fit of the sky spectrum is performed.

- Finally the fitted thermal background is subtracted from the sky spectrum.

- To properly compute the shift between object and sky only wavelengths corresponding to vibrational or rotational transitions are considered. This removes the effect from improper spectrum features.

- The pixel shift between the sky and the object spectra is computed by cross correlating the two spectra and the sky spectrum and the sky cube are correspondingly shifted.

- It is computed the mean of the sky spectrum intensity. If the mean is negative the spectrum is flipped.
Figure 11.1.5: The fit of a sky spectrum is shown on the left panel. The sky background (black) is first smoothed (green) and then fit by a Boltzmann function (red). On the right panel we compare un-corrected (black) and corrected (red) sky subtracted object spectra. Rotational transitions affect the spectrum quality by only 1-5%. Not only can one see 1-0S(1) at 2.12 µm and Brγ at 2.17 µm, but now the H2 1-0Q(1) and 1-0Q(3) emission lines are clearly seen longward of 2.4 µm, and several CO band-heads are also visible at 2.3-2.4 µm.

- Too low sky intensities are flagged.
- Line picks are identified.
- A Gaussian fit centered at each line pick is performed to compute the line wavelength position of the object and the sky lines, the line object and sky intensity and its associated error. Only lines of width greater than \texttt{skycor-line_hw} are considered.
- A kappa sigma clipping of the differences of the line positions as determined in the object and in the sky spectrum is performed.
- A uniform function is fit to the residual line position differences.

If the user has estimated the sky-object spectral shift independently, or would like to overwrite the value computed by the pipeline, can set this quantity (in pixel units) with the parameter \texttt{skycor-pshift}.

- The sky spectrum and the sky cube are correspondingly shifted.
- The scaling factor that is applied to the sky emission lines in order to remove them from the object spectrum is computed.

The object and sky sub-spectra are extracted over a wavelength range corresponding to each vibrational transition. For each wavelength range:

- Are used only pixels where the spectrum is finite.
- Sky lines are identified. Line and continuum regions are disentangled.
– If the line and continuum regions contain enough points ($2^{*}\text{skycor-line\_hw}$) it is computed the line ratio between object and sky line spectra. Two methods are possible: amoeba fit ($\text{skycor-scale\_method}=0$) or maximum likelihood ($\text{skycor-scale\_method}=1$). After a first line ratio computation it is performed a kappa sigma clipping of spectrum outliers, and eventually the fit is repeated.

In this way it is computed a wavelength dependent scaling factor. In first approximation, this factor is derived from the line contributions only due to vibrational transitions.

If the parameter $\text{skycor-rot\_cor}$ is set to TRUE, then the recipe computes a scaling factor based on the contribution of the rotational transitions. Later the two factors are combined and the shifted sky spectrum is accordingly rescaled and subtracted from the object spectrum, to obtain an object spectrum cleaned of sky lines (see Fig. 11.1.5, right panel).

• The shifted and scaled sky cube is finally subtracted from the object cube. The resulting cube is thus corrected for the sky residuals (see Fig. 11.1.6, right panels).

11.1.27 Efficiency computation

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

**Method used in $\text{sinf0\_utl\_eff}$ recipe** The efficiency at a given wavelength $\lambda$ is computed as:

$$EFF(\lambda) = \frac{\text{COUNTS\_BKG}(\lambda) \cdot 10^{\Delta PEXT(\lambda)-(AIRPRIM-AIRMASS(\lambda))} \cdot GAIN \cdot PEPH(\lambda)}{EXPTIME \cdot TEL\_AREA \cdot PREF(\lambda)} \cdot \text{factor}$$

Where $\text{COUNTS\_BKG}(\lambda)$ is the extracted standard star spectrum as observed by SINFONI, corrected for the contribute from the sky background, at a given wavelength $\lambda$, $PEXT(\lambda)$ is the atmospheric extinction value, $AIRM ASS(\lambda)$ is the airmass at which the object (star) was actually observed, $AIRPRIM$ is a parameter to indicate if the efficiency is be computed at airmass=0 (no-atmosphere) or at a given value (usually the one at which the reference standard star spectrum may be tabulated). $GAIN$ indicates the SINFONI detector’s gain (assumed to be 2.42), and $PEPH(\lambda)$ is the energy of one photon ($PEPH(\lambda) = \frac{10^7}{\lambda \cdot 1.986 \cdot 10^{19}} \cdot J \cdot um^{-1}$), $EXPTIME$ is the total exposure time in seconds, $TEL\_AREA$ is the UT telescope collecting area ($51.2 \cdot 10^7 \cdot cm^2$), $PREF(\lambda)$ is the flux calibrated spectrum of the reference source. $\text{factor}$ is a multiplicative number that corrects for the fact that in the previous formula some quantity has been expressed in different units.

**Standard method** In order to use in operations the previously described method the pipeline requires as input a catalog of standard star fluxes observed during SINFONI operations. We are in the process of building it. Thus it is still valid at this stage to evaluate the efficiency using the following, less accurate method (that assumes the observed standard star emits like a black body).

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 cm²,
  (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. \(10^{\frac{M_{\text{mag}}}{2.5}}\),
  (e) dividing by the dispersion (Å per pixel),
  (f) multiplying by the energy of the photon (\(\frac{hc}{\lambda}\)).

Figure 11.1.6: Comparison between un-corrected (left) and corrected (right) object cubes. On the top are displayed the cube plane corresponding to 2.151 \(\mu\)m for a case in which no sky frame is present in input. On the bottom are displayed the cube plane corresponding to 1.915 \(\mu\)m for a case in which a sky frame is present. Images on the left have the same intensity scale of images in the right.
This corresponds to multiplying the spectrum by the following factor:

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

where the following numerical values for the physical constants are used:

- \( h = 6.62618 \times 10^{-34} \text{ Js} \) (Planck constant)
- \( c = 2.998 \times 10^8 \text{ m/s} \) (Speed of Light)
- \( k = 1.3807 \times 10^{-23} \text{ J/K} \) (Boltzmann constant)

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

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

- The ratio between the flux from a blackbody at wavelength \( \lambda \) and the flux from the blackbody with \( T=10,000 \text{ K} \) at wavelength \( \lambda_c \) is computed.
- This curve is scaled so that the value at the central flux is \( f_0 \), were \( f_0 \) is for a 0\textsuperscript{th} magnitude star in J, H, K and "HK":

<table>
<thead>
<tr>
<th>band</th>
<th>( \lambda_c (\mu\text{m}) )</th>
<th>( f_0 ) (ergs/s/cm\textsuperscript{2}/Å)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>1.225</td>
<td>3.11e-10</td>
</tr>
<tr>
<td>H</td>
<td>1.675</td>
<td>1.15e-10</td>
</tr>
<tr>
<td>K</td>
<td>2.175</td>
<td>4.10e-11</td>
</tr>
<tr>
<td>HK</td>
<td>1.950</td>
<td>8e-11</td>
</tr>
</tbody>
</table>

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.

11.1.28 Response computation

11.2 Instrument response determination

Initially the flux table corresponding to the observed standard star is extracted from an input catalogue. We correct the wavelength scale of the reference spectrum (which is a stellar model spectrum) to the same radial velocity as the observed spectrum and then interpolate the reference spectrum to the same steps as the observed spectrum. For VIS and NIR data we apply a correction of the telluric absorption by looking for the best fitting spectrum within a catalogue of telluric model spectra. The best telluric model is the one that minimises the mean of a “correction” spectrum computed in the following intervals:

This spectrum is obtained via the following steps:
1. adjust telluric model spectrum to wavelength scale of observed spectrum (small shifts may occur due to imperfect wavelength calibration)

2. convolve the telluric spectrum to the same resolution as the observed spectrum

3. divide the observed spectrum by the shifted and convolved telluric spectrum

4. fit the continuum of this ratio at pre-defined wavelength points

5. divide ratio by fit

Since the model spectra are provided for a large range of water vapor content, but only for few values of CO2, we mask CO2 regions when fitting the response later. The response is computed by dividing this telluric-corrected model spectrum of the standard star (in $\text{erg cm}^{-2}\text{s}^{-1}\text{Å}^{-1}$) by the 1D extracted observed spectrum of the standard star corrected for gain, exposure time, atmospheric extinction (the atmospheric extinction table is interpolated to get the same binning) and for telluric absorption. To reduce the noise of the resulting ratio spectrum we apply a median filter of eleven pixels half width. Then we apply a cubic-spline fit to the points defined in RESP_FIT_POINTS_CAT_ARM to get the final response. Regions of very high telluric absorption (between J and H and between H and K in the NIR arm) or CO2 absorption (see above) are masked and interpolated.

The response is obtained with the following equation:

$$\text{Response [erg/e}^{-} / \text{cm}^2] = \frac{\text{STD}_{\text{flux table}}[\text{erg/s/cm}^2/\text{Å}^{-1}] \times \text{exp time}[\text{s}] \times \text{gain}[\text{ADU/e}^{-}]}{\text{STD}_{\text{observed}}[\text{ADU/pix}[\text{Å}]] \times 10^{(0.4 \cdot \text{airmass} \cdot \text{ext})}}$$

(16)

Where we have taken into account of the size of a pixel. The response is derived from the order merged flux standard spectrum and applied on the merged science spectrum.

### 11.2.1 Flux calibration

If the user provides the instrument response and the atmospheric extinction tables in input of a science recipe, the merged 2D and 1D spectra are then flux calibrated. This operation is performed by first dividing the observed spectra by exposure time and the detector gain and correction for atmospheric extinction, and then multiplying those by the instrument response.
\[ I_{\text{erg/s/cm}^2/\text{Å}} = \frac{I_{\text{ADU/pixel/Å}}} {\text{gain[ADU/e-]} \times \text{Exptime[s]} \times \text{bin_size}} \times \frac{\text{Response[erg/e^-/cm}^2]} {10^{(0.4 \cdot \text{airmass \cdot ext})}} \] (17)

Where we evidence the fact we correct for the size in pixels of the integration bin.

11.2.2 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.

As the SINFONI possible FOV are either too narrow or too large for most of the calibration stars used to measure the instrument Strehl, it has been defined a special template in which the same PSF standard is observed first in the 0.025 mas scale, to accurately determine the object image Point Spread Function maximum, and the with the 0.100 mas scale to determine the star flux and sky background levels.

Normally the Strehl is determined as ratio:

\[ \text{Strehl} = \frac{I_{\text{max}}/\text{flux}}{PSF_{\text{max}}/\text{flux}_{\text{PSF}}} \]

Where \( I_{\text{max}} \) is the maximum intensity of the object (sky background subtracted), and \( \text{flux} \) the corresponding flux, while \( PSF_{\text{max}} \) is the maximum intensity of a corresponding theoretical PSF corresponding to a telescope with the same diameters for the primary and secondary mirrors, at a given wavelength and in a given wavelength range, at a given pixel scale, with a given size.

This formula is applied on both images of the PSF standard taken with two camera pixel scales (25 and 100 mas).

The SINFONI pipeline implements the same algorithm as in the Paranal Multi Strehl Meter GUI.

Initially it is performed a 2D Gaussian fit to the image resulting averaging the STD star cube along the wavelength (z) axis. This fit determines the approximate position of the STD star image, the FWHM along the major and minor axes and the orientation of the elliptic isophotes. Then the sky background is subtracted by taking the median value of the four image plane’s corner regions, \( 8 \times 8 \) pixels each. Finally it is determined the maximum of the resulting sky corrected image.

The star flux is defined as the flux of the STD star above the sky background.

Then it is built the instrument PSF by taking into account also of the possible anamorphysm and occultation introduced by the secondary mirror and the PSF’s maximum and flux are determined.

11.2.3 Standard star position detection

The recipe sinfo_rec_jitter 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.

### 11.2.4 Atmospheric refraction algorithm

**Creating the calibration table** The shift in the North-South direction as a function of airmass and wavelength is computed on a set of standard stars observations for each band (J, H, K, H+K) and spatial scales (25, 100, 250) (See Fig.11.2.1).

![Figure 11.2.1: Offset by North-South direction (red) and the polynomial fit (blue) (J band, 0.25 mas).](image1)

![Figure 11.2.2: Offset by Y (North-South) direction in HK band 250 spatial scale.](image2)

The corresponding shifts measured for each cube plane along the EAST-WEST direction have values mostly in the range (-0.25, +0.25) pixels as shown by Fig.11.2.3.

The computation of the shift in the North-South direction is done for each plane by the 2D Gaussian fit of the object PSF on the intermediate cubes and is a function on NORTH-SOUTH direction angle (ABSROT START and ABSROT END parameters in the fits header). The coefficients of the corresponding 2D polynomial fit are saved into a calibration table (for example ATM_REF_CORR_025_H.poly.fits, and in similar tables for other bands and spatial scales).

For the 250 mas scale this calibration procedure is very complex, because of large fluctuations of the offset in the North-South direction (See Fig. 11.2.2), and anyway the shift in pixels is of the order of 1 spatial bin.

**Size of the each intermediate cube** is 64 x 64 pixels. In our dataset we have observed the shift could be up to 7 pixels (J Band, 25 mas). The current implementation can lead to a usually small loss of some information at the edge of some planes after correction (usually at the beginning and the end of the wavelength range), as we decided to keep the cube size fixed. A solution of this not very frequent problem may be require over-sizing each cube component but would require changing the mosaic reconstruction algorithm, and imply as a side
11.2.5 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.

11.2.6 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 ($\text{FWHM}=0.5*(\text{FWHM}_X+\text{FWHM}_Y)$) of the observed standard star.

11.3 Recipes

11.3.1 Detector linearity and non-linear bad pixel map determination: sinfo_rec_detlin

The sinfo_rec_distortion recipe computes the detector non-linearity coefficient as described in 11.1.9 (using methods 1 and 2). The detector gain is also computed as described in 11.1.11. 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 11.1.9).

11.3.2 Detector linearity and gain computation: sinfo_rec_lingain

We remind the user to 11.1.5 and 11.1.4.

11.3.3 Master dark and bad pixel map determination: sinfo_rec_mdark

A set of input raw dark frames is stacked in a cube. An average with rejection (parameters $bp\_noi.se.l ow\_rejection$ and $bp\_noi.se. h i g h\_rejection$), yields 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 $dark\_h i g h\_rejection$) 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 11.1.12. 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 11.1.13.

11.3.4 Master dark determination: sinfo_rec_mdark_detmon

We remind the user to 11.1.1.

11.3.5 Master flat and threshold pixels (bad pixel map) determination: sinfo_rec_mflat

- The input flat field frames are stacked. An average with rejection (parameters $lamp\_flats. low\_rejection$ and $lamp\_flats. h i g h\_rejection$) 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 $lamp\_flats.l lx$, $lamp\_flats.l ly$, $lamp\_flats.urx$, $lamp\_flats.ury$, 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 \([\text{clean_mean} - \text{lamp_flats.mean_factor} \times \text{clean_stdev}]\). Else no filter is applied. This results in a reference image. A median filter with a radius equal to \(\text{clean_stdev} \times \text{lamp_flats.factor}\) 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_fpnymax1`, 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 (\(\text{lamp_flats.qc_thresh_min}\) and \(\text{lamp_flats.qc_thresh_max}\)).

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 \([\text{bp_norm.min_cut}, \text{bp_norm.max_cut}]\) 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 (\(\text{bp_norm.llx}, \text{bp_norm.lly}, \text{bp_norm.urx}, \text{bp_norm.ury}\)) rectangular zone is determined in a way that the extreme low (\(\text{bp_norm.low_cut}\)) and the extreme high (\(\text{bp_norm.high_cut}\)) 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 (\(\text{bp_norm.low_rejection}, \text{bp_norm.high_rejection}\)) 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 \(\text{bp_norm.mean_factor} \times \text{bp_norm.sigma}\) 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 11.1.10.

- 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.
11.3.6 Optical distortion and slitlet distances determination: sinfo_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 11.3.7 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_displ1`, `distortion.guess_displ2`, `distortion.min_diff_col_tilt_int`, `distortion.half_width`, `distortion.sigma`) as described in 11.1.14.
  - A wavelength calibration is performed (the result depends on the following parameters: `distortion.guess_displ1`, `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 11.1.15.
  - A possible line shift is computed as described in 11.1.16. 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 11.1.18.
  - The slitlet distances are computed with a north-south test as described in 11.1.19. 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) as described in 11.1.20.

- 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: \texttt{north\_south\_test.half\_width}, \texttt{north\_south\_test.fwhm}, \texttt{north\_south\_test.min\_diff}, \texttt{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.

11.3.7 **Wavelength solution determination: sinfo\_rec\_wavecal**

**Frame stacking** This macro step is common to several recipes: sinfo\_rec\_distortion, sinfo\_rec\_wavecal, sinfo\_rec\_jitter.

- 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 \texttt{stacked.flat\_index==TRUE} the result is flat-fielded using the input master flat field.

- Static bad pixels are indicated (\texttt{stacked.mask\_ind=1}) or not (\texttt{stacked.mask\_ind=0}). If \texttt{stacked.mask\_ind=1}:
  - If \texttt{stacked.ind\_index} is FALSE bad pixels are interpolated over a given range defined by \texttt{stacked.mask\_rad} and using the information on the slitlet edge positions contained in the SLIT\_POS frame.
  - If \texttt{stacked.ind\_index} is TRUE bad pixels are indicated by multiplying the frame by the bad pixel map.

- If \texttt{stacked.warpfix\_ind==TRUE} the distortion is corrected using a given kernel (\texttt{stacked.warpfix\_kernel}).

- Finally some quality control is performed on the resulting stacked frame by monitoring the number of saturated pixels (\texttt{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 \texttt{wavecal.slitpos\_bootstrap\_switch} to TRUE. For the sake of robustness, it is suggested to use \texttt{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 \texttt{wavecal.calib\_indicator == TRUE} and \texttt{wavecal.wave\_map\_ind == TRUE} the input arc lamp reference line list is loaded.
• If one needs to do the wavelength calibration (\texttt{wavecal.calib\_indicator} == TRUE and \texttt{wavecal.wave\_map\_ind} == FALSE), the recipe finds the emission lines present in each arc lamp spectrum as described in 11.1.14.

• Then the dispersion relation is determined as described in 11.1.15.

• The overall line shift between the solution found from the fit as described in 11.1.16 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 \texttt{wavecal.write\_coeff\_ind}==TRUE the polynomial fit coefficients are saved in a FITS table.

• If \texttt{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 (\texttt{wavecal.wave\_map\_ind}==TRUE) and a dispersion relation does not need to be found (\texttt{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 \texttt{wavecal.calib\_indicator}==TRUE and \texttt{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 11.1.18.

In the end the recipe generates also a resamples the arc lamp stacked frame to control the quality of the performed wavelength calibration.

11.3.8 Science observations: \texttt{sinfo\_rec\_jitter}

• 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 11.1.25. Each pair is stacked.

• Each stacked frame is resampled using the information from the input calibration image (WAVE_MAP) doing a polynomial interpolation with \texttt{objnod.n\_coeffs} parameters to generate an output 3D frame as described in 11.1.21. 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 \texttt{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.

• If the user provides in input a reference atmospheric refraction correction catalog frame (part of this kit distribution) tagged as ATM\_REF\_CORR, the recipes applies the atmospheric correction. This correction is important in particular in 25 mas setting where otherwise, after cube reconstruction, the object PSF center may move up to several spaxels (see 11.2.4).
• The 3D frame is refined using different methods (\texttt{objnod.fine\_tuning\_method}). Possible values are P (polynomial of order \texttt{objnod.order}), S (Spline). The 3D frame is stretched in Y to make it appear as a parallelepipedus with a square cross-section. We usually call this a “3D cube”.

• If \texttt{objnod.sky\_cor} is TRUE and if the recipe parameter \texttt{product-density} is set to 3 sky line residuals are corrected as described in 11.1.26. We suggest to use this setting only after having verified that otherwise the object spectrum contains sky line residuals. A more personalized correction of sky line residuals would follow the following steps: reduce the data without correcting for residuals and setting \texttt{product-density} to 2. Then correct each corresponding obj-sky pair with \texttt{sinfo\_utl\_skycor} and proper parameter setting. Finally use \texttt{sinfo\_utl\_cubes\_combine} to reconstruct a cube mosaic.

• If \texttt{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 \texttt{objnod.kernel\_type} as described in 11.1.22 possibly scaling the sky background (if \texttt{objnod.scale\_sky} == TRUE) and doing a kappa sigma clipping (if \texttt{objnod.kappa\_sigma} == TRUE) of intensity outliers.

11.3.9 STD star data reduction: \texttt{sinfo\_rec\_jitter}

The initial data reduction is the same as the one of a normal science frame (\texttt{sinfo\_rec\_jitter}). Then if the input object is a PSF or flux standard star:

• 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 \texttt{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 \texttt{std\_star.conversion\_index}==TRUE a conversion magnitude to counts/seconds is performed. The intensity conversion factor (magnitude per count/s) is determined by fitting 2D Gaussian to the collapsed image of the observed standard star of known brightness. Then the resulting Gaussian is integrated and the counts are divided by the exposure time. The integration is made on a rectangular box of a given size (half box size is 16 pixels), centered at a given position (llx=lly=8). In case of more than an input standard star, it is computed a clean mean of the clean factors obtained by each standard star.

• The instrument efficiency is determined as described in 11.1.27.

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

11.3.10 PSF data reduction: \texttt{sinfo\_rec\_jitter}

The initial data reduction is the same as the one of a normal science frame. Then if the input object is a PSF or flux standard star, the main PSF standard parameters, the instrument Strehl and the encircled energy are determined as described in 11.2.2 and 11.2.5.
11.3.11 PUPIL data reduction: sinfo_rec_pupil

The initial data reduction is the same as the one of a normal science frame. Then the X and Y centroid of the cube’s median image are determined.
A Workflow technicalities

This section will present some detailed explanations of the FLAMES-UVES Reflex-based workflow.

A.1 Data requirements and optional inputs

In order to run the workflow with a given data set, a number of calibrations and science data must be present. The workflow functions with the concept of science data sets: a set of files that contain a coherent set of science and calibration data.

The minimum coherent fiber mode data set is composed of:

- Object source science frame.
- A set of linearity flat frames.
- A set of frames to determine the detector’s distortions: 74 frames obtained illuminating the slitlets by a fibre fed by a flat lamp, two flats (lamp-on and lamp off) frames and two wavecal (lamp-on and lamp off) frames.
- A set of dark frames that can be applied to the science frame.
- Two flat field frames obtained with a flat calibration lamp (respectively switched on and off).
- Two wavelength calibration frames obtained with an arc lamp (respectively switched on and off).
- Static calibrations: a line reference tables, a data reduction setup table, a slitlet position table.

The DataSetChooser shows a nice tree diagram with all the associations.

A workflow accepts science and calibration data, as delivered to PIs in the form of PI-Packs (until October 2011) or downloaded from the archive using the CalSelector tool\(^{11}\) and organises them into DataSets, where each DataSet contains one science object observation (possibly consisting of several science files) and all associated raw and static calibrations required for a successful data reduction.

For a detailed explanation about how these calibrations are organised and associated, please refer to Section A.3.

A.2 Overall layout

The workflow structure has several parts:

- Top area. Contains annotations about the workflow and the main parameters to setup the workflow. Note that not all the parameters of the workflow are set here: each actor has its own parameters that are set individually.

\(^{11}\)http://www.eso.org/sci/archive/calselectorInfo.html
• Bottom area. This contains all the actors that actually execute or perform a workflow action. It is also composed of several parts:

– Initialisation actors on the left side. These actors prepare the rest of the workflow to start with the data reduction. It includes setup of required intermediate variables, the data organisation, the data selection and the routing of the data. Please note that the data organisation and selection is performed only once, while the routing is performed once for each data set that has been selected.

– Recipe execution. The middle part of the bottom area contains the actors that execute pipeline recipes. This is basically where the logic of the data reduction chain is implemented.

– Closing of the reduction for this data set. The right side of the workflow contains actors that perform the last actions needed after the reduction of a data set. This includes renaming of the final science data, an interactive data display (disabled by default) and some housekeeping of variables.

• Composite actors. Some of the actors are composite of other actors. That means that they contain another workflow inside. To inspect a composite actor, right-click on it, and select Open Actor.

A.3  OCA rules

The OCA rules are the mechanism used to Organise, Classify and Associate the data. These rules are stored in a human-readable file which contains several sections, one for each of the required tasks.

The DataOrganiser is the component that makes use of these rules. The parameter Oca File specifies the proper OCA rule file to use. It is advised to check that it points to the right location (although the installation procedure should take care of that). Figure A.3.1 shows the parameter to inspect to check that the OCA rules are ok.

Figure A.3.1: Parameter to change for the OCA file, used for classification, grouping and association of data.

Here we present a brief summary of the meaning of the current OCA rules provided with the workflow:

• The raw data is classified according to the DPR keywords.

• The products of the recipes are classified according to the PRO.CATG keyword.

• The raw darks are grouped by detector DIT and observation template.

• The raw flats are grouped by filter type, on-sky element resolution size and observation template.

• The raw wavecal frames are grouped by filter type, on-sky element resolution size and observation template.
• The science data are grouped by filter type, on-sky element resolution size, detector DIT, and observation template.

• All the required calibrations are directly associated to the science data. If several groups of calibration data match the rules, then the closest in time is chosen.

• The master dark is associated if the detector DIT match.

• The master slit flat are associated if filter type, on-sky element resolution size match.

• The wavelength solution is associated if filter type, on-sky element resolution size match.

A.4 Workflow fine tuning and hints

We have collected several hints to fine tune and exploit all the capabilities of the workflow.

• Check disk space before starting to reduce the data. The reduction of all the demo data sets will need at least 4 GB. However, at least 7 GB are recommended. This disk space requirement applies to directories pointed to by the TMP_PRODUCTS and END_PRODUCTS_DIR variables.

• Should the user find data reduction problems, the recipe parameter debug allows to increase recipe verbosity. Change this parameter for the recipe of interest.

• The ProductRenamer can be setup to create the desired filenames. Use the Rename keywords parameter to change the renaming scheme. It is possible to use keywords from the the header or literal strings (quoted by ”). Currently, there is only one occurrence in the workflow of the ProductRenamer, just after the science recipe. However, this actor can be placed in several places in the workflow (for instance, to store in the final directory the master flat).

• All the intermediate products created by the workflow are stored in TMP_PRODUCTS_DIR directory. The subdirectory structure is shown in Figure A.4.1. For each recipe instance there is a subdirectory, and inside this there is another subdirectory with the timestamp of the execution time.
Figure A.4.1: Structure of the REFLEX_PRODUCTS directory.
B  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 B.1. It is recommended reading through Section B.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 [www.eso.org/pipelines](http://www.eso.org/pipelines), for users who are not familiar with the installation of software packages.

B.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),
- Mac,

using the GNU C compiler (version 3.2 or newer). Gasgano is installed by the install_pipeline script and is supported provided the user has a proper installation of the Java Development Kit (version 1.6.0 or newer) and has set the JAVA_HOME environment variable properly during the kit installation. It is not sufficient to have the Java Realtime Environment to have successfully installed the libgasganocpl.* libraries needed to properly interface Gasgano with the installed sinfoni recipes. We recommend the user to look carefully at the log produced during installation and possible warnings.

B.2  Building the SINFONI pipeline

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

B.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 7.1.1 or newer is required. The CPL distribution can be obtained from www.eso.org/cpl.

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 5.2, page 20). At least one of them must be installed. The EsoRex and Gasgano packages are available at www.eso.org/cpl and www.eso.org/gasgano respectively.

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

### B.2.2 Compiling and installing the SINFONI pipeline

The SINFONI pipeline distribution kit 3.3.2 contains:

- sinfo-pipeline–manual-3.3.2.pdf: The SINFONI pipeline manual
- install_pipeline: Install script
- cfitsio3360.tar.gz: CFITSIO 3360
- cpl-7.1.1.tar.gz: CPL 7.1.1
- esorex-3.13.2.tar.gz: esorex 3.13.2
- gasgano-2.4.8.tar.gz: GASGANO 2.4.8
- sinfony-3.3.2.tar.gz: SINFONI 3.3.2
- sinfony-calib-3.3.2.tar.gz: SINFONI calibration files 3.3.2

Here is a description of the installation procedure:

1. Change directory to where you want to retrieve the SINFONI pipeline recipes 3.3.2 package. It can be any directory of your choice but not:

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

2. Download from the ESO ftp server, 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-3.3.2.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,
   
   ```bash
   cd sinfo-kit-3.3.2
   ```
   
   it is possible to perform a simple installation using the available installer script *(recommended)*:
   ```bash
   ./install_pipeline
   ```
   
   Note: on recent Mac OS in order to properly install the kit it may be useful to set the following environment variable:
   ```bash
   export JAVA_HOME=/System/Library/Frameworks/JavaVM.framework/
   ```
   
   (beware: the execution may take a few minutes on Linux and several minutes on SunOS).
   
   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.
C Abbreviations and acronyms

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

ANSI, 96
AO_PERFORMANCE, 62
ASCII, 96

CalibDB, 96
CPL, 96
   web page, 11

Data reduction cascade, 40, 42
Definition
   ANSI, 96
   ASCII, 96
   CalibDB, 96
   CPL, 96
   DFS, 96
   DMD, 96
   DRS, 96
   ESOREX, 96
   FITS, 96
   FOV, 96
   FPN, 96
   GUI, 96
   OB, 96
   PSO, 96
   QC, 96
   RON, 96
   SINFONI, 96
   SOF, 96
   SPIFFI, 96
   UT, 96
   VLT, 96
Detector gain, 46
Detector non linearity, 46
DFS, 96
DMD, 96
DRS, 96

ENCIRCLED_ENERGY, 62
ESO, 96
ESOREX, 96
EsoRex
   Configuration file, 23
   Example

Help, 23
Introduction, 22
Parameters listing, 23
Pipeline recipe configuration, 23
Pipeline recipe execution, 24
Pipeline recipe help, 23
Pipeline recipe man page, 23
Pipeline recipes listing, 23
Set of frames definition, 22
Syntax, 22
Usage, 22
web page, 11

FITS, 96
Fixed Pattern Noise, 47, 51
FOV, 96
FPN, 96
Gasgano
   File classification, 18
   File selection, 18
   Introduction, 18
   Pipeline Recipe execution, 19
   Usage, 18
   web page, 11
GUI, 96
Hot pixels, 48
IFS, 40
Input data, 41
Integral Field Spectroscopy, 40
Lamp flux, 50
MACAO, 15
OB, 96
Optical distortions coefficients, 54

Pipeline
  Cookbook, 17
  Example
    sinfo_rec_detlin, 27
    sinfo_rec_distortion, 27
    sinfo_rec_jitter, 29, 30
    sinfo_rec_mdark, 26
    sinfo_rec_mflat, 28
    sinfo_rec_wavecal, 28
  Input data, 41
  Overview, 11, 13
  Purposes, 13
  Recent improvements, 14
  Recipe execution, 24
  Recipes
    sinfo_rec_detlin, 17, 42, 45
    sinfo_rec_distortion, 17, 43, 52
    sinfo_rec_jitter, 17, 43, 61
    sinfo_rec_mdark, 17, 42, 47
    sinfo_rec_mflat, 17, 43, 49
    sinfo_rec_wavecal, 17, 43, 57
    sinfo_utl_bp_mask_add, 17, 65
    sinfo_utl_cube2ima, 17, 66
    sinfo_utl_cube2spectrum, 17, 67
    sinfo_utl_cube_arith, 68
    sinfo_utl_cube_arithm, 17
    sinfo_utl_cube_combine, 17
    sinfo_utl_ima_arith, 17, 66
    sinfo_utl_ima_skycor, 17
    sinfo_utl_ima_skymap, 17
    sinfo_utl_ima_spectrum_divide_by_blackbody, 17
    sinfo_utl_ima_spectrum_wavelength_shift, 17
    sinfo_utl_skymap, 64
    sinfo_utl_spectrum_divide_by_blackbody, 69
    sinfo_utl_spectrum_wavelength_shift, 70

PSO, 96

QC, 96

QFITS
  web page, 11

Raw data, 33
  Arc frames, 36

Arc frames used to compute optical distortions, 35
Dark current, 34
Detector linearity Flat Field, 35
Flat Field, 34
Flat Field used to compute optical distortions, 35
PSF standard star frames, 36
Science frames, 37
Sky frames taken with PSF standard star frames, 37
Sky frames taken with science frames, 38
Sky frames taken with telluric standard star frames, 37
Telluric standard star frames, 37

Read Out Noise, 47

Reference data
  DRS setup table, 39
  Line reference table, 39
  Reference bad pixel map, 39

Reference pixels shift, 54

RON, 96

Set of frames
  Example
    sinfo_rec_detlin, 24
    sinfo_rec_distortion, 25
    sinfo_rec_jitter, 25, 26
    sinfo_rec_mdark, 24
    sinfo_rec_mflat, 25
    sinfo_rec_wavecal, 25

SINFONI, 96

Image Slicer, 16
Instrument description, 15
MACAO, 15
Pipeline publication, 11
Pipeline User Manual, 11
Quality Control web page, 11
Raw data, 33
Science Verifications web page, 11
SPIFFI, 15
User Manual, 11
web page, 11

SLIT_POS, 58
Slitlets distances, 54
SOF, 96
SPIFFI, 15, 96
STD_STAR_SPECTRA, 62
Strehl, 62
Threshold pixels, 51
UT, 96
VLT, 96
WAVE_COEF_SLIT, 58
WAVE_FIT_PARAMS, 58
Wavelength calibration error, 59
Wavelength calibration offset, 59
Wavelength calibration solution, 59