EUROPEAN SOUTHERN OBSERVATORY

Organisation Européenne pour des Recherches Astronomiques dans l'Hémisphère Austral

Europäische Organisation für astronomische Forschung in der südlichen Hemisphäre

VERY LARGE TELESCOPE

GRAVITY Pipeline User Manual

VLT-MAN-ESO-19500-XXXX

Issue 1.8.0

Date 2024-12-03

Prepared:	ESO GRAVITY Pipeline Team	2024-12-03	
	Name	Date	Signature
Approved:	P.Ballester, M. Peron		
	Name	Date	Signature
Released:	M. Peron		
	Name	Date	Signature

This page was intentionally left blank

Change record

Issue/Rev.	Date	Section/Parag. affected	Reason/Initiation/Documents/Remarks
0.4	01/10/2016	All	Draft version
0.5	01/10/2017	All	Updated to version 1.0.7 of the pipeline
0.8	01/05/2019	All	Updated to version 1.3.0 of the pipeline

This page was intentionally left blank

Contents

1 Introduction

This manual is a complete description of the data reduction recipes implemented for the GRAVITY pipeline, reflecting the status of the GRAVITY pipeline as of now.

The main part of the document is focused on the main feature of the pipeline useful to the science user of GRAVITY. The detailled appendixes may be more useful to ESO staff for the purpose of long term re-calibration and data quality control.

1.1 Reference and applicable documents

[1] *OIFits Explorer*. *https://www.jmmc.fr/oifitsexplorer*. 64

[2] *QFitsView*. *https://www.mpe.mpg.de/ ott/QFitsView/*. 64

[3] ESO/SDD/DFS, *https://www.eso.org/cpl/*. *CPL home page*. 108

[4] T. A. Pauls, J. S. Young, W. D. Cotton, and J. D. Monnier. A Data Exchange Standard for Optical (Visible/IR) Interferometry. *PASP*, 117:1255–1262, November 2005. 16, 69, 70, 111

[5] C.Sabet P.Ballester. *VLTI Data Interface Control Document*. ESO, 1.0 edition, 3 June 2002. VLT-SPE-ESO-15000-2764. 64, 67

2 GRAVITY Instrument Description

The GRAVITY instrument has been developed under ESO contract by the GRAVITY consortium.

The main purpose of the instrument is to measure the angular separation between stars in the vicinity of the Galactic Center black hole, and even the position and motion of the infrared emission of the black hole itself during flares.

GRAVITY recombines the light beams collected by either the four 8.2 m Unit Telescopes or the four VLTI Auxiliary Telescopes. The general principle of the instrument is to recombine the light coming from one (single field mode) or two (dual field mode) astronomical targets in two different beam combiners: the Fringe Tracker (FT) and the Science Combiner (SC). In single field mode, the light from the target is split between the FT and SC channels using a beam splitter, while in dual field mode a mirror directs the light from each target to their respective beam combiners. The FT is optimized to record fringes at very high frequency (up to 1 kHz), in order to measure and compensate in real time the atmospheric piston effect using a dedicated actuator in the instrument. As the observed targets are both within the atmospheric isoplanetic patch, the correction of the atmospheric piston by the FT stabilizes the fringes of the SC channel. This gives the possibility to integrate for up to several tens of seconds, and therefore reach a high sensitivity, even at relatively high spectral dispersion, on the SC channel.

The properties of the interference fringes are measured separately in the FT and SC beam combiners. GRAVITY measures the classical interferometric observables of any source, as the previous VLTI instruments (VINCI, MIDI, AMBER and PIONIER). The FT spectral resolution is limited to 5 spectral channels over the K band. The SC has three available spectral resolutions: low (20), medium (500) and high (4000), providing approximately 10, 200 and 1800 spectral channels over the K band.

The phases of the SC beam combiner are referenced to the FT using a metrology system that encompasses the optics of the VLTI up to the secondary mirror of the telescopes. Thanks to this link between the two beam combiners, GRAVITY provides very accurate measurements of the differential position of the fringe pattern obtained, for each baseline, between a reference star (in the FT channel) and the target star (in the SC channel). Given the VLTI FOV, the angular separation between the reference object and the science target is limited to 5 arcsec with the ATs and 2 arcsec with the UTs. Within this restricted separation, the final accuracy on the relative astrometry is expected to be of a few tens of μarcsec.

3 Data flow overview

3.1 The concept of recipe

The pipeline is built over “recipes” that implement the basic steps of the reduction. The recipe input is a Set Of Files (SOF), which contains the list of files to reduce, associated with the required calibrations. Each recipe has a list of possible options. Each recipe produces one or several products. The recipes can be triggered by several mechanisms:

gasgano is the ESO user interface to classify, select files (= construct a SOF) and launch recipes.

esorex is the ESO command line tool to launch a recipe on any SOF manually written by the user.

reflex is the ESO environment allowing to reduce an entire directory in batch mode.

The GRAVITY consortium has also built a dedicated python script which browses the content of the current directory, classifies the files, writes the SOFs and automatically triggers the corresponding recipes via esorex.

3.2 The concept of SOF

A SOF is a consistent list of files to be sent to a recipe. Each file in the list shall be assigned a *DO Category*, which tells the recipe its purpose in the reduction. In the GRAVITY pipeline, the DO categories are trivially built from the DPR.TYPE, DPR.CATG and PRO.CATG keywords in the FITS header.

3.3 Instrument calibrations

In order to reduce interferometric observations, it is mandatory to calibrate the detectors and the combiners. These calibrations are obtained via dedicated observations of the internal source with all shutters closed, one shutter open at a time, two shutters open at a time, and all shutters open.

The recipe **gravity_dark** creates the DARK calibration product, which contains the mean detector bias and the detector readout noise. It is associated matching the detector and the optical setup of the observation. The dark phisically doesn’t depend on the optical setup, however in GRAVITY the detector gain is setup based on the resolution mode used.

The recipe **gravity_p2vm** creates the BAD (bad pixel), FLAT (internal transmission), P2VM (internal phase and contrast) and WAVE (wavelength map) calibration products. They shall match the optical setup of the observation for the SC and the FT, as well as the detector gain for the FT.

These products are all needed in order to reduce the interferometric observations of science target.

3.4 From raw data to raw visibilities

The first step is to reduce the raw interferometric observations into uncalibrated measurement of the visibilities and closure phases. This step is generally done file-per-file, that is each OBJECT exposure in the raw directory-has a corresponding files in product directory (although it is possible to reduce several files together). Note that this step includes the averaging over all the frames that compose a standard exposure. This step is rather long (several hours for an entire night).

The recipe is **gravity_vis**.

The product (*_VIS) is an uncalibrated OIFITS file, containing the flux, the square visibilities, the complex visibilities and the closure phases for both the SC and the FT combiners.

The recipe produces an optional product, called P2VMRED, which contains many intermediate signals of the processing. It is very useful to assess the data quality. It is also possible to restart the reduction from the P2VMRED product, thus saving time for users who want to explore several values of recipe parameters.

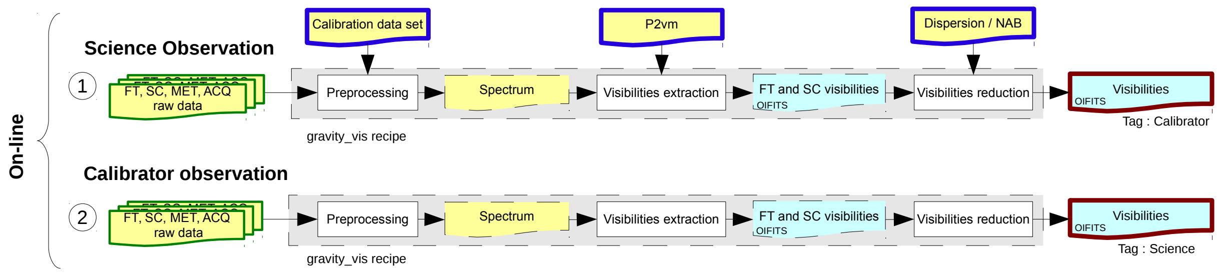

Figure 3.1: Data flow of gravity_vis recipe.

3.5 From raw visibilities to calibrated visibilities

The second step is to calibrate the observation of science objects with the observation of calibration stars. This step could be performed "globally", that is all the uncalibrated *_VIS oifits can be loaded first, then the pipeline searches for consistent sequences (same setup, same DIT, same wavelength table...), and then each sequence is calibrated independently with the calibration stars that could be found within it. This step is rather short (1min for an entire night).

The recipe is **gravity_viscal**.

The product (*_VIS_CALIBRATED) is a calibrated OIFITS file, ready for science.

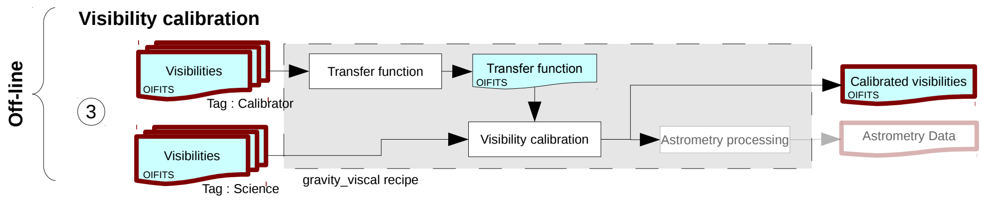

Figure 3.2: Data flow of gravity_viscal recipe.

4 Instrument Data Description

4.1 RAW science data

The RAW frames created when observing have the following DPR.TYPE:

OBJECT,DUAL are observations of a nearby pair of objects, one feeding the fringe-tracker (FT) and the other feeding the science combiner (SC). It can be of category SCI or CAL.

OBJECT,SINGLE are observations of a single object, feeding both the fringe-tracker (FT) and the science combiner (SC). It can be of category SCI or CAL.

SKY,SINGLE
SKY,DUAL are observation of an empty patch of the sky near the object in order to measure the sky brightness.

The OBJECT,DUAL and OBJECT,SINGLE types have a category DPR.CATG=SCI when observing a science target, and a category DPR.CATG=CAL when observing a calibration star used to monitor the transfer function.

4.2 RAW calibration data

The RAW frames used to calibrate the instrument on a daily-basis have the following DPR.TYPE:

DARK are observations with all shutters closed, in order to calibrate the detector dark level and the detector + dark level noise.

FLAT are observations of the internal source with one shutter open, in order to calibrate the positions of the spectra on the detectors and the internal transmission of the instrument.

P2VM are observations of the internal source with two shutters open, in order to calibrate the internal contrasts and phases of the instrument.

WAVE
WAVESC are observations of the internal source with all shutters open, in order to calibrate the wavelength table, and the internal closure phases. The WAVE data are recorded by scanning on both SC and FT, with FDDL in open loop and no fringe tracking (in order to calibrate the FT wavelengths). The WAVESC data are recorded by scanning only SC, and FDDL in close-loop and fringe tracking (to calibrate the SC wavelengths).

4.3 STATIC calibration

The STATIC calibration frames have the following DPR.TYPE:

DISP_MODEL is the model of the optical dispersive index $n(\lambda)$ of the fiber differential delay lines (FDDL) of the instrument.

DISP_VIS is an intermediate product when building DISP_MODEL, used to visualise the quality of the FDDL stretching sequence.

DIAMETER_CAT is the catalog of stellar diameters used to estimate the transfer function.

EOP_PARAM is a list of Earth Orientation Parameters (EOP) and DUT1 versus time. These corrections are only needed for the most demanding astrometric measurements.

DIODE_POSITION contains the position of the metrology receivers

4.4 PRODUCT calibration data

The PRODUCT of the calibration by the recipes **gravity_dark** and **gravity_p2vm** are identified by the following PRO.CATG keyword:

DARK contains images with the dark level and variance for the SC and FT detectors.

BAD contains images with the identified bad pixels for the SC and the FT detectors.

FLAT contains images of the profiles used to extract the SC spectra from the detector. There is one extracted spectrum per output of the detector and per polarisation if split (thus 24 or 48 spectra for each SC and FT combiners).

WAVE contains tables with the effective wavelengths of each channel of every spectra extracted with the profile. These tables are necessary to re-align the different spectra (outputs of the detector) onto a common wavelength grid.

P2VM contains tables with the internal transmission, contrast and phase of every output of the detector versus wavelength. These form the so-called pixel-2-visibility matrix used to extract the interferometric visibility from the spectra.

4.5 PRODUCT science data

The products of the science reduction by the recipe **gravity_vis** are identified by the following PRO.CATG keywords:

SINGLE_SKY **DUAL_SKY**	contains the mean brightness of the sky and its variance. Contrarily to DARKs, these quantities possibly depend on the instrument mode because different optics are used.
SINGLE_SCI_VIS **SINGLE_CAL_VIS** **DUAL_SCI_VIS** **DUAL_CAL_VIS**	are OIFITS files [4] with the uncalibrated flux, squared visibilities, complex visibilities and closure phases extracted from the raw observation of an object. SCI/CAL corresponds to a science target or a calibration star used to monitor the transfer function.
SINGLE_SCI_P2VMRED **SINGLE_CAL_P2VMRED** **DUAL_SCI_P2VMRED** **DUAL_CAL_P2VMRED**	are the raw data already processed through the P2VM algorithm. They contain flux per beam, and coherent flux per baseline, for each individual frame of the exposure. As such, they are intermediate products between the RAW data and the final, averaged, OIFITS. They also contain many intermediate results of the processing. The file size is huge (>200Mb). It is meant to assess the overal data quality and tune the reduction parameters. It is not used for science. Its format is inspired by OIFITS, but it is not strictly compliant.
SPECTRUM **PREPROC**	contains the RAW data already corrected for cosmetic and collapsed into one spectrum per combiner output. In SPECTRUM, the data are not yet re-interpolated into a common spectral wavelength grid, while this step is done in PREPROC. As such, they are debug-level intermediate products between the RAW data and the final, averaged, OIFITS.

The PRODUCT of the final calibration step by the recipe **gravity_viscal** are identified by the following PRO.CATG keywords:

SINGLE_SCI_VIS_CALIBRATED **DUAL_SCI_VIS_CALIBRATED**	are the final OIFITS file of the reduction, science ready. They contain the interferometric observations calibrated with the transfer function.
SINGLE_CAL_TF **DUAL_CAL_TF**	are OIFITS files containing the transfer function value estimated by the corresponding observation of a calibration star. It is the observed visibility of a calibrator divided by its visibility estimated from its diameter.
SINGLE_SCI_TF **DUAL_SCI_TF**	are OIFITS files containing the estimated (interpolated) value of the transfer function at the time of the corresponding science target observation.

5 Data Reduction

5.1 Graphical overview of the cascade

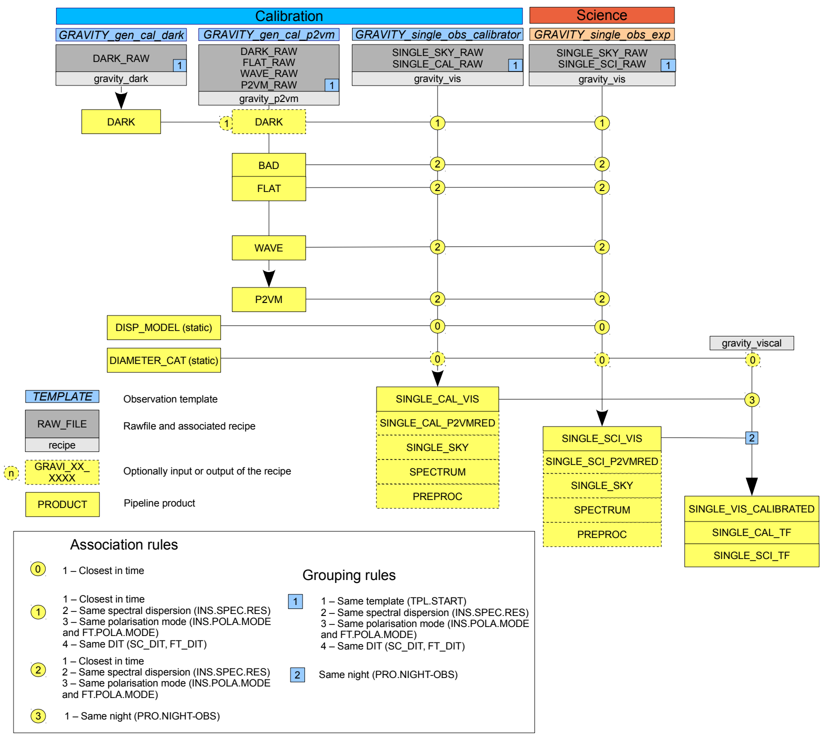

Figure 5.1: Data reduction cascade for observations in mode SINGLE. As of now, the cascade is the same for the mode DUAL.

5.2 Using Gasgano

Gasgano, provides a 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 &
```

Use the *Add/Remove Files* entry of the *File* menu to load data. The data are hierarchically organised as preferred by the user. More information about a single frame can be obtained by clicking on its name.

Frames can be selected from the main window for being processed by the appropriate recipe. Before launching the recipe, its configuration may be 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.

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

5.3 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. Users are free to define manually the input SOF and the appropriate configuration parameters.

A SOF for *EsoRex* is a simple ASCII file listing the files and their *DO Category*. Examples of SOF are given in the description of each recipe. Note that '#' is the comment character.

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

```
esorex [esorex_options] recipe_name [recipe_options] set_of_frames.sof
```

To get more information on how to customise *EsoRex* (see also [7]), or on a specific recipe, run the commands:

```
esorex  -h
esorex  -h  recipe_name
esorex  --man-page recipe_name
```

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

5.4 Using run_gravi_reduce.py python script

The consortium has written a simple python script that classifies the files in the current directory, associates them with calibrations, and runs the corresponding recipes.

Instrument calibration and data reduction is triggered by the following script:

```
run_gravi_reduce.py [options]
```

The script performs the following steps:

1. Trigger the recipe **gravity_dark** on all standalone DARK.
2. Trigger the recipe **gravity_p2vm** on all sequences of 1 x DARK, 4 x FLAT, 6 x P2VM, 1 x WAVE, 1 x WAVESC.
3. Trigger the recipe **gravity_vis** on all OBJECT,* files.

The script will search for (static) calibrations in the current directory if any. It will also search in the upper directory `../common_calibration/`. The products are created in the `reduced/` subdirectory.

Before triggering a recipe, the script writes the corresponding SOF and the esorex command in the `reduced/` subdirectory. Thus one can manually reproduce a given reduction by executing:

```
./reduced/GRAVI.XXXX_esorex.sh
```

To get more information on syntax and options, run the commands:

```
run_gravi_reduce.py -h
```

Transfer function and trending can be then triggered with the following script:

```
cd reduced/
run_gravi_trend.py [options]
```

The script performs the following steps:

1. Trigger the recipe **gravity_viscal** on all *CAL_VIS.
2. Trigger the recipe **gravity_viscal** on all *SCI_VIS.
3. Produce trending plots to check the transfer function.

The products are created in the `calibrated/` and `trend/` subdirectories.

6 Known Problems

6.1 Spectral calibration

The spectral calibration between baselines is accurate to 0.1nm. It corresponds to half a pixel in spectral direction (thus 1/4 of the spectral resolution element) in HR.

This uncertainty can generate biases in the closure phase, which amount to ≈3deg when observing at a group-delay of 40 μm.

The absolute spectral calibration is accurate to 0.5nm, which corresponds to one spectral resolution elements in HR.

6.2 Uncertainties in products

The uncertainty of product data contains the statistical noise only, computed by bootstrapping over the NDIT samples when possible. It does not contain the calibration uncertainty.

When the number of valid DIT within an exposure is lower than 5, the statistic to compute the final error bars also include additional MonteCarlo realisation of the *theoretical* photon and detector noise (to reach 5 samples). These uncertainties are thus less realistic.

6.3 Metrology and polarization

The metrology snr does not follow strictly the polar. s/ polar. p brightness ratio of the beam combiners when inserting the linear polarizer and rotating the half-wave plates. The reason is that the final polarization alignment is a compromise between best possible SC/FT s/p nulling of a polarized source, and the metrolgy snr (= alignment between FT/SC metrology and 3rd beam). In the "detector real time display reference frame" the metrology polarization is therefore aligned with the left FT channels and the lower SC channels, that is, with the P1 polarization signals in the GRAVITY data files.

7 Pipeline Recipe Interfaces

7.1 List of all recipes

We here list the role of each recipe. The input, output, options and QC parameters are detailled in each dedicated subsection.

gravity_astrometry	Perform phase referencing for dual-field astrometry.
gravity_badpix	Detect the bad pixels on the detectors.
gravity_biasmask	*Not Offered* Determine which pixels can be used to measure the bias of SC detector.
gravity_dark	Calibrate the detector noise and background level.
gravity_disp	Calibrate the linearity and the dispersion of the differential delay lines.
gravity_eop	Download the last values of the Earth Orientation Parameters and DUT from IERS.
gravity_p2vm	Detect the bad pixels on the detectors, calibrate the wavelength tables, calibrate the interferometric contrast and phase.
gravity_pcacal	Generate static calibration files for flattening phase visibility data using the PCA method.
gravity_piezo	Calibrate the response of the piezo actuators.
gravity_postprocess	Post-process the products, to fine-tune their content.
gravity_vis	Compute the visibilities from raw observation of OBJECT.
gravity_vis_from_p2vmred	Compute the visibilities from P2VMRED intermediate product.
gravity_viscal	Calibrate visibilities with the transfer function (atmospheric interferometric response).
gravity_wavelamp	Measure the position of the Argon lines in the spectra.

7.2 gravity_astrometry

This recipe computes phase and amplitude referencing for dual-field astrometric observations. It implements the computations described in section 10.27 for on-axis, off-axis and off-axis swap observing strategies.

Please note that this recipe is to be considered EXPERIMENTAL at present.

1. If swaps are present: obtain astrometric solution and compute swap phase reference.
2. Compute phase reference for the target, using the on-star and/or swap frames.
3. Save the product.

Input

DO.CATG	short description
ASTRO_CAL_PHASEREF	ASTRO_REDUCED file(s) from on-star observations
ASTRO_TARGET	ASTRO_REDUCED file(s) from on-planet observations
ASTRO_SWAP (optional)	ASTRO_REDUCED file(s) obtained when using the SWAP template

Output

PRO.CATG	short description
ASTRO_PHASE_CALIBRATED	ASTRO_REDUCED file(s) with correct phase reference

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–use-swap-fiber-pos	use fiber position for swap rather than computing an astrometric solution. [FALSE]
–ra-lim-swap	specify the RA range (in mas) over which to search for the astrometry of the swap. Default specifies entire field of view. [-1.0]
–nra-swap	number of points over the RA range for the swap. [50]
–dec-lim-swap	specify the dec range (in mas) over which to search for the astrometry of the swap. Default specifies entire field of view. [-1.0]
–ndec-swap	number of points over the dec range for the swap. [50]
–average-over-dits	Average over DITs before reducing astrometry for speed. [FALSE]
–zoom-factor	Factor to magnify ra/dec limits by after initial fit to find precise solution. [1.0]
–ft-mean-flux	remove all data with FT flux below this factor times the mean. [0.2]

–calib-strategy	how to calculate the reference coherent flux. See table below for explanation of options. <NONE | ALL | SELF | SWAP | NEAREST> [NONE]

Name	short description
NONE	do not use an amplitude reference
ALL	use all files
SELF	calibrate each file individually
SWAP	use the swap files
NEAREST	use the nearest two (in time) files.

7.3 gravity_badpix

The recipe creates a BAD calibration map from raw DARKs and raw FLATs observations. Since it is not associated with the calibration of the instrumental transmission, more specific darks or flats can be used. Such as very long darks, fore better statistic; and/or defocused flats to illuminate more pixels. The create BAD map can then be used as an input for further calibration (P2VM) and observations.

1. Load input files
2. Compute badpixel from dark rms, dark median, and flat value
3. Save the product

Input

DO.CATG	short description
DARK_RAW	raw dark, all shutters closed (DPR.TYPE=DARK)
FLAT_RAW x4	raw flats, one sutter open (DPR.TYPE=FLAT)

Output

PRO.CATG	short description
BAD	badpixel calibration (PRO.CATG=BAD)

Parameters

Name	short description

–static-name	Use static names for the products (for ESO). [FALSE]
–bad-dark-threshold	the rms factor for dark bad pixel threshold. [10]

Pseudo code gravity_badpix

```
# (1) Load and Check the input frameset for 1 DARK_RAW and 4 FLAT_RAW
# Extract DARK and FLAT frameset
# Compute the dark
dark_map = gravi_compute_dark(data)  # See algorithm in section 10.2
# Load and identify the FLAT files
# Build the list of FLAT files and output flat_raw_data
# (2) Compute the BADPIX from the DARK and FLATs
badpix_map = gravi_compute_badpix(dark_map, raw_data, flat_frameset_size, paramete
# (3) Save the BADPIX Map
gravi_data_save_new(badpix_map, frameset,  parameters, GRAVI_BAD_MAP)
```

7.4 gravity_biasmask

UNOFFERED The recipe creates a binary mask (BIASPIX) indentifying which pixels of the SC detector are not illuminated, and thus could be used as bias-pixels in further processing. The idea would be to input such a mask, as static calibration, in all reductions. However this is not yet implemented, nor demonstrated as necessary.

1. Load the input files
2. Identify the mask
3. Write product

Input

DO.CATG	short description
DARK_RAW	raw dark, all shutters closed (DPR.TYPE=DARK)
FLAT_RAW x4	raw flats, one sutter open (DPR.TYPE=FLAT)

Output

PRO.CATG	short description
BIASMASK	biaspixel mask calibration

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]

7.5 gravity_dark

This recipe computes the DARK calibration for the SC, the FT and the ACQ detectors. The SC detector is first debiased using the biaspixels, before computing the dark mean and rms. For detectors, the mean dark level of each pixel and the stdev of each pixel are saved in the output product.

1. Loop on input dark files and concatenate them
2. Compute the median and rms of these concatenated files
3. Save the product (FT, SC, ACQ camera into same product)

Input

DO.CATG	short description
DARK_RAW	raw dark, all shutters closed (DPR.TYPE=DARK)

Output

PRO.CATG	short description
DARK	dark calibration

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–bias-method	Method to average the biaspixels when cleaning-up the SC detector (only applied to MED and LOW). Ideally the same value shall be used when reducing the DARK with gravity_dark and the OBJECT with gravity_vis. UTO is equivalent to MASKED_MEDIAN_PER_COLUMN if the data ontains in the IMAGING_DETECTOR_SC extension the EFT, HALFLEFT, CENTER, HALFRIGHT and RIGHT columns. therwise it is like MEDIAN. <AUTO | MEDIAN | MEDIAN_PER_COLUMN | MASKED_MEDIAN_PER_COLUMN> [AUTO]

–bias-subtracted-file	Save the BIAS_SUBTRACTED intermediate product. [FALSE]

Pseudo code gravity_dark

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# (1) Extract DARK frameset
dark_frameset = gravi_frameset_extract_dark_data(frameset)
nb_dark = size(dark_frameset)
# Merge the dark frames
for i in nb_dark:
    data = gravi_data_load_rawframe(dark_frameset[i])
    gravi_data_detector_cleanup(data, parameters)  # see algo 10.1
    if i=0:
        raw_dark = data
    else:
        gravi_data_append(raw_dark, data)
# (2) compute the reduced dark
dark_map = gravi_compute_dark(raw_dark)  # see algo 10.2
# (3) save the product
gravi_data_save_new(dark_map, parameters)
```

7.6 gravity_disp

This recipe is associated to the template GRAVI_all_disp. It measures the phases obtained on the internal source at the position of the Argon lines and various positions (= fiber stretch) of the Fibered Differential Delay Lines (FDDL). It deduces the linearity model and the dispersion model of the differential delay lines. These models are stored as polynomials versus wavelength.

1. Reduce all the input DISP files (see gravity_vis), write each product
2. Compute the dispersion parameters from this entire dataset
3. Write product

Input

DO.CATG	short description
FLAT	flat calibration (PRO.CATG=FLAT)
BAD	badpixel calibration (PRO.CATG=BAD)
WAVE	wave calibration (PRO.CATG=WAVE)

P2VM	p2vm calibration (PRO.CATG=P2VM)
DARK	dark calibration (PRO.CATG=DARK)
WAVELAMP	spectrum of Argon, with position of lines
DISP_RAW (>50)	raw dispersion

Output

PRO.CATG	short description
DISP_VIS	intermediate product
DISP_MODEL	dispersion model of FDDL

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–bias-subtracted-file	Save the BIAS_SUBTRACTED intermediate product. [FALSE]
–spectrum-file	Save the SPECTRUM intermediate product. [FALSE]
–preproc-file	Save the PREPROC intermediate product. [FALSE]
–p2vmreduced-file	Save the P2VMRED intermediate product. [FALSE]
–astro-file	Save the ASTROREDUCED intermediate product. [FALSE]
–vis-file	Save the VIS intermediate product. [FALSE]
–ditshift-sc	Shift the time of SC DITs by an integer value to account for lost frames in exposure (issue on the instrument side, report to instrument team). The time of all DITs in exposure are increased by ditshift x PERIOD. ditshift can be 0, positive (system has lost one SC DIT), or negative (SC desynchronized). [0]
–extra-pixel-ft	Include the 6th pixels ot the FT. [TRUE]
–acq-correction-delay	Delay between the end of ACQ frame and correction offset seen by the metrology diodes, in seconds. [0.25]
–use-fiber-dxy	Use the fiber position when computing OPD_TEL_CORR. [FALSE]
–use-met-rtc	Reduce metrology voltage with the real time algorithm nstead of using the pipeline's algorithm. [FALSE]
–smooth-faint	Adds an additional factor to the smoothing of he metrology voltages in faint mode. [1]
–preswitch-delay	Delay where metrology values are ignored before aser brightness is switched in faint mode, ms. [50]
–postswitch-delay	Delay where metrology values are ignored after aser brightness is switched in faint mode, ms. [200]

–chi2r-threshold	Threshold in chi2r of the fringe-fit to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [50.0]
–chi2r-sigma	Threshold in chi2r of the fringe-fit (in unit of the the std of chi2r in the spectral direction) to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [100.0]
–nsmooth-snr-ft	Number of samples to average coherently when computing the real-time SNR and GDELAY of the FT (shall correspond to the atmospheric coherence time). The integration window runs from -nsmooth -> +nsmooth. [5]
–phase-ref-sc-maxdeg	Maximum deg for the fit of PHASE_REF. [3]
–use-met-zero	Flag to add a constant value to OPD_DISP. This constant value is taken from the header. [FALSE]
–imaging-ref-met	Metrology source used for IMAGING_REF calculation: Use fibre coupler metrology (FC); Use fibre coupler metrology corrected from pupil motion (FC_CORR); Use telescope metrology (TEL). <FC | FC_CORR | TEL> [FC]
–snr-min-ft	SNR threshold to accept FT frames (>0). It raises the first bit («0) of column REJECTION_FLAG of FT. [30.0]
–global-state-min-ft	Minimum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [2.0]
–global-state-max-ft	Maximum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [4.0]
–state-min-ft	Minimum OPDC state per baseline to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [1.0]
–tracking-min-sc	Minimum ratio of accepted FT frames in order to accept a SC frames (0..1), that is, for each SC DIT, the fraction of the time the REJECTION_FLAG of the FT is not 0. It raises the first bit («0) of column REJECTION_FLAG of SC. [0.8]
–vfactor-min-sc	vFactor threshold to accept SC frame (0..1). [0.8]
–opd-pupil-max-sc	Maximum OPD_PUPIL (abs) to accept SC frames. It raises the third bit («2) of column REJECTION_FLAG of SC. [9999.0]
–opd-pupil-stddev-max-sc	Maximum OPD_PUPIL_STDDEV to accept SC frames. It raises the fourth bit («3) of REJECTION_FLAG of SC. [2.9e-07]
–max-frame	Maximum number of frames to integrate coherently into an OIFITS entry. [10000]
–force-same-time	Force all baseline/quantities to have strictly the same TIME and MJD columns. [FALSE]
–debias-sc	Subtract the V2 bias from SC. [TRUE]
–debias-ft	Subtract the V2 bias from FT. [TRUE]
–nboot	Number of bootstraps to compute error (1..100). [1]

–vis-correction-sc	Correction of SC visibility from losses due to long integration, using the measured visibility losses with the FT (VFACTOR and/or PFACTOR) or by forcing the SC visibilities to match those of the FT (FORCE). Possible choices are:. <VFACTOR | PFACTOR | VFACTOR_PFACTOR | FORCE | NONE> [NONE]
–vis-correction-ft	Correction of FT visibility from losses due to long integration, using a sliding window PFACTOR, or its square. Choices are: <NONE | PFACTOR | PFACTOR_SQUARED> [NONE]
–pfactor-window-length	Length of the sliding window used to calculate the FT PFACTOR. For each FT frame, the window will run from -window-length to +window-length inclusive. [40]
–phase-ref-sc	Reference phase used to integrate the SC frames. Use a self-estimate of the phase, fitted by poly. (SELF_REF) Use the FT phase only, interpolated in lbd (PHASE_REF) Use the FT+MET-SEP.UV phase (IMAGING_REF). <SELF_REF | PHASE_REF | IMAGING_REF | AUTO | NONE> [AUTO]
–output-phase-sc	With DIFFERENTIAL, the mean group-delay and mean phases are removed from the output VISPHI in the final OIFITS file. With ABSOLUTE, the VISPHI is kept unmodified. With SELF_VISPHI, the internal differential phase between each spectral channel and a common reference channel is computed. <DIFFERENTIAL | ABSOLUTE | AUTO | SELF_VISPHI> [AUTO]
–output-phase-channels	range (string in the form [min,max]) of channels o use a SELF_VISPHI phase reference. [0,0]
–outlier-fraction-threshold	Flag channels with more than this fraction of the frames affected by outliers or cosmics. These are typically detected with the thresholds options in chi2 of the fringe-fit. [0.5]

Pseudo code gravity_disp

```
# Identify the RAW and CALIB frames in the input frameset
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Check if a DISP_VIS is already existing
dispvis_frameset = gravi_frameset_extract_dispvis_data(frameset)
# Extract static parameters from the frameset
static_param_frameset = gravi_frameset_extract_static_param(frameset)
# Load the STATIC_PARAM Parameter
if static_param_frameset:
    static_param_data = gravi_data_load_frame(static_param_frameset[0], used_frame
# (1) If there is no DISP_VIS, reduce all data
if dispvis_frameset is empty:
```

```
# Identify the ARGON, P2VM, DISP, DARK, WAVE, FLAT, BADPIX frames
wavelampcalib_frameset = gravi_frameset_extract_wavelamp_map(frameset)
p2vmcalib_frameset = gravi_frameset_extract_p2vm_map(frameset)
wavecalib_frameset = gravi_frameset_extract_wave_map(frameset)
flatcalib_frameset = gravi_frameset_extract_flat_map(frameset)
badcalib_frameset = gravi_frameset_extract_bad_map(frameset)
darkcalib_frameset = gravi_frameset_extract_dark_map(frameset)
dark_frameset = gravi_frameset_extract_dark_data(frameset)
disp_frameset = gravi_frameset_extract_disp_data(frameset)
# Identify the DARK in the input frameset
if dark_frameset:
    gravi_data_detector_cleanup(dark_frameset, parameters)  # see algo \ref{c
    # Compute dark
    dark_map = gravi_compute_dark(dark_frameset)  # see algo \ref{dark}
    # Save the dark map
    gravi_data_save_new(dark_map, parameters)
elseif darkcalib_frameset:
    dark_map = gravi_data_load_frame(darkcalib_frameset, used_frameset)
# Identify the BAD in the input frameset
badpix_map = gravi_data_load_frame(badcalib_frameset, used_frameset)
# Identify the FLAT in the input frameset
profile_map = gravi_data_load_frame(flatcalib_frameset, used_frameset)
# Identify the WAVE in the input frameset
wave_map = gravi_data_load_frame(wavecalib_frameset, used_frameset)
# Identify the P2VM in the input frameset
p2vm_map = gravi_data_load_frame(p2vmcalib_frameset, used_frameset)
# Loop on input DISP files
for ivis in size(disp_frameset)
    data = gravi_data_load_rawframe(disp_frameset[ivis])
    # Extract spectrum see algo \ref{extract}
    preproc_data = gravi_extract_spectrum(data, profile_map, dark_map,
                                    badpix_map, parameters)
    # Rescale to common wavelength
    gravi_align_spectrum(preproc_data, wave_map, p2vm_map, parameters)
    # Option save the proproc file
    if gravity.dfs.preproc-file:
        gravi_data_save_new(preproc_data, parameters, frameset, frame)
    # Compute the flux and visibilities for each telescope and
    # per acquisition with the P2VM applied to preproc_data
    p2vmred_data = gravi_compute_p2vmred(preproc_data, p2vm_map,"gravi_single
    # Reduce the OPDC table
    gravi_compute_opdc_state(p2vmred_data)
    # Reduce the metrology
    gravi_metrology_reduce(p2vmred_data, static_param_data, parameters)
    # Find outliers
```

```
        gravi_compute_outliers(p2vmred_data, parameters)
        # Compute the SNR_SMT and GDELAY_SMT columns
        gravi_compute_snr(p2vmred_data, parameters)
        # Compute the additional signals for averaging
        gravi_compute_signals(p2vmred_data, disp_map, parameters)
        # Compute rejection flags for averaging
        gravi_compute_rejection(p2vmred_data, parameters)
        # Save the P2VMREDUCED
        if gravity.dfs.p2vmred-file:
            gravi_data_save_new(p2vmred_data, frameset, parameters)
        # Loop on the wanted sub-integration
        while size(current_frame) >= 0:
            # Visibility and flux are averaged and the followings
            # are saved in Visibility data in tables VIS, VIS2 and T3
            tmpvis_data = gravi_compute_vis(p2vmred_data, parameters, current_fram
            # Compute QC parameters
            gravi_compute_vis_qc(tmpvis_data)
            # Save the VIS
            if gravity.dfs.vis-file:
                gravi_data_save_new(tmpvis_data, frameset, parameters)
            # Merge with already existing
            if vis_data == NULL:
                vis_data = tmpvis_data
            else:
                gravi_data_append(vis_data, tmpvis_data, 1)
        # Save the astro file, which is a lighter version of the p2vmreduced
        if gravity.dfs.astro-file:
            gravi_data_clean_for_astro(p2vmred_data)
            gravi_data_save_new(p2vmred_data, frameset, parameters)
        # End loop on the input files to reduce
        # Recompute the TIME column from the MJD column
        # in all OIFITS tables to follow standard
        gravi_vis_mjd_to_time(vis_data)
        # Identify the WAVELAMP in the input frameset
        argon_data = gravi_data_load_frame(wavelampcalib_frameset[0])
        # Duplicate POS_ARGON into the VIS file
        gravi_data_copy_ext(vis_data, argon_data, "POS_ARGON")
        # Save the output data file based on the first frame of the frameset
        cpl_frameset_join(used_frameset, disp_frameset)
        frame = cpl_frameset_get_position(disp_frameset, 0)
        gravi_data_save_new(vis_data, frameset, frame, parameters)
else:
    # Load the DIS_VIS already computed
    vis_data = gravi_data_load_frame(dispvis_frameset[0])
    disp_frameset = cpl_frameset_duplicate(dispvis_frameset)
```

```
# Compute the dispersion table of a set of disp frames
gravi_disp_cleanup(vis_data)
disp_map = gravi_compute_disp(vis_data)
# Create product frame
frame = cpl_frameset_get_position(disp_frameset, 0)
# Save the DISP_MODEL
gravi_data_save_new(disp_map, frameset, parameters)
```

7.7 gravity_eop

This recipe downloads the latest version of the Earth Orientation Parameter and DUT from the IERS site. File is created in the current directory. A web connection is required.

1. Download the IERS data
2. Convert into CPL table
3. Write product

Input

DO.CATG	short description
None	No input

Output

PRO.CATG	short description
EOP_PARAM	EOP calibration file (gravity_eop_calib.fits)

Parameters

Name	short description
–eop_host	FTP Host to retrieve the EOP from. [ftp.iers.org]
–eop_urlpath	FTP URL path of the EOP file to retrieve. [/products/eop/rapid/standard/finals2000A.data]

7.8 gravity_p2vm

This recipe reduces the internal calibrations. As a special sequence of shutter opening is required, it is advised to always build the SOF with a complete sequence of files obtained within a single execution of the p2vm calibration template. However it is still possible to input a SOF with DARK_RAW only, or DARK_RAW and FLAT_RAW only. It is also possible to input a SOF with some already processed calibration (e.g WAVE).

1. Compute the dark, write product
2. Compute the flat, write product
3. Compute the badpixels, write product
4. Compute the spectral calibration, write product
5. Compute the p2vm, write product

Input

DO.CATG	short description
DARK_RAW	raw dark, all shutters closed (DPR.TYPE=DARK)
FLAT_RAW x4	raw flats, one shutter open (DPR.TYPE=FLAT)
P2VM_RAW x6	raw p2vms, two shutters open (DPR.TYPE=P2VM)
WAVE_RAW	raw wavelength calibration for FT (DPR.TYPE=WAVE)
WAVESC_RAW	raw wavelength calibration for SC (DPR.TYPE=WAVE,SC)

Output

PRO.CATG	short description
DARK	dark calibration
FLAT	flat calibration
BAD	badpixel calibration
WAVE	wave calibration
P2VM	p2vm calibration

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–debug-file	Save additional debug file(s). [FALSE]
–preproc-file	Save the PREPROC intermediate product. [FALSE]

–bias-method	Method to average the biaspixels when cleaning-up the SC detector (only applied to MED and LOW). Ideally the same value shall be used when reducing the DARK with gravity_dark and the OBJECT with gravity_vis. UTO is equivalent to MASKED_MEDIAN_PER_COLUMN if the data ontains in the IMAGING_DETECTOR_SC extension the EFT, HALFLEFT, CENTER, HALFRIGHT and RIGHT columns. therwise it is like MEDIAN. <AUTO | MEDIAN | MEDIAN_PER_COLUMN | MASKED_MEDIAN_PER_COLUMN> [AUTO]
–acq-correction-delay	Delay between the end of ACQ frame and correction offset seen by the metrology diodes, in seconds. [0.25]
–use-fiber-dxy	Use the fiber position when computing OPD_TEL_CORR. [FALSE]
–use-met-rtc	Reduce metrology voltage with the real time algorithm nstead of using the pipeline's algorithm. [FALSE]
–smooth-faint	Adds an additional factor to the smoothing of he metrology voltages in faint mode. [1]
–preswitch-delay	Delay where metrology values are ignored before aser brightness is switched in faint mode, ms. [50]
–postswitch-delay	Delay where metrology values are ignored after aser brightness is switched in faint mode, ms. [200]
–bad-dark-threshold	the rms factor for dark bad pixel threshold. [10]
–profile-mode	Method to compute the extraction profile. PROFILE corresponds to the pixel intensities measured in the FLAT files (Gaussian like with FWHM of approx 1.5 pixel). This is the AUTO option for the Low and Med spectral resolution. GAUSS corresponds to a Gaussian fit of the (non-zero) pixel intensities measured in the FLAT files. BOX corresponds to a box-card of 6 pixels centered on the spectra measured in the FLAT files. This is the AUTO option for High spectral resolution. <AUTO | PROFILE | GAUSS | BOX> [AUTO]
–force-badpix-to-zero	Force the badpixel to zero in profile. [TRUE]
–profile-width	Width of the detector window extracted around the default position of each spectrum, and on which the profile will be applied to perform the extraction. [6]
–force-wave-ft-equal	Force the spatial order of the wavelength 2D fit for FT to zero (so all region share the same calibration). This is used to build the P2VM calibration of the TAC real-time code running on the instrument ifself. [FALSE]

–ditshift-sc	Shift the time of SC DITs by an integer value to account for lost frames in exposure (issue on the instrument side, report to instrument team). The time of all DITs in exposure are increased by ditshift x PERIOD. ditshift can be 0, positive (system has lost one SC DIT), or negative (SC desynchronized). [0]
–extra-pixel-ft	Include the 6th pixels ot the FT. [TRUE]
–phase-calibration	This option changes the phase reference of the P2VM: NONE defines phiA(lbd) at zero for all baselines (P2VM calibrates only the internal phase-shift of the beam combiner); CLOSURE defines phiA(lbd) at zero for baselines 01, 02 and 03 (P2VM calibrates the phase-shift and the closure-phase of the beam combiner); DISP defines phiA(lbd) to have zero mean and minimum GD for baselines (01,02,03); (P2VM calibrates the phase-shift, the closure-phase and the spectral-dispersion of the beam combiner); FULL defines phiA(lbd) to have zero-GD for baselines (01,02,03). <NONE | CLOSURE | DISP | FULL> [FULL]

Quality control

QC in BAD	short description
BADPIX ACQ	Total number of bad pixels on the ACQ detector
BADPIX SC	Total number of bad pixels on the SC detector
BADPIX FT	Total number of bad pixels on the FT detector
BADPIX_DARK SC/FT	Pixels with weird mean level
BADPIX_RMS SC/FT	Pixels with weird noise level
BADPIX_FLAT SC/FT	Pixels non-responding to illumination
QC in FLAT	short description
PROFILE_CENTER SC1 MED	[pixel] position of the first spectra on SC detector
PROFILE_WIDTH SC1 MED	[pixel] width of the first spectra on SC detector
PROFILE_CENTER SC13 MED	[pixel] position of the 13d spectra on SC detector
PROFILE_WIDTH SC13 MED	[pixel] width of the 13d spectra on SC detector
MEANGAIN SC	Mean gain [ADU/e] for SC detector
MEANGAIN FT	Mean gain [ADU/e] for FT detector
QC in P2VM	short description
FLUX_SCi AVG	[e/DIT/chanel/output] flux in SC (mean of files)
FLUX_FTi AVG	[e/DIT/chanel/output] flux in FT (mean of files)
P2VM_COHERENCE_AVG_SC	Average instrumental contrast of SC
P2VM_COHERENCE_AVG_FT	Average instrumental contrast of FT
P2VM_COHERENCE_SCij	Average instrumental contrast of SC for pair ij
P2VM_COHERENCE_FTij	Average instrumental contrast of FT for pair ij
QC in WAVE	short description
REFWAVE1	Reference wavelengh [m] for the below parameters

REFPOS1 SCi	Position [pix] of the REFWAVE1 in output SCi
REFPOS1 FTi	Position [pix] of the REFWAVE1 in output FTi
REFWAVE2	Reference wavelengh [m] for the below parameters
REFPOS2 SCi	Position [pix] of the REFWAVE2 in output SCi
REFPOS2 FTi	Position [pix] of the REFWAVE2 in output FTi
WAVE_CORR	Model to convert the glass wavelength in vacuum wavelength
WAVE_CORR N0	Paramater of above model
WAVE_CORR N1	Paramater of above model
WAVE_CORR N2	Paramater of above model
MINWAVE SC/FT	Min wavelength [m] of SC/FT channels
MAXWAVE SC/FT	Max wavelength [m] of SC/FT channels
RMSWAVE SC/FT	Rms of residuals during polynomial wavelength fit

Pseudo code gravity_p2vm

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Extract DARK frameset
dark_frameset = gravi_frameset_extract_dark_data(frameset)
darkcalib_frameset = gravi_frameset_extract_dark_map(frameset)
# Extract FLAT frameset
flat_frameset = gravi_frameset_extract_flat_data(frameset)
flatcalib_frameset = gravi_frameset_extract_flat_map(frameset)
# Extract BAD frameset
badcalib_frameset = gravi_frameset_extract_bad_map(frameset)
# Extract WAVE frameset
wave_frameset = gravi_frameset_extract_wave_data(frameset)
wavesc_frameset = gravi_frameset_extract_wavesc_data(frameset)
wavecalib_frameset = gravi_frameset_extract_wave_map(frameset)
# Extract P2VM frameset
p2vm_frameset = gravi_frameset_extract_p2vm_data(frameset)
# (1) Compute or load the DARK file
if dark_frameset:
    data = gravi_data_load_rawframe(dark_frameset[0])
    gravi_data_detector_cleanup(data, parameters)   # see algo 10.1
    dark_map = gravi_compute_dark(data)  # see algo 10.2
    gravi_data_save_new(dark_map, parameters)
elseif darkcalib_frameset:
    dark_map = gravi_data_load_frame(darkcalib_frameset[0])
# (2) Compute or load the BAD pixel file
if badcalib_frameset:
    badpix_map = gravi_data_load_frame(badcalib_frameset[0])
elseif dark_frameset and flat_frameset is not None:
    for i in nb_flat_frameset:
        raw_flat[i] = gravi_data_load_rawframe(flat_frameset[i])
```

```
        gravi_data_detector_cleanup(raw_flat[i], parameters)  # see algo 10.1

    badpix_map = gravi_compute_badpix(dark_map, raw_flat)
    gravi_data_save_new(badpix_map, parameters)
# (3) Compute or load the FLAT file
if flatcalib_frameset:
    profile_map = gravi_data_load_frame(flatcalib_frameset[0])
elseif flat_frameset:
    for i in nb_flat_frameset:
        raw_flat[i] = gravi_data_load_rawframe(flat_frameset[i])
        gravi_data_detector_cleanup(raw_flat[i], parameters)  # see algo 10.1

    profile_map = gravi_compute_profile(raw_flat, dark_map, badpix_map,
                            nb_frame_gain, parameters)  # see algo 10.3
    gain = gravi_compute_gain(raw_data, nb_frame_gain, dark_map)
    gravi_data_save_new (profile_map, parameters)
# (4) Compute or load the WAVE file
if wavecalib_frameset:
    wave_map = gravi_data_load_frame(wavecalib_frameset[0])
elseif wave_frameset:
    wave_data = gravi_data_load_rawframe_ext(wave_frameset)
    gravi_data_detector_cleanup(wave_data, parameters)  # see algo 10.1
    spectrum_data = gravi_extract_spectrum (wave_sc_data, profile_map,
                        dark_map, badpix_map, parameters)  # see algo 10.3
    wave_map.p2vm_met = gravi_metrology_compute_p2vm(wave_data.met_table)
    gravi_wave_compute_opds(spectrum_data, wave_data.met_table)  # see algo 10.4.
    gravi_compute_wave(wave_map, spectrum_data, GRAVI_FT, parameters)
    if wavesc_frameset:
        wavesc_data = gravi_data_load_rawframe_ext(wavesc_frameset)
        gravi_data_detector_cleanup(wavesc_data, parameters)  # see algo 10.1
        spectrum_data = gravi_extract_spectrum(wavesc_data, profile_map,
                            dark_map, badpix_map, parameters)  # see algo 10.3
        gravi_wave_compute_opds(spectrum_data, wavesc_data.met_table)  # see algo
        gravi_compute_wave(wave_map, spectrum_data, GRAVI_SC, parameters)
        gravi_data_save_new(wave_map, parameters)
# (5) Compute the p2vm
check(dark_map + bad_map + flat_map + wave_map)
p2vm_frameset = p2vm_frameset + wave_frameset + flat_frameset
check(size(p2vm_frameset) == 11)
# (6) Loop on files of the p2vm frameset
# Construction of the p2vm data.
# read wave parameter from calibration file - Load the WAVE_PARAM Parameter
p2vm_map = gravi_create_p2vm(wave_map)
for i in size(p2vm_frameset):
    data = gravi_data_load_rawframe_ext(p2vm_frameset[i])
```

```
    if gravi_data_check_shutter(hdr_data, 1,1,1,1):
        i_wave = i
        skip_file
    gravi_data_detector_cleanup (data, parameters)  # see algo 10.1
    preproc_data = gravi_extract_spectrum (data, profile_map, dark_map,
                        badpix_map, parameters)  # see algo 10.3
    gravi_align_spectrum(preproc_data, wave_map, p2vm_map, parameters)
    gravi_compute_p2vm(p2vm_map, preproc_data, valid_trans, valid_CP,
                        GRAVI_DET_SC)  # see algo 10.6
# (7) P2VM normalization
gravi_p2vm_normalisation(p2vm_map)
# (8) Analyse the WAVE to get the phase correction
# and the internal spectrum to latter correct
data = gravi_data_load_rawframe_ext(p2vm_frameset[i_wave])
gravi_data_detector_cleanup(data, parameters)  # see algo 10.1
preproc_data = gravi_extract_spectrum (data, profile_map, dark_map,
                                    badpix_map, parameters)  # see algo 10.3
gravi_align_spectrum(preproc_data, wave_map, p2vm_map, parameters)
p2vmred_data = gravi_compute_p2vmred(preproc_data, p2vm_map, parameters)
if parameters.phase-calibration == "CLOSURE":
    gravi_p2vm_phase_correction(p2vm_map, p2vmred_data, 0)
elseif parameters.phase-calibration == "DISP":
        gravi_p2vm_phase_correction (p2vm_map, p2vmred_data, 1)
elseif parameters.phase-calibration == "FULL":
        gravi_p2vm_phase_correction (p2vm_map, p2vmred_data, 2)

gravi_p2vm_transmission(p2vm_map, p2vmred_data)
gravi_data_save_new (p2vm_map, parameters)
```

7.9 gravity_pcacal

This recipe produces a PCA calibration file from a set of calibration frames to be used for flattening phase visibility data.

1. Select good input frames using tracking ratio criterion.
2. Compute PCA decomposition for each baseline and polarisation channel
3. Fit component model and write calibration product

Input

DO.CATG	short description

SINGLE_CAL_VIS (>=20)	calibration frames

Output

PRO.CATG	short description
PHASE_PCA	PCA calibration

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–pca-components	The number of PCA components to compute. [1]
–pca-tracking-ratio	The minimum tracking ratio to accept calibration frames for. [90]
–pca-clean-size	The window size to use for outlier cleaning. [20]
–pca-clean-nstd	The sigma-clip n_std for outlier cleaning. [5.0]
–pca-fit-type	The method to use for fitting the PCA components. <POLYNOMIAL | SPLINE> [SPLINE]
–pca-fit-degree	The polynomial fit degree, or number of spline coefficients. [20]
–pca-save-residuals	Also save the residuals from the PCA fitting for inspection. [FALSE]

Pseudo code gravity_pcacal

```
min_tracking_ratio = "gravity.calib.pca-tracking-ratio"
# Get the input frameset
vis_cal_frameset = gravi_frameset_extract_vis_calib(frameset)
# Get header data from first frame
data_tmp = gravi_data_load_frame(vis_cal_frameset[0])
header_first = gravi_data_get_header(data_tmp)
telescope = header_first["TELESCOP"]
pola_mode = gravi_pfits_get_pola_mode(header_first, GRAVI_SC)
npol = gravi_pfits_get_pola_num(header_first, GRAVI_SC)
spec_res = gravi_pfits_get_spec_res(header_first)
# Check on time First frame not too old
# and Select latest epoch date that precedes observation date
# Get length of wavelength axis
wave_plist = gravi_data_get_oi_wave_plist(data_tmp, GRAVI_SC, 0, npol)
nwave = wave_plist["NWAVE"]
# Select all frames for compatibility and reject if below tracking ratio
```

```
nframes = vis_cal_frameset.size
for n in range(nframes):
    data_tmp = gravi_data_load_frame(vis_cal_frameset[n])
    # Check for uniform wavelength axis
    hdr = gravi_data_get_header(data_tmp)
    wv_plisti = gravi_data_get_oi_wave_plist(data_tmp, GRAVI_SC, 0, npol)
    nwavei = wv_plisti["NWAVE"]
    # Check for matching telescope telescope == hdr["TELESCOP"]
    # Check for matching polarisation mode pola_mode == hdr[GRAVI_SC]
    # Check for matching resolution spec_res == gravi_pfits_get_spec_res(hdr)
    # Check for matching epoch time_mjd_obs = hdr["MJD-OBS"]
    # Check if visibilities are all zeroes (bad data?)
    # Check if tracking ratio for file exceeds threshold and discard if not
pca_data = gravi_compute_pca(data_accepted, naccept, parlist)  # see algo \ref{pc
# Add filenames for accepted files to output
# timestamp the ouput
#save the frameset
gravi_data_save_new(pca_data, frameset, product_filename, parameters,
                    used_frameset, "gravity_phase_pca", GRAVI_PHASE_PCA)
```

7.10 gravity_piezo

This recipe compute the response (open loop transfer function) of the piezo actuators used to fringe-track in GRAVITY.

1. Compute the piezo TF QC parameter* Write product

Input

DO.CATG	short description
PIEZOTF_RAW	dedicated observations (DPR.CATG=PIEZOTF)

Output

PRO.CATG	short description
PIEZOTF	Response of the piezo

Parameters

Name	short description

–static-name	Use static names for the products (for ESO). [FALSE]

Quality control

QC in PIEZOTF	short description
FT KAL PZ_FIT	Standard deviation of the residual of the fit of the piezo response [rad].
FT KAL PX_GAIN	Static gain of Piezo number X [rad/Volts]
FT KAL PX_DELAY	Pure delay of Piezo number X [ms]
FT KAL PX_STDEV	standard deviation error between the value calculated and the value which are used by the Kalman RTC [rad]
FT KAL PX_RESPY	Response value of Piezo number X at step number Y (AR5 decomposition) [rad/Volts].

Pseudo code gravity_piezo

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Dispatch the frameset
recipe_frameset = gravi_frameset_extract_piezotf_data(frameset)
# Check the frameset
# Loop on input RAW frames to be reduced
nb_frame = size(recipe_frameset)
for i_file in nb_frame:
    # Reduce the File
    piezo_data = gravi_compute_piezotf(recipe_frameset[i_file], parameters)
    # Save the PIEZOTF which is in fact a P2VMREDUCED
    gravi_data_save_new(piezo_data, frameset, parameters )
```

7.11 gravity_postprocess

This recipe allows to manipulate the product of the GRAVITY pipeline, mostly the VIS. It permits to merge several files together into a single VIS file with all observations; to average the observations of one or several VIS file to increase the SNR; to remove some data (FT, SC); and to resample the SC observation with spectral binning. The list of input files can be P2VMRED, VIS, VIS_CALIBRATED (or even RAW for some parameters). However they should all be compatible in term of setup and observed objets !! Note that the recipe performs only litle checks of the input file content and structure. Thus the user shall ensure the input files are conformable (same polarisation and spectral mode for instante)

1. Load the files
2. Execute request from user

3. Write product

Input

DO.CATG	short description
Input files	see above

Output

PRO.CATG	short description
POSTPROCESSED	Output file

Parameters

Name	short description
–average-vis	Average the results from the different input files (if any) in the output product, instead of simply appending them. [FALSE]
–fluxerr-sc	Force the uncertainty in FLUX of SC. [0.0]
–visamperr-sc	Force the uncertainty in VISAMP of SC. [0.0]
–visphierr-sc	Force the uncertainty in VISPHI of SC. [0.0]
–vis2err-sc	Force the uncertainty in VIS2 of SC. [0.0]
–copy-fluxdata	Duplicate FLUX into FLUXDATA for OIFITS2 gravity.postprocess. [TRUE]
–force-merge	Force merging even if inconsistent data. [FALSE]
–remove-ft	Remove FT extensions. [FALSE]
–remove-sc	Remove SC extensions. [FALSE]
–remove-opdc	Remove OPDC extensions. [FALSE]
–remove-met	Remove METROLOGY related extensions. [FALSE]
–nbin-lambda-sc	Bin SC extensions in spectral dimension. [0]

Pseudo code gravity_postprocess

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# To use this recipe the frameset must not be empty and have at least two frames
# (1) (2) Loop on frames to append them
for f in nframe:
    # Load the frame
    data  = gravi_data_load_frame(frame[f], used_frameset)
    # Remove some data
    if gravity.postprocess.remove-ft:
```

```
        gravi_data_erase_type(data, "_FT")
    if gravity.postprocess.remove-sc:
        gravi_data_erase_type(data, "_SC")
    if gravity.postprocess.remove-opdc:
        gravi_data_erase_type(data, "OPDC")
    if gravity.postprocess.remove-met:
        gravi_data_erase_type(data, "METROLOGY")
        gravi_data_erase_type(data, "VIS_MET")
    # Force uncertainties
    gravi_force_uncertainties(data, parameters)
    # force the merging
    if gravity.postprocess.force-merge:
        force = True
    if f == 0:
        # Use the first frame for merging
        frame_merged = frame
        data_merged = data
    else:
        # Merge
        gravi_data_append(data_merged, data, force)
# Co-add them if required
if gravity.postprocess.average-vis:
    # Average the different observations = EXPERIMENTAL
    gravi_average_vis(data_merged)
# Resample them if required
if gravity.postprocess.nbin-lambda-sc > 1:
    # Resamp the SC data = EXPERIMENTAL
    gravi_vis_resamp(data_merged, nbin-lambda-sc)
# Add the FLUXDATA column for OIFITS2 standard
if gravity.postprocess.copy-fluxdata:
    gravi_vis_copy_fluxdata(data_merged)
# Recompute the TIME column from the MJD column
# in all OIFITS tables to follow standard
gravi_vis_mjd_to_time(data_merged)

# (3) Save the output data file based on the first frame of the frameset
gravi_data_save_new(data_merged, frameset, parameters )
```

7.12 gravity_vis

This recipe is associated to the observations template. Its reduces the raw data acquired on calibrator or science targets and computes the uncalibrated visibilities, saved in an OIFITS file. If several OBJECT are provided, the recipe will reduce all of them and merge the resulting data into a single OIFITS. If several SKY_RAW are provided, the recipe reduces the first OBJECT with the first SKY file. Then each new OBJECT with the next

SKY. When the number of SKYs is reached, the recipe loops back to first SKY file (so if the number of SKYs is larger than the number of OBJECTs, the last SKY won't be used). The recipe will reduce the data even if no SKY or no DARK is provided. However this will lead to wrong estimate of the visibility and squared visibility of the object. If the file DIAMETER_CAT is not provided, the recipe will use the diameter provided in the header to compute the transfer function QC parameters. The tag in the DO category can be SINGLE/DUAL and CAL/SCI. They should reflect the instrument mode (SINGLE or DUAL) and the DPR.CATG of the observation (SCIENCE or CALIB). The tag in the PRO.CATG category will be SINGLE/DUAL and CAL/SCI depending on the input tag.

1. Load the input file (loop on input OBJECT files)
2. Extract the spectra (use BAD, DARK, SKY, FLAT files)
3. Interpolate the spectra into a common wavelength table (use WAVE file)
4. Compute the real-time visibilities (use P2VM file)
5. Compute additional real-time signals (SNR, GDELAY...)
6. Compute selection flags (= flag frames with SNR lower than threshold, vFactor lower than threshold...)
7. Average the real-time visibilities, considering the selection flag
8. Write the product

Input

DO.CATG	short description
FLAT	flat calibration (PRO.CATG=FLAT)
BAD	badpixel calibration (PRO.CATG=BAD)
WAVE	wave calibration (PRO.CATG=WAVE)
P2VM	p2vm calibration (PRO.CATG=P2VM)
DARK	dark calibration (PRO.CATG=DARK)
SINGLE_SCI_RAW	raw object (DPR.TYPE=OBJECT,SINGLE)
SINGLE_SKY_RAW	raw sky (DPR.TYPE=SKY,SINGLE)
DISP_MODEL (opt)	fiber dispersion model (PRO.CATG=DISP_MODEL)
DIODE_POSITION (opt)	met receiver position (PRO.CATG=DIODE_POSITION)
DIAMETER_CAT (opt)	catalog of diameter (PRO.CATG=DIAMETER_CAT)

Output

PRO.CATG	short description

SINGLE_SCI_VIS	OIFITS file with uncalibrated visibilities
SINGLE_SKY (opt)	sky map
SINGLE_SCI_P2VMRED (opt)	intermediate product (see detailled description of data)
SPECTRUM (opt)	intermediate product (see detailled description of data)
PREPROC (opt)	intermediate product (see detailled description of data)

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–bias-subtracted-file	Save the BIAS_SUBTRACTED intermediate product. [FALSE]
–spectrum-file	Save the SPECTRUM intermediate product. [FALSE]
–preproc-file	Save the PREPROC intermediate product. [FALSE]
–p2vmreduced-file	Save the P2VMRED intermediate product. [FALSE]
–astro-file	Save the ASTROREDUCED intermediate product. [FALSE]
–average-vis	Average the results from the different input files (if any) in the output product, instead of simply appending them. [FALSE]
–bias-method	Method to average the biaspixels when cleaning-up the SC detector (only applied to MED and LOW). Ideally the same value shall be used when reducing the DARK with gravity_dark and the OBJECT with gravity_vis. AUTO is equivalent to MASKED_MEDIAN_PER_COLUMN if the data contains in the IMAGING_DETECTOR_SC extension the LEFT, HALFLEFT, CENTER, HALFRIGHT and RIGHT columns. Otherwise it is like MEDIAN. <AUTO | MEDIAN | MEDIAN_PER_COLUMN | MASKED_MEDIAN_PER_COLUMN> [AUTO]
–acq-correction-delay	Delay between the end of ACQ frame and correction offset seen by the metrology diodes, in seconds. [0.25]
–use-fiber-dxy	Use the fiber position when computing OPD_TEL_CORR. [FALSE]
–use-met-rtc	Reduce metrology voltage with the real time algorithm instead of using the pipeline's algorithm. [FALSE]
–smooth-faint	Adds an additional factor to the smoothing of the metrology voltages in faint mode. [1]
–preswitch-delay	Delay where metrology values are ignored before laser brightness is switched in faint mode, ms. [50]
–postswitch-delay	Delay where metrology values are ignored after laser brightness is switched in faint mode, ms. [200]
–demodulate-metrology	Perform demodulation on the raw metrology data. [TRUE]
–use-dark-offsets	Use diode zeros measured from the DARK when demodulating metrology. [TRUE]

–chi2r-threshold	Threshold in chi2r of the fringe-fit to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [50.0]
–chi2r-sigma	Threshold in chi2r of the fringe-fit (in unit of the the std of chi2r in the spectral direction) to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [100.0]
–nsmooth-snr-ft	Number of samples to average coherently when computing the real-time SNR and GDELAY of the FT (shall correspond to the atmospheric coherence time). The integration window runs from -nsmooth -> +nsmooth. [5]
–phase-ref-sc-maxdeg	Maximum deg for the fit of PHASE_REF. [3]
–use-met-zero	Flag to add a constant value to OPD_DISP. This constant value is taken from the header. [FALSE]
–imaging-ref-met	Metrology source used for IMAGING_REF calculation: Use fibre coupler metrology (FC); Use fibre coupler metrology corrected from pupil motion (FC_CORR); Use telescope metrology (TEL). <FC | FC_CORR | TEL> [FC]
–snr-min-ft	SNR threshold to accept FT frames (>0). It raises the first bit («0) of column REJECTION_FLAG of FT. [3.0]
–global-state-min-ft	Minimum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [2.0]
–global-state-max-ft	Maximum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [4.0]
–state-min-ft	Minimum OPDC state per baseline to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [1.0]
–tracking-min-sc	Minimum ratio of accepted FT frames in order to accept a SC frames (0..1), that is, for each SC DIT, the fraction of the time the REJECTION_FLAG of the FT is not 0. It raises the first bit («0) of column REJECTION_FLAG of SC. [0.8]
–vfactor-min-sc	vFactor threshold to accept SC frame (0..1). [0.1]
–opd-pupil-max-sc	Maximum OPD_PUPIL (abs) to accept SC frames. It raises the third bit («2) of column REJECTION_FLAG of SC. [9999.0]
–opd-pupil-stddev-max-sc	Maximum OPD_PUPIL_STDDEV to accept SC frames. It raises the fourth bit («3) of REJECTION_FLAG of SC. [2.9e-07]
–max-frame	Maximum number of frames to integrate coherently into an OIFITS entry. [10000]
–force-same-time	Force all baseline/quantities to have strictly the same TIME and MJD columns. [FALSE]
–debias-sc	Subtract the V2 bias from SC. [TRUE]
–debias-ft	Subtract the V2 bias from FT. [TRUE]
–nboot	Number of bootstraps to compute error (1..100). [20]

<table>
<tr><td>–vis-correction-sc</td><td>Correction of SC visibility from losses due to long integration, using the measured visibility losses with the FT (VFACTOR and/or PFACTOR) or by forcing the SC visibilities to match those of the FT (FORCE). Possible choices are:. <VFACTOR | PFACTOR | VFACTOR_PFACTOR | FORCE | NONE> [VFACTOR]</td></tr>
<tr><td>–vis-correction-ft</td><td>Correction of FT visibility from losses due to long integration, using a sliding window PFACTOR, or its square. Choices are: <NONE | PFACTOR | PFACTOR_SQUARED> [NONE]</td></tr>
<tr><td>–pfactor-window-length</td><td>Length of the sliding window used to calculate the FT PFACTOR. For each FT frame, the window will run from -window-length to +window-length inclusive. [40]</td></tr>
<tr><td>–phase-ref-sc</td><td>Reference phase used to integrate the SC frames. Use a self-estimate of the phase, fitted by poly. (SELF_REF) Use the FT phase only, interpolated in lbd (PHASE_REF) Use the FT+MET-SEP.UV phase (IMAGING_REF). <SELF_REF | PHASE_REF | IMAGING_REF | AUTO | NONE> [AUTO]</td></tr>
<tr><td>–output-phase-sc</td><td>With DIFFERENTIAL, the mean group-delay and mean phases are removed from the output VISPHI in the final OIFITS file. With ABSOLUTE, the VISPHI is kept unmodified. With SELF_VISPHI, the internal differential phase between each spectral channel and a common reference channel is computed. <DIFFERENTIAL | ABSOLUTE | AUTO | SELF_VISPHI> [AUTO]</td></tr>
<tr><td>–output-phase-channels</td><td>range (string in the form [min,max]) of channels to use a SELF_VISPHI phase reference. [0,0]</td></tr>
<tr><td>–outlier-fraction-threshold</td><td>Flag channels with more than this fraction of the frames affected by outliers or cosmics. These are typically detected with the thresholds options in chi2 of the fringe-fit. [0.5]</td></tr>
<tr><td>–flat-flux</td><td>Normalize the flux (stored in OI_FLUX binary extension) with instrument transmission recorded in the nput P2VM calibration map. Consequently, the flux quantity is either the intensity level recorded n the detector, thus including the instrument transmission (FALSE); or the intensity level at the instrument entrance (TRUE). [FALSE]</td></tr>
<tr><td>–average-sky</td><td>Average the SKYs into a master SKY. If FALSE, the recipe loops over the SKY to reduce each OBJECT with a different SKY. [FALSE]</td></tr>
<tr><td>–reduce-acq-cam</td><td>If TRUE, reduced ACQ_CAM images. [FALSE]</td></tr>
<tr><td>–color-wave-correction</td><td>If TRUE, creates a new OI_WAVELENGTH_EFF with corrected wavelength. [FALSE]</td></tr>
</table>

Pseudo code gravity_vis

```
# (1) To use this recipe the frameset must contain the p2vm, wave and
# gain calibration file.
frameset = recipe_input_frameset
parameters = recipe_input_parameters

# Dispatch the frameset
p2vmcalib_frameset = gravi_frameset_extract_p2vm_map(frameset)
darkcalib_frameset = gravi_frameset_extract_dark_map(frameset)
wavecalib_frameset = gravi_frameset_extract_wave_map(frameset)
dark_frameset = gravi_frameset_extract_dark_data(frameset)
flatcalib_frameset = gravi_frameset_extract_flat_map(frameset)
badcalib_frameset = gravi_frameset_extract_bad_map(frameset)
dispcalib_frameset = gravi_frameset_extract_disp_map(frameset)
metpos_frameset = gravi_frameset_extract_met_pos(frameset)
diamcat_frameset = gravi_frameset_extract_diamcat_map(frameset)
eop_frameset = gravi_frameset_extract_eop_map(frameset)
patch_frameset = gravi_frameset_extract_patch(frameset)
static_param_frameset = gravi_frameset_extract_static_param(frameset)
recipe_frameset = gravi_frameset_extract_fringe_data(frameset)
sky_frameset = gravi_frameset_extract_sky_data(frameset)

# (2) Identify the DARK in the input frameset
if dark_frameset:
    data = gravi_data_load_rawframe(dark_frameset[0])
    gravi_data_patch(data, patch_frameset)
    gravi_data_detector_cleanup(data, parameters)  # see algo 10.1
    # Compute dark
    dark_map = gravi_compute_dark(data)  # see algo 10.2
    FREE(gravi_data_delete, data)
    # Save the dark map
    gravi_data_save_new(dark_map, frameset, NULL, NULL, parameters )
elseif darkcalib_frameset:
    dark_map = gravi_data_load_frame(darkcalib_frameset[0])

# Identify the BAD in the input frameset
badpix_map = gravi_data_load_frame(badcalib_frameset[0])
# Identify the FLAT in the input frameset
profile_map = gravi_data_load_frame(flatcalib_frameset[0])
# Identify the WAVE in the input frameset
wave_map = gravi_data_load_frame(wavecalib_frameset[0])
# Identify the P2VM in the input frameset
p2vm_map = gravi_data_load_frame(p2vmcalib_frameset[0])

# get extrapixel parameter
param_extrapixel = gravi_pfits_get_extrapixel_param(gravi_data_get_header(p2vm_ma
```

```
# Load the DISP_MODEL in the input frameset
if dispcalib_frameset:
    disp_map = gravi_data_load_frame(dispcalib_frameset[0])

# Load the DIODE_POSITION in the input frameset
if metpos_frameset:
  frame = cpl_frameset_get_position(metpos_frameset, 0)
  diodepos_data = gravi_data_load_frame(metpos_frameset[0])

# Load the EOP_PARAM
if eop_frameset:
    eop_map = gravi_data_load_frame(eop_frameset[0])

# read constant parameter from calibration file
# Load the STATIC_PARAM Parameter
if static_param_frameset:
  static_param_data = gravi_data_load_frame(static_param_framese[0])

# Load the DIAMETER_CAT
if diamcat_frameset:
    diamcat_data = gravi_data_load_frame(diamcat_frameset[0])

# Select the PRO CATG (based on first frame)
# Mode for the SKY
# Loop on input SKY frames to be reduced
nb_sky   = size(sky_frameset)
sky_maps = nb_sky * size(gravi_data)

for isky in nb_sky:
    # Load the raw SKY
    data = gravi_data_load_rawframe(sky_frameset[isky])
    gravi_data_patch(data, patch_frameset)
    gravi_data_detector_cleanup(data, parameters)  # see algo 10.1
    # Compute the SKY map
    sky_maps[isky] = gravi_compute_dark(data)  # see alog 10.2
    # Save the SKY map
    if not gravity.preproc.average-sky:
        gravi_data_save_new(sky_maps[isky], frameset, parameters)

# Average the sky if requested
if gravity.preproc.average-sky:
    msky_map = gravi_average_dark(sky_maps, nb_sky)
    gravi_data_save_new(msky_map, frameset, parameters)
    # Add all sky to used_frameset, and move pointers
```

```
    cpl_frameset_join(used_frameset, sky_frameset)
    nb_sky = 1

# Loop on input RAW frames to be reduced
nb_frame = recipe_frameset.size
for ivis in nb_frame:
    # Identify the SKY for this OBJECT
    isky = nb_sky>0 ? ivis % nb_sky : 0
    if nb_sky == 0:
        # No SKY in the frameset
    elseif gravity.preproc.average-sky:
        # Use master SKY already computed, already in frameset
    else:
        # SKY already computed, add in the used_frameset
        frame = sky_frameset[isky]
        # Add this frame to the current_frameset as well
        cpl_frameset_insert(current_frameset, cpl_frame_duplicate(frame))
    # Reduce the OBJECT
    data = gravi_data_load_rawframe(recipe_frameset[ivis])
    gravi_data_patch(data, patch_frameset)
    gravi_data_detector_cleanup(data, parlist)  # see algo 10.1
    # Option save the bias-subtracted file
    if gravity.dfs.bias-subtracted-file:
        gravi_data_save_new(data, frameset, parameters)
    # Open the shutters
    gravi_data_check_shutter_open(data)
    # Extract spectrum see algo 10.3
    preproc_data = gravi_extract_spectrum(data, profile_map, dark_map,
                                          badpix_map, sky_maps[isky],
                                          parlist, GRAVI_DET_ALL)
    # Option save the spectrum file
    if gravity.dfs.spectrum-file:
        gravi_data_save_new(preproc_data, frameset, paremters)

    # (3) Rescale to common wavelength see algo 10.5
    gravi_align_spectrum(preproc_data, wave_map, p2vm_map, GRAVI_DET_ALL)
    # Option save the spectrum-aligned file
    if gravity.dfs.spectrum-file:
        gravi_data_save_new(preproc_data, frameset, paremters)
    # Preproc the Acquisition Camera
    if gravity.test.reduce-acq-cam:
        gravi_preproc_acqcam(preproc_data, data, badpix_map)
    # Option save the preproc file
    if gravity.dfs.preproc-file:
        gravi_data_save_new(preproc_data, frameset, paremters)
```

```
# Copy metrology and subtract background to preproc
gravi_data_move_ext(preproc_data, data, GRAVI_METROLOGY_EXT)
if dark_map:
    gravi_subtract_met_dark(preproc_data, dark_map)
# Move extensions from raw_data and delete it

# (4) Compute the flux and visibilities for each telescope and
# per acquisition with the P2VM applied to preproc_data
p2vmred_data = gravi_compute_p2vmred(preproc_data, p2vm_map, mode, parameters
# Reduce the Acquisition Camera and delete data
if gravity.test.reduce-acq-cam:
    gravi_reduce_acqcam(p2vmred_data, preproc_data, sky_maps[isky], static_pa
# Move extensions and delete preproc

# (5) compute additional signals
# Reduce the OPDC table
gravi_compute_opdc_state(p2vmred_data)
# Reduce the metrology into OI_VIS_MET
gravi_metrology_reduce(p2vmred_data, eop_map, static_param_data, diodepos_dat
# Compute the uv and pointing directions with ERFA
gravi_compute_pointing_uv(p2vmred_data, eop_map)
# Compute the QC0 about tau0 from piezo signals
gravi_compute_tau0(p2vmred_data)
# Compute QC for the Fringe Tracker injection
gravi_compute_qc_injection(p2vmred_data)
# Compute QC for the Fringe Tracker OPD calculation
gravi_compute_qc_ft_opd_estimator(p2vmred_data)
# Find outliers
gravi_compute_outliers(p2vmred_data, parlist)
# Compute the SNR_BOOT and GDELAY_BOOT
gravi_compute_snr(p2vmred_data, parlist)
# Compute the signals for averaging
gravi_compute_signals(p2vmred_data, disp_map, parlist)
# Compute rejection flags for averaging
gravi_compute_rejection(p2vmred_data, parlist)
# Save the p2vmreduced file
if gravity.dfs.p2vmred-file:
    gravi_data_save_new(p2vmred_data, frameset, parameters)
# Loop on the wanted sub-integration
current_frame = 0
while current_frame >= 0:
    # Visibility and flux are averaged and the followings
    # are saved in tables VIS, VIS2 and T3
    tmpvis_data = gravi_compute_vis(p2vmred_data, parlist, &current_frame)
    # Set the mean TIME and mean MJD if required
```

```
        if gravity.vis.force-same-time:
            # Force same time for all quantities/baselines
            gravi_vis_force_time(tmpvis_data)

        # Copy the acquisition camera if requested.
        if current_frame < 0 and gravity.test.reduce-acq-cam:
            gravi_data_copy_ext_insname(tmpvis_data, p2vmred_data,
                                        GRAVI_IMAGING_DATA_ACQ_EXT, INSNAME_ACQ)

        # Merge with already existing
        if not vis_data:
            vis_data = tmpvis_data
            tmpvis_data = NULL
        else
            # "Merge with previous OI_VIS"
            gravi_data_append(vis_data, tmpvis_data, 1)

    # Save the astro file, which is a lighter version of the p2vmreduced
    if gravity.dfs.astro-file:
        gravi_data_clean_for_astro(p2vmred_data)
        gravi_data_save_new(p2vmred_data, frameset, parameters)
# End loop on the input files to reduce

# Compute QC parameters
gravi_compute_vis_qc(vis_data)
# Compute the QC parameters of the TF only for CALIB star
gravi_compute_tf_qc(vis_data, diamcat_data)
# Eventually flatten the OI_FLUX
if gravity.vis.flat-flux:
    # Flatten the FLUX with the internal P2VM spectrum
    gravi_flat_flux(vis_data, p2vm_map)
# Perform the normalisation of the SC vis2 and visamp
# to match those of the FT
if gravity.vis.vis-correction-sc != FORCE:
    #Align the SC visibilities on the FT
    gravi_normalize_sc_to_ft(vis_data)
# Correct the wavelength due to target color shifting
if gravity.vis.color-wave-correction:
    gravi_wave_correct_color(vis_data)
# Co-add the observations if requested
if gravity.postprocess.average-vis:
  gravi_average_vis(vis_data)
# Recompute the TIME column from the MJD column
# in all OIFITS tables to follow standard
gravi_vis_mjd_to_time(vis_data)
```

```
# (8) Save the output data file based on the first frame of the frameset
gravi_data_save_new(vis_data, frameset, parameters, proCatg)
```

7.13 gravity_viscal

This recipe calibrates the visibilities acquired on science target using visibilities acquired on a calibrator target. If the DIAMETER_CAT is not provided, the recipe will use the diameter provided in the header to compute the transfer function QC parameters. The corresponding keywords are INS.SOBJ.DIAMETER and FT.ROBJ.DIAMETER. The OI_FLUX data are not yet calibrated. The tag in the DO category can be SINGLE/-DUAL and CAL/SCI. The tag in the PRO.CATG category will be SINGLE/DUAL and CAL/SCI depending on the input tag.

1. Loop on all input CALIB files, compute the TF for each of them and write the corresponding product
2. Loop on all input SCIENCE files, interpolate the TF at that time, calibrate, and write the corresponding product

Input

DO.CATG	short description
SINGLE_SCI_VIS (>=1)	visibilities on sciences
SINGLE_CAL_VIS (>=1)	visibilities on calibrators
SINGLE_CAL_VISPHI (opt)	visibility on dedicated calibrator for science visibility phase
DIAMETER_CAT (opt)	catalog of stellar diameters

Output

PRO.CATG	short description
SINGLE_SCI_VIS_CALIBRATED	calibrated science visibilities
SINGLE_CAL_TF	Transfer Function (TF) estimated on calibrators
SINGLE_SCI_TF	TF interpolated at the time of sciences

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–delta-time-calib	Delta time to interpolate the TF [s]
. [3.6e+03]	
–force-calib	Force the calibration, don’t check setup. [FALSE]

–nsmooth-tfvis-sc	Smooth the TF spectrally by this number of spectral bin, to enhance SNR (only apply to VIS2, VISPHI, VISAMP, T3PHI, T3AMP). This parameter is ignored in spectral mode LOW. [0]
–nsmooth-tfflux-sc	Smooth the TF spectrally by this number of spectral bin, to enhance SNR (only apply to FLUX, RVIS, IVIS). This parameter is ignored in spectral mode LOW. [0]
–maxdeg-tfvis-sc	Fit the TF spectrally by a polynomial to enhance SNR (only apply to VIS2, VISPHI, VISAMP, T3PHI, T3AMP). This parameter is ignored in spectral mode LOW. [5]
–calib-flux	Normalize the FLUX by the calibrator. [FALSE]
–smoothing	control smoothing of transfer function (TF) TRUE do smoothing FALSE Skip smoothing. [TRUE]
–separate-phase-calib	Use a specific calibrator file, identified with the CAL_VISPHI tag, for calibrating the visibility phase. The remaining calibrators shall be used for the other calibrated quantities. [FALSE]

Pseudo code gravity_viscal

```
# To use this recipe the frameset must contain
# at least one VIS_*_CAL frame or TF_*_CAL frame.
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Extract a set of vis_calib and vis_sci data frameset
vis_calib_frameset = gravi_frameset_extract_vis_calib(frameset)
vis_sci_frameset = gravi_frameset_extract_vis_science(frameset)
tf_calib_frameset = gravi_frameset_extract_tf_calib(frameset)
diamcat_frameset = gravi_frameset_extract_diamcat_map(frameset)
# Get the number of the frames contained in the frameset
# Load or compute of the transfer function
# Load the DIAMETER_CAT
if diamcat_frameset:
    diamcat_data = gravi_data_load_frame(diamcat_frameset[0])

# Loop on the TF to compute
for j in nb_frame_calib:
    # Load the VIS data and compute TF
    vis_data = gravi_data_load_frame(vis_calib_frameset[j])
    vis_calib = gravi_compute_tf(vis_data, diamcat_data)
    # Smooth the TF if required
    if gravi_data_get_spec_res(vis_calib) != "LOW":
        # "LOW spectral resolution -> don't smooth the TF"
    elseif not gravity.viscal.smoothing:
        # "smoothing parameter == FALSE -> don't smooth the TF"
```

```
    else:
        smooth_vis_sc = "gravity.viscal.nsmooth-tfvis-sc"
        smooth_flx_sc = "gravity.viscal.nsmooth-tfflux-sc"
        maxdeg_sc = "gravity.viscal.maxdeg-tfvis-sc"
        gravi_vis_smooth(vis_calib, smooth_vis_sc, smooth_flx_sc, maxdeg_sc)
    # Save the TF file
    gravi_data_save_new(vis_calib, frameset, parameters)
    # Store this successfull TF
    vis_calibs[nb_calib] = vis_calib
    # Update the frameset -- now used as calibration

# Loop on the TF to load
for j in nb_frame_tf:
    vis_calib = gravi_data_load_frame(tf_calib_frameset[j])
    # Store this successfull TF
    vis_calibs[nb_calib] = vis_calib

# Compute the zero of the metrology if several TF are availables
if nb_calib > 1:
  # Compute the zero of the metrology
  zero_data = gravi_compute_zp(vis_calibs, nb_calib)
  gravi_data_save_new(zero_data, frameset, "output.fits", parameters)

# Apply the TF to the SCIENCE files of the frameset
# Loop on the SCI files to calibrate
for i in nb_frame_sci:
    vis_data = gravi_data_load_frame(vis_sci_frameset[i])
    tf_science = gravi_data_duplicate(vis_data)
    calibrated = gravi_calibrate_vis(vis_data, vis_calibs, nb_calib, tf_science, p
    # Save calibrated visibilities
    data_mode = gravi_data_frame_get_mode(frame)
    gravi_data_save_new(calibrated, frameset, parameters)
    # Save TF interpolated at the science visibilities
    data_mode = gravi_data_frame_get_mode(frame)
    gravi_data_save_new(tf_science, frameset, parameters)
```

7.14 gravity_vis_from_p2vmred

This recipe averages the real-time data of P2VMRED files into a VIS product. It allows to run the reduction with different parameters (for instance for SNR thresholding) without having to re-reduce the files from scratch. Typically the reduction is 4x faster when started from this intermediate product. The tag in the DO category can be SINGLE/DUAL and CAL/SCI. They should reflect the mode (SINGLE or DUAL) and the DPR.CATG of the observation (SCIENCE or CALIB). The tag in the PRO.CATG category will be SINGLE/DUAL and CAL/SCI depending on the input tag.

1. Load the input file (loop on input files)
2. Update the selection flag
3. Average the real-time visibilities
4. Write the product

Input

DO.CATG	short description
SINGLE_SCI_P2VMRED	Input intermediate product

Output

PRO.CATG	short description
SINGLE_SCI_VIS	OIFITS with uncalibrated visibilities

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]
–average-vis	Average the results from the different input files (if any) in the output product, instead of simply appending them. [FALSE]
–chi2r-threshold	Threshold in chi2r of the fringe-fit to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [50.0]
–chi2r-sigma	Threshold in chi2r of the fringe-fit (in unit of the the std of chi2r in the spectral direction) to declare a bad value. This is usefull to detect outliers or cosmic in individual frames. [100.0]
–nsmooth-snr-ft	Number of samples to average coherently when computing the real-time SNR and GDELAY of the FT (shall correspond to the atmospheric coherence time). The integration window runs from -nsmooth -> +nsmooth. [5]
–snr-min-ft	SNR threshold to accept FT frames (>0). It raises the first bit («0) of column REJECTION_FLAG of FT. [3.0]
–global-state-min-ft	Minimum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [2.0]
–global-state-max-ft	Maximum OPDC state to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [4.0]

–state-min-ft	Minimum OPDC state per baseline to accept FT frames (>=0) It raises the second bit («1) of column REJECTION_FLAG of FT. [1.0]
–tracking-min-sc	Minimum ratio of accepted FT frames in order to accept a SC frames (0..1), that is, for each SC DIT, the fraction of the time the REJECTION_FLAG of the FT is not 0. It raises the first bit («0) of column REJECTION_FLAG of SC. [0.8]
–vfactor-min-sc	vFactor threshold to accept SC frame (0..1). [0.1]
–opd-pupil-max-sc	Maximum OPD_PUPIL (abs) to accept SC frames. It raises the third bit («2) of column REJECTION_FLAG of SC. [9999.0]
–opd-pupil-stddev-max-sc	Maximum OPD_PUPIL_STDDEV to accept SC frames. It raises the fourth bit («3) of REJECTION_FLAG of SC. [2.9e-07]
–max-frame	Maximum number of frames to integrate coherently into an OIFITS entry. [10000]
–force-same-time	Force all baseline/quantities to have strictly the same TIME and MJD columns. [FALSE]
–debias-sc	Subtract the V2 bias from SC. [TRUE]
–debias-ft	Subtract the V2 bias from FT. [TRUE]
–nboot	Number of bootstraps to compute error (1..100). [20]
–vis-correction-sc	Correction of SC visibility from losses due to long integration, using the measured visibility losses with the FT (VFACTOR and/or PFACTOR) or by forcing the SC visibilities to match those of the FT (FORCE). Possible choices are:. <VFACTOR | PFACTOR | VFACTOR_PFACTOR | FORCE | NONE> [VFACTOR]
–vis-correction-ft	Correction of FT visibility from losses due to long integration, using a sliding window PFACTOR, or its square. Choices are: <NONE | PFACTOR | PFACTOR_SQUARED> [NONE]
–pfactor-window-length	Length of the sliding window used to calculate the FT PFACTOR. For each FT frame, the window will run from -window-length to +window-length inclusive. [40]
–phase-ref-sc	Reference phase used to integrate the SC frames. Use a self-estimate of the phase, fitted by poly. (SELF_REF) Use the FT phase only, interpolated in lbd (PHASE_REF) Use the FT+MET-SEP.UV phase (IMAGING_REF). <SELF_REF | PHASE_REF | IMAGING_REF | AUTO | NONE> [AUTO]

–output-phase-sc	With DIFFERENTIAL, the mean group-delay and mean phases are removed from the output VISPHI in the final OIFITS file. With ABSOLUTE, the VISPHI is kept unmodified. With SELF_VISPHI, the internal differential phase between each spectral channel and a common reference channel is computed. <DIFFERENTIAL | ABSOLUTE | AUTO | SELF_VISPHI> [AUTO]
–output-phase-channels) of channels o use a SELF_VISPHI phase reference. [[0,0]]	range (string in the form [min,max]
–outlier-fraction-threshold	Flag channels with more than this fraction of the frames affected by outliers or cosmics. These are typically detected with the thresholds options in chi2 of the fringe-fit. [0.5]
–reduce-acq-cam	If TRUE, reduced ACQ_CAM images. [FALSE]
–use-existing-rejection	Use existing rejection flags (ignore related options). [FALSE]

Pseudo code gravity_vis_from_p2vmred

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Dispatch the frameset
recipe_frameset = gravi_frameset_extract_p2vmred_data(frameset)
# To use this recipe the frameset must contain a P2VMREDUCED file.
# check it
# Insert calibration frame into the used frameset
# Select the PRO CATG(based on first frame)
frame_tag = cpl_frame_get_tag(cpl_frameset_get_position(recipe_frameset, 0))
if(frame_tag is GRAVI_P2VMRED_DUAL_CALIB):
    proCatg = GRAVI_VIS_DUAL_CALIB
    mode = "gravi_dual"
elif(frame_tag is GRAVI_P2VMRED_DUAL_SCIENCE):
    proCatg = GRAVI_VIS_DUAL_SCIENCE
    mode = "gravi_dual"
elif(frame_tag is GRAVI_P2VMRED_SINGLE_CALIB):
    proCatg = GRAVI_VIS_SINGLE_CALIB
    mode = "gravi_single"
elif(frame_tag is GRAVI_P2VMRED_SINGLE_SCIENCE):
    proCatg = GRAVI_VIS_SINGLE_SCIENCE
    mode = "gravi_single"
else
    proCatg = "UNKNOWN"
    mode = "gravi_single"
# Loop on input RAW frames to be reduced
```

```
nb_frame = size(recipe_frameset)
for ivis in nb_frame:
    p2vmred_data = gravi_data_load_frame(recipe_frameset[ivis])
    # Compute rejection flags for averaging
    if("gravity.signal.use-existing-rejection"):
        # Dont recompute SNR and selection, use the existing one
    else:
        # Find outliers
        gravi_compute_outliers(p2vmred_data, parameters)
        # Compute the SNR/GDELAY
        gravi_compute_snr(p2vmred_data, parameters)
        # Compute rejection flags for averaging
        gravi_compute_rejection(p2vmred_data, parameters)
    # Loop on the wanted sub-integration
    cpl_size current_frame = 0
    while(current_frame >= 0):
    # Visibility and flux are averaged and the followings
    # are saved in Visibility data in tables VIS, VIS2 and T3
    tmpvis_data = gravi_compute_vis(p2vmred_data, parameters, &current_frame)
    # Set the mean TIME and mean MJD if required
    if("gravity.vis.force-same-time")
        #Force same time for all quantities/baselines
        gravi_vis_force_time(tmpvis_data)
    # Copy the acquisition camera if requested
    if(current_frame < 0 && "gravity.test.reduce-acq-cam"):
        # Copy ACQ into the VIS file
        gravi_data_copy_ext_insname(tmpvis_data, p2vmred_data, GRAVI_IMAGING_DATA_
    # Merge with already existing
    if not vis_data:
        vis_data = tmpvis_data tmpvis_data = NULL
    else:
        # Merge with previous OI_VIS
        gravi_data_append(vis_data, tmpvis_data, 1)
# End loop on the input files to reduce
# Compute QC parameters
gravi_compute_vis_qc(vis_data)
# Perform the normalisation of the SC vis2 and visamp
# to match those of the FT
if gravity.vis.vis-correction-sc is not FORCE :
    # Align the SC visibilities on the FT
    gravi_normalize_sc_to_ft(vis_data)
else
    # Dont align the SC visibilities on the FT
# Co-add the observations if requested
if gravity.postprocess.average-vis:
```

```
    # Average the different observation(if any) = EXPERIMENTAL
    gravi_average_vis(vis_data)
else:
    # Dont average the different observation(if any)
# Recompute the TIME column from the MJD column
# in all OIFITS tables to follow standard
gravi_vis_mjd_to_time(vis_data)
# Save the output data file based on the first frame of the frameset
frame = cpl_frameset_get_position(recipe_frameset, 0)
gravi_data_save_new(vis_data, frameset, parameters)
```

7.15 gravity_wavelamp

This recipe is associated to the template gravity_wavelamp. It reduces the raw file obtained with the Argon lamp (WAVELAMP) and process it so that it can be used to calibrate the fiber dispersion (recipe gravity_disp).

1. Extract the spectra of the Argon exposure
2. Interpolate the spectra into a common wavelength table
3. Measure the wavelength position of known Argon lines
4. Write the product

Input

DO.CATG	short description
FLAT	flat calibration (PRO.CATG=FLAT)
BAD	badpixel calibration (PRO.CATG=BAD)
WAVE	wave calibration (PRO.CATG=WAVE)
P2VM	p2vm calibration (PRO.CATG=P2VM)
WAVELAMP_RAW	long exposure of Argon lamp
DARK_RAW	dark of Argon exposure

Output

PRO.CATG	short description
WAVELAMP	spectrum of Argon, with position of lines

Parameters

Name	short description
–static-name	Use static names for the products (for ESO). [FALSE]

Pseudo code gravity_wavelamp

```
frameset = recipe_input_frameset
parameters = recipe_input_parameters
# Identify the input framesets
wavelamp_frameset = gravi_frameset_extract_wavelamp_data(frameset)
dark_frameset      = gravi_frameset_extract_dark_data(frameset)
darkcalib_frameset = gravi_frameset_extract_dark_map(frameset)
p2vmcalib_frameset = gravi_frameset_extract_p2vm_map(frameset)
wavecalib_frameset = gravi_frameset_extract_wave_map(frameset)
flatcalib_frameset = gravi_frameset_extract_flat_map(frameset)
badcalib_frameset = gravi_frameset_extract_bad_map(frameset)
wave_param_frameset = gravi_frameset_extract_wave_param(frameset)

# Identify the DARK in the input frameset
if(dark_frameset):
    data = gravi_data_load_rawframe(dark_frameset[0])
    gravi_data_detector_cleanup(data, parameters)  # see algo 10.1
    # Compute dark
    dark_map = gravi_compute_dark(data)  # see algo 10.2
    # Save the dark map
    gravi_data_save_new(dark_map, frameset, parameters)
elif darkcalib_frameset:
    dark_map = gravi_data_load_frame(darkcalib_frameset[0])

# Identify the BAD in the input frameset
badpix_map = gravi_data_load_frame(badcalib_frameset[0])
# Identify the FLAT in the input frameset
profile_map = gravi_data_load_frame(flatcalib_frameset[0])
# Identify the WAVE in the input frameset
wave_map = gravi_data_load_frame(wavecalib_frameset[0])
# Identify the P2VM in the input frameset
p2vm_map = gravi_data_load_frame(p2vmcalib_frameset[0])

# Load input WAVELAMP_RAW
data = gravi_data_load_rawframe(wavelamp_frameset[0])
gravi_data_detector_cleanup(data, parameters)  # see algo 10.1

# Collapse ARGON
gravi_data_get_cube(argon_data, GRAVI_IMAGING_DATA_SC_EXT).median
```

```
# (1) Extract spectrum
preproc_data = gravi_extract_spectrum(argon_data, profile_map, dark_map,
                                      badpix_map, NULL, parameters,
                                      GRAVI_DET_ALL)  #see algo 10.3
# (2) Rescale to common wavelength
gravi_align_spectrum(preproc_data, wave_map, p2vm_map, GRAVI_DET_ALL)  # see algo
if wave_param_frameset:
  wave_param = gravi_data_load_frame(wave_param_frameset[0])
# (3) Compute position
gravi_compute_argon_pos(preproc_data, wave_param)
# (4) Save the output data file
gravi_data_save_new(preproc_data, frameset, parameters)
```

8 Re-creating the Static Calibration

8.1 Dispersion model

The **DISP_MODEL** static calibration can be recomputed with the recipes **gravity_wavelamp** and **gravity_disp**. The principle is to accurately measure the interferometric phases obtained for various position of the FDDL (= various strechting of the fibers) at the wavelengths of known Argon lines. The following dedicated RAW data are required:

WAVELAMP is a spectrum of the internal argon lamp.

DISP are interferometric observations of the internal source for various position of the FDDL (= different strech of the fibers).

8.2 Earth Orientation Parameters

The **EOP_PARAM** static calibration can be recomputed with the recipe **gravity_retrieve_eop**, which shall query the IERS webpage to obtain the best estimate of the past and futur EOP.

8.3 Metrology diode positions

There is no recipe to create a new calibration file, but the **DIODE_POSITION** static calibration can be update by editing the fits file.

9 Detailed description of the data content

9.1 Recommended tools to browse data

RAW files can be conveniently opened with the last version of *QFitsView* [2] from Thomas Ott.

OIFITS product files can be conveniently open with the last version of *OIFits Explorer* [1] from the JMMC.

The consortium has built a python script to visualise most of the GRAVITY data, *run_gravi_visual.py*.

9.2 Table structure common to all data

The INSNAME header keyword specifies the combiner to which a table refers to, and thus allows to cross-reference with other tables. The EXTVER keyword specifies the instance of a table repeated in the OIFITS file to allow fast search in the tables (see python FITS class for instance). They can take the values: GRAVITY_SC' (10), GRAVITY_SC_P1' (11), GRAVITY_SC_P2' (12), GRAVITY_FT' (20), GRAVITY_FT_P1' (21), GRAVITY_FT_P2' (22).

The polarisation 'P1' in output products correspond to the 'S' regions in IMAGING_DETECTOR tables. The polarisation 'P2' in output products correspond to the 'P' regions in IMAGING_DETECTOR tables.

The IMAGING_DETECTOR_SC and IMAGING_DETECTOR_FT tables store the detector configurations based on the VLTI interface control document [5].

The IMAGING_DATA_SC and IMAGING_DATA_FT extensions store the detector data. The SC data are stored as image list, while the FT data are stored as tables.

The OI_WAVELENGTH tables store the wavelength table following the OIFITS standard. They shall be associated to the SC or FT using the INSNAME or EXTVER keywords.

OIFITS tables storing quantities per-beam (OI_FLUX) have a total of NDIT x 4 rows (or NEXP x 4 rows for final product). In these tables, the four beams are always ordered following 12341234...

OIFITS tables storing quantities per-baseline (OI_VIS, OI_VIS2) have a total of NDIT x 6 rows (or NEXP x 6 rows for final product). In these tables, the six baselines are always ordered following 123456123456... The baselines 1 to 6 are always the pair of the beams 1-2,1-3,1-4,2-3,2-4,3-4.

9.3 RAW calibration data

RAW calibration data are set of data acquired on internal calibration source to be reduced by the recipe gravi_all_dark and gravi_all_p2vm in order to produce master calibration frame (DARK, FLAT, BAD PIXEL and WAVE maps) characterising the detector pixels and P2VM.

The data contain the following tables :

OPTICAL_TRAIN : Optical train configuration (see [5])

IMAGING_DATA_ACQ : Data of the imaging camera (image cube)

IMAGING_DETECTOR_SC : Configuration of the SC detector

IMAGING_DETECTOR_FT : Configuration of the FT detector

IMAGING_DATA_SC : Images of the SC camera (image cube)

IMAGING_DATA_FT : Images of the FT camera

OPDC : OPD Controler data (fringe tracker)

FDDL : Fiber Delay Line position

METROLOGY : Metrology data

IMAGING_DETECTOR_SC and IMAGING_DETECTOR_FT

The REGNAME column must refer to an output of the integrated optic. It is defined as the following [1..3][2..4]-[A,B,C,D]-[S-P]. The 2 first numbers define the 2 entrance ports connected to the output, the median letter correspond to the phase shift A, B, C or D, and the final letter is the polarisation of the region (S or P). Ex : 13-A-S is the A output of the T1-T3 pair with the polarisation S. The IMAGING_DETECTOR_SC tables contain the following columns:

Column	Size	Unit	Description
REGION	I		The region number that is being described by this row.
DETECTOR	I		The detector that is on this region, index defined in INS_DESCRIPTION.
CORRELATION	I		Correlation type: 0=background (no signal), 1=photometric, 2=interferometric.
REGNAME	16A		Detector region name, to match the IMAGING_DATA table.
CENTER	2I	PIXEL	This gives the position of the centre of the spectrum in the x and y dimension.
LEFT	2I	PIXEL	SC ONLY - This gives the position of the left of the spectrum in the x and y dimension.
HALFLEFT	2I	PIXEL	SC ONLY -This gives the position of the halfleft of the spectrum in the x and y dimension.
RIGHT	2I	PIXEL	SC ONLY - This gives the position of the right of the spectrum in the x and y dimension.
HALFRIGHT	2I	PIXEL	SC ONLY - This gives the position of the half right of the spectrum in the x and y dimension.

IMAGING_DATA_FT

The IMAGING_DATA_FT tables contain the following columns:

Column	Size	Unit	Description
REGION	I		The region number that is being described by this row.
REGNAME	16A		Detector region name, to match the IMAGING_DATA table.

OPDC

This table contains the data coming from the OPD controller. It is mainly the command applied to the PIEZO and VLTI delay lines.

The OPDC tables contain the following columns:

Column	Size	Unit	Description
TIME	J	μs	Time tag for this exposure, the effective centroid from the MJD_OBS date.
STATE	J		
STEPS	J		
BASELINE_STATE	J		
PIEZO_DL_OFFSET	4E	V	Command sent to GRAVITY's internal actuator at the current iteration.
VLTI_DL_OFFSET	4E	m	Command sent to the main VLTI delay lines at the current iteration.
KALMAN_PIEZO	4E	rad	Impact of GRAVITY's internal actuator on OPD at the current iteration.
OPD	6E	rad	Phase residual measured at the current iteration.
KALMAN_OPD	6E	rad	Phase residual predicted by the Kalman for the current iteration.

The $\mathrm{T2B}$ matrix converts telescope quantities to baseline quantities.

$$\mathrm{T2B} = \begin{pmatrix} 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \tag{1}$$

The internal modulation of GRAVITY is derived from

$$\mathrm{modulation} = \frac{\pi}{8}\left((\mathrm{STEPS} \gg 4i)\&15\right) \quad i \in [0,1,2,3], \tag{2}$$

where $\gg$ is the bitwise "right shift" operator, and $\&$ is the bitwise "and" operator.

The phase residuals, without the modulation, wrapped in the interval $[-\pi;+\pi]$, are obtained with

$$\mathrm{residuals} = (\mathrm{OPD} - \mathrm{T2B} \cdot \mathrm{modulation} + \pi)\%(2\pi) - \pi. \tag{3}$$

The pseudo open loop disturbances are reconstructed with

$$\text{disturbances} = \text{T2B}\cdot\text{KALMAN\_PIEZO}+(\text{OPD}-(\text{KALMAN\_OPD}-\pi))\%(2\pi)+(\text{KALMAN\_OPD}-\pi). \quad (4)$$

FDDL

This table contains the data coming from the FDDL controller. It is mainly the command applied to the fibered differential delay lines.

The FDDL tables contain the following columns:

Column	Size	Unit	Description
TIME	J	μs	Time tag for this exposure, the effective centroid from the MJD_OBS date.
FT_POS	4E	V	
SC_POS	4E	V	
OPL_AIR	4E	m	

METROLOGY

This table contains metrology measurement: phase differences between reference star and target star as function of time.

The METROLOGY tables contain the following columns:

Column	Size	Unit	Description
TIME	J	μs	Time tag for this exposure, the effective centroid from the MJD_OBS date.
VOLT	80E	V	
POWER_LASER	E	mV	
LAMBDA_LASER	E	m	

9.4 RAW science data

RAW science data are produced during on sky observation of calibrator or science target by the instrument. They are reduced by the recipe gravity_vis.

The data contain the following tables :

ARRAY_DESCRIPTION : Description of the telescope array (see [5])

ARRAY_GEOMETRY : Positions of the used telescope (see [5])

OPTICAL_TRAIN : Optical train configuration (see [5])

IMAGING_DATA_ACQ : Data of the imaging camera (image cube)

IMAGING_DETECTOR_SC : Configuration of the SC detector

IMAGING_DETECTOR_FT : Configuration of the FT detector

IMAGING_DATA_SC : Images of the SC camera (image cube)

IMAGING_DATA_FT : Images of the FT camera

OPDC : OPD Controler data (fringe tracker)

FDDL : Fiber Delay Line position

METROLOGY : Metrology data

9.5 P2VM product

Visibility to pixels matrix contains the beam combiner calibration matrix in P2VM table (transmission, coherence and phase) for the three data sources (P2VM_SC, P2VM_FT and P2VM_MET).

The products contain the following tables :

IMAGING_DETECTOR_SC : copied from raw data

IMAGING_DETECTOR_FT : copied from raw data

OI_WAVELENGTH : computed from the minimum and the maxiumum wavelength and the spectral resolution of the mode

OI_FLUX : store the internal instrument transmission, measured on the internal light, following the OIFITS standard, in arbitrary units. These tables are used to later calibrate the measured flux of the object using the spectral shape of the internal transmission.

P2VM_SC : P2VM of the SC beam combiner

P2VM_FT : P2VM of the FT beam combiner

P2VM_MET : P2VM of the metrology

P2VM_SC and P2VM_FT tables

The P2VM_SC and P2VM_FT tables contain the following columns:

Column	Size	Unit	Description

REGNAME	16A		Detector region name, to match the IMAGING_DATA table.
TRANSMISSION	ntel × nwave [E]		For each region (= output of the combiner), a ntel × nwave image with the transmission of each input beam in this region. Since the combination scheme is pairwise, normaly only 2 rows of this image shall be non-zero.
COHERENCE	nbase × nwave [E]		For each region (= output of the combiner), a nbase × nwave image with the instrumental visibility of each pair of input beam in this region. Since the combination scheme is pairwise, normally only one single rows of this image shall be non-zero.
PHASE	nbase × nwave [E]		For each region (= output of the combiner), a nbase × nwave image with the instrumental phase in radian of each pair of input beam in this region. Since the combination scheme is pairwise, normaly only one single rows of this image shall be non-zero.
C_MATRIX	nbase × nwave [E]		Normalisation matrix

9.6 *_VIS and *_TF products

The files with PRO.CATG=*_VIS and *_TF follow the OIFITS standard, version 2. All information can be found in [4].

The OI_VIS, OI_VIS2 and 0I_T3 tables contain one row per baseline and per corresponding RAW exposure, that is all the frames from the individual exposure are averaged together.

The OI_FLUX tables contain one row per telescope and per corresponding RAW exposure.

The OIFITS tables are associated to the SC or FT using the INSNAME (and/or EXTVER) keywords.

9.7 SPECTRUM, PREPROC products

These are intermediate products used to debug the pipeline.

Both files contain the extracted spectra of each output of the combiners. In SPECTRUM, the spectra are in pixel space, thus on different wavelength grids. In PREPROC, the spectra have been re-interpolated into a common wavelength grid.

The products contain the following tables :

IMAGING_DETECTOR_SC/FT : copied from raw data

SPECTRUM_DATA_SC/FT : Computed spectra in pixels space for SPECTRUM product, or re-interpolated in PREPROC product.

SPECTRUM_FLAT_SC : Computed spectra from the FLAT in pixels space for SPECTRUM product, or re-interpolated in PREPROC product.

OI_WAVELENGTH : Only for PREPROC file, this table is copied form the WAVE used for re-interpolation.

Columns in the SPECTRUM_DATA tables

Column	Size	Unit	Description
TIME	J	μs	time of the frame, in [us], from the PRC.ACQ.START time from header (RMN recording start).
DATAi	NWAVE*D	e	The spectrum of the flux from output i of the combiner.
DATAERRi	NWAVE*D	e	The spectrum of the theoretical error of the flux from output i of the combiner, including detector and photonic variances.

Columns in the SPECTRUM_FLAT_SC tables

Column	Size	Unit	Description
DATAi	NWAVE*D	e	The spectrum of the flux from output i of the combiner.
DATAERRi	NWAVE*D	e	The spectrum of the theoretical error of the flux from output i of the combiner, including detector and photonic variances.

9.8 *_P2VMRED products

The files with PRO.CATG=*_P2VMRED use elements of the OIFITS format [4], but are non-standard for the TIME colums. They also include many additional columns to store intermediate signal.

Beware that the OIFITS tables contain one row per frame: the number of row can be huge for the FT tables.

The product contains the following tables :

OI_WAVELENGTH : copied form the WAVE used for re-interpolation.

OI_TARGET : created by the pipeline from the template parameters (see [4] for description).

OI_ARRAY : created by the pipeline from ARRAY_GEOMETRY table of the raw data (see [4] for description).

OI_VIS : computed visibilitites adapted from [4] see below.

OI_FLUX : computed flux adapted from [4] see below.

METROLOGY : copied form the RAW data

OI_VIS_MET : computed phase of the metrology see below.

FDDL : copied form the RAW data

OPDC : copied form the RAW data

If the ACQUISITION camara data are reduced :

IMAGING_DATA_ACQ : reduced images of the acquisition camera (see below)

OI_VIS_ACQ : computed data from the acquisition camera images (see below)

Columns in the OI_VIS table of the SC

TARGET_ID : id listed in OI_TARGET

TIME [μs] : time of the frame, in [us], from the PRC.ACQ.START time from header (RMN recording start).

MJD [day] :

INT_TIME [s] : integration time of this frame

VISDATA [e,e] : complex coherent flux of SC in this frame

VISERR [e,e] : theoretical complex error on the coherent flux

UCOORD [m] : uv-plane of this SC frame

VCOORD [m] : uv-plane of this SC frame

STA_INDEX : station index in the OI_ARRAY

FLAG : T if flagged, F if not flagged

E_U, E_V, E_W : Local celestial {u,v,w} (East, North, Toward observer = OIFITS standard) expressed in local terrestrial (East, North, Up) at Paranal reference. It is useful for recomputing the projected baseline from physical baseline.

E_Az : Vector product of the E_W and the Zenith directions, expressed in the local terrestrial (East, North, Up) at Paranal reference. Sitting on telescope, looking at the target, E_Az points toward left in the horizontal plane.

E_Zd : Vector product of the E_W and E_Az, expressed in the local terrestrial (East, North, Up) at Paranal reference. Sitting on telescope, looking at the target, E_Az points toward Nadir in the plane perpendicular to pointing direction.

SNR : real-time SNR

GDELAY_BOOT [m] : best GD estimate, taking into account closing triangles

SNR_BOOT : best SNR estimate, taking into account closing triangles

FIRST_FT : first FT frame in this SC frame

LAST_FT : last FT frame in this SC frame

NFRAME_FT :

FIRST_MET : first MET frame in this SC frame

LAST_MET : last MET frame in this SC frame

NFRAME_MET :

P_FACTOR : predicted square visibility loss of this SC frame due to photometric flickering (based on the real-time photometry of the FT)

F1F2 Estimate of geometric flux of this frame.

PHASE_MET_FC [rad] : unwrapped FT-SC phase as computed by the DRS algorithm

PHASE_MET_TEL [rad] : unwrapped FT-SC phase as computed by the DRS algorithm, mean of 4 diodes

PHASE_MET_TELFC : differential phase between FC and TEL diodes, average in the complex phasor space. This correspond to an astrometric phase error to be applied to compute astrometry.

OPD_MET_FC [m] : unwrapped SC-FT delay as computed by the TAC algorithm

OPD_MET_TEL [m] : unwrapped SC-FT delay as computed by the TAC algorithm, 4 diodes

OPD_MET_FC_CORR : same as OI_VIS_MET table

OPD_MET_TELFC_MCORR : same as OI_VIS_MET table

OPD_MET_TELFC_CORR : same as OI_VIS_MET table

VISDATA_FT [e,e] : <VISDATA> spectra of FT (integrated in this SC frame)

VISVAR_FT [e**2] : <|VISERR|**2> spectra of FT (integrated in this SC frame)

VISPOWER_FT [e**2] : <|VISDATA|**2> spectra of FT (integrated in this SC frame)

V_FACTOR : predicted square visibility loss of this SC frame (re-interpolation of V_FACTOR_FT on the SC wavelengths)

V_FACTOR_FT : measured visibility loss on the FT

V_FACTOR_WL :predicted square visibility loss in white light for the SC.

OPD_DISP [m] : spectra of OPD introduced by fibers, including dispersion.

GDELAY_DISP [m] :

PHASE_DISP [m] :

GDELAY [m] : real-time GD computed from VISDATA

GDELAY_FT [m] : real-time GD computed from VISDATA_FT

SELF_REF [rad] : self-reference phase.

SELF_REF_COEFF [rad] :

PHASE_REF [rad] : reference phase from FT, actually -1*argVISDATA_FT, re-interpolated in the SC wavelength.

PHASE_REF_COEFF [rad] : polynomial coefficients fit to argVISDATA_FT and used to extrapolate to the SC wavelengths, in units of $(\lambda_{mean}/\lambda - 1)/(\lambda_{max} - \lambda_{min}) * \lambda_{mean}$.

IMAGING_REF [rad] : reference phase for dual-mode, which include the PHASE_REF, the metrology and the sidereal motion.

FRINGEDET_RATIO : fraction of FT frame accepted in this SC frame

REJECTION_FLAG : this frame is accepted/rejected

PUPIL_U [m] : lateral shift of pupil (in uv reference).

PUPIL_V [m] : lateral shift of pupil (in uv reference).

PUPIL_W [m] : focus shift of pupil (in uv reference).

Columns in the OI_VIS table of the FT

TARGET_PHASE [rad] : target phase of the loop, including the Sylvester modulation

STATE : baseline tracking state as reported by OPDC

OPDC_STATE :

SNR : real-time SNR

GDELAY_BOOT [m] : best GD estimate, accounting closing triangles

SNR_BOOT : best SNR estimate, accounting closing triangles

F1F2 Estimate of geometric flux of this frame.

SELF_REF [rad] : self-reference phase.

GDELAY [m] : real-time GD computed from VISDATA

REJECTION_FLAG : this frame is accepted/rejected

Columns in the OI_FLUX table of the SC

TARGET_ID : id listed in OI_TARGET

TIME [us] : time of the frame, in [us], from the PRC.ACQ.START time from header (RMN recording start).

MJD [day] :

INT_TIME [s] : integration time of this frame

FLUX [e] : flux

FLUXERR [e] : theoretical error on flux

STA_INDEX : station index in the OI_ARRAY

CHI2 : reduced chi2 of the fit of the raw data with the P2VM model

FLAG : T if flagged, F if not flagged

FIRST_FT : first FT frame in this SC frame

LAST_FT : last FT frame in this SC frame

NFRAME_FT :

FIRST_MET : first MET frame in this SC frame

LAST_MET : last MET frame in this SC frame

NFRAME_MET :

FIRST_FDDL : first FDDL frame in this SC frame

LAST_FDDL : last FDDL frame in this SC frame

NFRAME_FDDL :

OPD_MET_FC [m] : unwrap SC-FT delay as computed by the TAC algorithm

OPD_MET_TEL [m] : unwrap SC-FT delay as computed by the TAC algorithm, 4 diodes.

PHASOR_MET_TELFC : average over the SC DIT of PHASOR_TELFC in OI_VIS_MET

OPD_MET_FC_CORR :

OPD_MET_TELFC_MCORR :

OPD_MET_TELFC_CORR :

FT_POS [V] : mean FT FDDL strain gauge voltage during this frame

SC_POS [V] : mean SC FDDL strain gauge voltage during this frame

OPL_AIR [m] : optical path length calculated as the scalar product between the telescope position and the target direction

TOTALFLUX_SC [e] : total flux of SC in this SC frame (integrated over spectrum)

TOTALFLUX_FT [e] : total flux of FT in this SC frame (integrated over spectrum)

FDDL [m] : The mean of SC_FDDL and FT_FDDL, where these value are the FDDL strain gauge measurements, corrected from non-linearity and converted in [m].

Columns in the OI_FLUX table of the FT

STATE : telescope state as reported by OPDC

Columns in the OI_VIS_MET table

The table has the same structure as the OI_FLUX table, that is one row per beam (12341234124...), and thus nsample x 4 rows.

PHASE_FC_DRS [rad] : phases at combiner, unwrap by pipeline algorithm (FT-SC)

PHASE_TEL_DRS [rad] : 4 diodes phases at telescope, unwrap by pipeline algorithm (FT-SC)

PHASE_FC_TAC [rad] : phases at combiner, unwrap by TAC algorithm (FT-SC)

PHASE_TEL_TAC [rad] : 4 diodes phases at telescope, unwrap by TAC algorithm (FT-SC)

FLAG_FC, FLAG_TEL : flags computed by TAC algorithm

OPD_FC [m] : OPD at telescope, unwrap by TAC algorithm (SC-FT)

VAMP_FC_FT , VAMP_FC_SC, VAMP_TEL_FT, VAMP_TEL_SC: Volt amplitudes

OPD_TEL [m] : 4 diodes OPD at telescope, unwrap by TAC algorithm (SC-FT)

OPD_FC_CORR [m] : OPD error to astrometry caused by pupil displacements and static abberations

OPD_TEL_CORR [m] : OPD predicted on telescope diodes caused by astigmatism and tip-tilt

OPD_TELFC_CORR [m] : OPD measured on the telescope diodes with respect to FC and the 2 correction terms FC_CORR and TEL_CORR.

OPD_TELFC_MCORR [m] : mean of the 4 telescope diodes OPD_TELFC_CORR.

OPD_PUPIL [m] : Expected OPD introduced by the measured pupil shift, re-aligned in time with the MET sampling.

Columns in the OI_VIS_ACQ table

The table has the same structure as the OI_FLUX table, that is one row per beam (12341234124...), and thus nsample x 4 rows.

TIME [us] : time of the frame, in [us], from the PRC.ACQ.START

PUPIL_NSPOT : number of spot detected in the pupil sensor (maximum is 16 = 4 diodes x 4 sub-apertures).

PUPIL_X [pix] : horizontal shift of pupil (in detector).

PUPIL_Y [pix] : vertical shift of pupil (in detector).

PUPIL_Z [pix] : focus shift of pupil (in detector).

PUPIL_R [deg] : rotation of pupil diode (in detector).

PUPIL_U [m] : lateral shift of pupil (in uv reference).

PUPIL_V [m] : lateral shift of pupil (in uv reference).

PUPIL_W [m] : focus shift of pupil (in uv reference).

OPD_PUPIL [m] : Expected OPD introduced by the measured pupil shift.

Columns in the OPDC table

TIME [us] : time of the frame, in [us], from the PRC.ACQ.START time from header (RMN recording start).

STATE : global fringe tracking state

STEPS : target phase modulation per baseline (scrambled), in units of pi/8

BASELINE_STATE : fringe tracking state per telescope and baseline (scrambled)

PIEZO_DL_OFFSET : same as raw data

VLTI_DL_OFFSET : same as raw data

VLTI_FDDL_OFFSET : same as raw data

OPD : same as raw data

KALMAN_PREDICT : same as raw data

9.9 *_ASTROREDUCED products

This is a lighter version of the P2VMRED file. It is used for the astrometric mode of GRAVITY. For more detail on how to do it, see Section 10.27

9.10 DISP_MODEL

The data are stored in the table DISP_MODEL. There are one line per GRAVITY beam. They are used to calculate OPD_DISP (see section 10.24). The columns are:

WAVE0 : λ_0, reference wavelength to calculate OPD_DISP model.

NMEAN : Mean optical index of the SC and FT fibers. The vector contains the $nmean_i$ coefficients such that the index at wavelength λ is given by : $n(\lambda)/n(\lambda_{MET}) = \sum_i (\text{ nmean}_i \, (\frac{\lambda_0}{\lambda} - 1)^i \,)$.

NDIFF : Differential optical index between the SC and the FT fibers. The vector contains the $ndiff_i$ coefficients such that the differential index at wavelength λ is given by : $n(\lambda)/n(\lambda_{MET}) = \sum_i (\text{ ndiff}_i \, (\frac{\lambda_0}{\lambda} - 1)^i \ )$.

LIN_FDDL_FT : Linearity coefficients to convert the FDDL signal in [V] to stretching length in [m]. The vector contains the a_i coefficients such that $L(V) = \sum_i a_i \, V^i$.

LIN_FDDL_SC : Same for the SC.

10 Algorithms

In the following, f is the index of individual DIT (that is from 0 to NDIT-1, whose typicall values are 30 for SC and 300000 for FT), ij are the pixel indices on the detector (i being the spatial direction, and j being the spectral direction), l is for the spectral channel, o is the combiner output (so called region in the code, [0..23] or [0..47] if polarisations are split), b is the baseline [0..5], and t is the telescope/beam [0..4].

10.1 Correction of detector bias

The SC frames are corrected for the detector bias using the value of some specific pixels, supposedly non-illuminated. In LOW and MED spectral mode, the bias per frame is estimated as the mean over all the pixels of the *bias lines* interleaved between the spectra of each region.

In HIGH spectral mode, the bias per frame is estimated as the mean over all pixles of the *bias columns* at the edge of the detector.

In both case, there is thus a single, scalar bias value for the entire frame for each frame.

10.2 Dark map

The dark map is computed from a set of images acquired with all shutters closed and with the same exposure time as the raw image and as close in time as possible. The dark is computed for FT, SC and MET.

The dark map is the median image of this set of images.

$$D_{ij} = median_f(X_{fij}) \tag{5}$$

10.3 Spectrum extraction

The implemented spectrum extraction Y_{foj} from the 2D image X_{fij} is based on a profile image p_{oij}.

Profile definition

When computing the profile from the sequence of 4 FLAT files, the first step is to add the 4 median of the 4 files. Then for each output a Gaussian fit is perform for each column (spectral element) over the specified profile_width pixels.

Depending of the profile-mode option the used profile can be either the gausian fit (GAUSS), the mesured pixels intensity (PROFILE), or boxcar (BOX). AUTO is the default option value and means PROFILE for LOW and MED mode, and BOX for HIGH mode.

In LOW and MED spectral modes, the profile is identical to the one observed with the sequences of FLAT files. It resembles a Gaussian function with FWHM of 1.5 pixel. To ensure the overall flux is conserved in the extraction, we apply the following normalization which assumes the shape of the object spectrum is perfectly matched by the profile itself:

$$p_{oij} = p_{oij} \cdot \frac{\sum_i p_{oij}}{\sum_i p_{oij}^2} \tag{6}$$

In HIGH mode, the profile is a boxcar of 6 pixels around the center of the best-fit Gaussian on the observed profile in the FLAT files. To ensure flux conservation, this boxcar is either 0 (outside) or 1 (inside).

Bad pixels in profile

The bad pixels are forced to zero in the profile. Consequently, a profile with bad pixels will lead to a reduced amount of detected flux. This effect is calibrated by the P2VM algorithm because the P2VM coefficients and the data are affected by the same amout of flux losses. A worst, for some spectral channels, the spectra of one output (e.g A) can be forced zero if all the pixels are bad. The P2VM then relies on the remaining BCD outputs only.

Extracted spectrum and variance for SC

$$Y_{foj} = g \sum_i (X_{fij} - S_{ij})\, p_{oij} \tag{7}$$

where $S_{i,j}$ is the mean image measured on the SKY, and g is the conversion gain from [ADU] to [e]. The sum $\sum$ is obviously performed in the compressed spatial direction.

Introducing the photon and detector noise, the variance of the extracted spectrum is:

$$\mathrm{var}(Y)_{foj} = g \sum_i (X_{fij} - D_{ij})\, p_{oij}^2 \; + \; g^2 \sum_i \sigma_{ij}^2 \, p_{oij}^2 \tag{8}$$

where $D_{i,j}$ is the mean level measured on the DARK, and σ_{ij}^2 is the variance measured on the DARK.

If no SKY is available, it is replaced by the DARK in Eq.7. Note that this can bias the result since the background estimate will miss the true sky brightness, which is significant for long DIT and at the end of the K-band. If no DARK is available, it is replaced by the SKY in Eq.8. Note however that the number of frames in the SKY may be insufficient to properly estimate the variance, especially for the long DIT. Therefore it is strongly recommended to always reduce SC science observation with both a DARK with high statistic, *and* a SKY taken close in time and observing condition.

Extracted spectrum and variance for FT

$$Y_{foj} = g \sum_i (X_{fij} - S_{ij})\, p_{oij} \tag{9}$$

$$\mathrm{var}(Y)_{foj} = g \sum_i (X_{fij} - S_{ij})\, p_{oij}^2 \; + \; g^2 \sum_i \sigma_{ij}^2 \, p_{oij}^2 \tag{10}$$

(a) X versus Y before fitting

(b) Y' versus X' after fitting

Figure 10.1: Correction of $X = C - A$ and $Y = D - B$ with the fitted ellipse parameters

where g is the conversion gain from [ADU] to [e], $S_{i,j}$ is the mean level measured on the SKY, and σ^2_{ij} is the variance measured on the SKY.

If no SKY is available, it is replaced by the DARK in Eq.9 and Eq.10. Using a DARK or SKY makes litle difference for the FT because the sky brightness is negligeable at the FT frame rate, and because there is always enough statistic. However, it is critical to use a DARK or SKY calibration taken *close in time*, and with *exactly* the same FT setup.

10.4 Wavelength calibration

10.4.1 Compute the phase from ABCD

To compute the phase from the A, B, C and D measurements without knowing the P2VM, the 2 quantities $X = C - A$ and $Y = D - B$ must be corrected to compensate for non-perpendicularities of A, B, C and D. An ellipse with equation 11 is fitted to the raw data.

$$\sqrt{(aX + bY + c)^2 + (dY + e)^2} = 1 \tag{11}$$

Knowing the fitted parameters (a, b, c, d and e), we compute the corrected $X' = aX + bY + c$ and $Y' = dY + e$. The corrected points are now on a centered and normalized circle (10.1).

The phases are now computed as :

$$\varphi = \arctan\left(\frac{X'}{Y'}\right) \tag{12}$$

The phase values are between 0 and 2π , to reconstruct the continuous evolution of the phases one needs to unwrap the numbers.

10.4.2 Evaluation of the OPD

To do the wavelength calibration we have to compute the OPD_{FT} and OPD_{SC} from the FT and SC data and the metrology measurement. This is possible because the scanning of the FT and SC FDDL are at different time scale. The relation linking $OPD_{FT} = a\widetilde{\varphi}_{FT}$, $OPD_{SC} = b\widetilde{\varphi}_{SC}$ and the differential metrology is the following:

$$a\widetilde{\varphi}_{FT} - b\widetilde{\varphi}_{SC} + c = dOPL_{MET} \tag{13}$$

We compute $\widetilde{\varphi}_{FT}(t_{FT})$ from FT data as the mean phase. This is done by computing the phase of each spectral element with ellipse method, and then computing the mean of the phase for the whole wavelengths.

We compute $\widetilde{\varphi}_{SC}(t_{FT})$ from SC data as the mean phase, by the same way, and scalled at the time of the FT data.

We compute $dOPD_{MET}$ at the time scale of the FT data.

$$dOPD_{MET} = \varphi_{MET} * \lambda_{MET}/2\pi$$

$$a\widetilde{\varphi}_{FT} - b\widetilde{\varphi}_{SC} + c = dOPD_{MET}$$

Find a and b by fitting eq 13 on the metrology data. This equation can be written with the following matrix:

$$\begin{pmatrix} OPD^t_{METj} - OPD^t_{METi} \\ \vdots \end{pmatrix} = \begin{pmatrix} \widetilde{\varphi}^t_{FTij} & -\widetilde{\varphi}^t_{SCij} & 1 \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

$\begin{pmatrix} \widetilde{\varphi}^t_{FT} & -\widetilde{\varphi}^t_{FT} & 1 \\ \vdots & \vdots & \vdots \end{pmatrix}$ can be inverted via singular value decomposition and a, b and c computed as

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \widetilde{\varphi}^t_{FTij} & -\widetilde{\varphi}^t_{SCij} & 1 \\ \vdots & \vdots & \vdots \end{pmatrix}^{-1} \begin{pmatrix} OPD^t_{METj} - OPD^t_{METi} \\ \vdots \end{pmatrix}$$

For each baseline we compute the following

$$OPD_{FT} = a\widetilde{\varphi}_{FT}$$
$$OPD_{SC} = b\widetilde{\varphi}_{SC}$$

10.4.3 Spectral calibration

The wavelength of each spectral element is computed by comparing the measured phases of this spectral element with the realized OPD: $\mathrm{OPD_{FT}}$ or $\mathrm{OPD_{SC}}$.

The measured phases are computed from the A, B, C and D measurements with ellipse methode.

For each computed phase we know the expected OPD, $\mathrm{OPD_{FT}}$ or $\mathrm{OPD_{SC}}$ from the metrology The slope of the phase versus OPD gives us the wavelength of the spectral element.

This wavelength is the one of the ABCD set of spectral elements for a given position, which is the barycentre of these four spectral elements. When all spectral element wavelengths are computed we have two sets of calibrated points, one for each polarization. On each of these two sets, a model of lambda versus position on the detector is fitted. And from this the wavelength of each spectral element of each spectrum is computed and put in the wavelength map.

10.5 Re-interpolation to a common wavelength

Modified target wavelength for SC

For the output (region) o, let's call j_{ol} the sample just below the target wavelength λ_l, and $j_{ol}+1$ the sample just above. Of course j depends on the region because their wavelength tables are different.

For the SC, this target wavelength is slightly modified, in a different way, for each region:

$$\lambda'_{ol} = \lambda_{o\,j_{ol}} + \frac{(\lambda_l - \lambda_{o\,j_{ol}})\,(\lambda_{o\,j_{ol}+1} - \lambda_{o\,j_{ol}})}{(\lambda_l - \lambda_{o\,j_{ol}}) + (\lambda_{o\,j_{ol}+1} - \lambda_l)\,.\,\frac{F_{o\,j_{ol}+1}}{F_{o,j_{ol}}}} \tag{14}$$

where F_{oj} is the flat measured on the internal light, extracted the same way as the data. This modification ensures that we later interpolate to a common *effective wavelength* for all regions. It also ensures that spectral channels whose interpolation includes a bad-pixel (forced to zero) are all set to zero.

Modified target wavelength for FT

For the FT, we don't modify the target wavelength:

$$\lambda'_{ol} = \lambda_l \tag{15}$$

Interpolation of flux and variance

The following coefficient a_{ol}

$$a_{ol} = \frac{\lambda_{o\,j+1} - \lambda'_{ol}}{\lambda_{o\,j+1} - \lambda_{o\,j}} \tag{16}$$

allows to linearly interpolate the fluxes:

$$Y_{fol} = a_{ol}\,Y_{fo\,j_{ol}} \;\;+\;\; (1-a_{ol})\,Y_{fo\,j_{ol}+1} \tag{17}$$

and the variances:

$$\mathrm{var}(Y)_{fol} = a_{ol}^2\,\mathrm{var}(Y)_{fo\,j_{ol}} \;\;+\;\; (1-a_{ol})^2\,\mathrm{var}(Y)_{fo\,j_{ol}+1} \tag{18}$$

10.6 Computation of the P2VM

The P2VM theory is inspired by the amber data reduction (Tatulli et al. 2007), which is working fine. It has been adapted to the integrated optics specificities (Lacour et al. 2008).

Figure 10.2 represents the generalized view of the transfer function of an integrated optics component. E_n is the complex electric field entering the component via input n , and S_k is the resulting field on output number k . $T_{k,n}$ is a two dimensional complex matrix linking S_k to E_n .

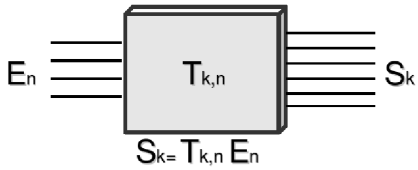

Figure 10.2: Generalization of the transfer function of an integrated optics component

The flux q received by the detector from the output k at wavelength λ and instant t is written as

$$q_k^{\lambda,t} = \left|S_k^{\lambda,t}\right|^2 = \left|\sum_n T_{k,n}^{\lambda} E_n^{\lambda,t}\right|^2 \tag{19}$$

with the electric fields $E_n^{\lambda,t}$ and $S_k^{\lambda,t}$ as function of time and wavelength and of input n and output k respectively. $T_{k,n}^{\lambda}$ is the complex matrix function of wavelength linking $E_n^{\lambda,t}$ and $S_k^{\lambda,t}$.

With introduction of $V_{n,m}$ (coherence of the incoming electric field) and $C_{k,n,m}^{\lambda}$ (integrated optics conservation of light coherence) equation (7) can be developed as:

$$\left|S_k^{\lambda,t}\right|^2 = R\left[\sum_n \left|T_{k,n}^{\lambda}\right|^2 \left|E_n^{\lambda,t}\right|^2 + \sum_n \sum_{m>n} 2T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda} E_n^{\lambda,t} E_m^{\lambda,t*} V_{n,m}^{\lambda}\right] \tag{20}$$

or

$$\begin{aligned}\left|S_k^{\lambda,t}\right|^2 = \sum_n \left|T_{k,n}^{\lambda}\right|^2 \left|E_n^{\lambda,t}\right|^2 + \sum_n \sum_{m>n} \Re 2T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda} \Re E_n^{\lambda,t} E_m^{\lambda,t*} V_{n,m}^{\lambda} \\ - \sum_n \sum_{m>n} \Im 2T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda} \Im E_n^{\lambda,t} E_m^{\lambda,t*} V_{n,m}^{\lambda}\end{aligned} \tag{21}$$

Using the matrix expression:

$$\begin{pmatrix} \left|S_1^{\lambda,t}\right|^2 \\ \vdots \\ \left|S_K^{\lambda,t}\right|^2 \end{pmatrix} = R\left[V2PM_C . \begin{pmatrix} \left|E_1^{\lambda,t}\right|^2 \\ \vdots \\ \left|E_N^{\lambda,t}\right|^2 \\ E_1^{\lambda,t} E_2^{\lambda,t} V_{1,2} \\ \vdots \\ E_{N-1}^{\lambda,t} E_N^{\lambda,t} V_{N-1,N} \end{pmatrix}\right] = V2PM_R . \begin{pmatrix} \left|E_1^{\lambda,t}\right|^2 \\ \vdots \\ \left|E_N^{\lambda,t}\right|^2 \\ R[E_1^{\lambda,t} E_2^{\lambda,t} V_{1,2}] \\ \vdots \\ R[E_{N-1}^{\lambda,t} E_N^{\lambda,t} V_{N-1,N}] \\ I[E_1^{\lambda,t} E_2^{\lambda,t} V_{1,2}] \\ \vdots \\ I[E_{N-1}^{\lambda,t} E_N^{\lambda,t} V_{N-1,N}] \end{pmatrix} \tag{22}$$

with N entries and K output. The complex V2PM is defined as:

$$V2PM_C = \begin{pmatrix} |T^{\lambda}_{1,1}|^2 & \cdots & |T^{\lambda}_{1,N}|^2 & 2T_{1,1}T_{1,2}C^{\lambda}_{1,1,2} & \cdots & 2T_{1,N-1}T_{1,N}C^{\lambda}_{1,N-1,N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ |T^{\lambda}_{K,1}|^2 & \cdots & |T^{\lambda}_{K,N}|^2 & 2T_{K,1}T_{K,2}C^{\lambda}_{K,1,2} & \cdots & 2T_{K,N-1}T_{K,N}C^{\lambda}_{K,N-1,N} \end{pmatrix} \quad (23)$$

Where the N first columns are the transmissions of the integrated optic, and the N(N-1)/2 others are the coherences. The real V2PM matrix which can be used for visibility computing is:

$$V2PM_R =$$
$$\begin{pmatrix} \left|T^{\lambda}_{1,1}\right|^2 & \cdots & \left|T^{\lambda}_{1,N}\right|^2 & R[2T_{1,1}T_{1,2}C^{\lambda}_{1,1,2}] & \cdots & R[2T_{1,N-1}T_{1,N}C^{\lambda}_{1,N-1,N}] & -I[2T_{1,1}T_{1,2}C^{\lambda}_{1,1,2}] & \cdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots \\ \left|T^{\lambda}_{K,1}\right|^2 & \cdots & \left|T^{\lambda}_{K,N}\right|^2 & R[2T_{K,1}T_{K,2}C^{\lambda}_{K,1,2}] & \cdots & R[2T_{K,N-1}T_{K,N}C^{\lambda}_{K,N-1,N}] & -I[2T_{K,1}T_{K,2}C^{\lambda}_{K,1,2}] & \cdots \end{pmatrix}$$

Compute the transmissions $\left|T^{\lambda}_{k,n}\right|^2$

This is done when only the shutter n is open. So we have $E^{\lambda,t}_m = 0 \forall m \neq n$. Eq (3) becomes:

$$\left|S^{\lambda,t}_k\right|^2 = \left|T^{\lambda}_{k,n}\right|^2 \left|E^{\lambda,t}_n\right|^2 \Leftrightarrow \left|T^{\lambda}_{k,n}\right|^2 = \frac{\left|S^{\lambda,t}_k\right|^2}{\left|E^{\lambda,t}_n\right|^2} \quad (24)$$

To avoid that $T^{\lambda}_{k,n}$ depends on the input flux, we assume that all incoming photons proceed to the exit of the optics, so:

$$\sum_k \left|S^{\lambda_0,t}_k\right|^2 \times \widetilde{E}(\lambda) = \left|E^{\lambda,t}_n\right|^2 \quad (25)$$

Now $T^{\lambda}_{k,n}$ can be expressed as:

$$\left|T^{\lambda}_{k,n}\right|^2 = \frac{\langle \left|S^{\lambda,t}_k\right|^2 \rangle_t}{\langle \sum_k \left|S^{\lambda_0,t}_k\right|^2 \rangle_t \times \widetilde{E}(\lambda)} \quad (26)$$

Compute the coherences $2T^{\lambda}_{k,n}T^{\lambda *}_{k,m}C^{\lambda}_{k,n,m}$

This is done when the shutter n and m are open and others closed. So we have $E^{\lambda,t}_i = 0 \forall i \neq m, n$.

A calibration source is used with $V^{\lambda}_{n,m} = 1$, so from eq (8) we have:

$$\left|S^{\lambda,t}_k\right|^2 = \left|T^{\lambda}_{k,n}\right|^2 |E^{\lambda,t}_n|^2 + \left|T^{\lambda}_{k,m}\right|^2 |E^{\lambda,t}_m|^2 + R\left[2T^{\lambda}_{k,n}T^{\lambda *}_{k,m}C^{\lambda}_{k,n,m}E^{\lambda,t}_n E^{\lambda,t *}_m\right] \quad (27)$$

This can be developed with $E_n^{\lambda,t} = \sqrt{I_n^{\lambda,t}} e^{-2i\pi \frac{OPD_n^t}{\lambda}}$ as:

$$|S_k^{\lambda,t}|^2 = \left|T_{k,n}^{\lambda}\right|^2 I_n^{\lambda,t} + \left|T_{k,m}^{\lambda}\right|^2 I_m^{\lambda,t} + 2\left|T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda}\right| \sqrt{I_n^{\lambda,t} I_m^{\lambda,t}} \cos 2\pi \frac{OPD_{m,n}^t}{\lambda} + \varphi_{k,n,m}^{\lambda} \tag{28}$$

With $OPD_{m,n}^t = OPD_m^t - OPD_n^t$ and $\varphi_{k,n,m}^{\lambda}$ is the phase of $T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda}$.

This equation can be fitted by a sinusoid with 3 free parameters (a, b and c)

$$\left|S_k^{\lambda,t}\right|^2 = a_{k,m,n}^{\lambda} \cos \frac{OPD_{m,n}^t}{\lambda} 2\pi + b_{k,m,n}^{\lambda} \sin \frac{OPD_{m,n}^t}{\lambda} 2\pi + c_{k,m,n}^{\lambda} \tag{29}$$

With:

$$c_{k,m,n}^{\lambda} = \left|T_{k,n}^{\lambda}\right|^2 I_n^{\lambda,t} + \left|T_{k,m}^{\lambda}\right|^2 I_m^{\lambda,t} \tag{30}$$

$$\varphi_{k,n,m}^{\lambda} = \arctan [\text{E09E?}] \frac{b_{k,m,n}}{a_{k,m,n}} [\text{E09F?}] \tag{31}$$

$$2\left|T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda}\right| \sqrt{I_n^{\lambda,t} I_m^{\lambda,t}} = \sqrt{{a_{k,m,n}^{\lambda}}^2 + {b_{k,m,n}^{\lambda}}^2} \tag{32}$$

The coherences ($2T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda}$) can be expressed by their phases and amplitudes. The phase is given by eq. (13) and the amplitude is given by eq. (14). But the amplitude depends on the $I_n^{\lambda,t}$ which can be evaluated with eq. (12).

Eq.(12)can be expressed as a matrix equation where the $\left|T_{k,n}^{\lambda}\right|^2$ matrix is already:

$$\begin{pmatrix} c_{1,m,n}^{\lambda} \\ \vdots \\ c_{K,m,n}^{\lambda} \end{pmatrix} = \begin{pmatrix} \left|T_{1,1}^{\lambda}\right|^2 & \cdots & \left|T_{1,N}^{\lambda}\right|^2 \\ \vdots & \ddots & \vdots \\ \left|T_{K,1}^{\lambda}\right|^2 & \cdots & \left|T_{K,N}^{\lambda}\right|^2 \end{pmatrix} \cdot \begin{pmatrix} I_1^{\lambda,t} \\ \vdots \\ I_N^{\lambda,t} \end{pmatrix} \tag{33}$$

This over determined system can be solved by inverting the matrix $\left|T_{k,n}^{\lambda}\right|^2$ with a singular value decomposition method. And knowing $I_n^{\lambda,t}$, we can compute the coherences amplitudes:

$$2\left|T_{k,n}^{\lambda} T_{k,m}^{\lambda *} C_{k,n,m}^{\lambda}\right| = \frac{\sqrt{{a_{k,m,n}^{\lambda}}^2 + {b_{k,m,n}^{\lambda}}^2}}{\sqrt{I_n^{\lambda,t} I_m^{\lambda,t}}} \tag{34}$$

This should be done for each base(n, m>n couple).

Phases calibrations

The phases $\varphi^{\lambda}_{k,n,m}$ of the integrated optics coherences $C^{\lambda}_{k,n,m}$ can be divided into two components: the modulation phases (ABCD-like modulation) $\psi^{\lambda}_{k,n,m}$ and the baseline phases $\varphi^{\lambda}_{n,m}$:

$$\varphi^{\lambda}_{k,n,m} = \psi^{\lambda}_{k,n,m} + \varphi^{\lambda}_{n,m} \tag{35}$$

Modulation phases The modulation phases $\psi^{\lambda}_{k,n,m}$ are defined as the instrumental phases introduced by the beam combiner between the two interfering beams n and m. For each baseline defined by the couple of beams (n, m), the corresponding modulations phases are defined relatively to one of the beams at the output $k = A_{(n,m)}$ acting as a reference, such that:

$$\psi^{\lambda}_{k,n,m} = \psi^{\lambda}_{k,n,m} - \psi^{\lambda}_{A_{(n,m)},n,m} \tag{36}$$

Absolute baseline phases The baseline phases $\varphi^{\lambda}_{n,m}$ are defined as the absolute instrumental phases introduced by the beam combiner on each baseline. They induce instrumental closure phases different from zero, and then bias the source closure phases.

However, the absolute values of the baseline phases cannot be measured, due to different reference phases and piston disturbances. The closure phases they induce do not depend on these biases, however. The baseline phases can be calibrated and included in the V2PM, in a way not to bias the closure phases of the source.

Thus, instead of measuring the absolute baseline phases $\varphi^{\lambda}_{n,m}$, relative baseline phases $\widetilde{\varphi}^{\lambda}_{n,m}$ can be determined, inducing the same closure phases as the absolute instrumental phases.

Closure phases and relative baseline phases Let $\Phi^{\lambda}_{n,m}$ and $\widetilde{\Phi}^{\lambda}_{n,m}$ be the vectors of respectively the 6 absolute and the 6 relative baseline phases, $\varphi^{\lambda}_{n,m}$ and $\widetilde{\varphi}^{\lambda}_{n,m}$, for the apertures (n, m)∈[301A?]$0, 3$[301B?]^2 :

$$\Phi^{\lambda}_{n,m} = (\varphi^{\lambda}_{0,1}\varphi^{\lambda}_{0,2}\varphi^{\lambda}_{0,3}\varphi^{\lambda}_{1,2}\varphi^{\lambda}_{1,2}\varphi^{\lambda}_{2,3})^T \tag{37}$$

$$\widetilde{\Phi}^{\lambda}_{n,m} = (\widetilde{\varphi}^{\lambda}_{0,1}\widetilde{\varphi}^{\lambda}_{0,2}\widetilde{\varphi}^{\lambda}_{0,3}\widetilde{\varphi}^{\lambda}_{1,2}\widetilde{\varphi}^{\lambda}_{1,2}\widetilde{\varphi}^{\lambda}_{2,3})^T \tag{38}$$

Let $\Xi^{\lambda}_{n,m,l}$ be the vector of the 4 resulting closure phases $\xi^{\lambda}_{n,m,l}$ between the 3 apertures (n, m, l)∈[301A?]$0, 3$[301B?]^3 :

$$\Xi^{\lambda}_{l,n,m} = (\xi^{\lambda}_{0,1,2}\xi^{\lambda}_{0,1,3}\xi^{\lambda}_{0,2,3}\xi^{\lambda}_{1,2,3})^T \tag{39}$$

such that:

$$\xi^{\lambda}_{n,m,l} = \varphi^{\lambda}_{n,m} + \varphi^{\lambda}_{m,l} - \varphi^{\lambda}_{n,l} \tag{40}$$

Three of these closure phases are independent, and the fourth is related to the other by:

$$\xi^{\lambda}_{0,1,2} + \xi^{\lambda}_{0,2,3} = \xi^{\lambda}_{0,1,3} + \xi^{\lambda}_{1,2,3} \tag{41}$$

Generalizing relation (22) to the vectors of closure phases and baseline phases leads to equation (24):

$$\Xi^{\lambda}_{n,m} = M\Phi^{\lambda}_{n,m} \tag{42}$$

with the matrix M:

$$M = \begin{pmatrix} 1 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & -1 & 1 \end{pmatrix} \tag{43}$$

The relative baseline phases $\widetilde{\varphi}^{\lambda}_{n,m}$ must result from a linear operation of the known closure phases, and lead to the same closure phases $\xi^{\lambda}_{n,m,l}$. Therefore the vector $\widetilde{\Phi}^{\lambda}_{n,m}$ must be linked to the closure phases $\Xi^{\lambda}_{n,m,l}$ by a matrix N such that:

$$\widetilde{\Phi}^{\lambda}_{n,m} = N\Xi^{\lambda}_{n,m} \tag{44}$$

and verify the same equation as(24):

$$\Xi^{\lambda}_{n,m} = M\widetilde{\Phi}^{\lambda}_{n,m} \tag{45}$$

Thus, combining equations (26) and (27), the relative baseline phases are defined by the matrix N such that:

$$\Xi^{\lambda}_{n,m} = MN\Xi^{\lambda}_{n,m} \tag{46}$$

N is not unique and we suggest to use the N matrix such that:

$$N = \frac{1}{4}M^T = \frac{1}{4}\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & -1 & -1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \tag{47}$$

Equation (26) easily leads to the vector of relative baselines phases:

$$\widetilde{\Phi}^{\lambda}_{n,m} = \frac{1}{4}\begin{pmatrix} \xi^{\lambda}_{0,1,2} + \xi^{\lambda}_{0,1,3} \\ -\xi^{\lambda}_{0,1,2} + \xi^{\lambda}_{0,2,3} \\ -\xi^{\lambda}_{0,1,3} + \xi^{\lambda}_{0,2,3} \\ \xi^{\lambda}_{0,1,2} + \xi^{\lambda}_{1,2,3} \\ \xi^{\lambda}_{0,1,3} - \xi^{\lambda}_{1,2,3} \\ \xi^{\lambda}_{0,2,3} + \xi^{\lambda}_{1,2,3} \end{pmatrix} \tag{48}$$

Compute the closure phases This is done when all the shutters are open together, with a calibration source with $V^{\lambda}_{n,m} = 1$. Using an intermediate V2PM with null instrumental closure phases and baseline phases, from the intensities $q^{\lambda,t}_k$ we can compute the vector (eq. (4)):

$$\begin{pmatrix} \left|E^{\lambda,t}_1\right|^2 \\ \vdots \\ \left|E^{\lambda,t}_N\right|^2 \\ R[E^{\lambda,t}_1 E^{\lambda,t}_2 \exp(i\varphi^{\lambda}_{1,2})] \\ \vdots \\ R[E^{\lambda,t}_{N-1} E^{\lambda,t}_N \exp(i\varphi^{\lambda}_{N-1,N})] \\ I[E^{\lambda,t}_1 E^{\lambda,t}_2 \exp(i\varphi^{\lambda}_{1,2})] \\ \vdots \\ I[E^{\lambda,t}_{N-1} E^{\lambda,t}_N \exp(i\varphi^{\lambda}_{N-1,N})] \end{pmatrix} \tag{49}$$

The phases $\varphi^{\lambda}_{n,m}$ can be expressed by the equation:

$$\varphi^{\lambda,t}_n + \varphi^{\lambda,t}_m + \varphi^{\lambda}_{n,m} = \arctan\left(\frac{I[E^{\lambda,t}_n E^{\lambda,t}_m \exp(i\varphi^{\lambda}_{n,m})]}{R[E^{\lambda,t}_n E^{\lambda,t}_m \exp(i\varphi^{\lambda}_{n,m})]}\right) \tag{50}$$

With $\varphi^{\lambda,t}_n$ the phase of the electric field $E^{\lambda,t}_n$. Equation (24) therefore leads to the instrumental closure phases, free of the electric fields phases $\varphi^{\lambda,t}_n$. The relative baseline phases $\widetilde{\varphi}^{\lambda}_{n,m}$ can then be deduced from equation (30).

10.7 Extraction of the coherent fluxes and telescope fluxes via P2VM

The fluxes of each telescope F_{ftl} and the complex coherent flux of each base $R_{fbl} + iI_{fbl}$ are extracted from a matricial analysis of the profiles, based on the P2VM calibration

$$(F_{ftl}, R_{fbl}, I_{fbl}) = P2VM^{o}_{b/tl} \times Y_{fol} \tag{51}$$

The variances are propagated assuming no correlation between the input Y_{fol}.

$$(\mathrm{var}(F)_{ftl}, \mathrm{var}(R)_{fbl}, \mathrm{var}(I)_{fbl}) = (P2VM^o_{b/tl})^2 \times \mathrm{var}(Y)_{fol} \tag{52}$$

The $P2VM$ is a well conditioned matrix thanks to the design of the integrated beam combiner. To demonstra the underlying reasoning, let's consider a perfect $P2VM$. The four regions related to baseline b (say $0, 1, 2, 3$, also called ABCD regions of baseline b) can be combined together to build the following quantities:

- $R_{fbl} = Y_{f0l} - Y_{f2l}$,
- $I_{fbl} = Y_{f1l} - Y_{f3l}$ and
- $F_{ft_1l} + F_{ft_2l} = Y_{f0l} + Y_{f1l} + Y_{f2l} + Y_{f3l}$.

The two first are directly the complex coherent flux of baseline b, while the latter, combined with the constraints of the 5 other baselines, easily solve for the flux F_{ftl} of individual beams. In practice, the actual $P2VM$ matrix takes into account the exact interferometric phase-shift between the four ABCD regions, and the relative photometric throughputs. It is a global fit of the 24 outputs into 6 × 2 coherent fluxes (R_{fbl}, I_{fbl}) and 4 photometric fluxes (R_{ftl}), for each frame, each wavelength channel, and each polarisation.

10.8 Computation of reduced chi2

Once the fluxes quantities (R_{fbl}, I_{fbl} and R_{ftl}) have been estimated via propagation through the $P2VM$, it is possible to recompute the corresponding expected output values:

$$Z_{fol} = V2PM_o^{b/tl} \times (F_{ftl}, R_{fbl}, I_{fbl}) \tag{53}$$

From these, we can compute a reduced chi2 with respect to the measurements:

$$CHI2_{fl} = \sum_o \frac{(Y_{fol} - Z_{fol})^2}{\mathrm{var}(Y)_{fol}} \frac{1}{24 - 16} \tag{54}$$

10.9 Computation of SNR

Individual SNR

The Signal to Noise Ratio (SNR) of each baseline and each frame of the the FT is computed using a running mean of the complex coherent flux over 10 consecutive samples. The complex coherent flux of the two polarisations, if any, are also averaged together after having recentered them to a common mean phase.

$$SNR_{fb} = \frac{(\sum_{f_r\,l} R_{f_r\,bl})^2 + (\sum_{f_r\,l} I_{f_r\,bl})^2}{\sum_{f_r\,l} \mathrm{var}(R)_{f_r\,bl} + \sum_{f_r\,l} \mathrm{var}(I)_{f_r\,bl}} \tag{55}$$

where f_r is the f index running in the interval $\in \{f-5\ ,\ f+5\}$ in order to implement the smoothing over 10 samples.

Bootstrapped SNR

If fringes are detected on two consecutive baselines, then, by nature of light, fringes are detected on the closing baseline. The purpose of bootstrapping is to increase the confidence on the signal for the baselines with lowest SNR (whose SNR may well be null for astrophysical reason), by looking at these closing baseline.

A "bootstraped" SNR is computed for each baseline and each frame, as the maximum between the SNR of this baseline and all closing triangles. For instance for the baseline $b = b_{ij}$ among the beams i, j, k, l:

$$SNRB_{fb_{ij}} = max\{\ SNR_{fb_{ij}},\ min\{SNR_{fb_{ik}}\ ,\ SNR_{fb_{kj}}\}, min\{SNR_{fb_{il}}\ ,\ SNR_{fb_{lj}}\}\ \} \tag{56}$$

The quantities SNR and SNRB are also computed for the SC, although they are not used in the processing.

10.10 Computing the vFactor

The purpose of the vFactor is to estimate the visibility loss of each individual SC frame due the phase jittering, from an analysis of the FT real-time data. This visibility loss is derived as the ratio between the coherent integration (squared norm of complex sum over DITs) and the incoherent integration (sum over DITs of complex squared norm) of the complex coherent flux of FT, across each SC DIT.

A white-light vFactor is first computed for each SC frame f with the FT data:

$$v_{fb} = \frac{(\sum_{f_{FT} l} R_{f_{FT} bl})^2 + (\sum_{f_{FT} l} I_{f_{FT} bl})^2 - \sum_{f_{FT} l} \mathrm{var}(R)_{f_{FT} bl} - \sum_{f_{FT} l} \mathrm{var}(I)_{f_{FT} bl}}{\sum_{f_{FT}} (\sum_l R_{f_{FT} bl})^2 + \sum_{f_{FT}} (\sum_l I_{f_{FT} bl})^2 - \sum_{f_{FT} l} \mathrm{var}(R)_{f_{FT} bl} - \sum_{f_{FT} l} \mathrm{var}(I)_{f_{FT} bl}} \times \frac{1}{\mathrm{n}_{f_{FT}}} \tag{57}$$

where the sum over l is over the 6 spectral channels of the FT and the sum over f_{FT} is over the FT frames acquired *during* the corresponding SC frame. This white-light vFactor at λ_0 (the central wavelength of the FT) is then extrapolated to the SC channels with:

$$\widetilde{v}_{fbl} = \exp(\,-\ln(v_{fb})\,\frac{\lambda_0^2}{\lambda_l^2}\,) \tag{58}$$

This vFactor correction is proved to be very efficient as long as the FT astrophysical visibility remain larger than 0.1. For fully resolved baselines, the vFactor results into a indefinit 0/0 ratio. In such a sitation, and if on-axis, one could simply rescale the SC visibilities to the one of the FT (see options of the recipes). In off-axis case with a fully resolved object on the FT, there is not much to be done however.

10.11 Computing the pFactor

The purpose of the pFactor is to estimate the visibility loss of each individual SC frame due to flux flickering, from an analysis of the FT real-time flux data. This visibility loss is derived as the ratio between the sum over DITs of the geometry mean, and the geometric mean of the sum over DITs of the photometric flux of FT, across each SC DIT.

We compute a white-light pFactor with the following formula:

$$p_{bf} = \frac{\left[\sum_{f_{FT}} \sqrt{(\sum_l \text{flux}_{f_{FT} t_1 l})(\sum_l \text{flux}_{f_{FT} t_2 l})} \right]^2}{(\sum_{f_{FT} l} \text{flux}_{f_{FT} t_1 l})(\sum_{f_{FT} l} \text{flux}_{f_{FT} t_2 l})} \tag{59}$$

This pFactor is computed in the P2VMRED product but not used so far.

10.12 Frame rejection

A FT frame is rejected if any of the conditions are met:

- its bootstrapped $SNRB$ is below the threshold.
- the OPDC state of this baseline is below the threshold. The OPDC states are: 1 = IDEL, 2 = GD_TRACKING, 3 = PHASE_TRACKING, 4 = SEARCHING, 5 += internal calibrations.

A SC frame is rejected if any of the conditions are met:

- the fraction of accepted FT frame during this SC frame is below the threshold.
- its computed vFactor is below the treshold.

The frame selection is done on a per-baseline basis. That is the baselines have a different selection map, and thus will have a different effective time after the averaging process.

10.13 Phase referencing

Self-referencing the FT phase

The phase reference of a FT frames is the runnning mean phase of the FT itself over few samples:

$$\text{P\_REF}_{fbl} = \arctan(\sum_{f_r} I_{f_r\, bl} \ , \ \sum_{f_r} R_{f_r\, bl}) \tag{60}$$

where f_r is in the interval $\in \{f - 3 \ , \ r + 3\}$, exluding $f_r = f$ to avoid biases. Note that this phase is not unwrapped neither temporally nor spectrally (SNR too low, and FT supposed to be near constant phase all the time thanks to real-time tracking).

Referencing the SC phases in single case

The phase reference of a SC frame is the mean phase of the FT during this SC frame:

$$\text{P\_REF}_{fbl} = \arctan(\sum_{f_{FT}} I_{f_{FT} bl} \ , \ \sum_{f_{FT}} R_{f_{FT} bl}) \tag{61}$$

where the sum over f_{FT} is over the FT frames acquired during the corresponding SC frame.

This P_REF, which have 6 spectral channels only, is interpolated/extrapolated into the wavelengths of the SC with a polynomial fit of order 2 (after properly unwrapping the phase along the spectral direction).

Referencing the SC phases in dual case

In case of dual-field observation, the previous reference phase is modified to account for the sideral motion of the binary separation $(\mathrm{dE}, \mathrm{dN})$ and the metrology measurements:

$$\mathrm{P\_REF\_IMG}_{fbl} = \mathrm{P\_REF}_{fbl} \; + \; \frac{2\pi}{\lambda_l} \, (\mathrm{UCOORD}_{fb} \, \mathrm{dE} \; + \; \mathrm{VCOORD}_{fb} \, \mathrm{dN} \; - \; \mathrm{OPD\_DISP}_{fbl}) \tag{62}$$

10.14 Geometric flux

The geometric flux of a baseline is the product of the photometric fluxes of the two beams of this baseline. It is needed to then normalise the complex coherent fluxes into complex visibilities.

For the FT

The FT real-time photometry is affected by large photometric fluctuations, which make it going near zero (flux loss event) or even going to negative values because of noise. Since the geometric flux will enter the visibility normalisation in the denominator, we have to avoid as much as possible these near-zero events.

The photometric fluctuations are highly corrected in the spectral direction. On the other side, the important property of the fluctuations to be extracted are the temporal correlations. Therefore we first compute a broad-band sum of the real-time flux of each beam, that we is temporally smooth:

$$F'_{ft} = \frac{\sum_{f_r \, l} F_{f_r \, tl}}{11} \tag{63}$$

where f_r is in the interval $\in \{f - 5 \; , \; f + 5\}$ (time smoothing). We also compute a normalized time-averaged spectrum for each beam:

$$F''_{lt} = \frac{\sum_f F_{ftl}}{\sum_{fl} F_{ftl}} \tag{64}$$

Only then the geometric mean is computed from these two quantities:

$$FF_{fbl} = F'_{ft_1} \, F''_{f \, t_1 \, l} \; \times \; F'_{ft_2} \, F''_{f \, t_2 \, l} \tag{65}$$

For the SC

For the science, the geometric flux is simply computed as:

$$FF_{fbl} = F_{f \, t_1 \, l} \; \times \; F_{f \, t_2 \, l} \tag{66}$$

10.15 Averaged flux estimator

The previous sections describe how the real-time quantities are extracted from every single FT and SC frame. We here describe the process of averaging these real-time quantities into final product. For the flux, all frames are simply co-added. There is no frame selection at all.

$$\widetilde{\text{flux}}_{tl} = \sum_f F_{ftl} \tag{67}$$

Hence the final flux $\widetilde{\text{flux}}_{tl}$ is the *sum* of all electron-events collected during the entire exposure and across all regions, for the beam t at channel l.

10.16 Averaged complex visibility estimator

For a given baseline, the averaged is computed only with accepted frames of this baseline.

The complex coherent flux vector is first rotated with the computed reference phase P_REF (or with P_REF_IMG in the case of dual-observations):

$$R'_{fbl} = \cos(\text{P\_REF}_{fbl})\, R_{fbl} \ - \ \sin(\text{P\_REF}_{fbl})\, I_{fbl} \tag{68}$$

$$I'_{fbl} = \sin(\text{P\_REF}_{fbl})\, R_{fbl} \ + \ \sin(\text{P\_REF}_{fbl})\, I_{fbl} \tag{69}$$

The visibilities of each frame are averaged together accounting for the visibility loss expected from the vFactor (only for SC, that is $v_{fbl} = 1.0$ for FT). Note that the coherent flux and the photometric flux are actually averaged first, before normalisation. The noise property is better than simply averaging the real-time normalised visibilities (Cauchy statistic).

Visibility amplitude

$$\widetilde{\text{visAmp}}_{bl} = \frac{\sqrt{(\sum_f R'_{fbl})^2 + (\sum_f I'_{fbl})^2}}{\sum_f \sqrt{FF_{fbl}\ v_{fbl}}} \tag{70}$$

Visibility phase

$$\widetilde{\text{visPhi}}_{bl} = \arctan(\sum_f I'_{fbl}\ ,\ \sum_f R'_{fbl}) \tag{71}$$

The mean spectral slope (stored in the GDELAY quantity) and mean spectral value (stored in the PHASE quantity) are removed from the VISPHI quantity.

10.17 Average squared visibility estimator

The square visibilities of each frame are averaged together accounting for the visibility loss expected from the vFactor (only for SC, that is $v_{fbl} = 1.0$ for FT).

For a given baseline, the averaging is performed only with accepted frames of this baseline.

$$\widetilde{\mathrm{vis2}}_{bl} = \frac{\sum_f R^2_{fbl} + \sum_f I^2_{fbl} - \sum_f \mathrm{var}(R)_{fbl} - \sum_f \mathrm{var}(I)_{fbl}}{\sum_f (FF_{fbl}\ v_{fbl})} \tag{72}$$

10.18 Average closure-phase estimator

The averaged bispectrum of triplet b_{ijk} is computed as the coherent integration of the bispectrum of each frame:

$$\widetilde{B}_{b_{ijk}\,l} = \sum_f (R_{fb_{ij}l} + i\,I_{fb_{ij}l}).(R_{fb_{jk}l} + i\,I_{fb_{jk}l}).(R_{fb_{ik}l} - i\,I_{fb_{ik}l}) \tag{73}$$

For a given triplet, the integration is performed only with frames for which *all* three baselines forming the triangles are accepted.

Bispectrum phase

The closure phase is computed:

$$\widetilde{\mathrm{t3Phi}}_{b_{ijk}l} = \arctan(\widetilde{B}_{b_{ijk}\,l}) \tag{74}$$

Bispectrum amplitude

The bispectrum amplitude is also computed. However this quantity has not been verified yet.

$$\widetilde{\mathrm{t3Amp}}_{bl} = \frac{\widetilde{B}_{b_{ijk}\,l}}{\sum_f (F_{ft_il}\ F_{ft_jl}\ F_{ft_kl}\ \sqrt{v_{fb_{ij}l}\, v_{fb_{ik}l}\, v_{fb_{jk}l}})} \tag{75}$$

10.19 Uncertainty on average quantities

The uncertainty on average quantities is computed by bootstrapping over the accepted frames. Basic fundamentals about this technic can be in the numerical recipes book, section 15.6.
See also: https://en.wikipedia.org/wiki/Bootstrapping_(statistics)

The implementation of the bootstraping method makes use of *segmentation* when the number of frame is larger than 100, or of *Monte-Carlo* when the number of frame is smaller than 5.

Segmentation (mostly for FT)

When the number of frames is larger than 100, the dataset is first split into a smaller number of *segments* (typically 20 to 100). First integration is done inside the segments. The final variance is estimated by bootstrapping over the segments. The number of segments does change the temporal sampling of the bootstrap, and thus the estimation of the uncertainties. We choose the number of segment so that each is about 1s length.

Note that this is mostly relevant for the FT, since the SC has often less than 100 frames (so each frame is a segment).

Monte-Carlo (mostly for very long DIT on SC)

When the number of frames is smaller than 5, the bootstrap method provides unrealistically small uncertainties. Hence the dataset is complemented with few fake frames (up to 5), on which a random realisation of the theoretical noise is added, using the theoretical variance.

This noise is added on the correlated flux quantity only, not on the photometric fluxes, vFactor, reference phase... Moreover it is clear that Monte-Carlo propagates the *fundamental* uncertainties only (photon and detector noise), but not the *atmospheric* noises (effect of injection fluctuation, tracking quality fluctuation).

As a matter of fact, the final uncertainty when the number of frames is very low can still be under-estimated.

10.20 Calibration with the TF

From CAL to TF estimations

For every observation c of an calibration star (VIS_CAL), the visibilities are converted into VIS_TF by dividing the VISAMP quantities by the expected visibility, accounting for the provided diameter. The uncertainty on the diameter is not propagated to the VIS_TF.

$$\widetilde{\mathrm{tfAmp}}_{bl\,c} = \frac{\widetilde{\mathrm{visAmp}}_{bl\,c}}{J1(\pi\,B_b/\lambda_l)/(\pi\,B_b/\lambda_l)} \tag{76}$$

$$\widetilde{\mathrm{tfPhi}}_{bl\,c} = \widetilde{\mathrm{visPhi}}_{bl\,c} \tag{77}$$

Interpolation of TF at the time of SCI

The TF are then averaged to obtain an estimation at the time of the SCI measurement:

$$\widetilde{\mathrm{tfAmp}}_{bl} = \frac{\sum_c W_{b\,c}\,\widetilde{\mathrm{visAmp}}_{bl\,c}}{\sum_c W_{b\,c}} \tag{78}$$

where the weight is a combination of the distance in time ($T - T_c$), and of the precision of the individual measurements:

$$W_{b\,c} = \frac{\exp(-2\,(T-T_c)^2/\Delta^2)}{\mathrm{median}_l(\sigma^2_{blc})} \tag{79}$$

The free parameter Δ corresponds to the expected stability of the instrumental + atmosphere responses. Calibrations recorded within this time interval are mostly averaged, calibrations separated by about this time interval are mostly interpolated, and calibrations far away from this time interval will be discarded (closed one becomes dominent).

A similar approach is used for the phases:

$$\widetilde{\mathrm{tfPhi}}_{bl} = \arg\big(\sum_{c} W_{b\,c} \, \exp(i \, \widetilde{\mathrm{visPhi}}_{bl\,c}) \big) \tag{80}$$

Calibration

$$\widetilde{\mathrm{visAmp}}_{bl} = \frac{\widetilde{\mathrm{visAmp}}_{bl}}{\widetilde{\mathrm{tfAmp}}_{bl}} \tag{81}$$

$$\widetilde{\mathrm{visPhi}}_{bl} = \arg\big(\exp(i \, (\widetilde{\mathrm{visPhi}}_{bl} - \widetilde{\mathrm{tfPhi}}_{bl})) \big) \tag{82}$$

10.21 Processing of ACQ

Pre-processing

The pre-processing re-interpolates the bad-pixels. For the pupil images, the background is computed from the median of the images and subtracted.

Analysis of pupil images

The pupil scale, rotation and position is computed for each frame by adjusting a spot model (16 spot on the expected grid) to the image. the degree of freedom are: rotation angle of pupil (same for the four sub-appertures), scaling of pupil (same for the four sub-appertures), mean x and y of each sub-apperture, FWHM (same for all diodes), and flux (one per diode).

The mean displacement of the pupil is computed (mean of the four sub-apperture) as PUPIL_X and PUPIL_Y. Then the expected opd introduced by the mean pupil displacement is computed by:

$$\mathrm{OPD\_PUPIL} = \mathrm{PUPIL\_S} \, . \, \mathrm{SEP} \, . \, (\mathrm{PUPIL\_X} \, \cos\Psi + \mathrm{PUPIL\_Y} \, \sin\Psi) \tag{83}$$

where Ψ is the angle of the binary separation in the ACQ camera frame, PUPIL_S is the pupil scale (mm/pix) in the ACQ images (read from header, DROTOFF position), and SEP is the binary separation in [rad] (read from header).

Analysis of field images

The positions on the acquisition camera detector of the SC and FT targets are measured by fitting a gaussian profile, after an initial guess based on the separation SOBJ.[X|Y], given in the main header, between the FT and

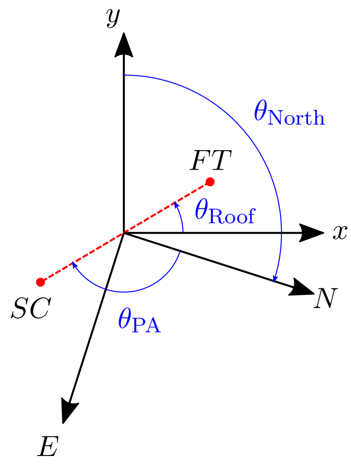

Figure 10.3: Geometry of the Acquisition Camera field image. The nominal positions of the FT and SC targets (red dots) are given in the acquisition camera field coordinates (x, y). The orientation of the FT/SC pair matches the orientation of the roof prism θ_{Roof} given by the keywords ESO.INS.DROTOFFn. The direction of North θ_{North} on the acquisition camera field is related to the position angle of the pair θ_{PA}, calculated from the offset keywords ESO.INS.SOBJ.[X|Y], and to the roof angle θ_{Roof}.

SC targets. The results of the fitting process is available in table OI_VIS_ACQ, columns FIELD_SC_[X|Y] and FIELD_FT_[X|Y], with associated errors FIELD_SC_[X|Y]ERR and FIELD_FT_[X|Y]ERR.

In true dual-field mode, and when the separation SOBJ.[X|Y] is not zero, the plate scale of the acquisition camera is measured for each frame as the ratio between the separation and the distance between the two detected objects:

$$\text{FIELD\_SCALE} = \frac{||\text{SOBJ.[X|Y]}||}{||\text{FIELD\_SC\_[X|Y]} - \text{FIELD\_FT\_[X|Y]}||}, \tag{84}$$

and associated error FIELD_SCALEERR.

In addition, an error signal FIELD_FIBER_D[X|Y] is generated between the detected SC/FT target relative positions and the SC/FT fibre relative positions, taking into account any dithering offset SOBJ.OFF[X|Y].

$$\begin{aligned} \text{FIELD\_D[X|Y]} = \quad & (\text{FIELD\_SC[X|Y]} - \text{FIELD\_FT[X|Y]}) \\ & + \text{SOBJ.OFF[X|Y]}/\text{FIELD\_SCALE} \\ & - (\text{ACQ.FIBER.SC[X|Y]} - \text{ACQ.FIBER.FT[X|Y]}) \end{aligned} \tag{85}$$

This quantity is later used to unwrap the telescope metrology diode signals.

Finally, a Strehl ratio is estimated from each frame and stored in the FIELD_STREHL column.

10.22 Demodulation of metrology signal

Description of metrology data

The metrology data is stored as a FITS Table in the `METROLOGY` HDU (10^{th}) as in table 10.1. The voltages `VOLT` are composed of 80 columns:

Name	Size	Type	TFORM
TIME		Int32	1J
VOLT	(80,)	Float32	80E
POWER_LASER		Float32	1E
LAMBDA_LASER		Float32	1E

Table 10.1: Metrology table

- 2 directions (x and y) per diodes
- 4 diodes per telescope (one on each spider)
- 1 fiber coupler diode (labeled `FC`) per telescope
- 4 telescopes
- 2 sides: `FT` and `SC`

The signals (the table column) are sampled at 500 Hz leading to a very large number of rows.

Metrology modulation

When the pupil modulation is on (`ESO INS PMC1 MODULATE` keyword is `true`), the metrology signal is modulated at a frequency of $f = 1\,\mathrm{Hz}$. This modulation does not affect the fiber coupler diode.

For each diode in {D1, D2, D3, D4}, we define the complex measurement $\boldsymbol{v} \in \mathbb{C}^N$ with $v_i = x_i + \jmath y_i\,, \quad \forall i \in \{1, N\}$ and N is the number of rows in the metrology data. The modulation as a function of time is:

$$\boldsymbol{v} = \exp\left(\jmath\, b \sin\left(\boldsymbol{\omega} + \phi\right)\right) \,. \tag{86}$$

where $\omega = 2\,\pi\,\boldsymbol{t}$ is the modulation pulsation built from the `TIME` column, the amplitude b and the phase ϕ are the modulation parameters that need to be estimated to demodulate the metrology signal.

For a diode, the measured modulated voltage data $\boldsymbol{d}$ can be modeled as follows:

$$\boldsymbol{d} = \left(\boldsymbol{c} + \boldsymbol{s} \times \exp\left(\jmath\, b \sin\left(\boldsymbol{\omega} + \phi\right)\right)\right) \times \exp\left(\jmath\, \Phi_{\mathrm{FC}}\right) + \boldsymbol{e}\,, \tag{87}$$

where $\times$ is the element-wise multiplication, $\boldsymbol{s}$ is the sought-after demodulated metrology signal, Φ_{FC} is the phase of the fiber coupler measurement accounting for FDDL movements. $\boldsymbol{c} = x_0 + \jmath\, y_0$ is the center of the pupil and $\boldsymbol{e}$ is a vector representing the measurement errors, which are assumed to be Gaussian centered, independent, and identically distributed.

Demodulation implementation

The pipeline implements a simplified version of the model in Eq. 87, such that the phase of the fiber coupler Φ_{FC} is assumed to be constant over a 'chunk' of duration `MAX_SECONDS_PER_CHUNK = 100` seconds. Each chunk is demodulated independently. Before processing, the metrology signal is first centered (such that

$\boldsymbol{c} = \mathbf{0}$). If the keyword `gravity.metrology.use-dark-offsets=TRUE`, centering is done using the dark file if it exists; otherwise, it uses hard-coded diode centers computed by S. Gillessen and set in the array `diode_zeros` in the file `gravi_demodulate.c`. Within each chunk, the metrology data is then averaged on a single second such that:

$$\bar{d}_k = \langle d_{k+i\,T}, \forall i \in \mathbb{N} \rangle \ , \tag{88}$$

where T is the number of steps in 1 second, equal to the metrology sampling rate and set by the hard-coded constant `STEPS_PER_SECOND=500`. The model of the metrology described by Eq. 87 is rewritten as:

$$\bar{\boldsymbol{d}} = a \times \exp\left(\jmath\, b \sin\left(\bar{\boldsymbol{\varpi}} + \phi'\right) + \phi\right) + \boldsymbol{e}\,, \tag{89}$$

where $\bar{\omega}_k = \frac{2\pi}{500}\, k$ is the modulation pulsation at the k^{th} step within one second. Note that within the pipeline, these variables are relabelled as `a`, `b`, `pha1`, `pha2` where `pha1`$= \phi'$, `pha2`$= \phi$ and $s = a \exp(\jmath\, \phi)$.

For each chunk, the parameters are estimated using the Nelder-Mead Simplex algorithm minimizing the chi-square χ^2 :

$$\chi^2\left(a, b, \phi, \phi'\right) = \left\| \bar{\boldsymbol{d}} - a \times \exp\left(\jmath\, b \sin\left(\bar{\boldsymbol{\varpi}} + \phi'\right) + \phi\right) \right\|^2 \tag{90}$$

To ensure a global minimum, this minimization is performed multiple times with different initializations:

- $a_0 = \text{std}(\bar{\boldsymbol{d}})$
- $b_0 = 0.25$
- $\phi_0 = \{-\pi/2, 0\}$
- $\phi'_0 = \{-\pi/2, 0\}$

The phases are then unwrapped if needed.

Once the parameters a, b, ϕ, ϕ' have been estimated for each chunk, the modulation ψ is computed over one second:

$$\psi_k = -\arctan(\bar{d}_k) - \phi \tag{91}$$

Finally, the metrology signal is demodulated, chunk-wise, with each demodulated sample p_i being:

$$p_i = d_i \exp\left(-\jmath\, \psi_{\text{mod}(i,T)}\right) \tag{92}$$

where $\text{mod}(i, T)$ is the modulo of i by T.

10.23 Processing of MET and FDDL

From Volts to real time SC-FT phases

The signals of all diodes are analysed independently, without prior correction of the voltage, and converted into a real-time SC-FT phase for each diode. This real-time SC-FT phase is unwrapped and forced to match the OCS.MET reference. Both the unwrapping and the absolute reference are done with fringe-integer corrections only. These quantities are labeled OPD_TEL and OPD_FC in the OI_VIS_MET table.

Averaging SC-FT phase inside SC DIT

The phase of a diode is averaged inside each SC DIT, as a scalar quantity (not phasor). This averaged signal is stored for each beam in OI_FLUX table and each base in OI_VIS table. These quantities are labeled OPD_MET_TEL and OPD_MET_FC.

Average the FDDL inside SC DIT

The mean SC and FT FDDLs strain gauge voltage during each frame of the SC, and per beam, is stored in the OI_FLUX table. Columns are labelled SC_POS and FT_POS.

10.24 Applying dispersion correction to MET

Correction of FFDL non linearity

The DISP_MODEL provides, for each beam t:

- linSC_{tm} : the non-linearity coefficients of order m of the SC FDDL.
- linFT_{tm} : the non-linearity coefficients of order m of the FT FDDL.

SC_POS and FT_POS are first corrected from non-linearity of the strain gauge, and then averaged to get the mean position of the two fibers:

$$\mathrm{FDDL}_{tf} = \frac{\sum_m (\ \mathrm{linSC}_{tm}\, \mathrm{SC\_POS}_{tf}^m\ ) \ +\ \sum_m (\ \mathrm{linFT}_{tm}\, \mathrm{FT\_POS}_{tf}^m\ )}{2} \tag{93}$$

Dispersion-included metrology signal

The DISP_MODEL allows to compute the *mean* refractive index of SC and FT FDDL at wavelength l for each beam t, normalized to the one at the metrology wavelength (hereafter called nmean_{tl}); and the *differential* refractive index between SC and FT FDDL wavelength l for each beam t, normalized to the one at the metrology wavelength (hereafter called ndiff_{tl}).

In practice, these two quantities are stored as a polynomial model versus wavenumber, centered in the middle of the K-band ($\lambda_0 = 2.2\mu$m):

$$\mathrm{nmean}_{tl} = \sum_m (\ \mathrm{nmean}_{tm}\, (\frac{\lambda_0}{\lambda_l} - 1)^m\ ) \tag{94}$$

$$\mathrm{ndiff}_{tl} = \sum_m (\ \mathrm{ndiff}_{tm}\, (\frac{\lambda_0}{\lambda_l} - 1)^m\ ) \tag{95}$$

where nmean_{tm} and ndiff_{tm} are the polynomial coefficients of order m of beam t, read from the DISP_MODEL.

From the dispersion and the metrology signal, we can compute the amount of delay introduced by the differential delay-line, for each spectral channel and each baseline. This quantity is called $\mathrm{OPD\_DISP}_{bfl}$:

$$\begin{aligned}\mathrm{OPD\_DISP}_{bfl} &= \mathrm{nmean}_{t_1\,l}\,\mathrm{OPD\_MET\_FC}_{t_1\,f} - \mathrm{nmean}_{t_2\,l}\,\mathrm{OPD\_MET\_FC}_{t_2\,f} + \\ &\quad \mathrm{ndiff}_{t_1\,l}\,\mathrm{FDDL}_{t_1\,f} - \mathrm{ndiff}_{t_2\,l}\,\mathrm{FDDL}_{t_2\,f} \end{aligned} \tag{96}$$

Dispersive group-delay and remaining phase

The signal $\mathrm{OPD\_DISP}_{bfl}$ contains a fraction of group-delay coded as a spectral slope. This shall be properly taken into account when attempting to compute astrometric/absolutes phases out of the dataset.

Therefore the pipeline also provide an additional representation of the same quantity, but decomposed into the total group-delay in the middle of the K-band ($\mathrm{GDELAY\_DISP}_{bf}$ in unit of distance, thus [m]), and the remaining phase ($\mathrm{PHASE\_DISP}_{lbf}$, in [rad]).

The total group-delay introduced by FDDL in the middle of the band is:

$$\mathrm{GDELAY\_DISP}_{bf} = \frac{\lambda_{l_1}^{-1}\,\mathrm{OPD\_DISP}_{l_1\,bf} - \lambda_{l_2}^{-1}\,\mathrm{OPD\_DISP}_{l_2\,bf}}{\lambda_{l_1}^{-1} - \lambda_{l_2}^{-1}} \tag{97}$$

where l_1 and l_2 are two consecutive wavelength channel in the middle of the band. The remaining phase is:

$$\mathrm{PHASE\_DISP}_{lbf} = \arctan(\,\exp(\frac{2i\pi}{\lambda_f}\,(\mathrm{OPD\_DISP}_{lbf} - \mathrm{GDELAY\_DISP}_{bf}))\,) \tag{98}$$

The $\mathrm{GDELAY\_DISP}_{bf}$ quantities are defined to within a constant: the so-called dispersed metrology zeros $\mathrm{Z\_DISP}_t$ (one per beam). When combined with the group-delays from SC and FT, the $\mathrm{GDELAY\_DISP}_{bf}$ can be used to construct a group-delay astrometry, e.g:

$$\begin{aligned}\vec{B_{b_{ij}}}.\vec{\delta} + (\mathrm{Z\_DISP}_i - \mathrm{Z\_DISP}_j) &= \mathrm{GDELAY\_DISP}_{b_{ij}f} \\ &\quad -(\mathrm{GDELAY\_SC}_{b_{ij}f} - \mathrm{GDELAY\_FT}_{b_{ij}f})\end{aligned} \tag{99}$$

In order to use the telescope metrologies, additional corrections need to be added to the astrometric equation above:

$$\begin{aligned}\vec{B_{b_{ij}}}.\vec{\delta} + (\mathrm{Z\_DISP}_i - \mathrm{Z\_DISP}_j) &= \mathrm{GDELAY\_DISP}_{b_{ij}f} \\ &\quad +(\mathrm{TELFC\_MCORR} + \mathrm{FC\_CORR}) \\ &\quad -(\mathrm{GDELAY\_SC}_{b_{ij}f} - \mathrm{GDELAY\_FT}_{b_{ij}f})\end{aligned} \tag{100}$$

10.25 Astrometric transformations and projected baseline

Due to its astrometric requirements, the pipeline uses the ERFA astrometric library (derived from SOFA) to transform coordinates between the Celestial and Observed reference frames. The SOFA documentation gives a good description of the transformations between the two reference frames; the present manual only describes its use.

For each observation, the mid-point (α_c, δ_c) between the FT and SC targets in celestial (ICRS) is first converted into barycentric (BCRS) by applying space motion (proper motion, parallax, and radial velocity).

$$(\alpha_c, \delta_c) \Rightarrow (\alpha_b, \delta_b) \equiv \vec{e}_{Wb} \tag{101}$$

The equatorial directions $(\vec{e}_{Ub}, \vec{e}_{Vb})$ are then computed from the pointing direction $\vec{e}_{Wb}$ and the direction of the pole $\vec{e}_{Zb}$ as follows:

$$\vec{e}_{Ub} = \frac{\vec{e}_{Zb} \times \vec{e}_{Wb}}{||\vec{e}_{Zb} \times \vec{e}_{Wb}||} \tag{102}$$

$$\vec{e}_{Vb} = \vec{e}_{Wb} \times \vec{e}_{Ub} \tag{103}$$

The unit vectors $\vec{e}_{Ub}$ and $\vec{e}_{Vb}$ point in the direction of increasing right ascension and declination respectively.

Since ERFA is only capable of transforming coordinates, a set of four coordinates at a small $\epsilon = \pm 10\text{arcsec}$ angle in right ascension and declination is used instead: the cardinal asterism. They are created by rotating the pointing direction $\vec{e}_{Wb}$ around the right ascension and declination directions $(\vec{e}_{Ub}, \vec{e}_{Vb})$ by this small ϵ angle.

$$(\alpha_b + \epsilon, \delta_b) \equiv \vec{e}_{W+Ub} = \mathcal{R}_{-\epsilon\vec{e}_{Vb}}(\vec{e}_{Wb}) \tag{104}$$

$$(\alpha_b - \epsilon, \delta_b) \equiv \vec{e}_{W-Ub} = \mathcal{R}_{+\epsilon\vec{e}_{Vb}}(\vec{e}_{Wb}) \tag{105}$$

$$(\alpha_b, \delta_b + \epsilon) \equiv \vec{e}_{W+Vb} = \mathcal{R}_{+\epsilon\vec{e}_{Ub}}(\vec{e}_{Wb}) \tag{106}$$

$$(\alpha_b, \delta_b - \epsilon) \equiv \vec{e}_{W-Vb} = \mathcal{R}_{-\epsilon\vec{e}_{Ub}}(\vec{e}_{Wb}) \tag{107}$$

ERFA is then used to transform the pointing origin and cardinal asterism coordinates from barycentric to observed reference frame. This transformation uses the Earth Orientation Parameters (UT1-UTC and polar motion) published by the IERS.

$$(\alpha_b, \delta_b) \Rightarrow (\alpha_o, \delta_o) \equiv \vec{e}_{Wo} \tag{108}$$

$$(\alpha_b + \epsilon, \delta_b) \Rightarrow (\alpha_o + \epsilon, \delta_o) \equiv \vec{e}_{W+Uo} \tag{109}$$

$$(\alpha_b - \epsilon, \delta_b) \Rightarrow (\alpha_o - \epsilon, \delta_o) \equiv \vec{e}_{W-Uo} \tag{110}$$

$$(\alpha_b, \delta_b + \epsilon) \Rightarrow (\alpha_o, \delta_o + \epsilon) \equiv \vec{e}_{W+Vo} \tag{111}$$

$$(\alpha_b, \delta_b - \epsilon) \Rightarrow (\alpha_o, \delta_o - \epsilon) \equiv \vec{e}_{W-Vo} \tag{112}$$

Then the pointing directions to the cardinal asterism are combined to generate the equatorial directions $(\vec{e}_{Uo}, \vec{e}_{Vo})$ in the observed reference frame.

$$\vec{e}_{Uo} = \frac{1}{2\epsilon}\vec{e}_{Wo} \times (\vec{e}_{W+Uo} \times \vec{e}_{W-Uo}) \tag{113}$$

$$\vec{e}_{Vo} = \frac{1}{2\epsilon}\vec{e}_{Wo} \times (\vec{e}_{W+Vo} \times \vec{e}_{W-Vo}) \tag{114}$$

Even though $(\vec{e}_{Ub}, \vec{e}_{Vb}, \vec{e}_{Wb})$ is an orthonormal basis in the barycentric reference frame, $(\vec{e}_{Uo}, \vec{e}_{Vo}, \vec{e}_{Wo})$ is not orthonormal in the observed reference frame. The pointing vector $\vec{e}_{Wo}$ is a unit vector, but the right ascension and declination vectors $(\vec{e}_{Vo}, \vec{e}_{Wo})$ are not: they carry a scaling factor associated to the effect of astronomical aberration.

The azimuth and zenith distance directions $(\vec{e}_{AZo}, \vec{e}_{ZDo})$ are also calculated from the observed pointing direction $\vec{e}_{Wo}$ and the zenith direction $\vec{e}_{Zo}$.

$$\vec{e}_{AZo} = \frac{\vec{e}_{Wo} \times \vec{e}_{Zo}}{||\vec{e}_{Wo} \times \vec{e}_{Zo}||} \tag{115}$$

$$\vec{e}_{ZDo} = \vec{e}_{Wo} \times \vec{e}_{Azo} \tag{116}$$

The vectors $\vec{e}_{Uo}, \vec{e}_{Vo}, \vec{e}_{Wo}, \vec{e}_{AZo}, \vec{e}_{ZDo}$ populate the columns **E_U**, **E_V**, **E_W**, **E_AZ**, **E_ZD** of the pipeline products.

Finally, the projected baseline is calculated as the scalar product between the physical baseline $\vec{B}_o$ in the observed reference frame and the equatorial directions $(\vec{e}_{Uo}, \vec{e}_{Vo})$ also in the observed reference frame.

$$B_{Ub} = \vec{B}_o \cdot \vec{e}_{Uo} \tag{117}$$

$$B_{Vb} = \vec{B}_o \cdot \vec{e}_{Vo} \tag{118}$$

The projections B_{Ub}, B_{Vb} populate the columns **UCOORD**, **VCOORD** of the pipeline products.

Last element of importance, the baseline vector $\vec{B}_o$ in the observed reference frame is supposed to be the physical or vacuum vector. However, the telescope positions reported in the fits headers correspond to OPD models that are scaled by the refractive index of air ($n = 1.0002028$) at the atmospheric pressure of Paranal and at the wavelength of the Helium Neon laser of the VLTI delay lines. The pipeline takes care of compensating this refractive index scaling when computing the observed baseline from the headers.

$$\vec{B}_o = 1/n\vec{B}_{header} \tag{119}$$

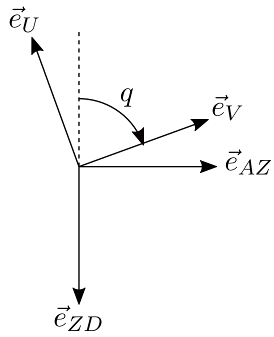

Figure 10.4: Geometry in the pupil, looking toward the target. It shows the azimuth and zenith distance directions $(\vec{e}_{AZ}, \vec{e}_{ZD})$, the right ascension and declination directions $(\vec{e}_U, \vec{e}_V)$, and the parallactic angle q.

10.26 The metrology data flow for phase astrometry

The metrology computations starts with the Volts values in the raw fits files. They are used to compute at 2kHz the PHASE_FC (1 value per telescope) and PHASE_TEL (4 values per telescope). To convert voltages to phases, two algorithms are implemented in the pipeline: the one from the RTC (which only look at the past), and the new algorithm from the DRS (which is doing zero lag temporal smoothing). The default algorithm is the one from the DRS, but that can be changed by the gravity_vis parameter "–use-met-rtc" (default = false).

These phases are converted into OPD using the mean wavelength from the METROLOGY table (the value is stored in the QC "MET LAMBDA MEAN" inside the header). They are used to produced 2 quantities, which are necessary to calculate the astrometry: OPD_DISP and PHASE_MET_TELFC.

The first quantity is OPD_DISP. This quantity is obtained from PHASE_FC, which is converted to OPD_FC in the OI_MET table. OPD_FC us then averaged to the SC DIT frequency and stored into the OI_FLUX table under the name OPD_MET_FC. Last, it is combined with the position of the FDDL sensor gauge, and using a model of the fiber dispersion, it is converted into a wavelength dependent OPD_DISP (in meters). To be noted, it is possible to add a constant value to this OPD_DISP calculation thanks to the –use-met-zero parameter. It is recommanded not to use this parameter because the keyword values in the files header are not always consistent between exposures.

The PHASE_MET_TELFC actually corresponds to the difference between the phase at the telescope center and the phase measured on the fiber coupler (OPD_FC). The issue at stake is to account from the fact that the metrology at the telescope level is not properly unwrapped, and 2π offsets can exist. To do so, several intermediary product are computed:

- OPD_FC_CORR, to project the diode at the fiber coupler to the center of the telescope
- OPD_TEL_CORR, to project the phase at the center of the telescope to the position of the diode sensors. This calculation includes a model of astigmatism, and an approximation of the separation of the FC/SC fibers.
- OPD_TEFC_CORR_XY, to account for residual astigmatism and the tip-tilt between the two fibers.
- OPD_TEFC_CORR, are the phase residuals, which will have to be included in the astrometry (hopefully, they are smaller than 2π to be properly averaged.

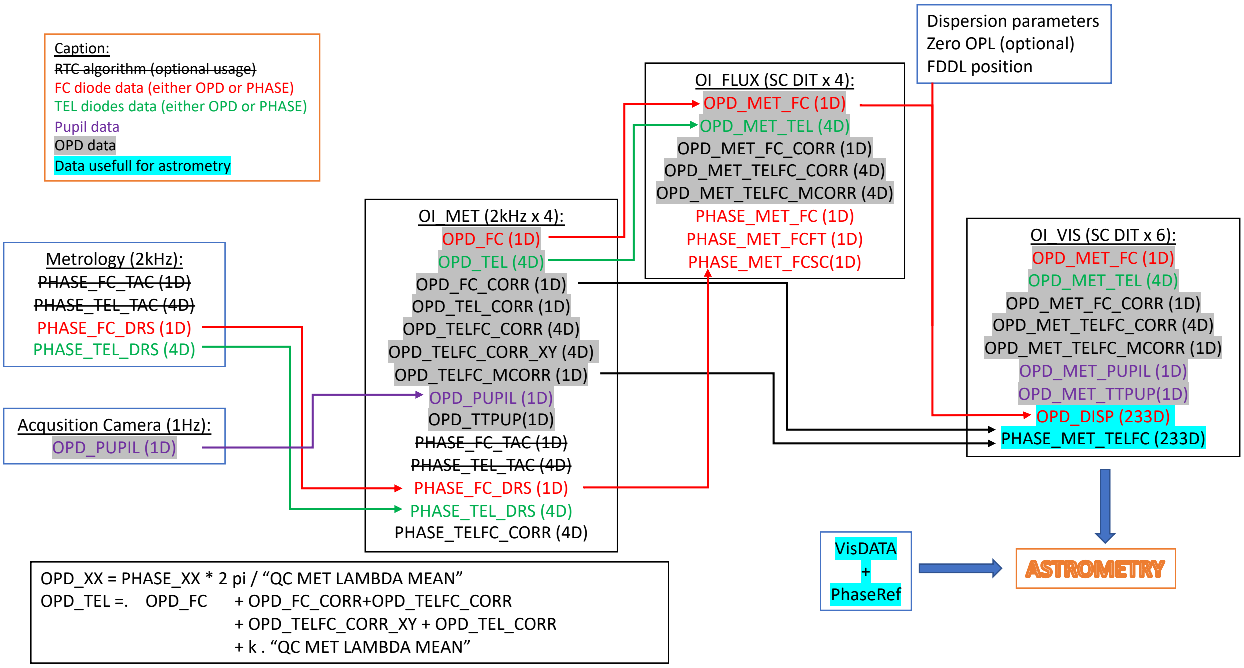

Figure 10.5: The data flow of the metrology, starting from the phase extracted from the metrology voltage, to the final products OPD_DISP and PHAS_MET_TEFC

The last part is to use the OPD_FC_CORR and OPD_TEFC_CORR value and add them together. The sum is then averaged as a wavelength dependant phasor over an SC DIT. The phase of the phasor is therefore of length Nwave (233 in MED resolution, as shown in Fig. 10.5). It is stored in the OI_VIS table.

The OPD_DISP and PHASE_MET_TELFC values, both stored in the astrored files, can then be used to perform astrometry as described in section 10.27.

10.27 Phase referencing the science visibilities for astrometry

10.27.1 Phasing the VISDATA

Astrometry is typically done with the *_ASTROREDUCED products. These products are intermediary fits that can be obtained using the "–astro-file=TRUE" option with esorex gravity_vis recipe (they are not produced by default). The advantage of this data product is that the Science DITs are not co-added. This is how astronomers have all latitude to co-add the complex visibilities coherently at any position in space. Alternatively, astronomers can use directly VISPHI in the dualvis files, but the coadding will be done for the X and Y positions which are in the header.

Of importance is the OI_VIS table in the *_ASTROREDUCED data product. Inside are stored the complex coherent amplitude (VISDATA), the differential fibered delay lines optical delay (OPD_DIS), the differential phase between Fiber Coupler and telescope diodes (PHASE_MET_TELFC), and the phase of the fringe tracker

(PHASE_REF). All theses values, in this table, are sampled at the frequency rate of the SC detector. They are also re-sampled to the spectral resolution of the science detector. It is by adding all these variable that one can phase the science VISDATA, and perform astrometry.

To be more explicit, the goal, to be able to do astrometry, is to reference the phase of the science complex visibility with respect to the phase of the object observed on the fringe tracker. This is done as follows:

$$VISDATA_{\mathrm{phasedFT}} = VISDATA \times exp\left(i\left[PHASE\_REF - PHASE\_MET\_TELFC - \frac{2\pi}{\lambda}OPD\_DISP\right]\right) \tag{120}$$

It is important to realise at this point that $VISDATA_{\mathrm{phasedFT}}$ is a highly wrapped quantity. It does not convey the notion of interferometric field. It is therefore often useful to reference the VISDATA to an arbitrary position in the field of the interferometer. To do so, one can add a specific phase offset, as follows:

$$VISDATA_{\mathrm{phasedXY}} = VISDATA_{\mathrm{phasedFT}} \times exp\left(i\frac{2\pi}{\lambda}(UCOORD \times X + VCOORD \times Y)\right) \tag{121}$$

using the UCOORD and VCOORD columns in the OI_VIS table. These values are the projected baseline coordinates, also described in the previous section by Eqs. (117) and (118). X and Y are the position coordinates, in radians, from the position of the fringe tracker star.

Because a few lines of python code are worth a thousands words, here is the code to reference the VISDATA to the fringe tracker object position.

```
extension = 10 # for COMBINED observations
wave=fits.getdata(f,'OI_WAVELENGTH',extension).field('EFF_WAVE')
opdDisp  = fits.getdata(f,'OI_VIS',extension).field('OPD_DISP')
phaseTelFc  = fits.getdata(f,'OI_VIS',extension).field('PHASE_MET_TELFC')
phaseRef  = fits.getdata(f,'OI_VIS',extension).field('PHASE_REF')
phaseFt = phaseRef - 2*pi/wave*opdDisp - phaseTelFc
visData  = fits.getdata(f,'OI_VIS',extension).field('VISDATA')
visData_phasedFT = visData * exp(1j*phaseFt)
```

with "f" the astroreduced file. If the user wants to reference the phase to the GRAVITY pointing position (value in the header), it can be done as follows:

```
X=fits.getheader(f)["HIERARCH ESO INS SOBJ X"]
Y=fits.getheader(f)["HIERARCH ESO INS SOBJ Y"]
X/=1000/(180/pi*3600)
Y/=1000/(180/pi*3600)
ucoord=fits.getdata(f,'OI_VIS',extension).field('UCOORD')
vcoord=fits.getdata(f,'OI_VIS',extension).field('VCOORD')
phaseXY=2*pi*( ucoord*X + vcoord*Y)[:,None]/wave
visData_phasedXY = visData_phasedFT * exp(1j*phaseXY)
```

$VISDATA_{\mathrm{phasedXY}}$ is now phased with respect to an arbitrary position: XY.

10.27.2 "Zeroing" the phase constant from the metrology

We now have two quantities, $VISDATA_{\rm phasedFT}$ and $VISDATA_{\rm phasedXY}$. To perform the final astrometry, it is now important to "zero" the astrometry. To "zero" the astrometry, we need 2 distinct observations. To start with a general case, we can assume that we have 2 distinct observations, of 2 different binary systems. The physical separation between the 2 objects are respectively (X_1, Y_1) and (X_2, Y_2). The 2 arguments of the VISDATA are therefore:

$$\arg(VISDATA_{\rm phasedFT1}) = \frac{2\pi}{\lambda}(UCOORD_1 \times X_1 + VCOORD_1 \times Y_1) + \phi_\lambda \tag{122}$$

$$\arg(VISDATA_{\rm phasedFT2}) = \frac{2\pi}{\lambda}(UCOORD_2 \times X_2 + VCOORD_2 \times Y_2) + \phi_\lambda \tag{123}$$

where ϕ_λ is the constant, internal, phase offset. The key, to do phase astrometry, is to determine this offset (we call that "zeroing" the metrology).

To do so, there are 2 options, depending if the observations are done on-axis or off-axis:

1. On-axis mode: the second observation is obtained on the FT target. In that case, $X_2 = 0$ and $Y_2 = 0$. If we have the possibility to get such an observation (it requires using the 50/50 beam splitter), the metrology phase offset can be obtained directly from the on-star observation:

$$\phi_\lambda = \arg(VISDATA_{\rm phasedFT2}) \tag{124}$$

2. Swap in off-axis mode: in this mode, we rotate the K-mirror to swap the fringe tracker source with the science combiner source: $X_2 = -X_1$ and $Y_2 = -Y_1$. In that situation:

$$\begin{aligned}\phi_\lambda &= \arg(VISDATA_{\rm phasedFT1} VISDATA^*_{\rm phasedFT2})/2 \\ &- \frac{2\pi}{\lambda}\left[\frac{UCOORD_1 - UCOORD_2}{2} \times X_1 + \frac{VCOORD_1 - UCOORD_2}{2} \times Y_1\right].\end{aligned} \tag{125}$$

   If the swap is done fast enough, $UCOORD_1 \approx UCOORD_2$ and $VCOORD_1 \approx VCOORD_2$, and to first order, the values of X_1 and Y_1 do not matter. For serious astrometry, astronomers should fit simultaneously ϕ_λ, X_1 and Y_1.

Once ϕ_λ is known, X_1 and Y_2 (or X_2 and Y_2) values can be obtained by fitting the Eq. (122) or (123) to the $VISDATA_{\rm phasedFT}$ measurement. The only remaining difficulty is that the fit is not convex because of the 2π wrapping. So it is important to start with a good initialisation of the X_1 and Y_2 values to avoid falling into a local minimums.

A Installation

One is advised to read the installation instructions delivered with the GRAVITY pipeline distribution kit. These release-specific instructions can be found in the file `README` located in the top-level directory of the unpacked GRAVITY pipeline source tree. The supported platforms are listed in Section A.1. It is recommended reading through Section A.2.2 before starting the installation.

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

A.1 Supported platforms

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

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

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

A.2 Building the GRAVITY pipeline

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

A.2.1 Requirements

To compile and install the GRAVITY pipeline one needs:

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

An installation of the Common Pipeline library (CPL) must also be available on the system. Currently the CPL version 2.1.1 or newer is required. The CPL distribution can be obtained from [3].

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

In order to run the GRAVITY 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 17). At least one of them must be installed. The *EsoRex* and *Gasgano* packages are available at `https://www.eso.org/cpl/esorex.html` and `https://www.eso.org/gasgano` respectively.

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

A.2.2 Compiling and installing the GRAVITY pipeline

The GRAVITY pipeline distribution kit 1.0 contains:

gravity-manual-1.0.pdf	The GRAVITY pipeline manual
install_pipeline	Install script
cpl-7.2.2.tar.gz	CPL 7.2.2
esorex-3.13.3.tar.gz	esorex 3.13.3
gasgano-2.4.8.tar.gz	GASGANO 2.4.8for Linux
gravity-1.8.0.tar.gz	GRAVITY 1.8.0
gravity-calib-1.8.0.tar.gz	GRAVITY calibration files 1.8.0

Here is a description of the installation procedure:

1. Change directory to where you want to retrieve the GRAVITY pipeline recipes 1.8.0package. It can be any directory of your choice but not:

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

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

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

4. Unpack using the following command:

   ```
   tar -xvf \pipename-kit-\pipelinevers.tar
   ```

   Note that the size of the installed software (including *Gasgano*) together with the static calibration data is about 1.3 Gb, mainly du to the test data.

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

   ```
   cd \pipename-kit-\pipelinevers
   ```

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

```
./install_pipeline
```

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

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

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

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

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

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

B Abbreviations and acronyms

ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CalibDB	Calibration Database
CPL	Common Pipeline Library
DFO	Data Flow Operations department
DFS	Data Flow System department
DHS	Data Handling Server
DO	Data Organizer
DMD	Data Management and Operations Division
DRS	Data Reduction System
ESO	European Southern Observatory
ESOREX	ESO-Recipe Execution tool
FDDL	Fibered Differential Delay Lines
FITS	Flexible Image Transport System
FOV	Field Of View
FT	Fringe-Tracker
SC	Science-Combiner
GUI	Graphical User Interface
OB	Observation Block
OIFITS	OIFITS format, see [4]
PSO	Paranal Science Operations
QC	Quality Control
RON	Read Out Noise
SOF	Set Of Frames
UT	Unit Telescope
AT	Auxiliary Telescope
VLTI	Very Large Telescope Interferometer