VIRTIS-H CALIBRATION

	NAME	FUNCTION	SIGNATURE	DATE
Prepared by	JM REESS F. HENRY	Engineer Engineer		
Approved by	JM REESS	V-H PM		
Authorized by	JM REESS	V-H PM		

1 Document change record

Issue	Date	Name	#	Paragraph	Description of the modification	Reason of the modification
1.0 1.1	21/03/13 16/06/13	JMR FH			First draft Corrections	

2 Contents

3 Scope of the document

This document describes how the science data of VIRTIS-H are calibrated from raw measurements.

Routines names used in this document are those of our calibration pipeline. As these routines cannot be used without the whole database, their source won't be furnished.

4 Applicable documents

To be completed

5 Reference documents

RD1. VIR-DLR-MA-001 : Virtis Software User Manual

RD2. VVX-LES-TN-2272: Definition update of the Virtis-H observation sequence parameters

RD3. VIR-INAF-IC-007: VIRTIS To Planetary Science Archive Interface Control Document (EAICD)

RD4. VVX-DLR-NC-002: Non-conformance report : H-PEM shutter synchronisation

6 List of acronyms

ADU	Arbitrary Digital Units
IAS	Institut d'Astrophysique Spatiale (Orsay, France)
LESIA	Laboratoire d'Etudes Spatiales et d'Instrumentation en Astrophysique (Observatoirede Paris, Meudon section, France)
ME	Main Electonics
PDS	Planetary Data System
SCET	SpaceCraft Elapsed Time
TC	TeleCommand
TM	TeleMetry

7 Virtis-H data

7.1 Detector size

The Virtis-H detector is a matrix of 438x270 pixels. During development, subsystem tests of Virtis-H were performed without ME, and the data obtained are 2d images of 438 pixels width by 270 pixels height.

When operated with the ME, only a window of 256 pixels width per 432 pixels height of the detector is used. The position of this window is set by the parameters Xwin and Ywin.

Detector	Xwin	Ywin
FM	3	7

Table 1: window position for each detector used.

7.2 File formats

A **qube** is a PDS formatted file that contains data. It is read by the IDL routine `virtispds('filename')`. This routine gives in output a structure.

A **raw qube** contains raw data, acquired in nominal mode. Relevant fields of the structure are:

- `qube`: 3d integer array containing the data. Dimensions are `spec_dim x 64 x n_frame`
  - `spec_dim` is the size of one spectrum, and is equal to 8 orders x 432 pixels = 3456
  - `64` is the number of spectra per frame
  - `n_frame` is the number of frame in the qube

  Note: The reason of this structure is that Virtis-H spectra are compress onboard by packets of 64 spectra. For more details, see RD1.
- `suffix`: 3d array containing the housekepings

A **raw backup qube** contains raw data acquired in backup mode. In this mode, the whole detector is read. Dimensions are `432 x 256 x n_acquisition`. The fields of the structure are the same as those of the raw qube.

A **calibrated qube** contains calibrated data. Units are W/m^2/sr/micron. Relevant fields of the structure are:

- `qube`: 3d float array containing the data. Dimensions are `3456 x n_acquisitions`
  - `n_acquisitions` is `64 x n_frame`
  - `suffix`: 3-columns array containing the time in SCET coded on 3 unsigned integers. One can converts into a real double precision value with the IDL routine `v_scet(s0, s1, s2)`.
- `table`: 3-columns array containing, for each pixel :
  - The wavelength, in micron
  - The spectral full width at half maximum (FWHM), in micron
  - The 1-σ error on the radiance, in W/m^2/sr/micron

A **dark qube** contains only the spectra acquired with the shutter closed. A dark qube can be raw or calibrated.

A **calibration qube** is the result if the internal calibration process (see § 8.2). Relevant fields are those of raw qube. Dimension of the field 'qube' is always `432 x 256 x 7`, that is to say 7 2d images of 432 x 256 pixels.

This is a summary to help the reader to better understand the calibration process. For further details, see RD3.

7.3 Acquisitions sequences

7.3.1 Observations and darks patterns

One telecommand (TC) sent to virtis-H leads to one session of acquisitions. Observations and dark acquisitions are interleaved, following this pattern:

- 1 dark
- `h_dark_rate x 1` observations
- 1 dark
- ...

Where `h_dark_rate` is specified at the TC stage. We describe here the usual observation mode. For further details, see RD2.

7.3.2 Acquisition_id

Each acquisition has its own acquisition_id during a session. The first acquisition (which is always a dark) has the number 1. The following `h_dark_rate x 1` observations have the numbers `2` to `(h_dark_rate + 1)`.

In the housekeepings of the observations, the reported acquisition_id is the one of the last spectrum of the current 64-spectra packet. Then all the 64 spectra of the same frame have the acquisition_id of the last one.

Concerning the darks, they are not sent within packets of 64, but sent each time they are performed. Thus each dark has its own acquisition_id reported in the housekeepings.

7.4 Corrupted darks

7.4.1 Description of the problem

When we received the first nominal spectrum from Virtis-H, we noticed a serious problem: some dark acquisitions contained signal from Venus. After investigations, it was a synchronization problem of the shutter (see RD4.).

Figure 1 shows an example. Regularly, dark acquisitions were performed with the shutter partially opened. Here, the amount of time with the shutter opened is growing with time.

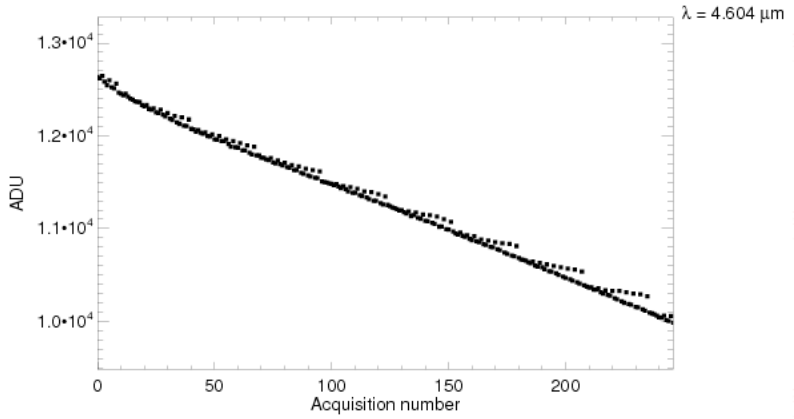


Figure 1: Example of dark corruption

7.4.2 Workaround

We have no mean to remove the Venus signal from the dark. The only solution is to identify the corrupted acquisitions and to ignore them during the data processing. The identifying process is partially automatic, and some corrupted darks may be still present.

For each dark qube, a table has been created (with the filemane VSxxxx_yy_BAD.TAB), containing the list of the corrupted darks.

The onboard software has been modified and uploaded on Venus Express on the 13th of June 2007. Data produced after this date are clean.

8 Virtis-H pixel map

8.1 Introduction

VIRTIS-H is a cross-dispersion spectrometer using 8 orders of a diffraction grating. Each order covers `432 x 5` pixels, 432 being the number of pixels in the spectrum, 5 being the number of pixels covered by the image of the slit per spectral element.

To locate the illuminated pixels on the detector a pixel map is used. This pixel map is the `8 x 3`-coefficient table of the 2-degree polynomial function per order that identify the pixels that are illuminated on the detector.

In nominal mode, only the `8 x 432 x 5` pixels identified by the pixel map are read.

This pixel map has to be checked and redefined from time to time.

8.2 Creating the pixel map from the internal calibration

A user interface named PixelMapCalculator can be used to generate the pixel map coefficients.

To create the pixel map, the program needs a calibration session This kind of session is a set of 7 acquisitions, all made with the cover closed:

- 1 shutter closed, calibration lamps switched off
- 2 for the slit spectral calibrations (1 shutter closed, 1 shutter opened)
- 2 telescope spectral calibrations (1 shutter closed, 1 shutter opened)
- 2 radiometric calibrations (1 shutter closed, 1 shutter opened)

In this mode, the full frame is read and transferred, in order to identify the pixels lighted through the spectrometer.

For each order, the program computes a set of coefficient to best fit a 2-degree polynomial function.

The process has to be done for each order.

8.3 PixelMapCalculator program

This program is written in Java (the user interface part) and IDL (the computation part).

8.3.1 Load a calibration session

The program can load one or more internal calibration sessions. For each loaded session, one can add or subtract one or more acquisitions together into a stack. Usually, the stack is composed of:

- + 1 telescope spectral calibration, shutter opened
- - 1 telescope spectral calibration, shutter closed
- + 1 radiometric calibration, shutter opened
- - 1 radiometric calibration, shutter closed

The stack is then displayed, and the user can adjust the contrast.

8.3.2 Compute the best fit for one order

For a chosen order, the user clicks on 4 points illuminated on the image along the spectrum. Using those 4 points, a set of 3 coefficients is calculated to fit a 2-degree polynomial function. The IDL pre-programmed function `svdfit()` is used for this purpose. The computation is made using double precision real values.

8.3.3 Optimize the fit

For each of the 432 pixels in column addressed by this polynomial function, a gauss fit (IDL `gaussfit()` function) is used in rows to find the maximum flux position.

At the end of this process, a set of 432 positions is identified for the given order. A new best fit (IDL `poly_fit()` function) by a 2-degree polynomial function is calculated using the set of 432 row positions. The 3 coefficients calculated are then stored in the pixel map file.

This optimization step is optional.

9 Virtis-H transfer function

9.1 Introduction

To recover the calibrated spectra from raw data, a transfer function is used. The transfer function has been calculated from measurements done on ground calibration at channel level and at instrument level. This function coded in IDL processes the raw data taking into account the detector response and the optical response. Those two responses are treated separately.

The optical response is essentially based on the optical transmission. This response is spectral and temperature dependent. It is linear in flux. It has been measured at instrument level at IAS/Orsay using calibrated blackbodies.

The detector response depends on the incident flux and the integration time. It is non linear at low flux and high flux. It is spectral, pixel and temperature dependent. Considering the functioning of VIRTIS-H, a cross-dispersion spectrometer lightning only 8 orders on the detector (each order being spread on `432 x 5` pixels), the calibration of the detector had to be done at sub-system level in order to calibrate both lighted and unlighted pixels. If the spectrum is shifted due to vibrations or thermal constraints, the newly lighted pixels calibration is used. The lighted pixels in VIRTIS-H are identified in the so-called pixel map.

9.2 The detector response

9.2.1 Method

The objective of this calibration is to recover the photon flux lightning the detector when the detector records ADU at a given integration time.

Two zones on the detector, due to two different filters, have different spectral responses. The two zones are named zone 1 and zone 2.

As the linearity of the detector depends on the integration time and the incident flux, two kinds of measurement have been done for each pixel.

- Number of ADU versus the integration time at constant incident flux. This constant incident flux is noted **`\ref`**.
- Number of ADU per second versus the incident flux with a constant charge (number of ADU). The integration time is adjusted for each measurement in order to keep the same ADU level. This ADU level is noted **`ADUref`**.

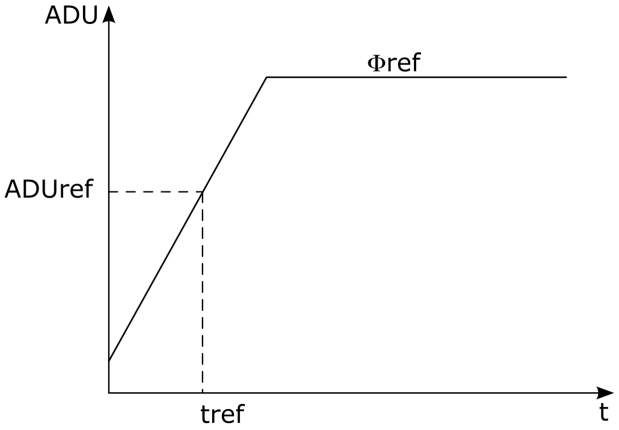


Figure 2: ADU vs. integration time @ constant flux

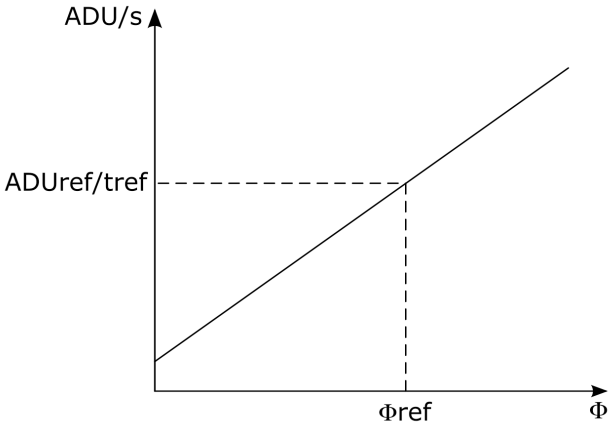


Figure 3: Number of ADU/s vs. flux @ constant charge

Note: on the curve **ADU=f(t)** (Figure 2), the **ADUref** level is chosen arbitrarily at half of the dynamic.

ADUref gives tref (as shown on Figure 2) that corresponds to **ϕref** on the curve **ADU/s=g(ϕ)** (Figure 3). This arbitrary level yields to the relative response pixel to pixel.

Let's consider now a measurement that gives for each pixel the number of ADU and an integration time (**ADUi, ti**). One has to find back the flux **ϕi**.

On the curve **ADU=f(t)**, ADUi corresponds to an integration time **ti_ref** for **ϕref** but it corresponds also to an integration time on the curve **ADU=f2(t)** for **ϕi** constant (**ϕi** is the flux to be found).

Using an affine transformation from the curve **ADU=f(t)** to the curve **ADU=f2(t)**, it is possible to find back the integration time **t2** that should have been used to reach **ADUref** with a flux **ϕi** (see Figure 4).

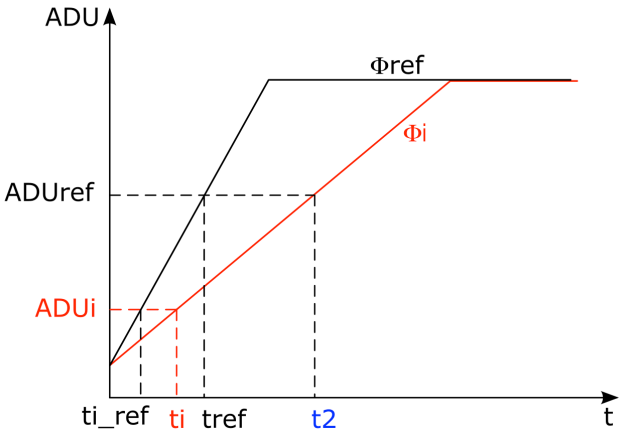


Figure 4: Recovering the integration time to get ADUref when lightning with ϕi

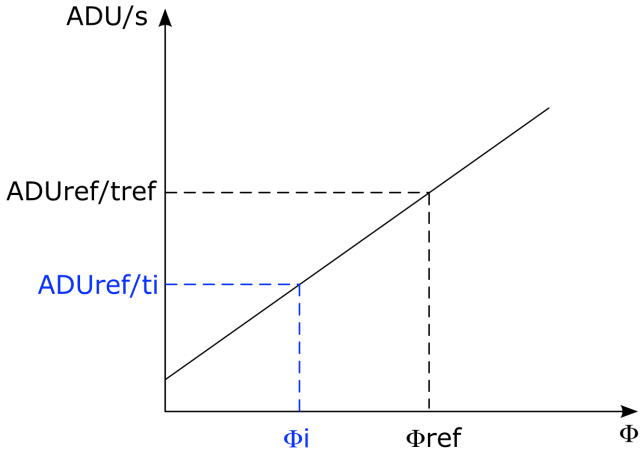


Figure 5: Recovering the incident flux ϕi

From **t2**, we find back **ϕi** on the curve **ADU/s=g(ϕ)** (see Figure 5).

For each pixel the functions **ADU=f(t)** are first considered linear. Deviation from the linear fit is estimated by Chebyshev polynomial coefficients, identical for all pixels but different for zone 1 and 2.

For each pixels the functions **ADU/s=g(ϕ)** are considered linear.

The procedure is the following:

1) For each pixel, the coefficient **at**, **bt** of the function **at*t+bt** are calculated.
2) For each zone the Chebyshev polynomial coefficients that fit the deviation to the linearity of the mean of the functions **at*t+bt** are calculated.
3) For each pixel, the **tref** corresponding to **ADUref** is found by inversion of the functions **at*tref+bt+Chebyshev-ADUref=0**.
4) For each pixel, the coefficients **af**, **bf** of the functions **ADU/s=af*ϕi+bf** are calculated.
5) For each pixel, **ti_ref** is found by inversion of the function **at*ti_ref+bt+Chebyshev-ADUi=0**
6) For each pixel **t2** is found using the transformation **t2/tref=ti/ti_ref**
7) For each pixel, **ϕi** is found by inversion of the function **af*ϕi+bf-ADUref/t2=0**

9.2.2 IDL programs

9.2.2.1 ADU_vs_ti()

It reads, via the routine `virhyacpds()`, the images acquired on the bench Yacadire and computes for each pixel, the curve shown in Figure 2.

These PDS files contain the detector response for several integration times considering a constant incident flux **ϕref**. For each pixel, a linear fit is performed with the IDL pre-programmed function `poly_fit()`. Coefficients **af** and **bf** of the formula: **ADU/s=af*ϕref+bf** are computed.

The following step should be performed for each pixel. In order to be less time consuming, the routine considers that all the pixels of the same parity and the same detector zone behave the same way. The parity refers to the column number parity. As the detector is read by column (one column from top to bottom, the next from bottom to top), odd and even columns do not behave in the same way. They have to be treated separately. In summary, 4 relevant pixels are chosen (1 odd and 1 even into zone 1 and zone 2).

For each relevant pixel, the deviation from the fit is modeled with Chebyshev polynomial coefficients. To avoid artifacts, only the 10 first coefficients are kept. Greater order coefficients are set to 0.

The matrices of **af** and **bf** coefficients are respectively written in the files `a_ADU_vs_t.fits` and `b_ADU_vs_t.fits`.

The sets of Chebyshev coefficients are written into `ChebZ1_pair.fits`, `ChebZ2_pair.fits`, `ChebZ1_impair.fits` and `ChebZ2_impair.fits`.

The integration times for which the measures have been done are written into `tiZ1.fits` and `tiZ2.fits`.

9.2.2.2 ADU_vs_flux()

It reads, via the routine `virhyacpds()`, the images acquired by the bench Yacadire, and computes for each pixel, the curve shown in Figure 3.

These PDS files are the result of the following experiment. For several lamps and several incident fluxes, the goal was to adjust the integration time in order to obtain the same charge in ADU **ADUref**. For a given lamp, the plot representing the charge in ADU divided by the integration time, versus the incident flux, is linear.

For each lamp and each pixel, a linear fit is performed with the IDL pre-programmed function `poly_fit()`. Coefficients **at** and **bt** of the formula: **ADU/s=at*ϕ+bt** are computed.

Plotting these curves for different lamps in the same plot leads to parallels. The slope of these parallels is the parameter we need. The routine normalizes all these plots by subtracting **bt** from each **ADU/s**. It gathers then all the normalized plots, and makes a global linear fit (`poly_fit()`) to compute the final **at** and **bt** coefficients for each pixel.

The resulting matrices are written into `a_ADU_vs_flux.fits` and `b_ADU_vs_flux.fits`.

9.2.2.3 InitRepDetecteur()

`InitRepDetecteur()` performs steps 1), 2), and 3) of § 9.2.1. It reads the .fits files created by `ADU_vs_flux()` and `ADU_vs_ti()`, and computes the global variables **`ADUref`** and **`tref`**, whose values only depend on the detector type (KIT or FM2). This routine is then executed only once at the beginning, or when the detector changes.

9.2.2.4 RepDetecteur()

Before calling RepDetecteur(), InitRepDetecteur() has to be called.

`RepDetecteur()` performs steps 4) to 7). It computes, for each pixel of each 2d image, the flux ϕ_i, in arbitrary units (ADU), seen by the detector.

9.3 The optical response

The overall optical response (including the detector) has been measurement at instrument level at IAS/Orsay and at LESIA/Meudon.

9.3.1 Calibration at LESIA/Meudon

Fine spectral calibration at instrument level.

Relative radiometric calibration at instrument level (no ME, detector alone).

9.3.2 Calibration in IAS/Orsay

Absolute radiometric calibration at instrument level

Rough spectral calibration at instrument level

Both calibrations at IAS/Orsay and LESIA/Meudon are used in the transfer function of the VIRTIS-H.

9.4 Building the transfer function

9.4.1 Synopsis

The transfer function is built using the optical and the detector response (see § 9.2 and 9.3). The process is shown Figure 6

The RepDetecteur function uses the detector calibration done at component level and transforms the blackbody calibration in an arbitrary unit (NIV/s). Using theoretical blackbody profile and the setup transmission, the transfer function is calculated.

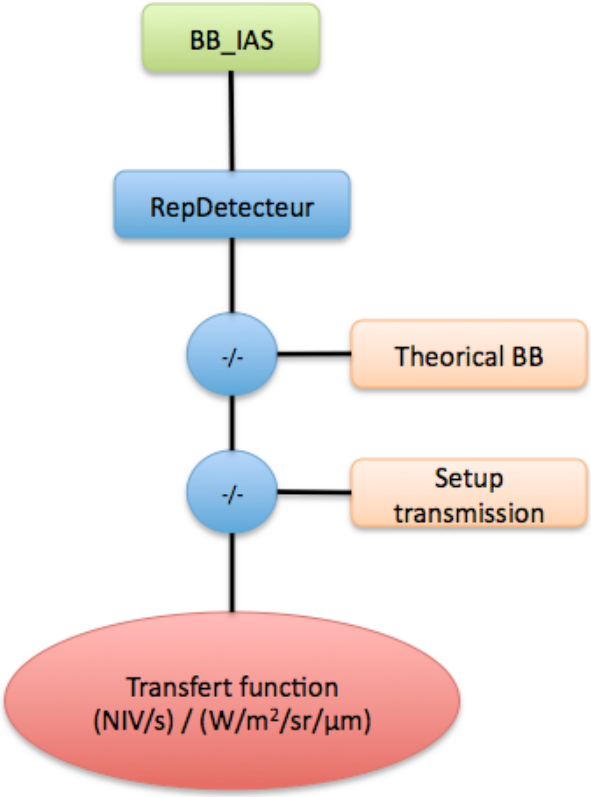


Figure 6: Transfer function synopsis

9.4.2 IDL programs

9.4.2.1 TransfertRos_Cal_IAS()

This function reads the acquisitions performed while observing the IAS black body at several temperatures (150K and 300K). Operations on orders 0 and 1 will use the acquisitions made at 150K, and those on orders 2 to 7 will use the acquisitions made at 300K. The FM2 detector was used.

Each acquisition is converted into a 438x270 2d image (`convert_438()` routine). The 8 orders spectra are extracted and transformed into spectra. The relative response of the detector is computed with `RepDetecteur()`. This part is handled by the `traceordrequb()` IDL routine.

Mean spectra are computed for each blackbody temperature. They are divided by the optical bench transmission, and by the black body simulated flux (`BlackBody()` routine). The black body simulation is done using the following formula :

$$flux(\lambda) = \frac{2hc^2/\lambda^5}{exp(\frac{ch}{\lambda kT}) - 1)}$$

Fluxes are first computed for 1µm to 6µm, with a resolution of 2.5e-3µm. Then, for each spectrum, the black body flux is computed for each wavelength with the IDL routine `interpol()`.

The result of these computations is the first order transfer function, and is written into the file `TransfertIAS.txt`. This transfer function has a good global level, but is polluted by many dying pixels.

9.4.2.2 TransfertRos_Lissage()

This routine smoothes the transfer function obtained by `TransfertRos_Cal_IAS()`. It uses the IDL routine `median()` with a width of 10 pixels. That means that each pixel of each spectrum is replaced by the median value computed from the 10 closest pixels. The resulting transfer function is written into `Transfert.txt`.

9.4.2.3 TransfertRos_RecouvOrdre()

This routine scales the 8 orders obtained with `TransfertRos_Lissage()`, in order to have a good agreement between overlapping wavelengths.

For this purpose, it reads acquisitions, the blackbody at 600K, and with different integration times (1.54ms and 10.24ms). The smallest integration time will be used for the orders 0, 1, and 2. The longest one will be used for the orders 3, 4, 5, 6, and 7.

The relative response of the detector is computed for each acquisition with `RepDetecteur()`. All the acquisitions are summed and darks are subtracted. The result is then divided by the number of acquisitions performed. The 8 orders spectra are extracted.

For every spectral resolution $\Delta\lambda$ shared by two consecutive orders, the routine computes the mean spectra ratio:

ratio(i, i-1) = mean(Δλ, order i-1) / mean(Δλ, order i)

Each order (from 7 to 1) is then multiplied by the ratio(i, i-1). Finally, all the orders are scaled by the same factor in order to keep the absolute level:

factor = original_spectrum(order 3, pixel 256) / new_spectrum(order 3, pixel 256)

This is the last step of the transfer function computation, and the result is written into `TransfertRecouvOrdre.txt`.

Figure 7 shows the transfer function plotted for each order.

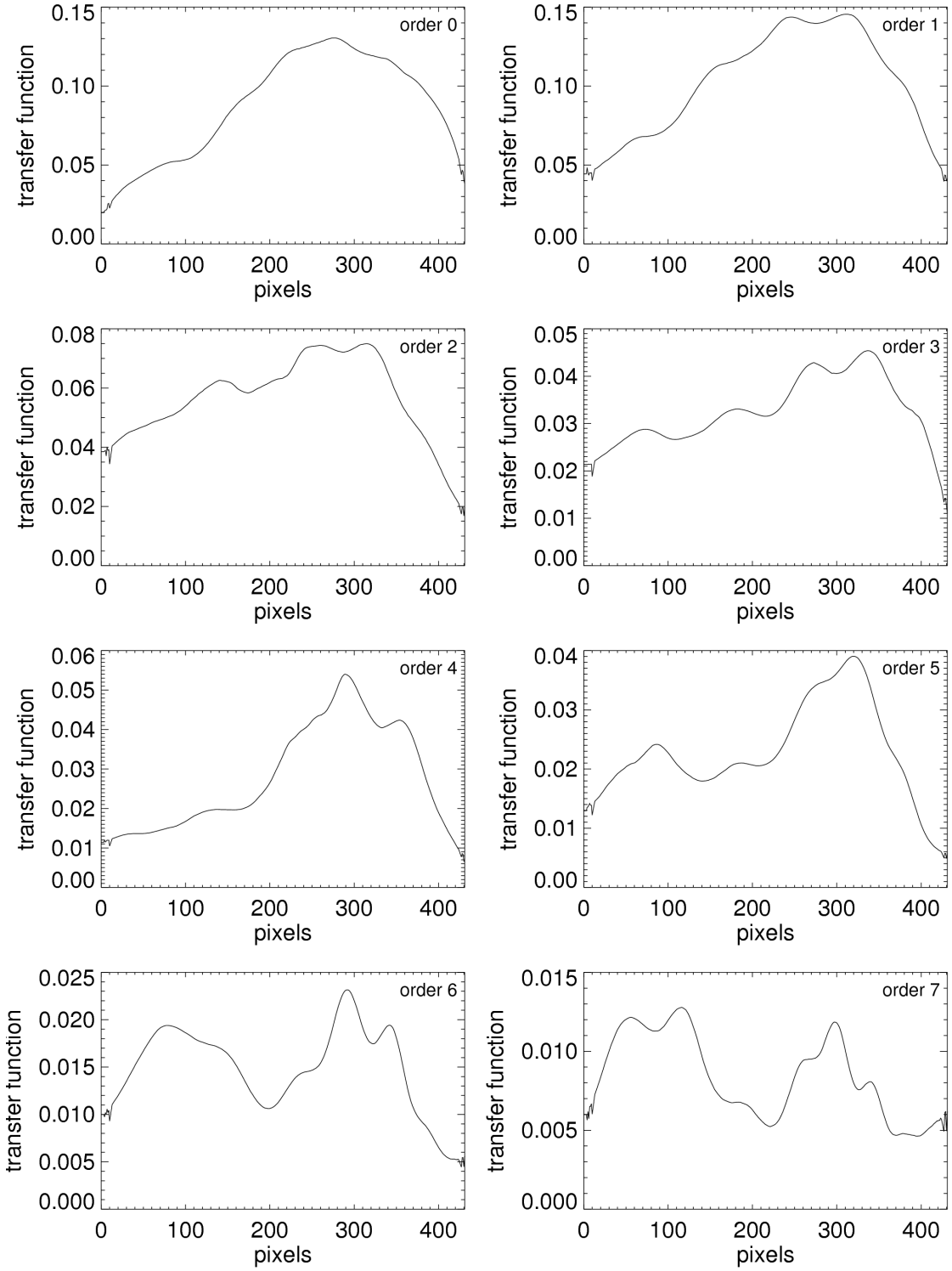


Figure 7: Plot of the transfer function for each order. Units are (Niv_KIT / s) / (W_IAS / sr / m^2 / µm).

9.5 Using the transfer function

9.5.1 Synopsis

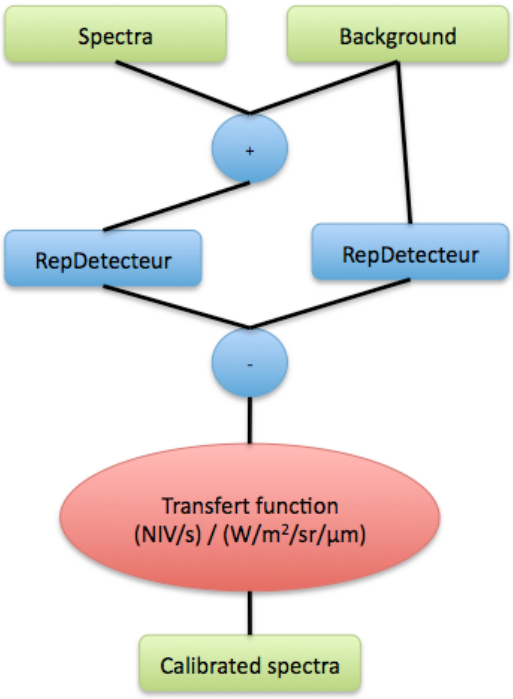


Figure 8: Raw data calibration

VIRTIS-H raw data are stored in qube files formatted in the PDS format. The latest dark acquired by Virtis-H is subtracted to the spectra. Background spectra are also stored in separated PDS qubes.

As the detector response is flux dependant, the first step of the calibration is to recover an optimized dark spectrum. Then both the background and the raw spectrum are processed in the `RepDetecteur()` function, to be corrected from the detector response defaults and to provide the contribution of their flux seen by the detector.

The raw spectrum and background are then subtracted again before being divided by the transfer function.

The calibrated spectrum is given in radiance unit.

The VIRTIS-H internal radiometric calibration is used to monitor and correct the drift of the optical transmission during time.

9.5.2 IDL programs

9.5.2.1 MakeCalibratedQube()

The program that calibrates qube is MakeCalibratedCube(). It reads a qube and its associated dark qube. The routine `fixhpds()` performs the calibration and `v_convlabel()` writes the file with the appropriate label. We will focus here on `fixhpds()`.

9.5.2.2 InitRepDetecteur()

Some of the computations needed by the calibration process are identical for all the acquisitions of a session, as they only depend on the detector model (KIT or FM2). The routine `InitRepDetecteur()` is doing that. The steps 1), 2), and 3) of § 9.2 are performed.

9.5.2.3 Fixhpds() : find the subtracted dark and add it

On board, for each observation (shutter opened) the last dark acquired has been subtracted (even corrupted ones). The routine computes for each acquisition the acquisition_id of its corresponding dark. As corrupted darks were subtrated onboard, they are not ignored here.

Dark acquisition_id	1									10									19	
Observation acquisition_id		2	3	4	5	6	7	8	9		11	12	13	14	15	16	17	18		20
Dark number	0									1										2
Obs. number		0	1	2	3	4	5	6	7		9	10	11	12	13	14	15	16		17
Corresponding dark number		0	0	0	0	0	0	0	0		1	1	1	1	1	1	1	1		2

Table 2: Example of the correspondence between acquisitions and darks for h_dark_rate=8.

The dark that has been subtracted onboard is added to the acquisition, in order to have the real response of the detector.

9.5.2.4 NomToBkp()

This function transforms a collection of spectra (3456 elements array) into a collection of 2d images (432x256).

Each pixel of the input spectrum is the column x of an order o. The pixel value is the mean of the 5 pixels around the position given by the pixel map for the column x of the order o. This operation was made on board by Virtis-H.

The routine spreads the pixel value over the 5 relevant pixels, using the slit transfer function.

The slit transfer function is stored in the file `FonctionFente.txt`. This file is a 2-columns array containing the transmitted flux (normalized to 1) along the slit. The first column is in milliradians. The routine transforms it into pixels, the resolution beeing 0.576mrad/pixel.

For each of the 5 pixels, the routine computes the percentage of the flux received with the following formula:

$$f(i) = \frac{\int_{i-0.5}^{i+0.5} slit(x)dx}{\int_{slit} slit(x)dx}$$

The function `slit()` is the slit transfer function oversampled to reach a resolution of 2 millipixel, with the IDL pre-programmed function `interpol()`.

Using the pixel map, the routine computes the position `(x, y)` of the corresponding pixel, and fills the range `[x,y-2:y+2]` with :

`pixel(x, y-i) = 5 * spectrum(x, o) * f(i)`

The resulting 432x256 image is returned.

9.5.2.5 RepDetecteur()

For each acquisition (observations and darks), the routine `RepDetecteur()` computes the detector response. Steps 4), 5), 6), and 7) of § 9.2 are performed.

9.5.2.6 Fixhpds() : Apply the transfer function

The transfer function derived from the optical response analysis (written in the file `TransfertRecouvOrdre.txt`) is applied. Each pixel value is divided by this 8 x 432 array.

9.5.2.7 BkpToNom()

This function transforms back a collection of 2d images into a collection of spectra.

It reshapes the input image into a 438x270 one, if needed, using the Xwin and Ywin parameters.

Then it computes for each pixel of each order the mean of the 5 pixels around the relevant position given by the pixel map.

The pixel range `[xwin:xwin+431]` is extracted for each order, and all the orders are concatenated, beginning with the order 0. The resulting 3456 elements array is returned.

9.5.2.8 Fixhpds() : compute the observation timestamp

For a given observation acquisition, the routine `fixhpds()` computes the associated timestamp (in SCET). The SCET in the relevant housekeepings is not the good one, but the one associated to the last spectrum of the current 64-spectra packet.

As virtis-H observation mode is in free run, the delay between 2 adjacent acquisitions (dark or observation) is always the same. We conclude that the plot **`acquistion_id`** versus SCET is linear.

The SCET of the acquisition numbered i is computed as follow:

`SCET = acquisition_id * (frame_scet – dark_scet) /`
`(frame_acquisition_id – dark_acquisition_id) + dark_scet`

Where:

- **`acquisition_id`** is the acquisition_id of the current acquisition ;
- **`frame_scet`** is the SCET of the last spectrum of the current 64-spectra packet ;
- **`dark_scet`** is the SCET of the last dark ;
- **`frame_acquisition_id`** is the **`acquisition_id`** of the last spectrum of the current 64-spectra packet ;
- **`dark_acquisition_id`** is the **`acquisition_id`** of the last dark ;

9.5.2.9 Fixhpds() : Interpolate the darks at every observation timestamp

In order to subtract the best dark from the observations, the dark acquisitions have to be interpolated at every timestamp computed above. A linear interpolation is performed using, for each acquisition, the preceding and the following dark acquisitions (ignoring the corrupted darks). Channel i of a spectrum is computed as follow:

```
s(i) = sp(i) – scet * (sn(i) – sp(i)) / (scet(sn) – scet(sp))
```

Where:

- **`sp(i)`** is the preceding dark spectrum value for channel i
- **`sn(i)`** is the next dark spectrum value for channel i
- **`scet(sp)`** is the scet of the preceding dark spectrum
- **`scet(sn)`** is the scet of the next dark spectrum
- **`scet`** is the timestamp where the dark has to be interpolated

Once all the darks are computed, they are subtracted from the acquisitions. The resulting spectra are then ready to be written into the calibrated qube.

The calibrated darks (obtained at § 9.5.2.6, before interpolation) are written separately into the calibrated dark qube. Both writing operations are performed by the `v_convlabel()` routine.

9.5.2.10 Fixhpds() : Compute the radiometric error

To compute the radiometric error, we use the dark acquisitions (ignoring the corrupted ones).

We first evaluate the noise level $\sigma_{ADU}(i)$ in ADU for each pixel *i* of the dark spectra with the following formula :

$$\sigma_{ADU}(i) = \sqrt{5 \times ADU(i) \times \eta \times \frac{32767}{2 \cdot 10^6}}$$

where :

- ADU(i) is the dark spectrum value for the pixel *i;*
- η is the quantic efficiency, and is equal to 0.6
- the ratio $32767/2.10^6$ is the conversion factor between the number of photons and the detector reponse in ADU.

- The factor 5 is due to the sum over 5 pixels (illuminated through the slit) made on board.

The noise is then computed for each observation timestamp, using a linear interpolation.

Finally, for each observation, the routine `RepDetecteur()` is called again to compute the radiance when the noise is added to the spectrum signal. The following relation is used:

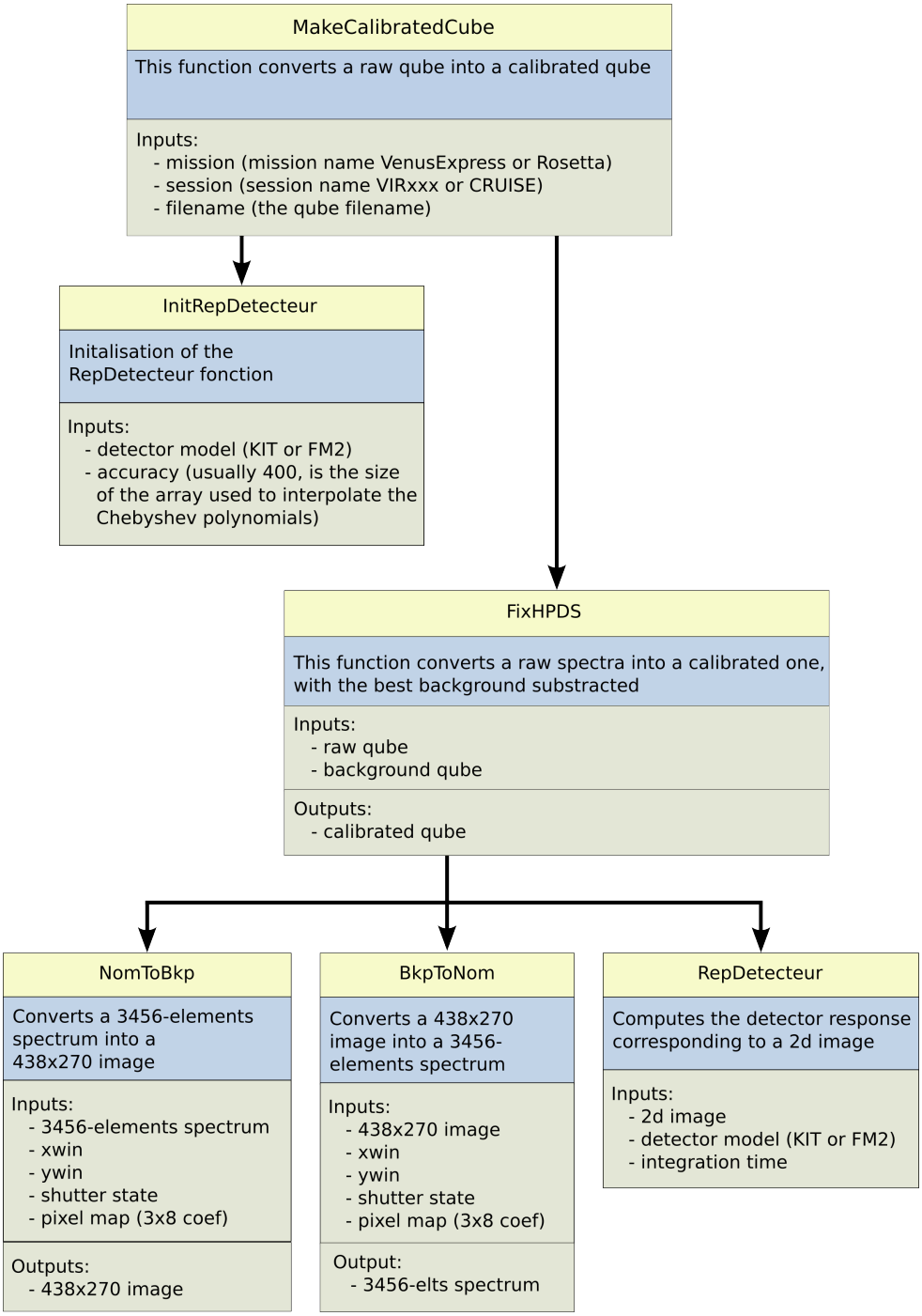
$$S(i) + \sigma(i) = RepDetecteur(ADU(i) + \sigma_{ADU}(i))$$

where:

- S(i) is the calibrated spectrum value for pixel i ($S(i) = RepDetecteur(ADU(i))$)
- ADU(i) is the raw spectrum value for the pixel i
- $\sigma_{ADU}(i)$ is the noise level for pixel i
- $\sigma(i)$ is the 1-σ error

The 1-σ error for each pixel is stored in the calibrated files.

9.5.3 IDL programs flow chart



10 Spectral calibration

10.1 Principle

The spectral calibration is determining the central wavelength of each pixel.

The internal spectral calibration is made by mean of a pre-calibrated Fabry-Perot with respect to the temperature. The Fabry-Perot emission gives between 20 and 10 lines depending of the grating order. This device allows finding back the spectral registration of the instrument. The absolute position of each known spectral lines on the detector is gauss fitted for each order and used to reconstruct the spectral registration of the instrument, which fits a 2-degree polynomial function.

Those polynomial functions are then used to give for each pixel the correspondence between the pixel number and the wavelength.

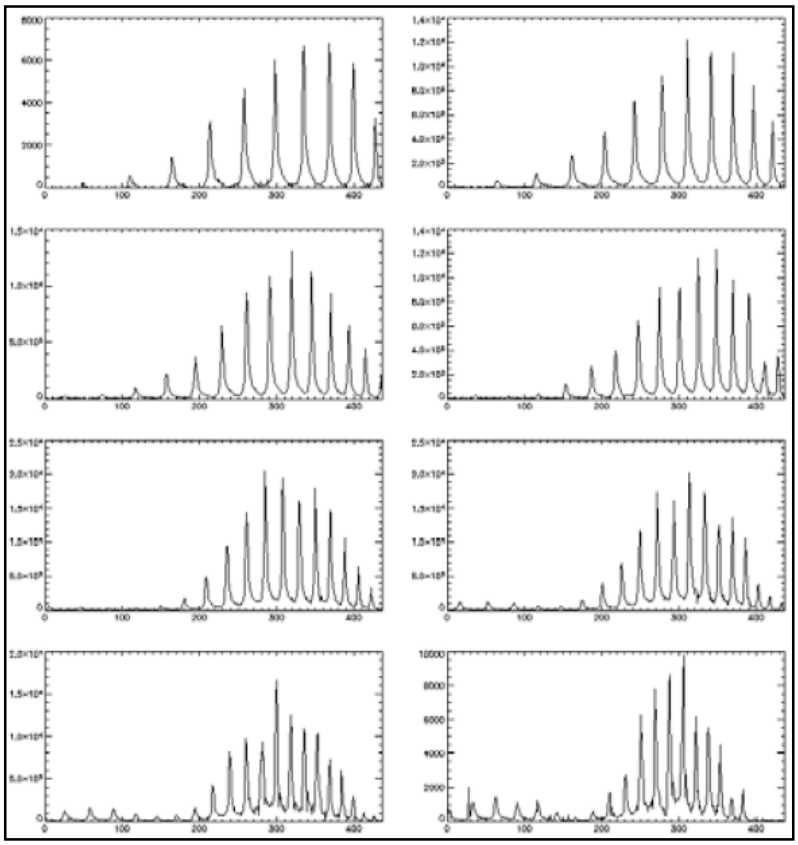


Figure 9: Emission of the Fabry-Perot spectral lines in each order

10.2 IDL program

The goal of this program is to calculate the PxlMapL. The PxlMapL is a table of 3 coefficients per order. Those coefficients are the coefficients of the 2-degree polynomial function giving the correspondence between the pixel number and the wavelengths.

The calculation is done within the `CreatePxlMapL_WithFP_Vex_Qub()` function.

For each order, the spectrum is scanned to find the FP spectral line having a level upper than `'Offset'` using a gauss fit function. The peak positions returned by the gauss fit function of each spectral line are stored in a table.

Using the FP temperature `'dT'` and the initial room temperature FP thickness `'e'`, a table of theoretical wavelengths emitted through the FP is generated.

For each order a table `'lambdamin'` of first expected wavelengths at the border of the detector is sent to the function. The maximum error tolerated is given by the difference between two consecutive FP spectral lines. If the table is false, the registration will be false by a factor that is the difference between two consecutive FP spectral lines. Nonetheless, the error will be seen immediately on a science spectrum. If so the `'lambdamin'` table has to be readjusted. This process is not dealt within the function and has to be done manually when needed.

Once the table of FP wavelengths and positions on the detector are created for each order, the 2-degree polynomial fit is used to generate the coefficients. Those coefficients are then stored in a file. The file name is `'PxlMapL'+'datePxlMap'` in the LESIA data center and `H_spectral_coef.TAB` in the PSA archive.

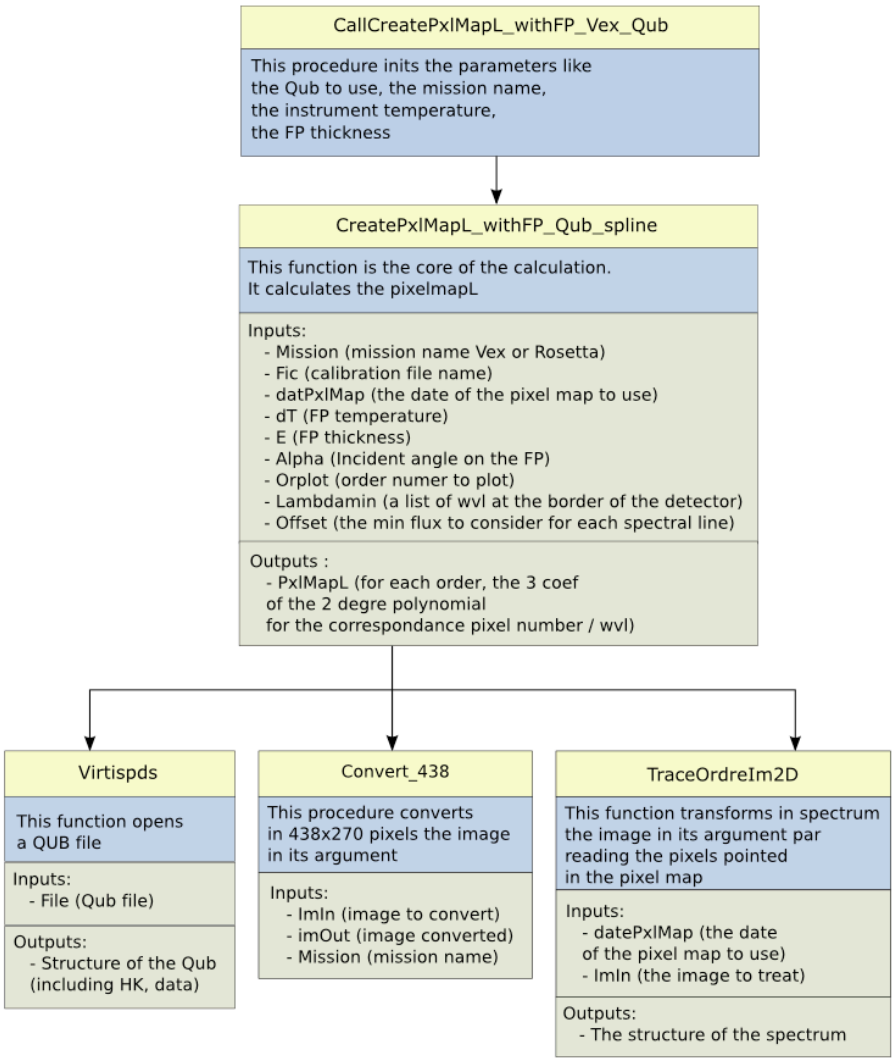


Figure 10: Flow chart of the spectral registration function

10.3 Accuracy of the spectral calibration process

The sources of errors to construct the registration table are:

- The accuracy of the gauss fit for each FP spectral line;
- The deviation from a 2 degree polynomial function for the registration;
- The initial room temperature thickness for the FP.

The addition of all the errors gives a maximum estimated global error for the registration of 1 pixel.

This errors can be decreased using science spectra having well known spectral lines and adjusting the parameters of the function.

11 Pixel spectral profiles

11.1 Introduction

The spectral registration given by the PixelMapL table does not take into account the spectral profile within the image of the slit. This profile is the spectral response of the instrument.

As VIRTIS-H is a high-resolution spectrometer, the spectral response varies significantly along the spectrum. No dedicate on-ground calibration were able to measure this spectral response. Nonetheless, as VIRTIS-H is hardly diffraction limited, this profile can be estimated by the diffraction theory using the optical parameter of the spectrometer.

11.2 Spectral profile calculation

The spectral response is calculated for each wavelength in each order using the diffraction theory.

It is the convolution of the diffraction profile by the width of the slit image

The formula used to calculate the spectral response are given above:

Max. intensity	$\beta(\lambda,k) = \arcsin\left(k \times n \times \lambda - \sin(\alpha)\right)$
Resolution per pixel	$\Delta\lambda(\lambda,k) = \frac{-\cos(\beta(\lambda,k))}{n \times k} \times \frac{pxl}{f}$
Energy profile due to diffraction by grating	$I(x,\lambda,k) = \mathrm{sinc}\left(\pi \times \frac{L \times \cos(\beta(\lambda,k))}{\lambda} \times \arctan\left(\frac{x}{f}\right)\right)^2$
Window function	$\Pi(x,a) = \begin{vmatrix} 1 & \text{if} \quad |x| < \frac{a}{2} \\ 0 & \text{otherwise} \end{vmatrix}$
Convolution of the energy a λ by a pixel	$Ic(x,\lambda,k) = \int_{-0.1\mathrm{mm}}^{0.1\mathrm{mm}} I(x-t,\lambda,k) \times \Pi(t,pxl)dt$

With: λ is the wavelength

k is the order number

α is the incident angle on the grating

n is the groove density of the grating

pxl is the pixel size

L is the width of the grating

f is the objective focal length

The following figures show the spectral response at two different wavelengths in two different orders. The blue line shows the geometrical slit width.

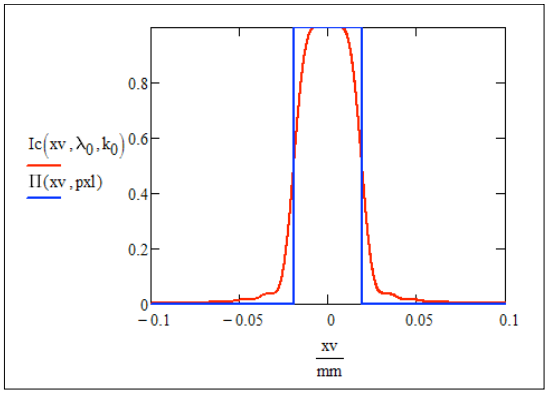


Figure 11:
Spectral response
@ 2µm, order 8

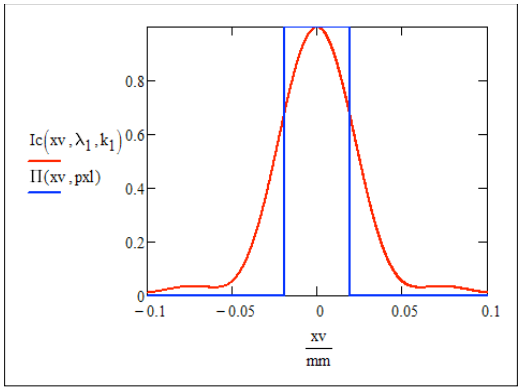


Figure 12:
Spectral response
@ 5µm, order 0

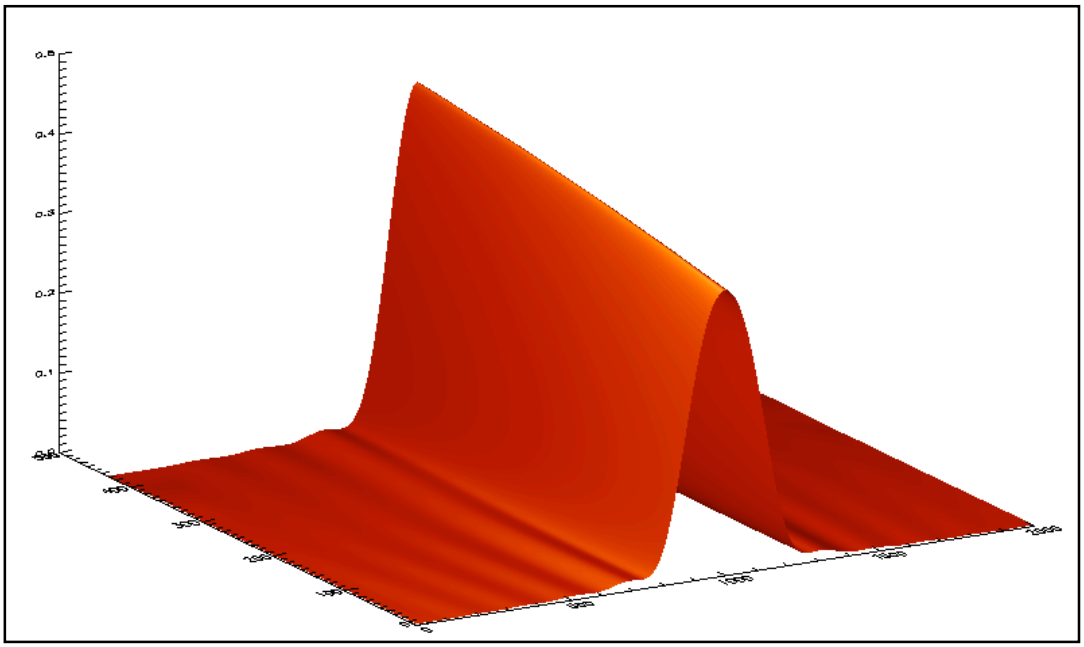


Figure 13: Figure 1. Plot of the spectral profile through order 0