Master thesis and internship[BR]- Master's thesis : Calibration of FLORIS instrument onboard ESA FLEX satellite. Study of the detector non-linearity key data parameters[BR]- Integration internship

Auteur : Tezel, Nursel
Promoteur(s) : Georges, Marc
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en aérospatiale, à finalité spécialisée en "aerospace engineering"
Année académique : 2020-2021
URI/URL : http://hdl.handle.net/2268.2/12964

Calibration of FLORIS instrument onboard ESA FLEX satellite. Study of the detector non-linearity key data parameters.

MASTER THESIS SUBMITTED FOR THE DEGREE OF
MASTER IN AEROSPACE ENGINEERING

TEZEL Nursel - s161443

UNIVERSITY OF LIÈGE
FACULTY OF APPLIED SCIENCES
ACADEMIC YEAR 2020-2021

_____ ABSTRACT

This work presents the study of KDP (Key Data Parameters) for the calibration of the instrument FLORIS (FLuORescence Imaging Spectrometer) onboard of the satellite FLEX (FLuorescence EXplorer). FLORIS is an hyperspectral imager that will be calibrated at CSL (Centre Spatial de Liège) which is a research center of the University of Liège. This project explains the calibration philosophy applied for this instrument, and focuses on the computation of KDP related to the non-linearity of the detector.

The first part presents the fluorescence mission, mission architecture and FLORIS overview and design.

The second part explains the calibration philosophy that will be applied to FLORIS at CSL and will also introduce the concept of KDP.

The last part focuses on the computation of the KDP related to the non-linearity of the detector. As there are no measurements available for FLORIS, the calibration will be done with the measurements of another instrument: 3MI. Four different methodologies are applied and compared for computing KDP related to the non-linearity.

ACKNOWLEDGEMENTS

I would like to thanks my supervisor Marc Georges for allowing me to undertake this internship and master thesis at CSL. His advice for the writing of this project were very useful.

I also wish to thanks Bogdan Vasilescu, Pascal Blain and Céline Michel, who work at CSL, for their valuable guidance. They helped me a lot to understand some complicated subjects.

I must also thanks my family and my friends for their profound belief in my abilities and in my work. They were good support.

Finally, many thanks to CSL for giving to me the access to every resources I needed to work on my thesis.

$$$$

CONTENTS

The Fluorescence Explorer (FLEX) mission takes part in ESA's Living Earth Programme. The goal of this mission is to show the fluorescence of the vegetation on global maps. As the fluorescence is inversely proportional to the photosynthetic activity, this last parameter will also be knwon. Additionally, scientists will be able to monitor the health of the plants. This is crucial for anthropomorphic reasons as the production of food is directly dependent on the vegetation. Moreover, the fluorescence of the vegetation will be a great indicator of the photosynthetic efficiency. This will allow to understand the role of plants in the global carbon cycle better. This mission is important as no other satellite has measured the photosynthetic activity from space before. The launch is planned for 2024 [1].

FLEX's only payload is an hyperspectral imager named FLORIS (FLuORescence Imaging Spectrometer). Leonardo, an Italian company that specialises in aerospace, defence and security, has designed the instrument and has charged CSL (Centre Spatial de Liège), a research center of the University of Liège, to calibrate it.

The calibration of an instrument is an important task in the conception of a mission. It will assure the user that the measurements are accurate. For this reason, the calibration of FLORIS is capital. This work will explain the calibration philosophy that will be applied for the calibration of FLORIS at CSL. KDPs (Key Data Parameters) will also be introduced as they are key concepts in calibration. In addition, this project will focus on the calibration of the non-linearity of the detector.

The first part of this work will present the mission in more details as well as an overview of FLORIS. The second part, will explain the calibration philosophy done at CSL for calibrating FLORIS. All the key concepts of calibration will be introduced. The third part will explain the algorithms necessary to compute the calibration of the non-linearity of the detector. Finally, these algorithms will be applied to an example, they will be compared and the best one will be chosen.

THE FLUORESCENCE EXPLORER MISSION

1.1   Mission architecture

FLEX will carry a single instrument: the FLuOrescence Imaging Spectrometer (FLORIS). It is a pushbroom hyperspectral imager that will cover a spectral range bewteen 500 nm and 780 nm. FLORIS is composed of two high spectral resolution spectrometers which cover a spectral range of 740 nm - 780 nm and 677 nm - 697 nm. This respectively corresponds to the two oxygen absorption bands O_{2A} and O_{2B}. Their spectral resolutions are respectively 0.3 and 0.7 nm. FLORIS is also composed of a low spectral resolution spectrometer with a range between 500 nm and 758 nm and has a spectral resolution of 2 to 3 nm [2].
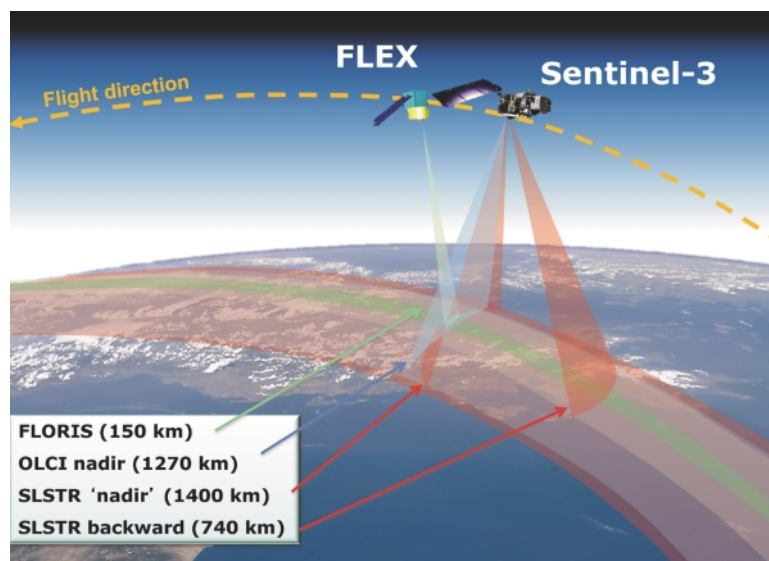
Figure 1.1: Formation flying of FLEX and Sentinel-3 [2]. With FLORIS' swath represented in green, OLCI's swath represented in blue and SLSTR's nadir and backward swath indicated by the red arrow.

As FLEX is only composed of an imager, additionnal data are needed to interpret the fluorescence signal. These data will be given by Sentinel-3. In fact, FLEX will orbit the Earth in tandem formation with Sentinel-3. They will fly at an altitude of 814km. Fig.1.1 shows the formation flying of FLEX and Sentinel-3 and the overlap of the swaths from FLORIS and Sentinel-3's Ocean and Land Colour Instrument (OLCI) and Sea and Land Surface Temperature Radiometer (SLSTR) instruments. The ground segment will be operated by the Earth Explorer ground segment infrastructure and it will be placed in orbit by the launcher VEGA/VESPA from Kourou in 2024. The mission lifetime is

expected to be 3.5 years and will have a propellant budget for up to 5 years.

The main characteristics of the mission's architecture are listed in Tab. 1.1.

Mission lifetime	3.5 years (nominal mission phase)
	ΔV and propellant budget computed up to 5 years
Mission phases	Launch at an early orbit phase and commissioning = 3 months
	Nominal mission phase = 3.5 years
	Possible mission extension = 1.5 years
	End of life phase = less than 3 months
Orbit	Sun-synchronous orbit: 14+7/27 ($H_{REF} = 800$ km),
	Local time of descending node = 10h00, repeat cycle = 27 days
Formation flying	Tandem with Sentinel-3

Table 1.1: Main characteristics of the mission architecture [2].

1.2 FLORIS overview and design

FLEX's only payload is FLORIS which is a pushbroom hyperspectral imager. The instrument optical layout is represented in Fig. 1.2 and its structure in Fig. 1.3. Moreover, Fig. 1.4 shows the mechanical layout.
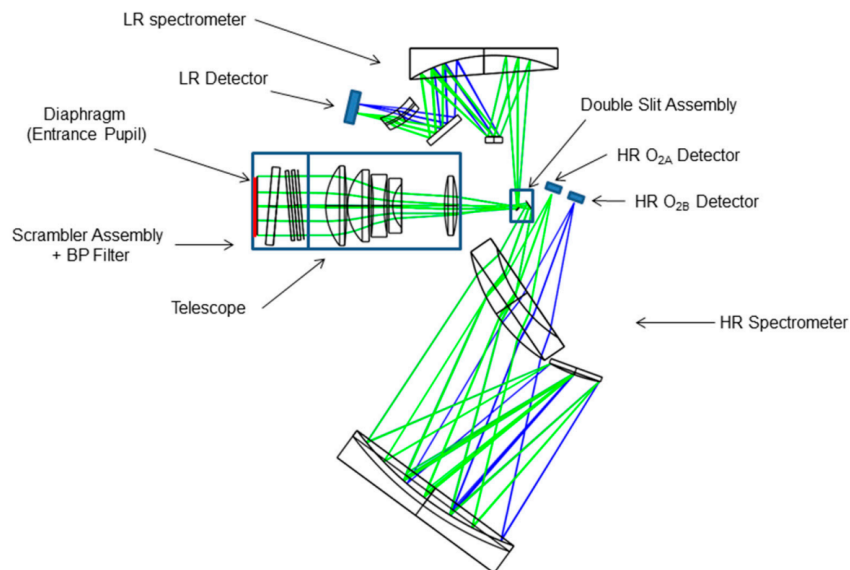
Figure 1.2: Optical layout of FLORIS [3].

Firstly, the radiation coming from the Earth enters the Calibration Unit (CU) which consists of a rotating carousel. This unit is responsible for the in-flight calibration. The rotating carousel can have three positions. The first one is nadir pointing, the second one points at a black target for dark calibration and the third one is sun oriented for sun radiometric calibration through the solar port. These positions are represented in Fig. 1.4 [2]. The CU is then followed by band pass filter that will filter the wanted spectral band.

Secondly, the telescope has a focal length of 234.5 mm and a f-number of 3.1. It has two slits corresponding to the high resolution (HR) and low resolution (LR) spectral bands. [3].

Earth

Calibration Unit ← Sun

Band Pass Filter + Scrambler

Telescope

Double Slit + folding Mirrors Assy

HR Spectrometer

LR Spectrometer

Detectors HR1 and HR2

Detector LR
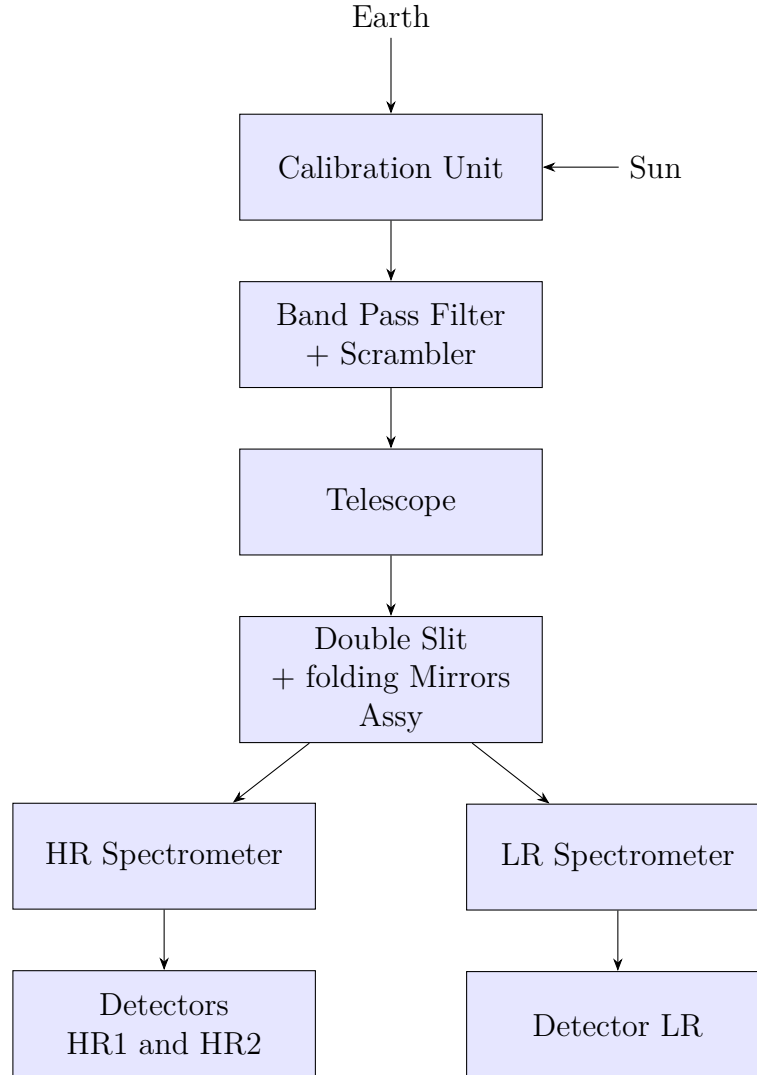
Figure 1.3: FLORIS instrument diagram [2].

The radiation enters then the HR and LR spectrometers. These are offner spectrometers that were modified to achieve the required spectral resolution. The high resolution spectrometer operates at a range between 677 nm and 780 nm. The corresponding optical layout is represented in the bottom right corner of Fig. 1.2. The pixel has a size of 28μm in the spectral direction and 42μm in the spatial direction. The low resolution spectrometer operates between 500 nm to 758 nm. The corresponding optical layout is represented in the top of Fig. 1.2. The pixel size and the optical design are the same as the HR spectrometer. Finally, the rays arrive to the CCD detectors. There are a total of three identical detectors, two for the HR spectrometer named HR1 and HR2 detectors and one for the LR spectrometer named LR detector. The format of the detectors are 1072×460 pixels^2 with a pixel size of 42 μm $\times$ 28 μm (spatial $\times$ spectral) [3].
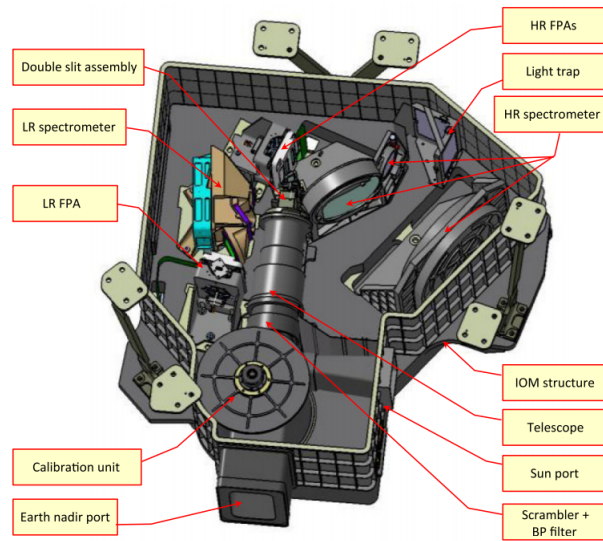
Figure 1.4: Mechanical layout of FLORIS [3].

PART 2

CALIBRATION

Calibration is essential for every satellite. In fact, instruments are subjected to response function of each component, optical aberrations, as well as noise sources or artefacts. In the case of FLEX, these errors are for example: the non-linearity of the detector, the presence of dark current, the presence of saturated or dead pixels, stray-light, and so on. All these contributions taken globally participate to lowering the global instrument performances. The goal of calibration is then to characterize the instrument response and artefacts and evaluate the influence of space environment (temperature mainly) on the instrument response. The performance of the instrument is then measured and has to satisfy the wanted instrument specifications. In this project, the company Leonardo charged Centre Spatial de Liège (CSL) to perform the calibration of FLORIS. My work is then to understand the process of calibration.

In this chapter, the overall calibration process of FLEX will be explained. All the essential steps are explained in section 2.5. For the sake of clarity some essential subjects will be first introduced. These are: key data parameters, data processing levels, instrument and calibration model.

2.1 Key data parameters

In calibration, Key Data Parameters (KDPs) are crucial. They consist of a set of parameters that characterize the instrument. In a calibration campaign these parameters are the ones to be calibrated. Once calibrated they are stored in the instrument characterisation and calibration data base (ICCDB). Each KDP is computed for different flight models and stored in a specific instance.

These KDPs are measured on-ground long before the launch of the satellite. However, some of them need to be updated in-flight. This is done thanks to the calibration unit discussed in section 1.1. KDP calibrated on-ground are measured thanks to Ground Support Equipments (GSEs). There are various type of GSE (electrical, mechanical, optical and thermal) depending on the type of KDP which needs to be calibrated. Some KDP can have a relatively simple form (e.g. a constant value) and can be directly written in the ICCDB. Some other KDP, more complex, require knowledge of other ones and are determined on the basis of the latter by computation following a theoretical description. This is called 'KDP determination'. The algorithms relative to these ones are written in the Algorithm Theoretical Baseline Document (ATBD).

For example, the dark current is a residual current that occurs when no pixel are illuminated due to thermal activity. The model for computing the dark current for every

pixel(i,j) in function of its KDPs is represented by Eq. 2.1:

$$DarkCurrent(i,j) = KDP_{Slope}(i,j) \cdot t_{int} + KDP_{Offset}(i,j), \qquad (2.1)$$

where, KDP_{Slope} is the KDP containing the slope of the dark current, KDP_{Offset} is the KDP containing the offset and t_{int} is the integration time.

Moreover, as it is impossible to calibrate the KDPs perfectly, specifications on performances of the instrument have been given. Thus, the GSEs need to satisfy these specifications as the quality of the KDPs directly depends on the accuracy of the GSEs to measure them. For this reason, requirements on GSEs will be applied to satisfy the wanted performances. [5]

2.2 Data Processing Levels

Another important subject is data processing levels. Throughout the whole processing chain, data will be processed at different levels. They range from level 0, which corresponds to raw data, to level 4, which corresponds to highly processed data. The main goal is to obtain data which will be easier to use in scientific work. The higher the level the easier the data will be used and the more complex it gets to compute them. For the mission FLEX the levels and their key parameters are summarized in Tab. 2.1.

Firstly, level 0 corresponds to data directly measured by the instrument. They are unprocessed and at full resolution. At this level, data is measured in digital number or DN. This corresponds to the numerical value measured by each pixel. This value refers to the intensity of the electromagnetic radiation received by the instrument [15]. Additional information is also included such as the orbital data, the time conversion and so on. Level 1b corresponds to data that have been radiometrically calibrated. At this level data are expressed in radiance ($W/(m^2.sr.nm)$). These data are also spectrally and geometrically characterized.

Secondly, level 2 is subdivided into 4 levels: a, b, c and d. Level 2-a corresponds to the orthorectification and collocation of FLEX and Sentinel-3's data. These data are then re-assembled into a grid. Level 2-b-c-d correspond to the derivation of geophysical parameters. The first geophysical parameter is the fluorescence emission of the two oxygen bands O_{2A} and O_{2B} (F_{687} and F_{760}) and peak values ($\lambda_{<685>}$, $F_{<685>}$ and $\lambda_{<740>}$, $F_{<740>}$). Finally, the total fluorescence emission will also be computed and is represented by: F_{TOT}. Level 2 data represent the central parameters of the mission.

Finally, the European Space Agency (ESA) is in charge of computing the levels up to level 2-d. Higher levels will be computed outside ESA's ground segment. Level 3 and Level 4 will give for example fluorescence quantum efficiency, vegetation stress and photosynthetic rate. These could be used in models such as the carbon cycle to have a better understanding of that cycle. Moreover, the calibration process aims at obtaining L1b data from L0 data. Thus, higher levels are not computed for the calibration. [3] [16]

Level	Description
L0	Raw data in DN
L1b	Calibrated data in Radiance
L2	O_{2A} and O_{2B} fluorescence emission
L2	Peak values ($\lambda_{<685>}$, $F_{<685>}$ and $\lambda_{<740>}$, $F_{<740>}$)
L2	Total fluorescence emission.
L3-L4	Fluorescence quantum efficiency
L3-L4	Vegetation stress
L3-L4	Photosynthetic rate

Table 2.1: Summary of the main data levels for FLEX.

2.3   Instrument Model

The instrument model (IMOD) is a numerical model which simulates the behavior of the instrument. It reproduces its acquisition chain and simulates the response function of each component, it adds the optical aberrations, noise sources or artefacts and aims at computing L0 data. These errors, such as simulating the non-linearity of the detector, adding dark current, adding stray-light, are reproduced thanks to their respective KDP. This instrument model is thus useful to predict how the satellite will behave with a specific input image. In calibration, this model is helpful to see how well the instrument is simulated. In fact, for a given input image, L0 data obtained numerically and L0 data obtained from the satellite will be compared. If they are close to each other this means that the KDPs have been accurately calibrated as they simulate the behavior of the instrument correctly. The diagram of the model is represented in Fig. 2.1. Note that each of these blue boxes implies one or several KDP.

The instrument model starts with an input image (in radiance) given by an image generator. The input image is spatially and spectrally oversampled. The first step is then to reduce the size of the image to reach the cardinality of the instrument. For this reason, spectral integration and ISRF application is first applied. This step aims at aggregating the spectrally oversampled image. In order to apply the ISRF, a related KDP_{ISRF} is used. The second step consists on subsampling the overally sampled matrix and applying the MTF by using the PSF which is in the ICCDB. After this step the image has the size of the active matrix or AM. As the image is in radiance it is important to do a radiometric calibration in order to convert it into DN. This is done in the third box. After the application of these three boxes comes the application of all the artefacts of the instrument. As a reminder, every artefacts implies one or several KDP previously calibrated. After applying all these errors, L0 data are obtained.

For example, in the instrument model the box in Fig. 2.1 named 'flat-field' adds the error concerning the flat-field effect to the image. The mathematical model of this box is represented by the following equation:

$$I_{FF}(i,j) = I(i,j) \times KDP_{FF}(i,j), \tag{2.2}$$

where, $KDP_{FF}(i,j)$ is the KDP related to the flat-field for pixel (i,j), $I(i,j)$ is the input image of the box which is flat-fielded, $I_{FF}(i,j)$ is the output image of the box which is not flat-fielded (the image is not corrected from the flat-field effect) [5].
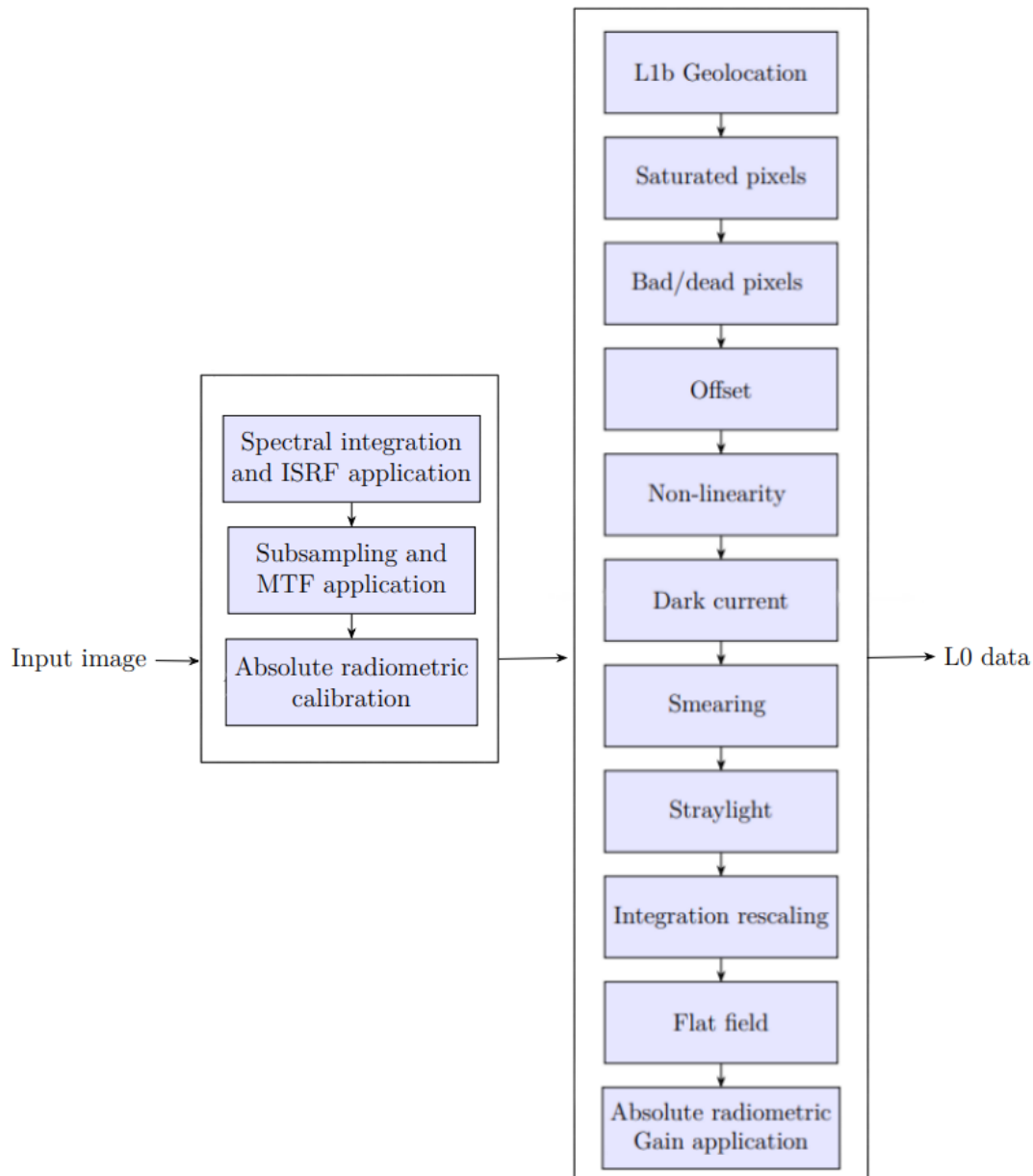
Figure 2.1: Instrument model of FLORIS.

2.4 Calibration Model

The calibration reverse model (CMOD) is a numerical model which calibrates L0 data and returns L1b data. This model is used to calibrate images coming from the satellite. The diagram of the CMOD is represented in Fig. 2.2. Every boxes implies one or several KDPs. L0 data are first corrected from all the response function of each component, noise sources or artefacts by using the previously calibrated KDP. After these corrections, the data is converted in radiance thanks to absolute radiometric calibration which is done in the last box.

For example, in the calibration model the box in Fig. 2.2 named 'flat-field' will correct

the image from the flat-field effect. This equation can be written as:

$$I(i,j) = \frac{I_{FF}(i,j)}{KDP_{FF}(i,j)},\tag{2.3}$$

where $I(i,j)$ is the output flat-fielded image and $I_{FF}(i,j)$ is the input image [5].
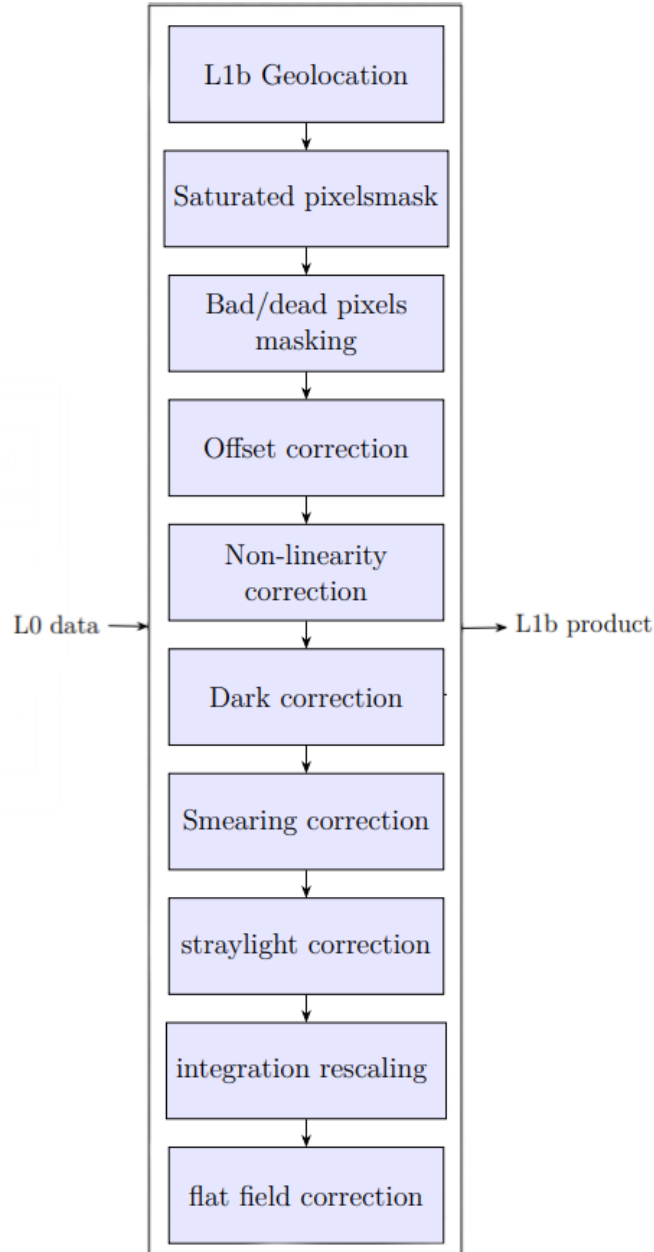
Figure 2.2: calibration model of FLORIS.

2.5   Calibration Philosophy

In this section, all the steps of the calibration philosophy will be explained. Fig. 2.3 shows the diagram representing the calibration philosophy [9]. The green boxes correspond to the work to be done by CSL and the blue boxes correspond to the work already done by

Leonardo but which must be verified by CSL. This diagram will be explained in three steps, each of them is represented in the figure.
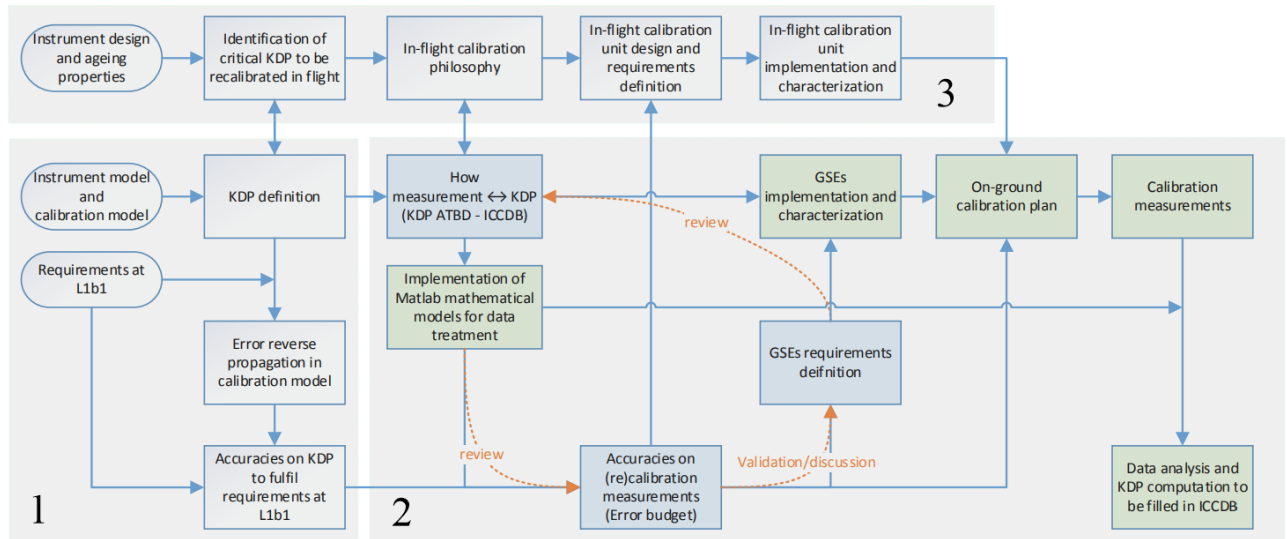
Figure 2.3: Calibration Philosophy [9].

Starting from the first part (indicated by the number 1). This part of the work is done by Leonardo. The aim is to find the accuracy needed on each KDP to fulfill the requirements on L1b data. The instrument and calibration model have already been implemented by Leonardo. The goal is then to define each KDP and to propagate errors for each one in the calibration model. If the resulting L1b data do not have the required accuracy, a smaller error is propagated in the model. After couple of iterations a required accuracy on each KDP is fixed. The accuracy on KDPs is chosen to fulfill the requirements on L1b data.

The second part (indicated by the number 2) of this figure consists on finding the requirements on GSEs. The first step is to define how the KDPs will be measured. This is done in the ATBD. Then, the mathematical models of each KDP need to be implemented. Thanks to the first part, the accuracy on the calibration measurements are known. By knowing this precision, the requirements on GSEs can be defined. In fact, the GSEs need to have strict requirements in order to measure KDP within the error budget. Finally, the requirements on GSEs are checked to see if there are feasible. When this phase is conclusive, the measures of the KDPs can be done. The GSEs are then implemented and characterized, and on-ground calibration can finally take place. Each KDP will be measured and will be filled in the ICCDB.

The third and final part (indicated by the number 3) concerns the in-flight calibration. It is important to define the needs of the KDP that will have to be calibrated in-flight beforehand. Once there are chosen, steps from before are applied. The calibration unit is then calibrated [9].

PART 3

KEY DATA PARAMETERS COMPUTATION RELATED TO THE NON-LINEARITY OF THE DETECTOR

This project will focus only on the computation of KDPs[1] related to the non-linearity of the detector. The goal of this chapter is to understand the non-linearity and to introduce the algorithms used to correct it. All the algorithms are hand coded in Matlab and can be found in the Annex. A.

Firstly, the physical principle behind the non-linearity will be introduced. Secondly, two different algorithms used to compute KDPs will be explained as well as the algorithm needed to correct the images from the non-linearity.

3.1 Physical Principle

When light reaches FLEX's detector, photons are converted in electrons that are stored in wells. The amount of photons converted depends on the integration time, or equivalently the time the detector has been illuminated. The photo-conversion is supposed to be linear. This means that the amount of electrons increases proportionally with increasing incoming photons. The linearity of the photo-conversion will be checked during on-ground calibration.

Then, the analog to digital converters (ADC) will convert the electrons into digital counts [DN]. Unfortunately, this conversion is not linear. This means that the increase in light intensity, or radiance, received is not proportional to the increase in signal of the detector. This effect is called the non-linearity of the detector. This issue will have to be corrected during the calibration of the instrument. [5]

Fig. 3.1 illustrates the non-linearity of the detector [2]. It shows the digital counts perceived by the instrument with respect to the integration time. The expected signal, represented in grey, is linear but the observed signal, represented in red, is not. Any deviation from this linear line will be considered as non-linearity and must be corrected. Moreover, it goes without saying that the quality of the measurements will be affected by this non-linearity issue. That is why it is important to correct it.

---

[1] In this part of the project, 'KDP' will always refer to the KDPs related to the non-linearity of the detector.

[2] This figure is an example that shows the non-linearity of a fictional detector. The observed line could be very different from the one depicted in this figure. The point here is to show a deviation from the expected linear line. Any type of deviation, even above the linear line, would be considered as a non-linearity.
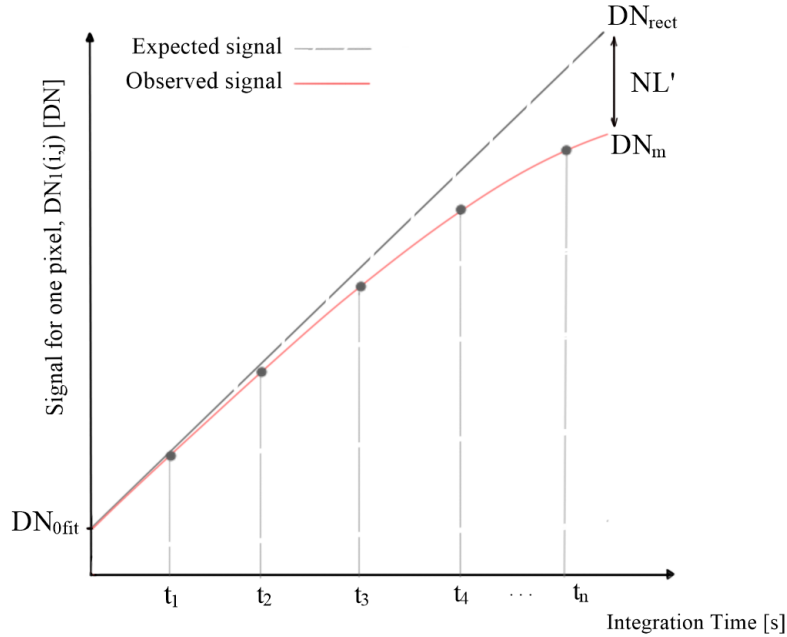
Figure 3.1: This figure represents the signal of one pixel (i,j) in [DN] with respect to the integration time in [s]. The observed signal, DN_m, is represented in red while the expected signal, DN_{rect}, is represented in grey. NL' represents the non-linearity and DN_{0fit} is the y-intercept of both lines.

3.2   KDP computation: algorithm 1

In this section, the first algorithm behind the computation of KDP will be introduced. This algorithm is inspired by the one described in the ATBD [6]. It has also been hand coded and can be found in the Annex. A.1.

The first part of this section will focus on the mathematical development of the algorithm. The second part will show the inputs and outputs and the last part consists on a verification of the code.

3.2.1   Mathematical development

Fig. 3.1 shows the non-linearity effect for one pixel. It represents the signal on the pixel (i,j), denoted by $DN_1(i, j)$ with respect to the integration time. Four integration times are represented: t_1, t_2, t_3 and t_4. The dashed line, DN_{rect}, represents the expected signal while the red line, DN_m, represents the observed signal. DN_{0fit} is the y-intercept of both lines. NL' represents the non-linearity. It corresponds to the observed signal minus the expected signal. The goal of the algorithm is to characterize this function. The following developments will consider the computation of the non-linearity for one spectral band only. The mathematical developments are the same for the three spectral bands.

Firstly, the inputs are a set of images with varying integration time, or t_{int}. In order to have better quality measurements a certain number of images N_{acq} is acquired, each with a specific integration time. Fig. 3.2 shows the inputs for one spectral band. The annotations t_1, t_2, t_3 and t_n represent different integration times. For each one several

acquisitions are taken. The number of acquisitions is represented respectively for each integration time by N_{acq1}, N_{acq2}, N_{acq3} and N_{acqn}. The arrow shows one image, or one acquisition, taken with an integration time of t_1. The total number of input images is then the sum of all the acquisitions.
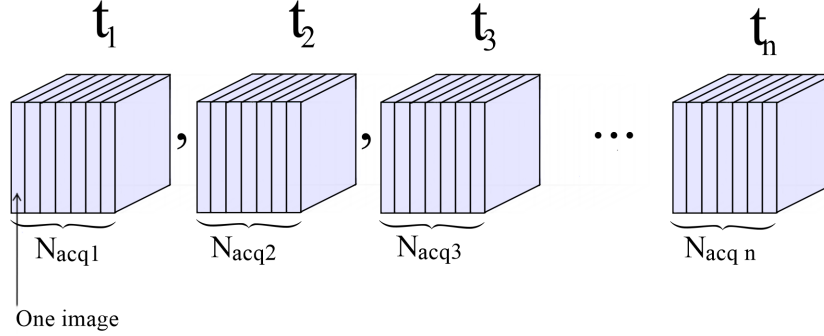
Figure 3.2: This figure represents the input images acquired by the instrument. t_1, t_2, t_3 and t_n represent different integration time and N_{acq1}, N_{acq2}, N_{acq3} and N_{acqn} represent respectively the number of acquisitions for each integration time.

As there are several acquisitions, it is necessary to compute the mean of all the acquisitions given an integration time. This will have to be done for all the integration time and for all the pixels. After this, each pixel will have n number of measurements. These measurements are represented by the dark dots in Fig. 3.1. They will be expressed by the function $DN(t_{int})$.

By using the function *Polyfit()* in Matlab these points can be fitted to a curve. The polynomial that has been fitted to these measurements is represented in red in Fig.3.1. This function can be written as Eq. 3.1. The order k of the useful polynomial will have to be determined. The first two terms of this equation represents the linear part $DN_{rect}(t_{int})$ while all the others represent the non-linearity. The order of the polynomial $DN_m(t_{int})$ must be chosen to fit as best as possible the data described by $DN(t_{int})$.

$$DN_m(t_{int}) = \underbrace{DN_{0fit} + Pt_1 . t_{int}}_{\text{Linear part}} + \underbrace{Pt_2 . t_{int}^2 + Pt_3 . t_{int}^3 + ... + Pt_k . t_{int}^k}_{\text{Non-linearity part}}, \quad (3.1)$$

$$DN_{rect}(t_{int}) = Pt1 . t_{int} + DN_{0fit} , \quad (3.2)$$

where DN_{0fit}, Pt_1, Pt_2, Pt_3 and Pt_k represent the polynomial coefficients.

The non-linearity can be written as Eq. 3.3. It represents the ratio of non-linearity with respect to $DN_{rect}(t_{int}) - DN_{0fit}$.

$$NL(t_{int}) = \frac{DN(t_{int}) - DN_{rect}(t_{int})}{DN_{rect}(t_{int}) - DN_{0fit}} = \frac{NL'(t_{int})}{DN_{rect}(t_{int}) - DN_{0fit}} \qquad (3.3)$$

$NL(t_{int})$ can then be fitted to a polynomial. The order, m, will have to be determined. Again $NL(t_{int})$ and $NL_m(t_{int})$ might not be equal as the polynomial might not perfectly fit the values of $NL(t_{int})$.

$$NL_m(t_{int}) = PNL_0 + PNL_1 . DN(t_{int}) + ... + PNL_m . DN(t_{int})^m, \qquad (3.4)$$

where, PNL_0, PNL_1 and PNL_m are the polynomial coefficient.

3.2.2   Inputs and Outputs

Firstly, Tab. 3.1 summarizes the necessary inputs for the algorithm (table given by Leonardo [6]). They consist of a set of images for each spectral band. These images are corrected from previous effects such as removing bad, dead and saturated pixels, and correcting the offset. Unfortunately, the computation of the KDP related to these effects will not be considered in this work. However, the correction of these effects will be applied during the real phase of calibration. The images are expressed in DN and their sizes correspond to the size of their respective detectors. These images will be captured by FLORIS thanks to GSE during the calibration campaign. At the moment, there are no images from FLEX available.

Finally, Tab. 3.2 represents the outputs in the case where Eq. 3.4 is a polynomial of order 1 (table given by Leonardo [6]). Each spectral band has three types of KDP which are: $DN0_{fit}$, PNL_0 and PNL_1. The number of outputs rely on the order of Eq. 3.4. For example, if the order was 4 we would have 6 outputs per spectral band. These outputs will then be useful in the calibration model later on (section. 3.4). They will permit to correct images from non-linearity.

Name	Data Type	Units	Size	Note
DN2 HR1	Integer	DN	530 rows, 140 cols	HR1 image corrected for previous effects
DN2 HR2	Integer	DN	530 rows, 268 cols	HR2 image corrected for previous effects
DN2 LR	Integer	DN	530 rows, 234 cols	LR image corrected for previous effects

Table 3.1: Inputs of the KDP computation algorithm. They correspond to images from each spectral band and are corrected from previous effects. The units are in DN and their size is the binning of the detector matrix of each spectral band.

Name	Data Type	Units	Size	Note
HR1 DN_{0fit}	Float	DN	530 rows, 140 cols	Fit parameters of DN vs t_{int} for HR1 spectral band
HR1 PNL_0	Float	$\sim$	530 rows, 140 cols	Fit parameter of NL vs DN for HR1 spectral band
HR1 PNL_1	Float	DN^{-1}	530 rows, 140 cols	Fit parameter of NL vs DN for HR1 spectral band
HR2 DN_{0fit}	Float	DN	530 rows, 268 cols	Fit parameters of DN vs t_{int} for HR2 spectral band
HR2 PNL_0	Float	$\sim$	530 rows, 268 cols	Fit parameter of NL vs DN for HR2 spectral band
HR2 PNL_1	Float	DN^{-1}	530 rows, 268 cols	Fit parameter of NL vs DN for HR2 spectral band
LR DN_{0fit}	Float	DN	530 rows, 234 cols	Fit parameters of DN vs t_{int} for LR spectral band
LR PNL_0	Float	$\sim$	530 rows, 234 cols	Fit parameter of NL vs DN for LR spectral band
LR PNL_1	Float	DN^{-1}	530 rows, 234 cols	Fit parameter of NL vs DN for LR spectral band

Table 3.2: Outputs of the KDP computation algorithm. The table gathers all the KDP which need to be computed. There are three KDP per spectral band: $DN0_{fit}$, PNL_0 and PNL_1. These are fit parameters obtained by DN vs t_{int} for the first one and by NL vs DN for the two last ones.

3.2.3   Verification of the code

A way to verify the code (Annex. A.1) is to derive input images from known polynomial coefficients. Considering that $DN_m(t_{int})$ is a second order polynomial, it can be written as:

$$DN_m(t_{int}) \; = \; DN_{0fit} \; + \; Pt_1 \, . \, t_{int} \; + \; Pt_2 \, . \, t_{int}^2 \, . \tag{3.5}$$

The polynomial coefficients are fixed at:

- $DN_{0fit} = 10 \; DN$,

- $Pt_1 = 30 \; DN.ms^{-1}$,

- $Pt_2 = -100 \; DN.ms^{-2}$.

Based on these known polynomial coefficients, input images are created. After the creation, the polynomial coefficients of DN_m are forgotten. The code then takes these images as inputs and computes the polynomial coefficients. If the code returns the exact same coefficients, the code is correct.

In order to verify $NL(t_{int})$, Eq. 3.6 is computed and compared with values given by the code. As all of these parameters have been verified it is easy to check if the values of $NL(t_{int})$ are the good ones.

$$NL(t_{int}) \; = \; \frac{DN(t_{int}) - DN_{rect}(t_{int})}{DN_{rect}(t_{int}) - DN_{0fit}} = \frac{DN(t_{int}) - Pt_1.t_{int} + DN_{0fit}}{Pt_1.t_{int} + DN_{0fit} - DN_{0fit}}$$

$$= \frac{DN(t_{int}) - Pt_1.t_{int} + DN_{0fit}}{Pt_1.t_{int}} \tag{3.6}$$

By applying this methodology the code as been verified. Fig. 3.3 shows the constructed points $DN(t_{int})$ and the fit $DN_m(t_{int})$ as well as the points $NL(t_{int})$ and the fit $NL_m(t_{int})$ for pixel(10,10). $NL_m(t_{int})$ do not fit the data perfectly because the order of the polynomial is not appropriate.
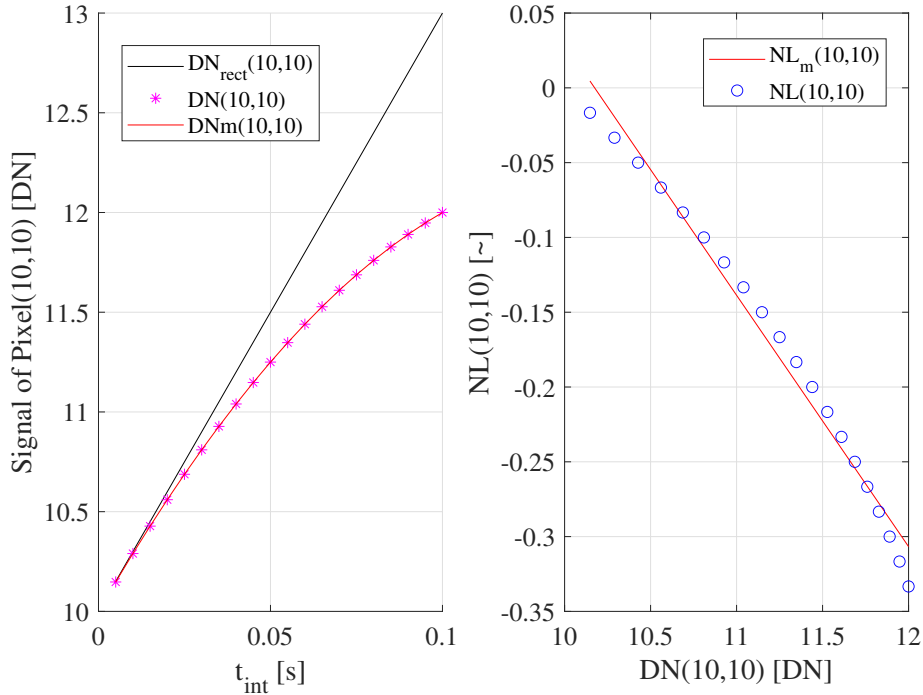
Figure 3.3: The left figure represents the signal in pixel(10,10) in [DN]with respect to t_{int} in [s]. The right figure represents the non linearity with respect to $DN(10, 10)$ in [DN].

3.3   KDP computation: algorithm 2

In this section, a second method for computing KDP will be introduced. This algorithm does not come from the ATBD. The goal is to compare both algorithms and see if the one presented in the ATBD is the best. The code of this algorithm can be found in Annex. A.2.

This section will explained the mathematical development behind the code. The inputs and outputs of this code are the same as for the first algorithm and are respectively represented by Tab. 3.1 and Tab. 3.2. The code has also been verified according to the methodology described in Sec. 3.2.3.

3.3.1   Mathematical development

The only step varying from the first algorithm is the way of computing the linearized signal: $DN_{rect}(t_{int})$. All the other steps remain the same. For this reason, only the devel-

opment of $DN_{rect}(t_{int})$ will be explained.

$DN_{rect}(t_{int})$ can be computed by assuming that the signal $DN_m(t_{int})$ is linear at an average signal value of DN_{midd} corresponding to an integration time of t_{midd}. Thus, a linear line can be constructed passing through this point. Fig. 3.4 shows $DN_{rect}(t_{int})$ and $DN_m(t_{int})$ as well as the average signal value DN_{midd} and t_{midd}. In order to characterize this linear line, we need the value of its slope and its intercept named p. Eq. 3.7 represents $DN_{rect}(t_{int})$. The following mathematical development shows how the slope and intercept are found.

$$DN_{rect}(t_{int}) \ = \ Slope \ . \ t_{int} \ + \ p \tag{3.7}$$
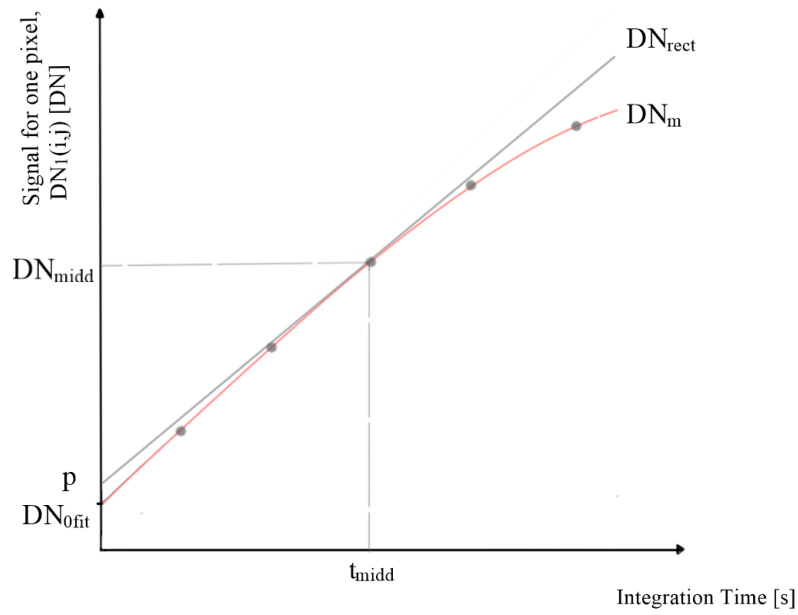
Figure 3.4: This figure represents the signal of one pixel (i,j) in [DN] with respect to the integration time in [s]. DN_m is represented in red and DN_{rect} is represented in grey. P is the y-interpect of DN_{rect} and DN_{0fit} is the y-intercept of DN_m.

In order to have the slope of $DN_{rect}(t_{int})$ we can compute the slope of $DN_m(t_{int})$ at t_{midd}. This is done simply by derivating $DN_m(t_{int})$:

$$DN_m(t_{int}) \ = \ DN_{0fit} + Pt_1 \ . \ t_{int} \ + Pt_2 \ . \ t_{int}^2 \ + \ Pt_3 \ . \ t_{int}^3 \ + \ ... + Pt_k \ . \ t_{int}^k \ , \tag{3.8}$$

$$Slope \ = \ Pt_1 \ + \ 2 \ . \ Pt_2 \ . \ t_{midd} \ + \ 3 \ . \ Pt_3 \ . \ t_{midd}^2 \ + \ ... + k \ . \ Pt_k \ . \ t_{midd}^{k-1} \ . \tag{3.9}$$

$DN_m(t_{midd})$ can be computed thanks to Eq. 3.8:

$$DN_m(t_{midd}) \ = \ DN_{0fit} + Pt_1 \ . \ t_{midd} \ + Pt_2 \ . \ t_{midd}^2 \ + \ Pt_3 \ . \ t_{midd}^3 \ + \ ... + Pt_k \ . \ t_{midd}^k \tag{3.10}$$

Eq. 3.10 in Eq. 3.7 will give the value of the intercept:

$$DN_m(t_{midd}) \ = \ Slope \ . \ t_{midd} \ + \ p \quad \rightarrow \quad p \ = \ DN_m(t_{midd}) \ - \ Slope \ . \ t_{midd} \tag{3.11}$$

By replacing Eq. 3.9 and Eq. 3.11 in Eq. 3.7 $DN_{rect}(t_{int})$ is found.

3.4 Calibration model related to non-linearity

This section will explain the calibration model for the non-linearity correction. All of the calibration model's code will not be explained but only the part where the non-linearity is being corrected. As the KDP have been computed, the image can now be corrected from the non-linearity. The code can be found in Annex. A.3. The first part will explain the inputs and outputs of the algorithm and the last part will explain the mathematical development behind the code.

3.4.1 Inputs and Outputs

The inputs of this code are the images already corrected from previous effects, described in Tab. 3.1, and the KDPs computed from the previous algorithms, described in Tab. 3.2. The outputs are represented in Tab. 3.3 (table given by Leonardo [6]). They correspond to images corrected from the non-linearity.

Name	Data Type	Units	Size	Note
DN3 HR1	Float	DN	530 rows, 140 cols	HR1 image corrected for the non-linearity
DN3 HR2	Float	DN	530 rows, 268 cols	HR2 image corrected for the non-linearity
DN3 LR	Float	DN	530 rows, 234 cols	LR image corrected for the non-linearity

Table 3.3: Outputs of the calibration model related to the non-linearity. They correspond to the corrected image for each spectral band. The units are in DN and their size is the binning of the detector matrix of each spectral band.

3.4.2 Mathematical development

The image $DN2$ is corrected using the KDP previously computed. The equation can be written as:

$$DN3 \ = \ \frac{DN2 \ - \ DN_{0fit}}{NL_m(t_{int}) \ + \ 1} \ + \ DN_{0fit} \tag{3.12}$$

By supposing that $NL(t_{int})$ is perfectly approximated by the polynomial $NL_m(t_{int})$, we can replace $NL(t_{int})$ into 3.12.

$$DN3 \ = \ \frac{DN2 \ - \ DN_{0fit}}{NL(t_{int}) \ + \ 1} + DN_{0fit} = \ \frac{DN2 \ - \ DN_{0fit}}{\dfrac{DN2 - DN_{rect}(t_{int})}{DN_{rect}(t_{int}) - DN_{0fit}} \ + \ 1} + DN_{0fit} \tag{3.13}$$

$$DN3 \ = \ \frac{DN2 \ - \ DN_{0fit}}{\dfrac{DN2 - Pt_1.t_{int} - DN_{0fit}}{Pt_1.t_{int} + DN_{0fit} - DN_{0fit}} + 1} + DN_{0fit} \tag{3.14}$$

$$DN3 \ = \ \frac{(DN2 - DN_{0fit}).Pt_1.t_{int}}{(DN2 - DN_{0fit})} + DN_{0fit} = Pt_1.t_{int} + DN_{0fit} \tag{3.15}$$

$$DN3 = DN_{rect}(t_{int}) \tag{3.16}$$

In the case the polynomial fits perfectly $NL(t_{int})$: $DN3$ is equal to DN_{rect}. However, as it is not always possible to perfectly fit a polynomial to its data, $DN3$ will not be equal to DN_{rect} and the correction will not be perfect. [6]

In this part of the work, the algorithm explained in the previous chapter will be applied. In order to correct the non-linearity of FLORIS's detector, on-ground measurements of the instrument (with the GSE) are required. These measurements collected for different integration times, with several acquisitions, will then be used as inputs of the algorithm to compute KDPs. However, as the calibration of this instrument is a project at an early stage, on-ground measurements have not been performed yet. This means that, in order to test the code with relevant inputs, images from another instrument are needed. This other instrument is 3MI, which stands for Multi-viewing Multi-channel Multi-polarisation imager. It is one of the ten instruments on board of the MetOp-Second Generation (MetOp-SG) satellite. The purpose of 3MI is study the aerosols and the chemical composition of the atmosphere. It also needs to be calibrated by CSL but the project is more advanced and some inputs are already available to compute KDPs related to non-linearity. The computed KDPs using 3MI's images will not be correct for FLORIS as the two instruments are different and need different calibration parameters. Nonetheless, it is a great way to be familiarized with the technique before the actual measurement on FLORIS.

This section will be divided in six parts. The first part will explain briefly what is 3MI and the format of the inputs. Then, the filters that need to be applied on the measurements will be explained. Finally, the last parts will focus on the results obtained with the algorithm 1 and the algorithm 2.

4.1 3MI's images

In order to calibrate the non-linearity of the detector, the instrument has been tested in CSL with the help of GSE. As explained earlier, in order to compute the non-linearity a set of images with different integration times are needed. First of all, we will explain briefly the setup used for 3MI and which will be similar in the case of FLORIS. The following figure shows the principle of the calibration setup.
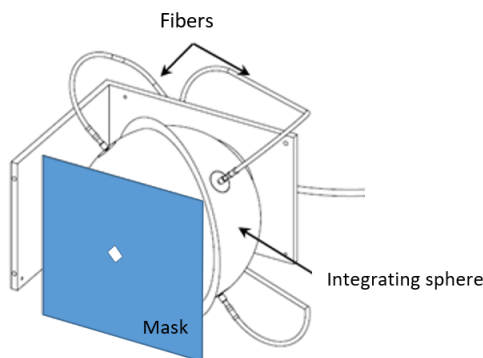
Figure 4.1: Calibration setup.

An integrating sphere is an optical component that has a spherical shape. It consists of a cavity with highly diffusive white interior coating. It has small holes (ports) for entrance and exit [13]. The light from a source (lamp or laser) transported to the entrance port. The goal of the integrating sphere is to obtain uniformly distributed radiation in all directions, so as the exit port appears uniform. At the output port of the integrating sphere, a square mask is placed which defines the field-of-view illuminating the instrument.

Fig. 4.2 shows an example of one image taken with the instrument 3MI. Due to the small field-of-view defined by the square mask, only the middle part of the detector is illuminated (in red). The integration time is 0.003s and it is the second acquisition. The field-of-view with which all the images have been taken is 0.0256°.
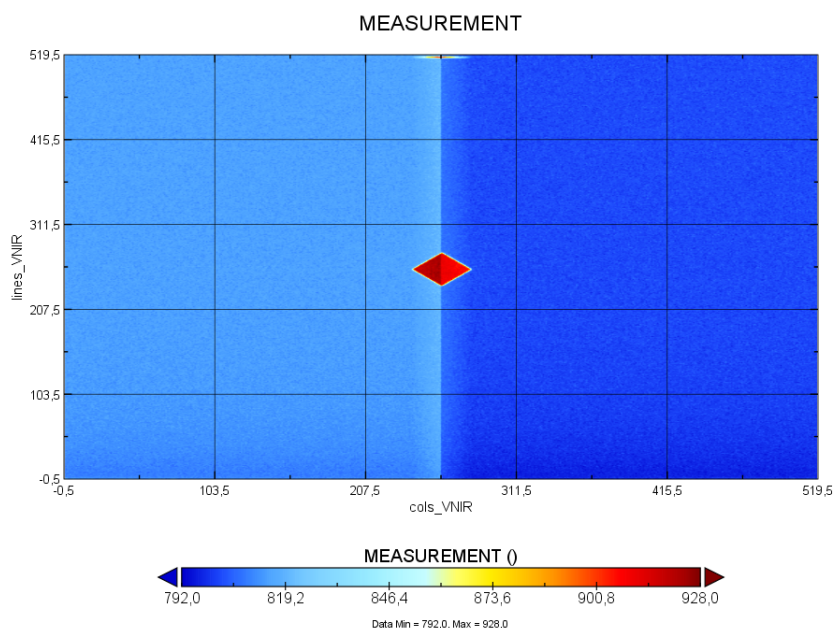
Figure 4.2: Image taken from 3MI. The instrument has been illuminated by an integrating sphere. The intensity of the radiation is shown thanks to colours, dark blue corresponding to less intensity, starting at 792 DN and red corresponding to higher intensity, the maximum being 928DN.

As there is only one part of the detector that is illuminated the computation of KDPs is only valid for these pixels. The non-linearity of the other non-illuminated pixels will have to be corrected with larger field-of-views (as larger field-of-view will illuminate more pixels). These images are not corrected from the other effects. For example, the dark current signal, the offset, are not corrected and bad, dead and saturated pixels are not eliminated. In fact, as only the middle part of the detector is illuminated, there would normally be only a signal in this part. But as can be seen in Fig. 4.2 there is a signal outside the illuminated area. Moreover, in the figure two parts can be seen: a lighter blue side and a darker blue side. This is due to the fact that 3MI has two readout registers and that each one of them has a different offset. FLORIS has four different readout registers, so if the images were taken by FLORIS, four different parts would be visible, each with their own offset.

3MI's images are NETCDF (network Common Data Form) files. This type of file is used to store scientific data and variables. Fig. 4.2 has been opened with the software Panoply. This software stretches the image in the x-direction. In fact, Fig. 4.2 is actually a square. In matlab, the NETCDF files containing the measurements are read thanks to the code in Annex. A.4. This code is also used to compute the mean of every acquisition given an integration time. Images will be displayed in gray levels thanks to the function Mat2gray() in Matlab. This function takes the maximum value of the signal and represents it in white while the minimum value of the signal is represented in black. Fig. 4.3 shows an image at integration time of 0.004 s. The function also transforms the values of the signal to a range between 0 (black) and 1 (white). As only the illuminated part is interesting, all the other pixels are fixed at a value of 900 DN which corresponds to the minimum value of the image.

Fig. 4.3 also shows the coordinates of the illuminated part. The pixels that forms the square are: (242,258), (261,238), (280,258) and (261,277). The pixel (261,258) corresponds to the middle of the illuminated region. It is important to notice that the borders of the square are formed by pixels that are not directly illuminated by the integrating sphere, but by scattering. This region is called the transition zone. As it is challenging to separate the transition zone from the illuminated zone, they have been both taken into account. The transition zone is represented in white in Fig. 4.2. The pixels from this region (for example pixel (277,260)) have a weaker signal than the signal from the directly illuminated pixels. The computation of KDP will also be performed for the pixels in the transition region but the result might not be representative of the reality. Again, a higher field-of-view is needed to be able to correctly characterize these pixels.
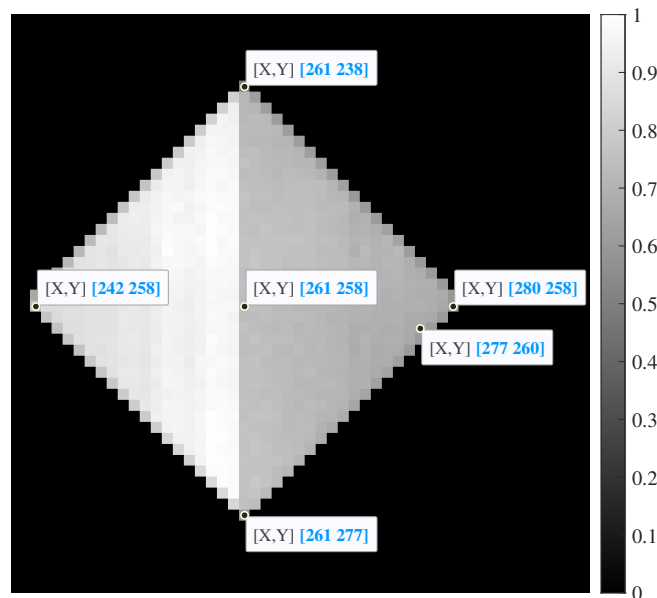
Figure 4.3: Representation of an image taken with the instrument 3MI with an integration time of 0.004s. White represents the maximum value of intensity and black represents the minimum value of intensity. In this figure the pixels in non-illuminated region have been fixed at 900 DN.

4.2   Filters

As other effects have not been calibrated, the data need to be pre-processed. According to [14] several filters need to be applied to the data before the computation of the KDP related to the non-linearity.

The first filter will remove the pixels that reach the maximum signal value during the two first integration times. In the illuminated area, there are no pixels of that kind to be removed. Nonetheless, in the non-illuminated area there are some pixels that show this kind of bad behavior. Fig. 4.4 shows the image at an integration time of 0.004 s. The red arrow shows the bad pixels. The image on the right is a zoom of the affected region. As said before, all the pixels in the non-illuminated area will be fixed at 900 DN and the non-linearity will not be corrected for them. So in the scope of this project these pixels will not affect the results. However, in future studies, when this region will be illuminated with a higher field-of-view, these pixels will have to be removed.
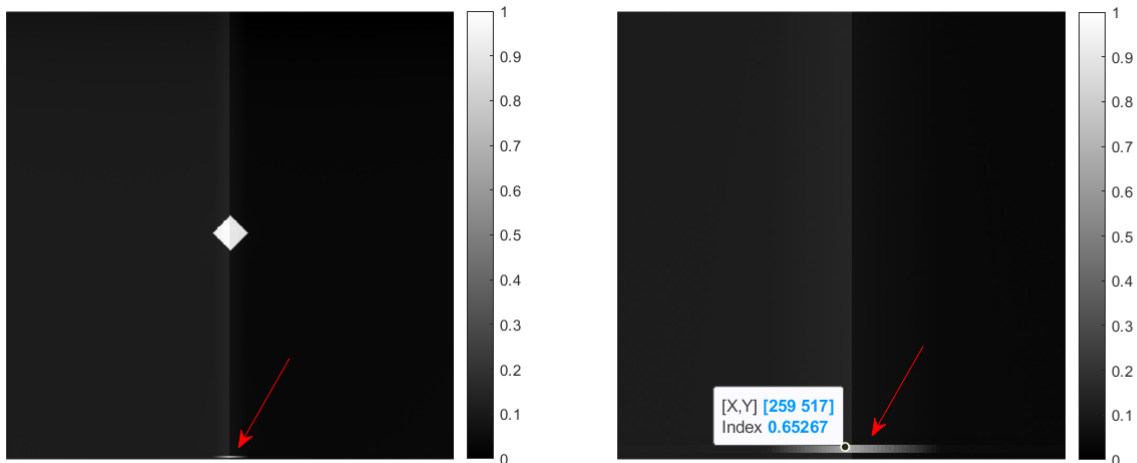
Figure 4.4: Representation of an image taken with the instrument 3MI with an integration time of 0.004s. The red arrow shows the bad pixels in the non-illuminated area. In this figure the pixels in the non-illuminated area have not been fixed at 900DN.

The second filter removes the dead pixels. According to [14] one pixel is considered dead if the signal emitted does not reach 100DN. There are no pixels of this kind in the illuminated area, so no pixels are removed.

The third filter computes for each pixels a function called $DN_{ramp}(t_{int})$. This function is a linear line constructed thanks to the three first measurements of $DN(t_{int})$. Any data that is at more than 25% away from the ramp will be ignored. This will limit odd measurements. Two examples are shown in Fig. 4.5. The figures represent $DN(t_{int})$ as well as $DN_{ramp}(t_{int})$ in function of the integration time for two different pixels (261,258) and (277,260). These pixels have been chosen as the majority of the other pixels have the same shape. In fact, in the illuminated area most of the pixels have the same shape as the middle pixel (261,258). And it is important to show how one of the transition pixel vary. In this project, there are no data in the illuminated region for any pixel to be ignored.
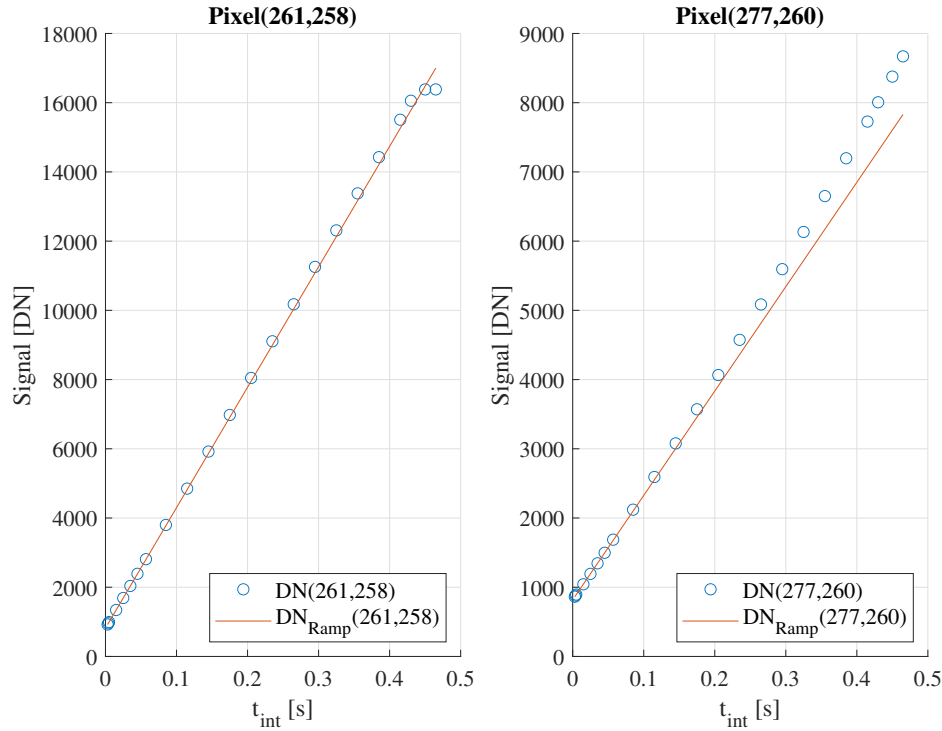
Figure 4.5: Signal in [DN] over t_{int} in [s]. The figures show DN_{ramp} in function of the measurements for two different pixels: (261,258) and (277,260).

4.3   Application of algorithm 1 as suggested in the ATBD

In this section the algorithm 1 will be used with the images taken by the instrument 3MI. The first part of the algorithm 1 consists on taking the mean of every acquisitions for each integration time and for each pixel. This is achieved with the code described in Annex. A.4 which will compute the function $DN(t_{int})$. The next steps of the algorithm will be done with the code described in Annex. A.1.

The order of the polynomials $DN_m(t_{int})$ and $NL_m(t_{int})$ will be taken at respectively 2 and 1 as suggested in the ATBD. Thus, Eq. 3.1 and 3.4 can be written as:

$$DN_m(t_{int}) \; = \; DN_{0fit} \; + \; Pt_1 \, . \, t_{int} \; + \; Pt_2 \, . \, t_{int}^2, \tag{4.1}$$

$$NL_m(t_{int}) \; = \; PNL_0 \; + \; PNL_1 \, . \, DN(t_{int}). \tag{4.2}$$

The following part consists on fitting the polynomial $DN_m(t_{int})$ to the data $DN(t_{int})$. Thanks to the function *Polyfit* in Matlab, the three polynomial coefficients DN_{0fit}, Pt_1 and Pt_2 are found. This step is performed for every pixels in the illuminated area. As the two first polynomial coefficients have been found, DN_{rect} can be computed:

$$DN_{rect} \; = \; Pt_1 \, . \, t_{int} \; + \; DN_{0fit}. \tag{4.3}$$

Fig. 4.6 shows $DN(t_{int})$, $DN_m(t_{int})$ and $DN_{rect}(t_{int})$ for the pixels (261,258) and (277,260). As explained earlier, the pixel (261,258) is very representative of all the pixels

25

in the illuminated region while the pixel (277,260) shows how the pixels in the transition region behave. For the pixel (261,258), it can be seen that the polynomial of order 2 does not fit perfectly the data $DN(t_{int})$. However, for the pixel (277,260), the polynomial seems to fit better. In order to measure how well the polynomial fits the data the function χ^2_{DN} is introduced. This function is written in Eq. 4.4. It represents the sum of every distance between the polynomial and $DN(t_{int})$ squarred over $DN(t_{int})$.

$$\chi^2_{DN} = \sum_{i=t_i}^{t_f} \frac{(DN_m(i) - DN(i))^2}{DN(i)} \tag{4.4}$$

For each pixel χ^2_{DN} is computed. The goal is then to minimize this value. In fact, the smaller χ^2_{DN}, the best the polynomial fits the data. Tab. 4.1 shows the χ^2_{DN} computed for the pixel (261,258) and (277,260). The last column of the table represents the mean of every χ^2_{DN} for every pixels in the illuminated area.

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
χ^2_{DN} [DN]	45.5771	0.1305	28.0115

Table 4.1: This table shows the value of χ^2_{DN} for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{DN} for every illuminated pixel.

Figure 4.6: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260).

Afterwards, the non-linearity is computed using Eq. 4.5. Then, the polynomial $NL_m(t_{int})$ will be fitted to the data $NL(t_{int})$. By using the function *Polyfit()* in Matlab, the two polynomial coefficients PNL_0 and PNL_1 are found. These are the outputs of algorithm 1. This step is performed for every pixel in the illuminated region. Fig. 4.7 shows $NL_{t_{int}}$

and $NL_m(t_{int})$ for the pixel (261,258) and (277,260). As can be seen in the figures, the polynomial fits poorly the data. The quantity χ^2_{NL} given by Eq. 4.6 needs to be computed to see how well the data have been fitted. Tab. 4.2 shows χ^2_{NL} computed for the pixel (261,258), (277,260) and the average value for every pixel.

$$NL(t_{int}) = \frac{DN(t_{int}) - DN_{rect}(t_{int})}{DN_{rect}(t_{int}) - DN_{0fit}} \tag{4.5}$$

$$\chi^2_{NL} = \sum_{i=t_i}^{t_f} \frac{(NL_m(i) + NL(i))^2}{NL(i)} \tag{4.6}$$

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X^2_{NL} [$\sim$]	7.1091	0.3645	6.3296

Table 4.2: This table shows the value of χ^2_{NL} for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{NL} for every illuminated pixel.
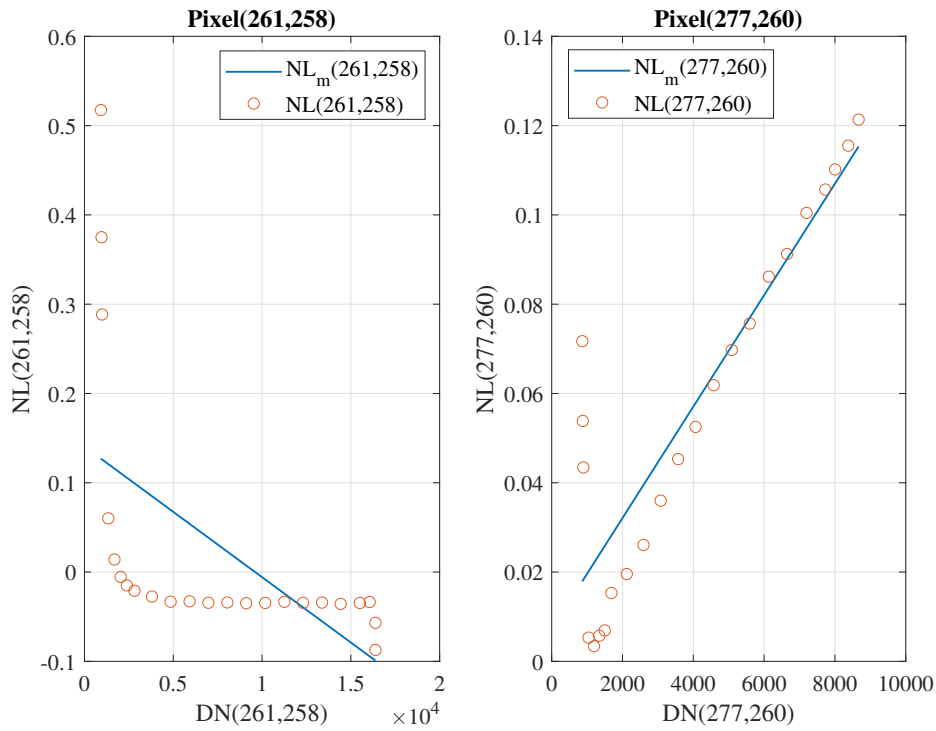
Figure 4.7: Representation of the non-linearity in [$\sim$] with respect to $DN(t_{int})$ in [DN] for the pixels (261,258) and (277,260).

Finally, the outputs of algorithm 1 are found: DN_{0fit}, PNL_0 and PNL_1. They will be used as inputs of the calibration model algorithm. This algorithm corrects the non-linearity for a given image. This code can be found in Annex. A.3. Fig. 4.8 shows the correction $DN_{corr}(t_{int})$ for each data in $DN(t_{int})$. The values are corrected perfectly if they are on the linear line $DN_{rect}(t_{int})$. So if $DN_{corr}(t_{int}) = DN_{rect}(t_{int})$. This is not the

case for pixel(261,258). However, the pixel (277,260) is nearly perfectly corrected. As before the value χ^2_{Err} will be computed thanks to Eq. 4.7.

$$\chi^2_{Err} \quad = \quad \sum_{i=t_i}^{t_f} \frac{(DN_{corr}(i) + DN_{rect}(i))^2}{DN_{rect}(i)} \tag{4.7}$$

Moreover, the percentage of error can also be computed. This is done thanks to Eq. 4.8. Tab. 4.3 shows the value of χ^2_{Err} and the Error for the pixel (261,258), (277,260) and the average values for every pixel.

$$Error = \frac{DN_{corr}(t_{int}) - DN_{rect}(t_{int})}{DN_{rect}(t_{int})} \cdot 100 \tag{4.8}$$

Pixel	(258,260)	(277,260)	Mean for every illuminated pixel
X^2_{Err} [DN]	390.4283	1.3364	239.3590
Error [%]	4.62	0.4	3.46

Table 4.3: This table shows the value of χ^2_{Err} and the percentage of error for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{Err} and the percentage of error for every illuminated pixel.
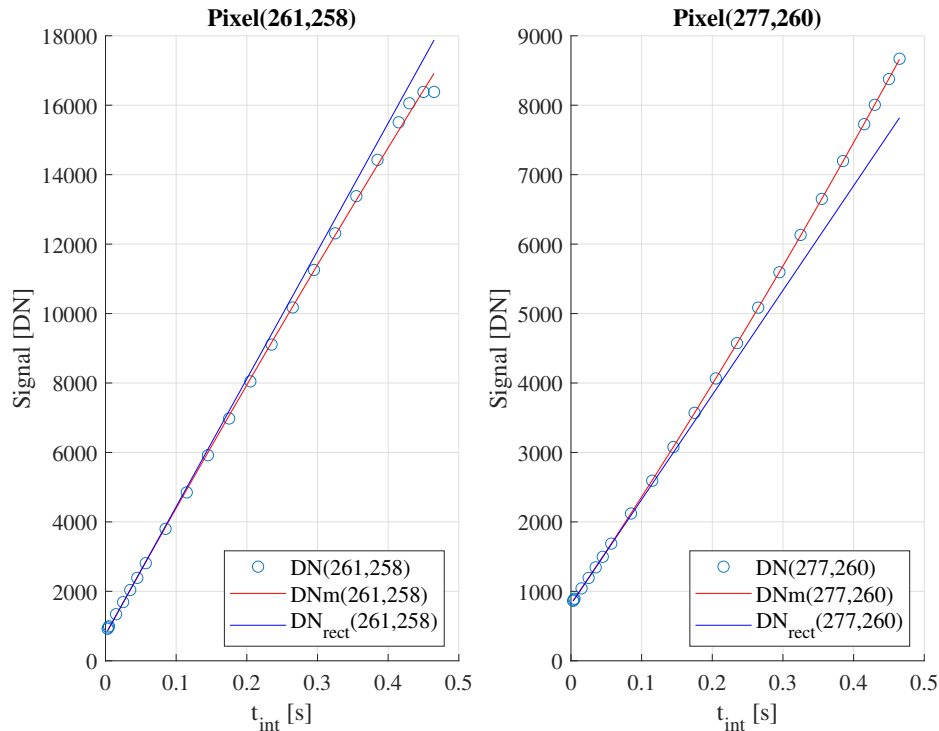

Figure 4.8: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260).

Fig. 4.9 shows the initial image with an integration time of 0.004 s on the left and the corrected image on the right. The corrected image is bad as it is not as homogeneous as the initial image. It should look smoother.
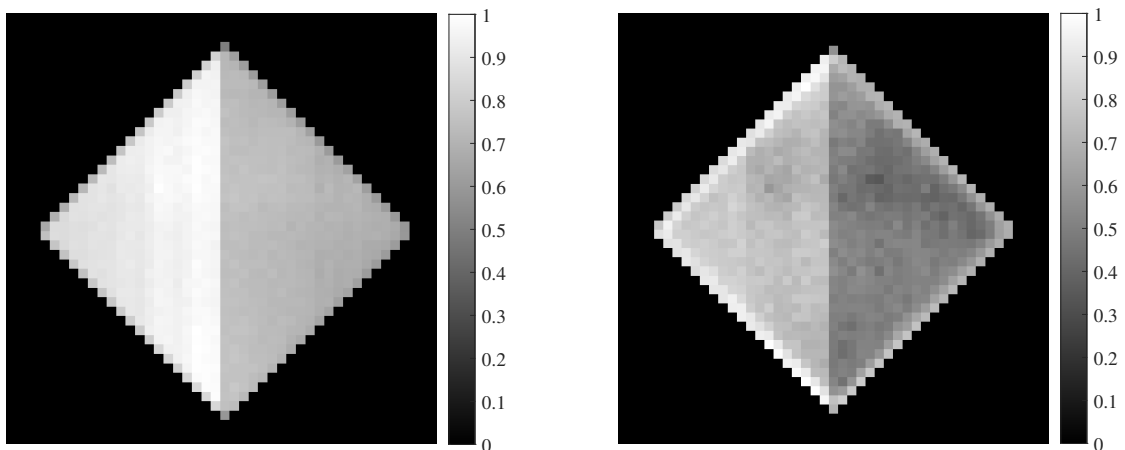
Figure 4.9: Difference between the initial image on the left and the corrected image on the right. The integration time of the image is 0.004 s.

4.3.1 Discussion

It can be seen that pixel (261,258) is not corrected properly. Most of the pixels follow the same path and are poorly corrected too. In fact, the average error for all the pixels is 3.46%. This is due to the fact that the orders for the polynomials suggested by the ATBD are not appropriate. In fact, $DN_m(t_{int})$ do not fit correctly the data $DN(t_{int})$. This leads to polynomial coefficients that do not represent well the data. As a result the function $DN_{rect}(t_{int})$ will be incorrect as it depends on the polynomial coefficients of $DN_m(t_{int})$. For the polynomial $NL_m(t_{int})$ the fit is even worse. Again, this means that the polynomial coefficients do not represent the data $NL(t_{int})$ correctly. As both polynomials are not well fitted to their corresponding data, the inputs of the calibration model algorithm: DN_{0fit}, PNL_0 and PNL_1 are not appropriate. This results in bad corrections.

To conclude, the rightness of the correction rely on how well the polynomials $DN_m(t_{int})$ and $NL_m(t_{int})$ are fitted to their respective data. For this reason, the orders proposed by the ATBD will not be considered anymore.

4.4 Application of algorithm 1 with varying polynomial order

In this section, the algorithm 1 will be used with the images taken by the instrument 3MI. The computation is done the same way as for Sec. 4.3. For this reason, all the steps of the algorithm will not be explained again. However, compared to Sec. 4.3 the order of the polynomials $NL_m(t_{int})$ and $DN_m(t_{int})$ will be different and need to be investigated.

The first step is to find the best suited order for the polynomial $DN_m(t_{int})$. The order that suits the best the polynomial would be the one that minimizes χ^2_{DN}. In fact, with a more appropriate order the polynomial and the data will be closer to each other which

will lead to a smaller χ^2_{DN}. Fig. 4.10 shows the variation of χ^2_{DN} with respect to the order of the polynomial $DN_m(t_{int})$.
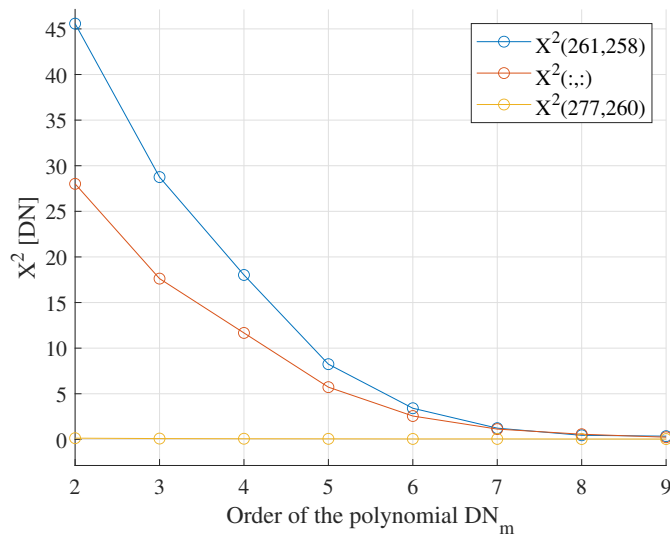
Figure 4.10: χ^2_{DN} in function of the order of the polynomial $DN_m(t_{int})$.

The blue line represents the pixel (261,258), the yellow line represents the pixel(277,260) and the red line represents the average value of all pixels. The polynomial of order 2 for the pixel(277,266) was already a good fit in (Sec. 4.3). For this reason, the value of χ^2_{DN} remains constant. However, most of the pixels and pixel(261,258) have high values of χ^2_{DN} for the small orders. The minimum is reached for a polynomial of order 9. Using the function *Polyfit()* in Matlab the polynomial $DN_m(t_{int})$ expressed by Eq. 4.9 can be fitted to the data $DN(t_{int})$. The 10 polynomial coefficients are then found.

$$DN_m(t_{int}) \;=\; DN_{0fit} \;+\; Pt_1 \,.\, t_{int} \;+\; ... \;+\; Pt_9 \,.\, t^9_{int}, \tag{4.9}$$

Then, $DN_{rect}(t_{int})$ can be expressed thanks to the first two polynomial coefficients just computed. Fig. 4.11 shows $DN(t_{int})$, $DN_m(t_{int})$ and $DN_{rect}(t_{int})$ in function of the integration time for the pixel (261,258) and (277,260). Tab. 4.4 shows the χ^2_{DN} for the pixel (261,258), (277,260) and for the average of every pixels. As can be seen, $DN_m(t_{int})$ fits much better the data in this section.

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X^2_{DN} [DN]	0.3510	0.0361	0.2363

Table 4.4: This table shows the value of χ^2_{DN} for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{DN} for every illuminated pixel.
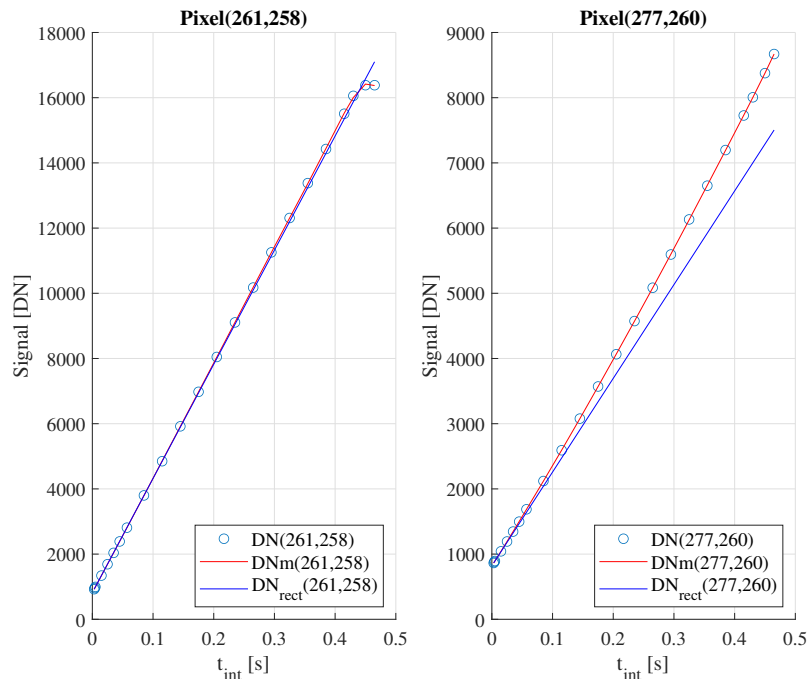
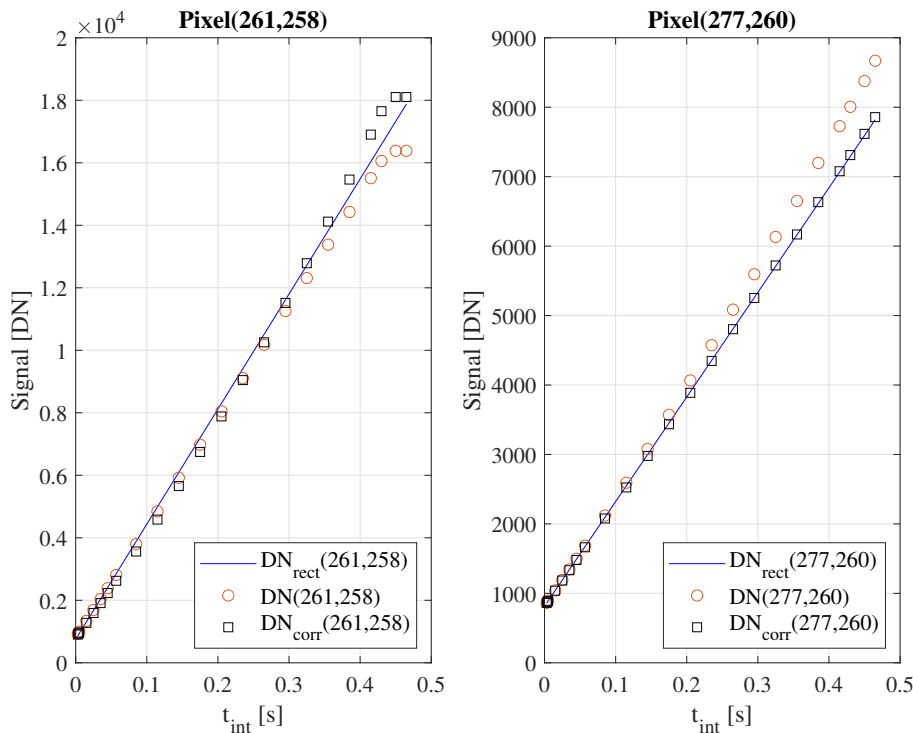**Figure 4.11:** Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260).

The second step is to find the best suited order for the polynomial $NL_m(t_{int})$. Again the order that suits the best the polynomial will be the order that minimizes χ^2_{NL}. Fig. 4.12 shows the variation of χ^2_{NL} with respect to the order of the polynomial $NL_m(t_{int})$. Again the blue line represents the pixel(261,258), the yellow line represents the pixel(277,260) and the red line represents the average for all pixels. The minimum value of χ^2_{NL} occurs at an order of 10. Thus the polynomial $NL_m(t_{int})$ can be written as Eq. 4.10. Fig. 4.13 shows how different order fits the data $NL(t_{int})$. The order 2, 4 and 8 are represented. Obviously, higher order polynomials give better approximation for the data $NL(t_{int})$.

$$NL_m(t_{int}) = PNL_0 + PNL_1 . DN(t_{int}) + ... + PNL_{10} . DN(t_{int})^{10} \qquad (4.10)$$

Figure 4.12: χ^2_{NL} in function of the order of the polynomial $NL_m(t_{int})$.

31
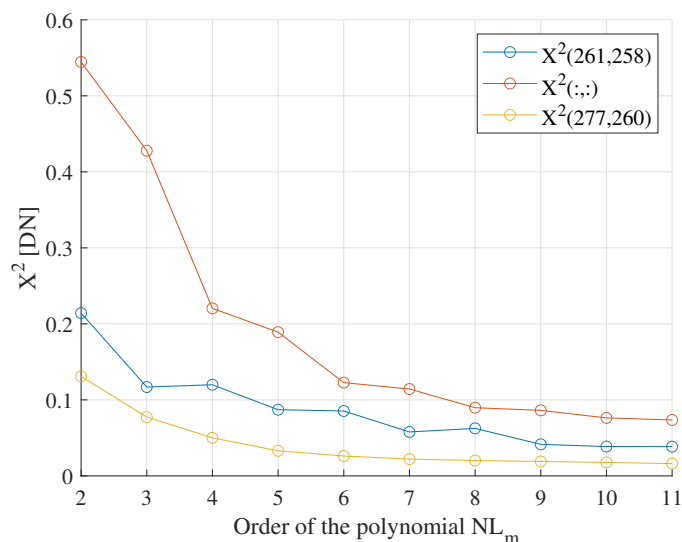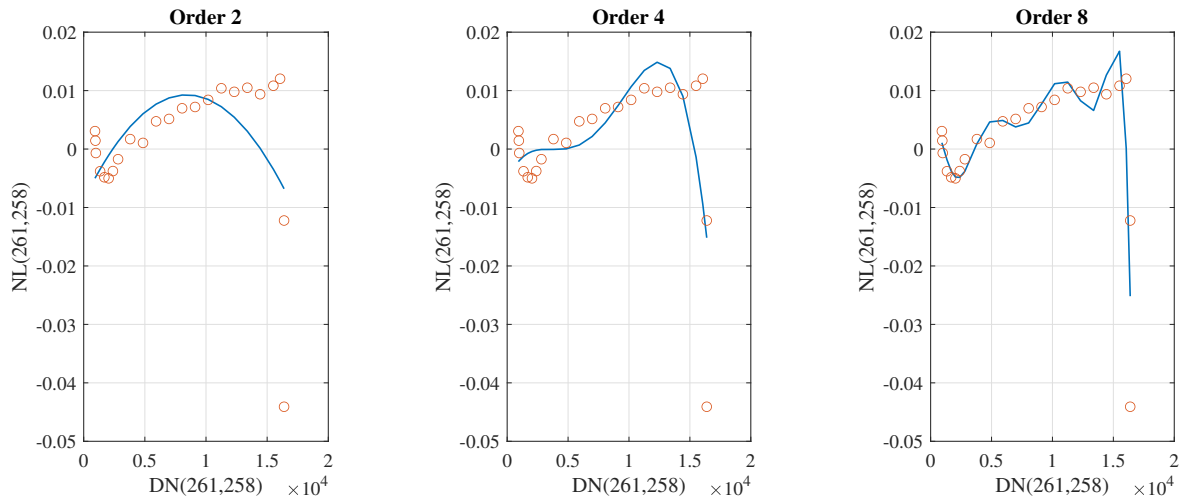
Figure 4.13: Representation of the non-linearity in [∼] with respect to $DN(t_{int})$ in [DN] for the pixel (261,258). Three polynomials $NL_m(t_{int})$ with the order 2, 4 and 8 are represented.

Fig. 4.14 shows $NL_m(t_{int})$ of order 10 and $NL(t_{int})$ with respect to $DN(t_{int})$ for pixel (261,258) and (277,260). Compared to Sec. 4.3 the polynomials fit better the data for both pixels. Tab. 4.5 shows χ^2_{NL} for the pixel(261,258), (277,260) and the average of every pixel. As expected, the values are smaller than the ones computed in Sec. 4.3.
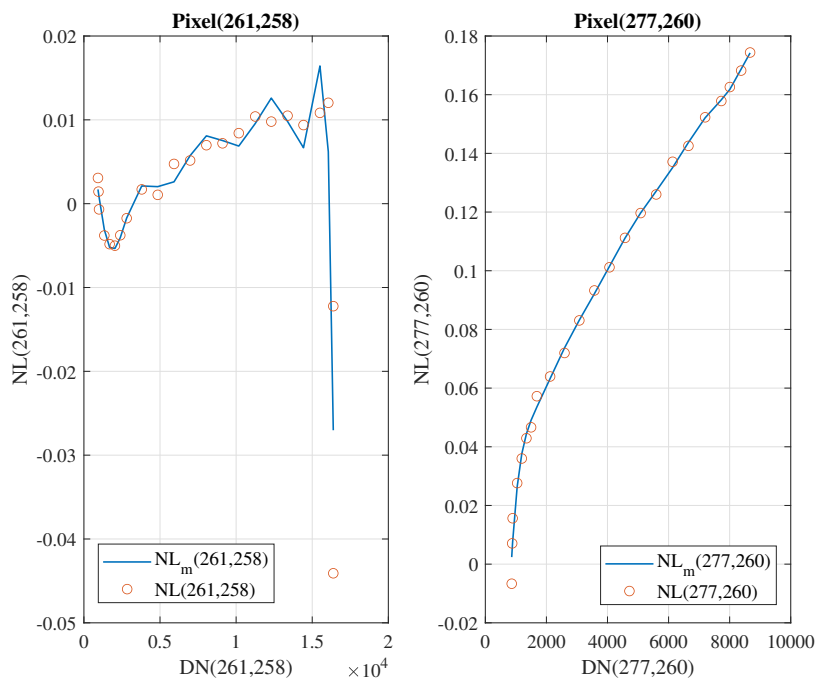
Figure 4.14: Representation of the non-linearity in [∼] with respect to $DN(t_{int})$ in [DN] for the pixels (261,258) and (277,260).

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X_{NL}^2 [$\sim$]	0.0386	0.0177	0.0763

Table 4.5: This table shows the value of χ_{NL}^2 for the pixels (261,258) and (277,260) as well as the mean value of χ_{NL}^2 for every illuminated pixel.

Finally, the outputs of algorithm 1 are found: DN_{0fit} and from PNL_0 to PNL_{10}. They will be used as inputs of the calibration model algorithm. This algorithm will be used to correct $DN(t_{int})$ to $DN_{corr}(t_{int})$. Fig. 4.15 shows the correction $DN_{corr}(t_{int})$, $DN(t_{int})$ and $DN_{rect}(t_{int})$ in function of the integration time. $DN_{corr}(t_{int})$ is perfectly corrected if it is equal to $DN_{rect}(t_{int})$. Compared to the previous section, the correction of $DN(t_{int})$ is much better. Tab. 4.4 shows χ_{Err}^2 and the error in percentage for the pixel(261,258), (277,260) and the average of every pixel. The average error is smaller than Sec. 4.3.

Pixel	(261,258)	(277,260)	Mean for every illuminated pixel
X_{Err}^2 [DN]	9.3717	0.0381	6.4915
Error [%]	0.2	0.05	0.24

Table 4.6: This table shows the value of χ_{Err}^2 and the percentage of error for the pixels (261,258) and (277,260) as well as the mean value of χ_{Err}^2 and the percentage of error for every illuminated pixel.
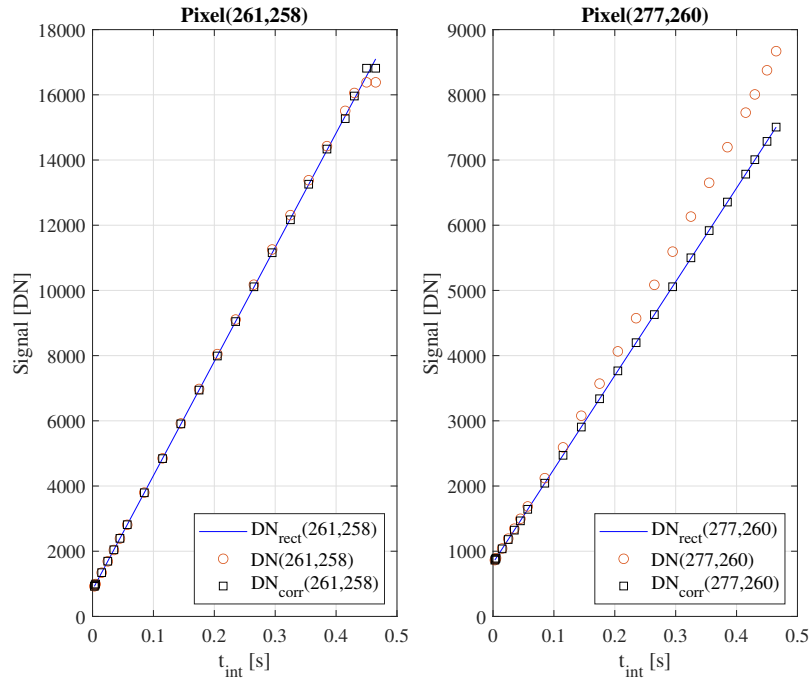
Figure 4.15: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260)

The calibration model algorithm is used again but this time for the correction of an image. Fig. 4.16 shows the initial image with an integration time of 0.004s on the left and

the corrected image on the right. The corrected image is smoother than the one presented in Sec. 4.3. This kind of correction is expected for the non-linearity.
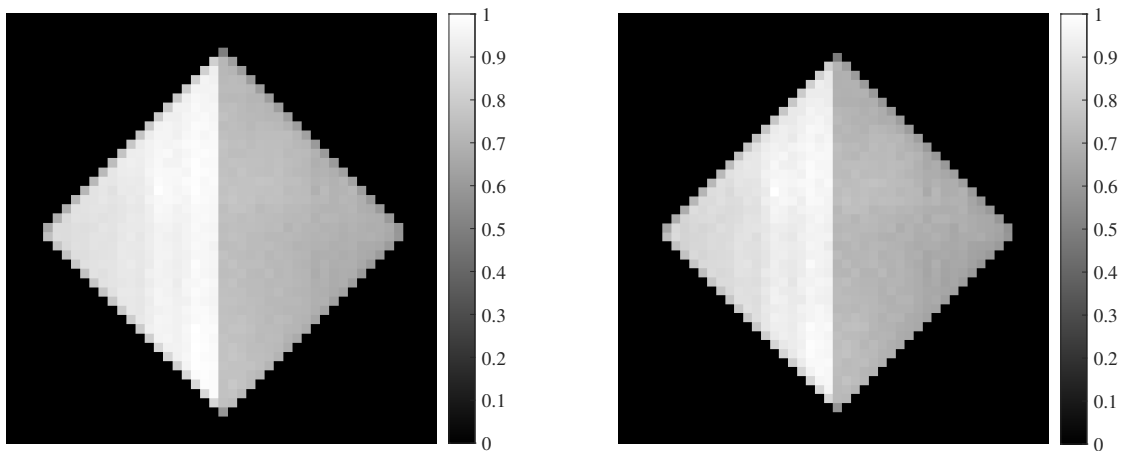
Figure 4.16: Difference between the initial image on the left and the corrected image on the right. The integration time of the image is 0.004 s.

4.4.1 Discussion

Firstly, polynomials of higher orders approximate better their respective data compared to the ones in Sec. 4.3. Concerning the polynomial $DN_m(t_{int})$, X_{DN}^2 is almost 10 times smaller for a polynomial of order 9 than for a polynomial of order 2. Regarding the polynomial $NL_m(t_{int})$, X_{NL}^2 is also almost 10 times smaller for the polynomial of order 10 than for the polynomial of order 1. The mean error was 3.46% in Sec. 4.3 while it is now 0.24%. This means that the order of the polynomials are more appropriate. The corrected image represented in Fig. 4.16 seems also more adequate than the one represented in Fig. 4.9 as the image in smoother with less contrast.

Secondly, because of the change in order for the polynomial $DN_m(t_{int})$, the linear line $DN_{rect}(t_{int})$ has not the same position as the one depicted in Fig. 4.6 from Sec. 4.3. This is due to the fact that the values of the first two polynomial coefficients have changed. It can be seen that $DN_{rect}(t_{int})$, computed with the polynomial $DN_m(t_{int})$ of order 9, is closer to the data $DN(t_{int})$. As the distance between the data and the linear line is smaller, the non-linearity between them is also smaller. This leads to less non-linearity to correct. In fact, in the methodology suggested by the ATBD the order of magnitude of $NL(t_{int})$ is about 0.1 and now the order of magnitude is about 0.01. To sum up, it is important to find an appropriate order for the polynomial $DN_m(t_{int})$, as it will determine the accuracy of the polynomial $DN_{rect}(t_{int})$.

Finally, there are still some issues regarding the correction of the last points of $DN(t_{int})$ in Fig. 4.15 for pixel (261,258). In fact, by looking at Fig. 4.11 it can be seen that the three last points in the data $DN(t_{int})$ have an odd position compared to the other points. Normally, with the increasing integration time they should give a signal of higher value. These three points show a behavior of saturation. For this reason, they should be ignored. In the next section the computation will be done without the three last points of the data $DN(t_{int})$ for each pixels. This will allow to have simpler data distribution $DN(t_{int})$ which

will lead to better fitting. It is important to note however that pixel (277,260) does not show this kind of behavior. In fact, pixels in the transition region do not receive the full intensity of the integration sphere.

4.5   Application of code 1 without the last three points

In this section, the algorithm 1 will be used with the images taken by the instrument 3MI. The computation is done the same way as for Sec. 4.3. For this reason, all the steps of the algorithm will not be explained again. However, for every pixel the last three points in the data $DN(t_{int})$ will be ignored. The order of the polynomials $NL_m(t_{int})$ and $DN_m(t_{int})$ will have to be investigated.

The first step is to find the best suited order for the polynomial $DN_m(t_{int})$. The order that suits the best the polynomial would be the one that minimizes χ^2_{DN}. Fig. 4.17 shows the variation of χ^2_{DN} with respect to the order of the polynomial $DN_m(t_{int})$. The blue line represents the pixel(261,258), the yellow line represents the pixel(277,260) and the red line represents the average of every pixel. Most of the pixels reaches a minimum χ^2_{DN} for a polynomial of order 8. By using the function *Polyfit* in Matlab, the polynomial $DN_m(t_{int})$ expressed by Eq. 4.11 can be fitted to the data $DN(t_{int})$. The corresponding 9 polynomial coefficients are then found.

$$DN_m(t_{int}) \;=\; DN_{0fit} \;+\; Pt_1 \,.\, t_{int} \;+\; ... \;+\; Pt_8 \,.\, t_{int}^8 \qquad (4.11)$$
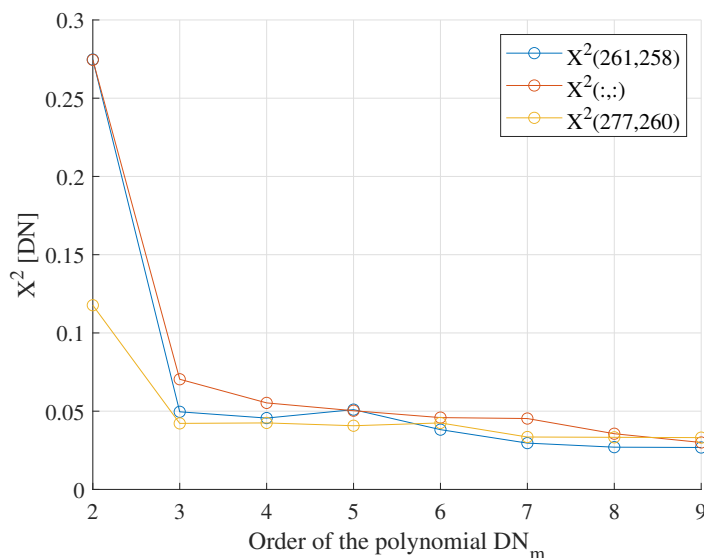
Figure 4.17: χ^2_{DN} in function of the order of the polynomial $DN_m(t_{int})$.

Then, $DN_{rect}(t_{int})$ can be expressed thanks to the first two polynomial coefficients of $DN_m(t_{int})$. Fig. 4.18 shows $DN(t_{int})$, $DN_m(t_{int})$ and $DN_{rect}(t_{int})$ in function of the integration time for the pixels (261,258) and (277,260). Tab. 4.7 shows the χ^2_{DN} for the pixel (261,258), (277,260) and for the average of every pixel. As expected, the value χ^2_{DN} for pixel(277,260) from Sec.4.4 do not change a lot from the value obtained in this section. This is because, there are no saturated data in the pixels in the transition region. In fact, they do not receive the full intensity emitted by the integration sphere. However, for

most of the pixels and for pixel (261,258) the polynomial fits better without the three last points of $DN(t_{int})$.

Figure 4.18: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260).

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X_{DN}^2 [DN]	0.027	0.0333	0.0356

Table 4.7: This table shows the value of χ_{DN}^2 for the pixels (261,258) and (277,260) as well as the mean value of χ_{DN}^2 for every illuminated pixel.

The second step is to find the best suited order for the polynomial $NL_m(t_{int})$. Again the order that suits the best the polynomial will be the order that minimizes χ_{NL}^2. Fig. 4.19 shows the variation of χ_{NL}^2 with respect to the order of the polynomial $NL_m(t_{int})$. The blue line represents the pixel (261,258), the yellow line represents the pixel (277,260) and the red line represents the average of every pixel. The minimum value of χ_{NL}^2 occurs at an order of 7. Thus the polynomial $NL_m(t_{int})$ can be written as Eq. 4.12. Fig. 4.20 shows how different order fits the data $NL(t_{int})$. The order 2, 4 and 6 are represented.

$$NL_m(t_{int}) = PNL_0 + PNL_1 \, . \, DN(t_{int}) + ... + PNL_7 \, . \, DN(t_{int})^7 \qquad (4.12)$$
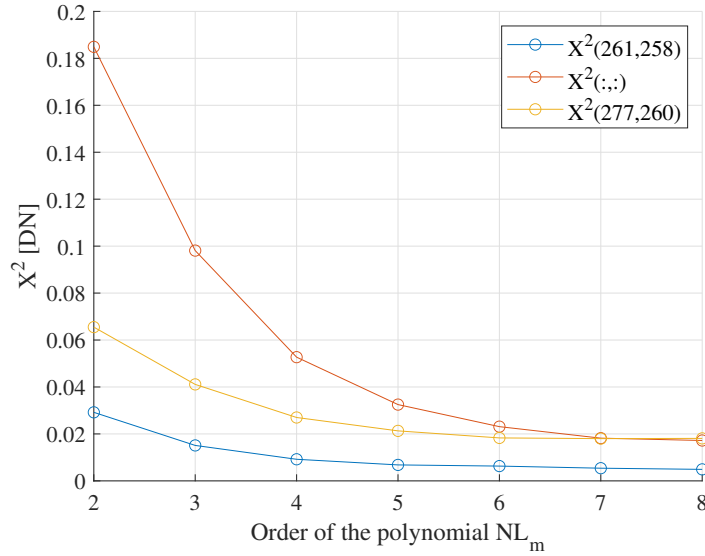
Figure 4.19: χ^2_{NL} in function of the order of the polynomial $NL_m(t_{int})$.
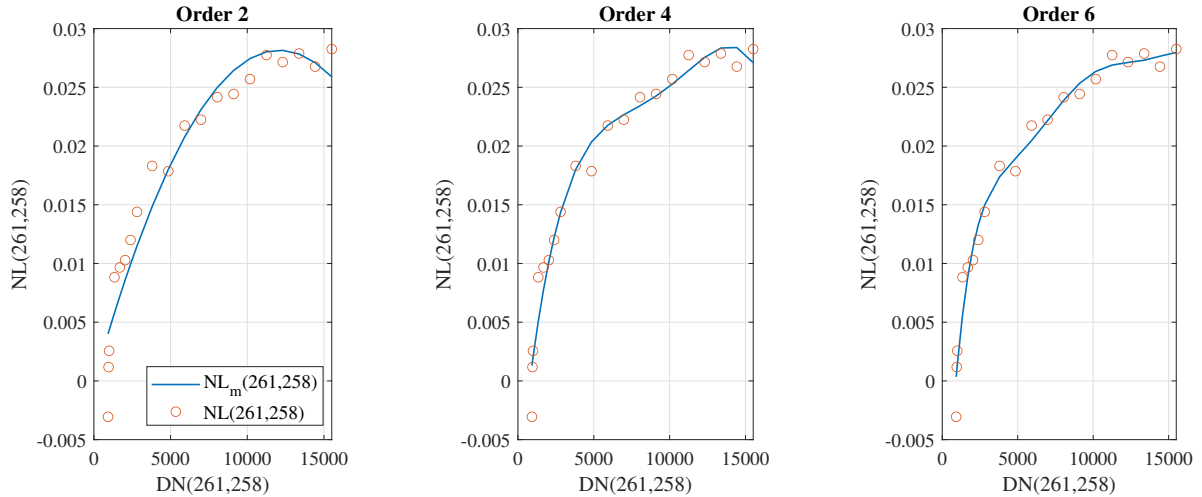
Figure 4.20: Representation of the non-linearity in [$\sim$] with respect to $DN(t_{int})$ in [DN] for the pixel (261,258). Three polynomials $NL_m(t_{int})$ with the order 2, 4 and 6 are represented.

Fig. 4.21 shows $NL_m(t_{int})$ of order 7 and $NL(t_{int})$ with respect to $DN(t_{int})$. Tab. 4.8 shows χ^2_{NL} for the pixel (261,258), (277,260) and the average of every pixels. As can be seen from the table, χ^2_{NL} of pixel (277,260) do not vary a lot from Sec. 4.4 however, most of the pixels and pixel (261,258) have a much better fit. This is again due to the fact that without the three last points the data fits much better to any polynomial.

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X^2_{NL} [$\sim$]	0.0054	0.018	0.0182

Table 4.8: This table shows the value of χ^2_{NL} for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{NL} for every illuminated pixel.
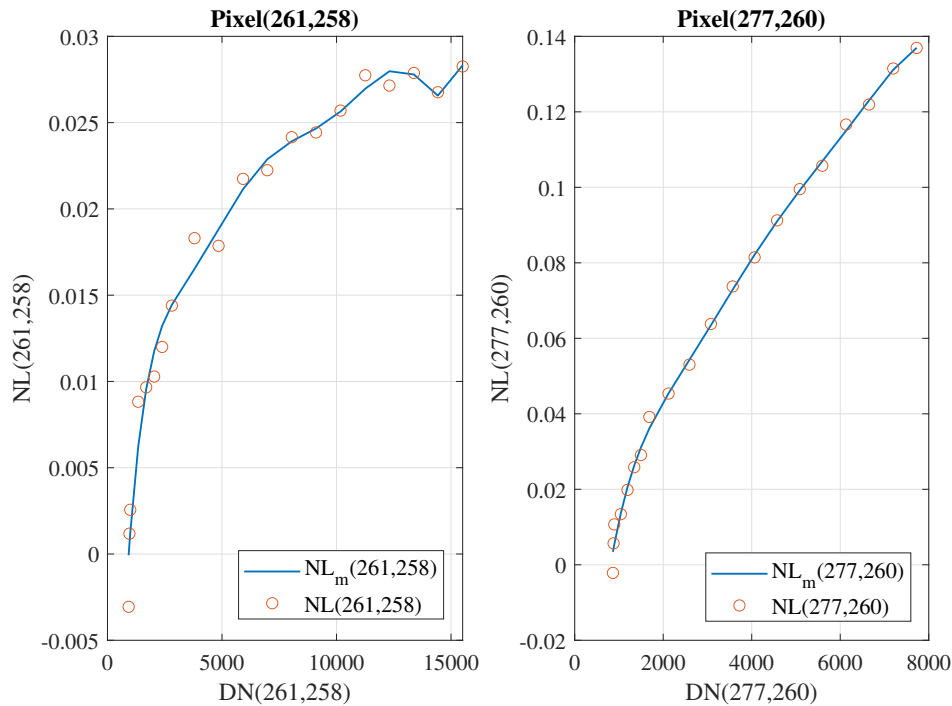
Figure 4.21: Representation of the non-linearity in [∼] with respect to $DN(t_{int})$ in [DN] for the pixels (261,258) and (277,260).

Finally, the outputs of algorithm 1 are found: DN_{0fit} and from PNL_0 to PNL_7. They will be used as inputs of the calibration model algorithm. This algorithm will be used to correct $DN(t_{int})$ and will return $DN_{corr}(t_{int})$. Fig. 4.22 shows the correction $DN_{corr}(t_{int})$, $DN(t_{int})$ and $DN_{rect}(t_{int})$ in function of the integration time. Compared to the previous section, the correction of $DN(t_{int})$ is much better. Tab. 4.5 shows χ^2_{Err} and the error in percentage for the pixel(261,258), (277,260) and the average of every pixels. The mean error in Sec. 4.4 was 0.24% and is now 0.056% which represents a net amelioration.

The calibration model algorithm is used again but this time for the correction of an image. Fig. 4.23 shows the initial image with an integration time of 0.004 s on the left and the corrected image on the right. The differences between both are barely visible. This is due to the fact that the non-linearity to be corrected was very small as the three points in the data $DN(t_{int})$ have been ignored.

Pixel	(261,258)	(277,260)	Mean for every illuminated pixel
X^2_{Err} [DN]	0.0314	0.0268	0.0541
Error [%]	0.044	0.047	0.056

Table 4.9: This table shows the value of χ^2_{Err} and the percentage of error for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{Err} and the percentage of error for every illuminated pixel.
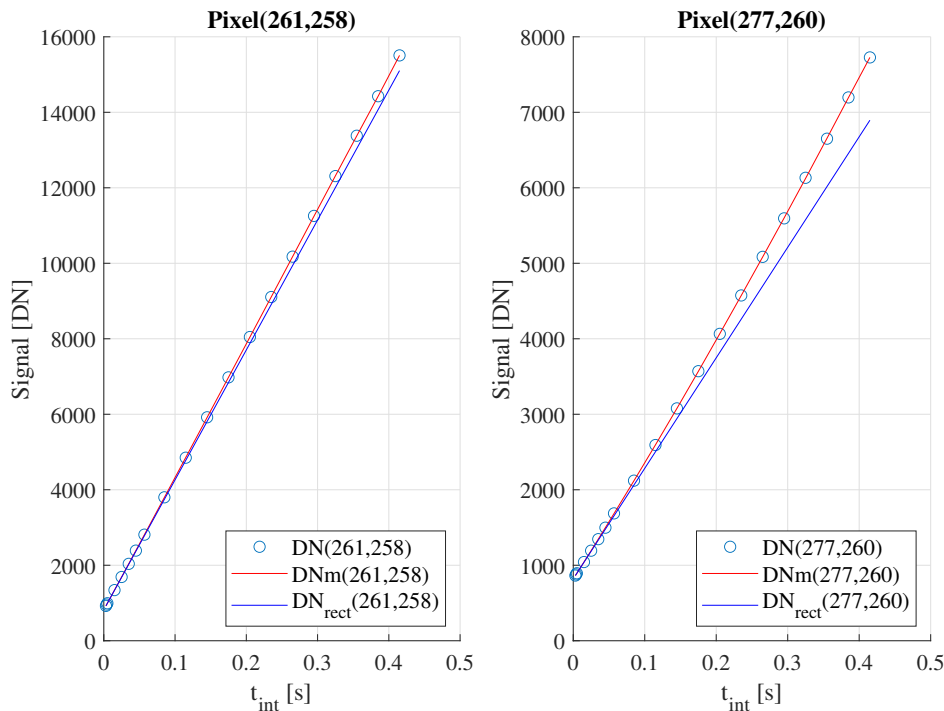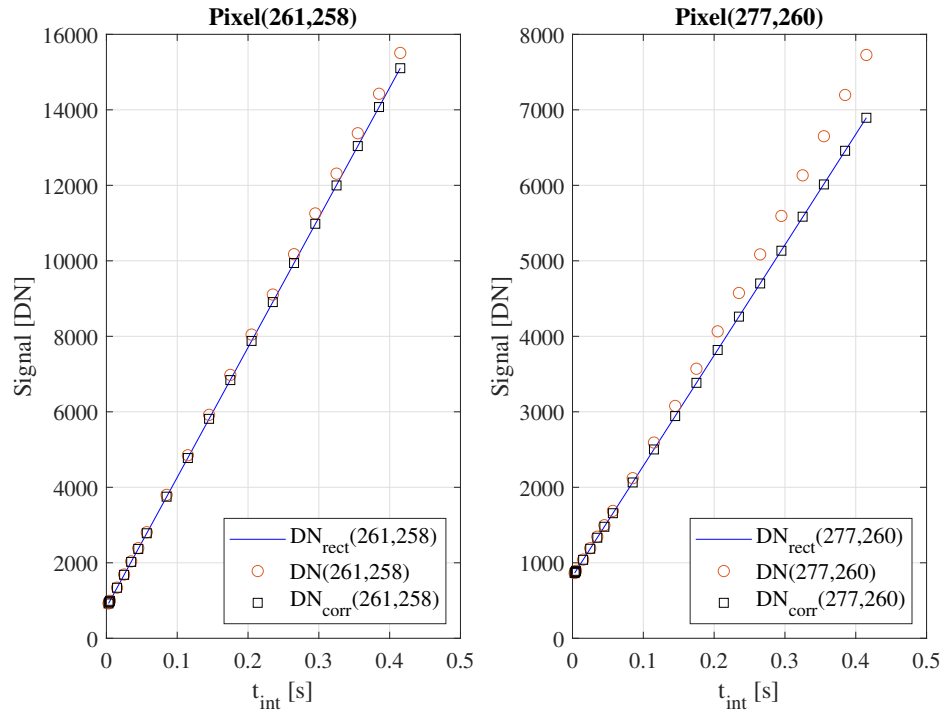
Figure 4.22: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260)
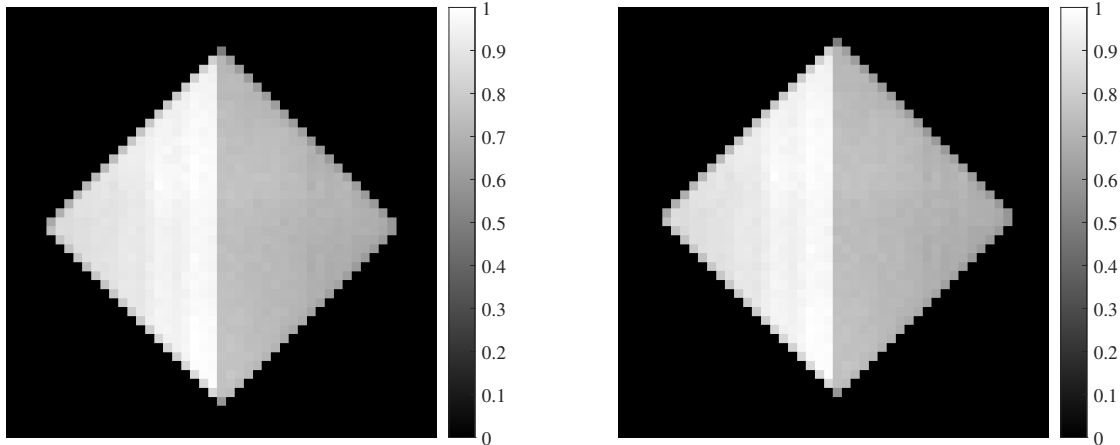
Figure 4.23: Difference between the initial image at the left and the corrected image at the right. The integration time of the image is 0.004s.

4.5.1   Discussion

By ignoring the three last points in the data $DN(t_{int})$, the mean value of χ^2_{DN} is almost 10 time smaller than the one computed in Sec. 4.4. This is the case even if the order of the polynomial $DN_m(t_{int})$ is smaller: the polynomial as an order of 8 here compared to 9 for Sec. 4.4. This is due to the fact that by ignoring the last three points, the shape of the data became simpler. Thus, the polynomial fits better the data. It can be seen that the accuracy of the fitting do not really change for the pixel in the transition part. In fact, as there were no saturated data, the fitting accuracy stay the same for both sections.

Concerning the mean error for all the pixels, it was 0.24% for Sec. 4.4 compared to 0.056% for this section which is a good amelioration.

4.6   Application of algorithm 2

In this section, the algorithm 2 will used with the images taken by the instrument 3MI. The only step varying from the algorithm 1 is the way of computing $DN_{rect}(t_{int})$. The first step is to determine t_{midd} for an average value of $DN(t_{int})$. For each pixel, the average value named $DN_{midd} = DN(t_{midd})$ corresponds to an integration time of : $t_{midd} = 0.175s$.

In this section all the points of $DN(t_{int})$ are considered and the order of the polynomial $DN_m(t_{int})$ will be taken to 9 and is represented in Eq. 4.13. In fact, the study has already been made in Sec. 4.4 and the best polynomial for fitting $DN(t_{int})$ has an order 9. Thus, the χ^2_{DN} for this polynomial are the same as the ones listed in Sec. 4.4.

$$DN_m(t_{int}) \; = \; DN_{0fit} \; + \; Pt_1 \, . \, t_{int} \; + \; ... \; + \; Pt_9 \, . \, t_{int}^9 \tag{4.13}$$

As explained in chapter 3, in order to compute $DN_{rect}(t_{int})$ we need the value of the slope and the intercept. As the value t_{midd} as been found, they can be computed for each pixels following the next equations:

$$Slope \; = \; Pt_1 \; + \; 2 \, . \, Pt_2 \, . \, t_{midd} \; + \; 3 \, . \, Pt_3 \, . \, t_{midd}^2 \; + \; ... + 9 \, . \, Pt_9 \, . \, t_{midd}^{9-1} \, . \tag{4.14}$$

$$p \; = \; DN_m(t_{midd}) \; - \; Slope \, . \, t_{midd} \tag{4.15}$$

$$DN_{rect} \; = \; Slope \, . \, t_{int} \; + \; p \tag{4.16}$$

Fig. 4.24 represents $DN(t_{int})$, $DN_m(t_{int})$ and $DN_{rect}(t_{int})$ for the pixels (261,258) and (277,260). As expected most of the non-linearity occurs for small and large integration time. This is due to the fact that the middle has been considered linear.

The next step is to find the best suited order for the polynomial $NL_m(t_{int})$. The order that suits the best the polynomial will be the order that minimizes χ^2_{NL}. Fig. 4.25 shows the variation of χ^2_{NL} with respect to the order of the polynomial $NL_m(t_{int})$. It can be seen that χ^2_{NL} of the pixel (277,260) is very unstable. The minimum value occurs for the order 11. The polynomial $NL_m(t_{int})$ can be written as Eq. 4.17. Fig. 4.26 shows how different order fits the data $NL(t_{int})$. The order 2, 4 and 8 are represented.

$$NL_m(t_{int}) \; = \; PNL_0 \; + \; PNL_1 \, . \, DN(t_{int}) \; + ... + \; PNL_{11} \, . \, DN(t_{int})^{11} \tag{4.17}$$

Figure 4.24: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260).
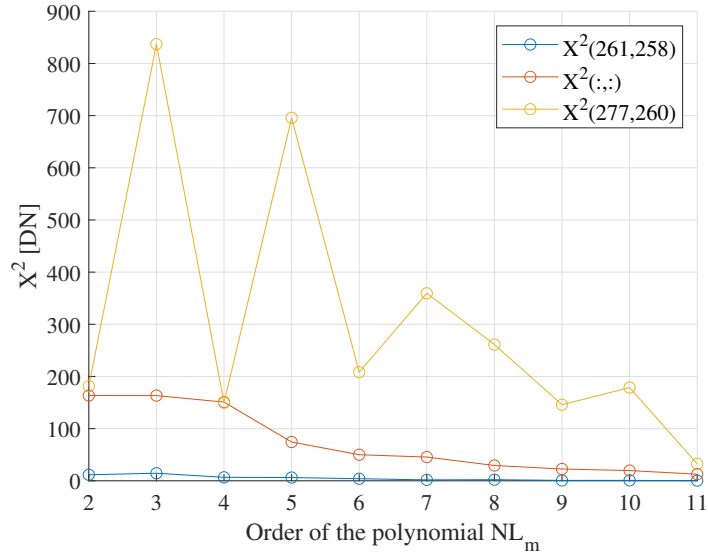
**Figure 4.25:** χ^2_{NL} in function of the order of the polynomial $NL_m(t_{int})$.
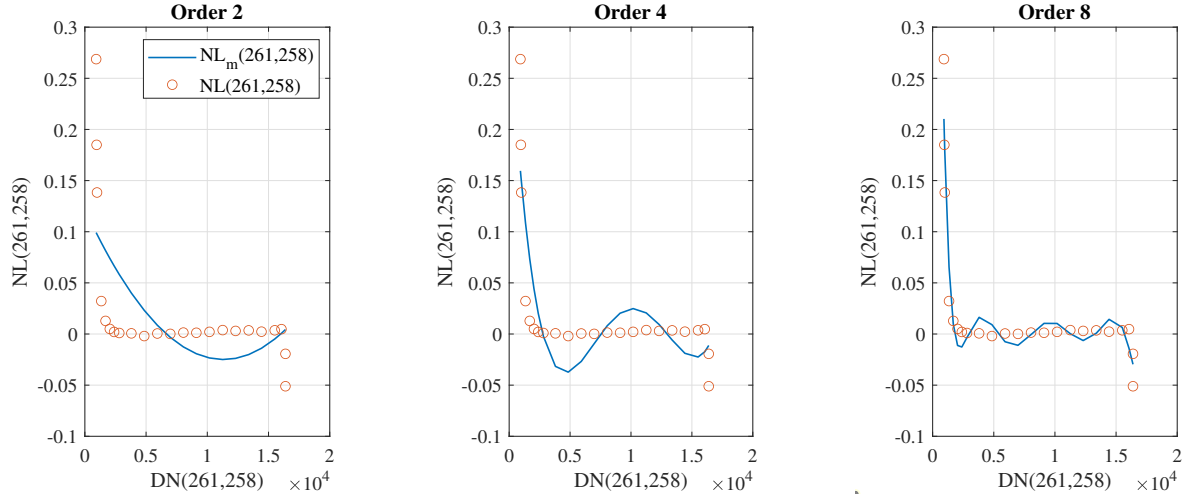
Figure 4.26: Representation of the non-linearity in [∼] with respect to $DN(t_{int})$ in [DN] for the pixel (261,258). Three polynomials $NL_m(t_{int})$ with the order 2, 4 and 8 are represented.

Fig. 4.27 shows $NL_m(t_{int})$ of order 11 and $NL(t_{int})$ with respect to $DN(t_{int})$. Tab. 4.10 shows χ^2_{NL} for the pixel (261,258), (277,260) and the average of every pixels. The values for most pixels and pixel (277,260) are abnormally big compared to the other sections.
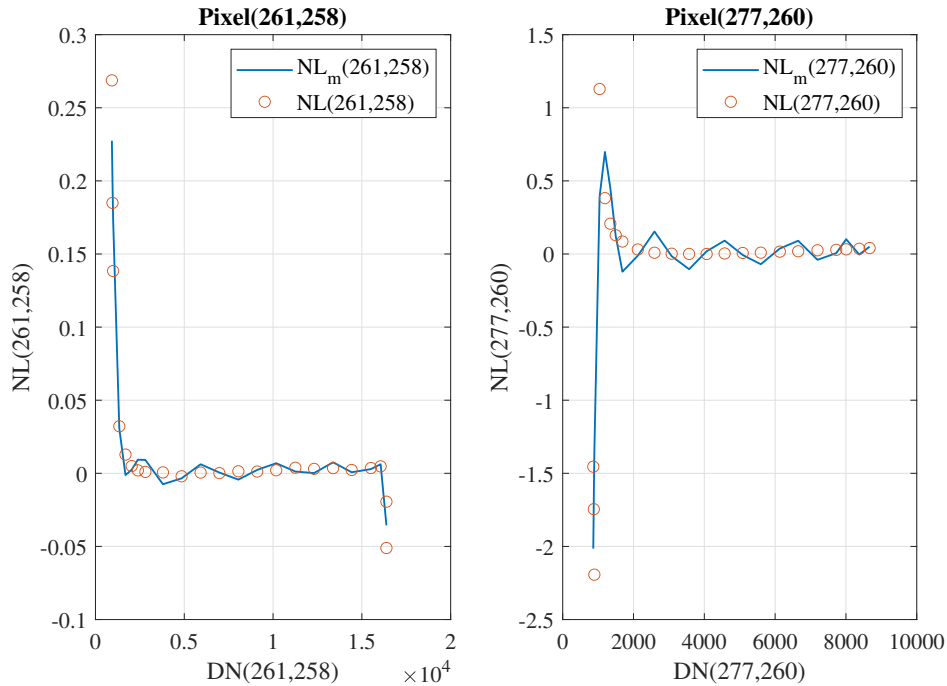
Figure 4.27: Representation of the non-linearity in [∼] with respect to $DN(t_{int})$ in [DN] for the pixels (261,258) and (277,260).

Pixel	(261,258)	(277,260)	Mean of every illuminated pixel
X^2_{NL} [∼]	0.3627	32.5042	12.6265

Table 4.10: This table shows the value of χ^2_{NL} for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{NL} for every illuminated pixel.

Finally, the outputs of algorithm 2 are found: DN_{0fit} and from PNL_0 to PNL_1. They will be used as inputs of the calibration model algorithm. This algorithm will be used to correct $DN(t_{int})$ and will return $DN_{corr}(t_{int})$. Fig. 4.28 shows the correction $DN_{corr}(t_{int})$, $DN(t_{int})$ and $DN_{rect}(t_{int})$ in function of the integration time. Tab. 4.11 shows χ^2_{Err} and the error in percentage for the pixel(261,258), (277,260) and the average of every pixel. The mean error is 0.98% which is larger than the error from Sec. 4.4 and Sec. 4.5.

Pixel	(261,258)	(277,260)	Mean for every illuminated pixel
X^2_{Err} [DN]	9.6804	250.0385	39.2650
Error [%]	0.42	5.16	0.98

Table 4.11: This table shows the value of χ^2_{Err} and the percentage of error for the pixels (261,258) and (277,260) as well as the mean value of χ^2_{Err} and the percentage of error for every illuminated pixel.
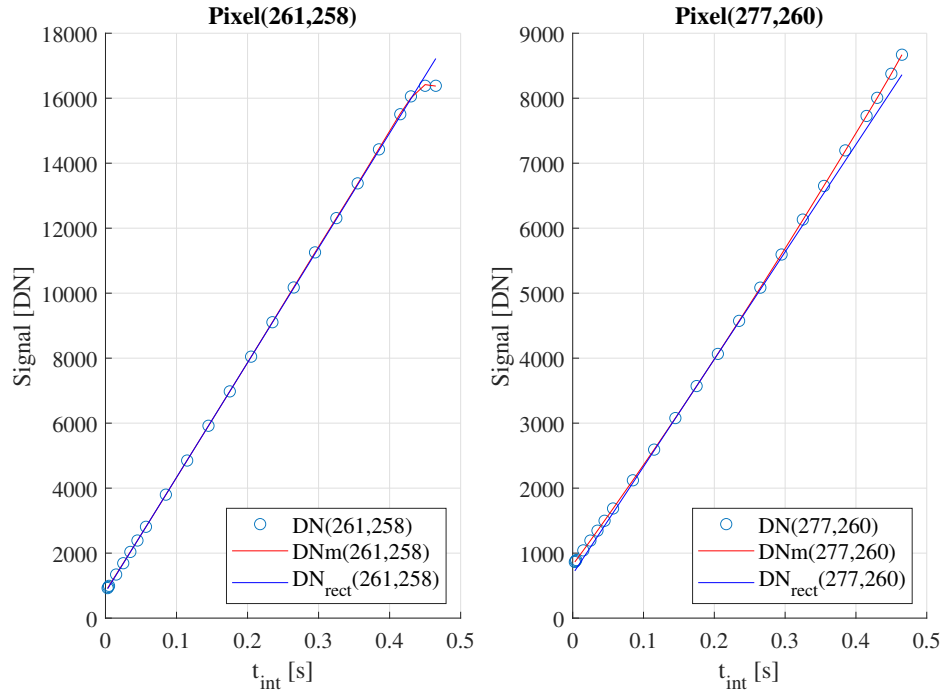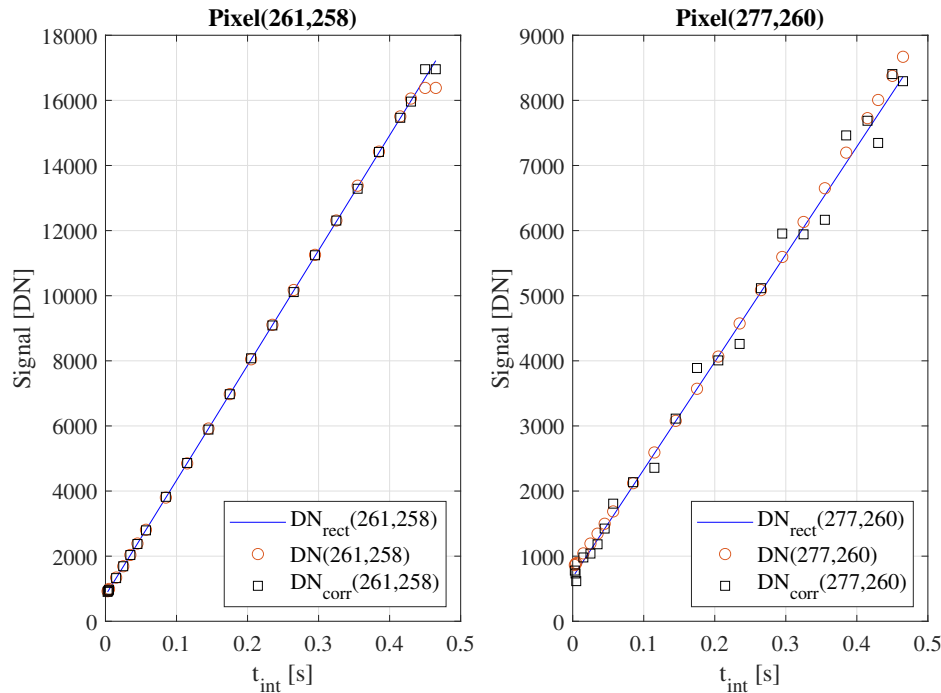

Figure 4.28: Representation of the signal in [DN] with respect to t_{int} in [s] for the pixels (261,258) and (277,260)

The calibration model algorithm is used again but this time for the correction of an image. Fig. 4.29 shows the initial image with an integration time of 0.004 s on the left and the corrected image on the right. The corrected image shows a very bad behavior. This correction is totally incorrect.
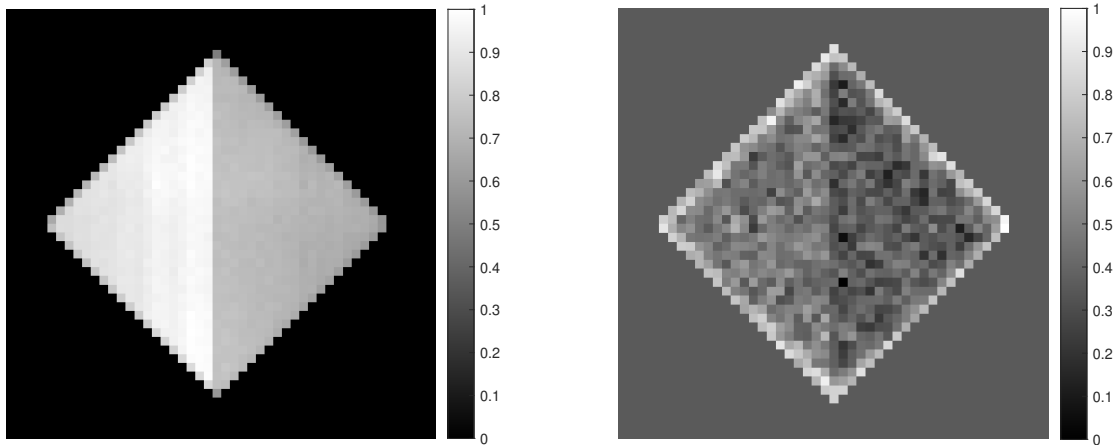
Figure 4.29: Difference between the initial image at the left and the corrected image at the right. The integration time of the image is 0.004s.

4.6.1 Discussion

Firstly, it can be seen in Fig. 4.29, that the corrected image is very odd. In fact, the non-illuminated pixels were fixed at 900 DN. They were always represented in black as 900 DN is supposed to represents the minimum value. However, in the corrected image presented here, some pixels in the illuminated region are even smaller than 900 DN. This is not correct as at an integration time of 0.004s the signal in the illuminated region is supposed to be around 950 DN. This means that the corrected image is not representative of the reality. Moreover, the mean error was computed at 0.98%, this seems very small compared to the errors in other sections. In fact, the other sections showed more appropriate corrected images. The error is computed with respect to $DN_{rect}(t_{int})$, and $DN_{rect}(t_{int})$ is computed very differently in this section than from the others. As the error is small and that the outcomes are very bad, this means that the algorithm 2 is a bad algorithm. The way of computing $DN_{rect}(t_{int})$ do not represents the reality.

Secondly, in Tab. 4.10, the χ^2_{NL} of pixel (277,260) was equal to 32.5042 which is very huge compared to the values obtained in the previous sections. This anomaly comes from the fact that at t_{midd}, $DN_m(t_{midd})$ and $DN(t_{midd})$ are not exactly equal. Theoretically the method supposes that $DN_m(t_{midd}) = DN(t_{midd})$, as it is not the case they are huge problems concerning the non-linearity computations.

In conclusion, the algorithm 2 is bad because the polynomial $DN_m(t_{int})$ can not fit the data at t_{midd} perfectly. Moreover, the way of computing DN_{rect} is incorrect because even with small errors, the corrected image shows odd behavior. The best method to apply for computing KDP related to non-linearity is the method described in Sec. 4.5 as it gave the best results.

This project presented the key concepts in the calibration of FLORIS instrument. FLORIS is the only payload of FLEX. This mission aims at measuring the fluorescence of the vegetation and representing it on global maps [1]. FLORIS, is an hyperspectral imager and is composed of two high spectral resolution spectrometers that covers a spectral range of 740nm - 780nm and 677nm-697nm. It is also composed of a low spectral resolution spectrometer that covers a spectral range of 500nm - 758nm [2]. As FLEX is only composed of one instrument, it needs additional data to interpret the fluorescence signal. Thus it will orbit the Earth in tandem formation with Sentinel-3 which will provide the necessary information. The company Leonardo has designed FLORIS and has charged CSL to calibrate it [9]. This work explained the essential steps in the calibration of FLORIS.

Firstly, key data parameters (KDPs) as well as the calibration philosophy was introduced. KDPs are parameters that characterize the instrument. These parameters are the ones that will need to be calibrated during the calibration campaign. In this project, only the KDP related to the non-linearity of the detector was computed. In this purpose, two algorithms have been introduced and compared. They only differ by the way of computing the linearized signal $DN_{rect}(t_{int})$. However, as the calibration of FLEX is a new project, the images from 3MI have been used to test the algorithms.

The first study used algorithm 1. The order for the polynomials $NL_m(t_{int})$ and $DN_m(t_{int})$ were taken at 1 and 2 respectively as suggested in the ATBD. It was demonstrated that these polynomials were not appropriate for their respective data $NL(t_{int})$ and $DN(t_{int})$. The second study tested algorithm 1 too but with larger orders. It was shown that this improved the results but at large integration times the data were not properly corrected. The third study tested the algorithm 1 but this time the data taken at large integration times were ignored because the signal was too small compared to what it should have been. This was due to saturation. The results from this study were the best, the corrected image was very close to the initial one. Finally, the last study tested algorithm 2. The results were very bad, even though the error in the correction was small.

In conclusion, study number 3 was the one which gave the best results. Moreover, the use of algorithm 2 must be avoided.

[1] European Space Agency, *FLEX Mission Overview*, `https://earth.esa.int/eogateway/missions/flex`, accessed 15 June 2021.

[2] Leonardo Electronics, *Floris instrument description summary*, FLEX earth explorer mission, Florence, 2019.

[3] Peter Coppo, Lucia Pettinato, Davide Nuzzi, *Instrument predevelopment activities for FLEX mission*, Optical Engineering, 2019.

[4] National Research Council, *Issues in the Integration of Research and Operational Satellite Systems for Climate Research: Part II. Implementation.*, The National Academies Press, Washington DC, 2000.

[5] Magellium, *3MI On-Ground Calibration, GPP Algorithm Theorical Baseline Document (ATBD)*, Toulouse, 2020.

[6] Leonardo Electronics, *FLEX Earth Explorer Mission, FLORIS L1 GPP ATBD*, Florence, 2020.

[7] Jose Lorenzo Alvarez, *Space Optical Systems Requirements definition*, ESA, Sardinia, 2017.

[8] Bernd Harnisch, *Space Optics Instrument Design and Technology, Spectrometer* , ESA, Sardinia, 2017.

[9] Centre Spatial de Liège, Leonardo, *Proposal FLEX On-Ground Calibration*, Liège, 2019.

[10] National Aeronautics and Space Administration, *Data Processing Levels*, `https://earthdata.nasa.gov/collaborate/open-data-services-and-software/data-information-policy/data-levels`, accessed 15 June 2021.

[11] European Space Agency, *FLEX and Sentinel-3 joining forces*, `https://sentinel.esa.int/web/sentinel/-/flex-and-sentinel-3-joining-forces/1.3?inheritRedirect=true&redirect=%2Fweb%2Fsentinel%2Fsearch%3Fq%3DFLEx`, accessed 15 June 2021.

[12] Wikipedia, *Imagerie hyperspectrale*, `https://fr.wikipedia.org/wiki/Imagerie_hyperspectrale`, accessed 15 June 2021.

[13] Wikipedia, *Integration sphere*, `https://en.wikipedia.org/wiki/Integrating_sphere`, accessed 15 June 2021.

[14] B. Hilbert, *Updated non-linearity calibration method for WFC3/IR*, July 2014.

[15] Sabins Jr, *Digital Image Processing*, `https://en.wikipedia.org/wiki/Integrating_sphere`, accessed 15 June 2021.

[16] European Space Agency, *Data Product Levels*, `https://www.esa.int/Applications/Observing_the_Earth/FLEX/Data_product_levels`, accessed 15 June 2021.

APPENDIX A

MATLAB FUNCTIONS AND SCRIPTS

A.1 Code for algorithm 1

 NL3_Code1.m

```matlab
%% Code for computing KDP related to non-linearity
clear all;
close all;

%% Part 1: Loading the images of 3MI
DN = load("HR1_DNm.mat");                    % Loading the input images
tint = 10^(-3)*[3,4,5,15,25,35,45,57,85,115,145,175,205,235,265,295,325....
    ,355,385,415,430,450,465];               % Integration time [s]
DN = DN.HR1_DNm ;                            % DN(t_int) [DN]

%% Part 2.1: Fit of DN, then extraction of the polynomial coefficients
% Initialization of the polynomial coefficients in the case where the
% order of the polynomial is 2.

data = zeros(length(tint),1);
DN0_fit = zeros(520,520);
Pt1 = zeros(520,520);
Pt2 = zeros(520,520);

for i=1:520
    for j=1:520
        data(:)= DN(i,j,:);
        ft = polyfit(tint(:),data(:),2);    % Polyfit of order 2
        DN0_fit(i,j) = ft(3);               % coefficient of t^0
        Pt1(i,j) = ft(2);                   % coefficient of t^1
        Pt2(i,j) = ft(1);                   % coefficient of t^2
    end
end

for i=1 : 1:520
    for j=1 :1:520
    % Construction of DNm for every pixel
    DNm(i,j,:) = DN0_fit(i,j) + Pt1(i,j).*tint(:) + Pt2(i,j).*tint(:).^2 ;
    end
end
```

```matlab
%% Part 2.2: Construction of NL and DNrect

% Initialization
DNrect = zeros(520,520,length(tint));
NL = zeros(520,520,length(tint));

for t=1:length(tint)
    for i=1:520
        for j=1:520
            % Construction of DNrect for every pixel
            DNrect(i,j,t) = DN0_fit(i,j) + Pt1(i,j)*tint(t);
            % Construction of NL for every pixel
            NL(i,j,t) = (DN(i,j,t)-DNrect(i,j,t))/(DNrect(i,j,t)- ....
                        DN0_fit(i,j));
        end
    end
end

%% Part 2.3: Fit of NL, then extraction of the polynomial coefficients

% Initialization of the polynomial coefficients in the case where the
% order of the polynomial is 1.

P_NL_1 = zeros(520,520);
P_NL_0 = zeros(520,520);
NLm = zeros(520,520,length(tint));

for i=1:520
    for j=1:520
        datax(:) = DN(i,j,:);
        datay(:) = NL(i,j,:);
        ft = polyfit(datax(:),datay(:),1) ;     % Polyfit of order 1
        P_NL_0(i,j) = ft(2)   ;                  % coefficient of t^0
        P_NL_1(i,j) = ft(1)   ;                  % coefficient of t^1

    end
end


for i=1:520
    for j=1:520
        % Construction of NLm
        NLm(i,j,:) = P_NL_0(i,j)+P_NL_1(i,j).*DN(i,j,:);
    end
end
```

A.2   Code for algorithm 2

NL3_Code2.m

```matlab
%% Code for computing KDP related to non-linearity
clear all;
close all;

%% Part 1: Loading the images of 3MI
DN = load("HR1_DNm.mat");                   % Loading the input images
tint = 10^(-3)*[3,4,5,15,25,35,45,57,85,115,145,175,205,235,265,295,325....
    ,355,385,415,430,450,465];              % Integration time [s]
DN = DN.HR1_DNm ;                           % DN(t_int) [DN]

%% Part 2.1: Fit of DN, then extraction of the polynomial coefficients
% Initialization of the polynomial coefficients in the case where the
% order of the polynomial is 2.

data = zeros(length(tint),1);
DN0_fit = zeros(520,520);
Pt1 = zeros(520,520);
Pt2 = zeros(520,520);

for i=1:520
    for j=1:520
        data(:)= DN(i,j,:);
        ft = polyfit(tint(:),data(:),2);    % Polyfit of order 2
        DN0_fit(i,j) = ft(3);               % coefficient of t^0
        Pt1(i,j) = ft(2);                   % coefficient of t^1
        Pt2(i,j) = ft(1);                   % coefficient of t^2
    end
end

for i=1 : 1:520
    for j=1 :1:520
    % Construction of DNm for every pixel
    DNm(i,j,:) = DN0_fit(i,j) + Pt1(i,j).*tint(:) + Pt2(i,j).*tint(:).^2 ;
    end
end

%% Part 2.2: Construction of NL and DNrect

% Initialization
DNrect = zeros(520,520,length(tint));
NL = zeros(520,520,length(tint));

for t=1:length(tint)
    for i=1:520
        for j=1:520
```

```matlab
            % Construction of DNrect for every pixel
            slope(i,j) = Pt1(i,j)+ 2*Pt2(i,j)*tint(12);
            intercept(i,j) = DN0_fit(i,j) + Pt1(i,j)*tint(12) + ....
                Pt2(i,j)*tint(12)^2 - pente(i,j) *tint(12);
            DNrect(i,j,:) = slope(i,j).*tint(:) + intercept(i,j);

            % Construction of NL for every pixel
            NL(i,j,t) = (DN(i,j,t)-DNrect(i,j,t))/(DNrect(i,j,t)- ....
                        DN0_fit(i,j));


        end
    end
end

%% Part 2.3: Fit of NL, then extraction of the polynomial coefficients

% Initialization of the polynomial coefficients in the case where the
% order of the polynomial is 1.

P_NL_1 = zeros(520,520);
P_NL_0 = zeros(520,520);
NLm = zeros(520,520,length(tint));

for i=1:520
    for j=1:520
        datax(:) = DN(i,j,:);
        datay(:) = NL(i,j,:);
        ft = polyfit(datax(:),datay(:),1) ;     % Polyfit of order 1
        P_NL_0(i,j) = ft(2)   ;                  % coefficient of t^0
        P_NL_1(i,j) = ft(1)   ;                  % coefficient of t^1

    end
end


for i=1:520
    for j=1:520
        % Construction of NLm
        NLm(i,j,:) = P_NL_0(i,j)+P_NL_1(i,j).*DN(i,j,:);
    end
end
```

A.3   CMOD for the correction of non-linearity

NL3_CMOD.m

```matlab
%% Calibration model for the correction of the non-linearity


for i=1:520
    for j=1:520
        DN_corr(i,j,:)=(DN(i,j,:)-DN0_fit(i,j))./....
            (NL_m(i,j,:)+1)+DN0_fit(i,j);
    end
end
```

A.4 Code for computing the mean of each acquisitions for every pixels

NL3_Mean.m

```matlab
%% Opens NETCDF files and computes the mean of the acquisitions for every
%% integration time and for every pixel
clear all;
close all;

tint = 10^(-3)*[3,4,5,15,25,35,45,57,85,115,145,175,205,235,265,295,325....
    ,355,385,415,430,450,465];    % Integration time in [s]

% Initialization
DN = zeros(520,520,length(tint));

%% Time integration 3
Nacq3 = 1000;
for i=1:Nacq3
    filename = strcat('meas_TINT_0003.0_',num2str(i-1),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_3(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,1) = mean(Imread_3(i,j,:));
    end
end

%% Time integration 4
Nacq4 = 609;
for i=1:Nacq4
    filename = strcat('meas_TINT_0004.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_4(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
```

```matlab
        DN(i,j,2) = mean(Imread_4(i,j,:));
    end
end
%% Time integration 5
Nacq5 = 390;
for i=1:Nacq5
    filename = strcat('meas_TINT_0005.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_5(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,3) = mean(Imread_5(i,j,:));
    end
end
%% Time integration 15
Nacq15 = 43;
for i=1:Nacq15
    filename = strcat('meas_TINT_0015.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_15(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,4) = mean(Imread_15(i,j,:));
    end
end
%% Time integration 25
Nacq25 = 16;
for i=1:Nacq25
    filename = strcat('meas_TINT_0025.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_25(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,5) = mean(Imread_25(i,j,:));
    end
end
%% Time integration 35
Nacq35 = 8;
for i=1:Nacq35
    filename = strcat('meas_TINT_0035.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_35(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
```

```matlab
        DN(i,j,6) = mean(Imread_35(i,j,:));
    end
end
%% Time integration 45
Nacq45 = 5;
for i=1:Nacq45
    filename = strcat('meas_TINT_0045.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_45(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,7) = mean(Imread_45(i,j,:));
    end
end
%% Time integration 57
Nacq57 = 3;
for i=1:Nacq57
    filename = strcat('meas_TINT_0057.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_57(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,8) = mean(Imread_57(i,j,:));
    end
end
%% Time integration 85
Nacq85 = 3;
for i=1:Nacq85
    filename = strcat('meas_TINT_0085.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_85(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,9) = mean(Imread_85(i,j,:));
    end
end
%% Time integration 115
Nacq115 = 3;
for i=1:Nacq115
    filename = strcat('meas_TINT_0115.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_115(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
```

```matlab
        DN(i,j,10) = mean(Imread_115(i,j,:));
    end
end
%% Time integration 145
Nacq145 = 3;
for i=1:Nacq145
    filename = strcat('meas_TINT_0145.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_145(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,11) = mean(Imread_145(i,j,:));
    end
end
%% Time integration 175
Nacq175 = 3;
for i=1:Nacq175
    filename = strcat('meas_TINT_0175.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_175(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,12) = mean(Imread_175(i,j,:));
    end
end
%% Time integration 205
Nacq205 = 3;
for i=1:Nacq205
    filename = strcat('meas_TINT_0205.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_205(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,13) = mean(Imread_205(i,j,:));
    end
end
%% Time integration 235
Nacq235 = 3;
for i=1:Nacq235
    filename = strcat('meas_TINT_0235.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_235(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
```

```matlab
        DN(i,j,14) = mean(Imread_235(i,j,:));
    end
end


%% Time integration 265
Nacq265 = 3;
for i=1:Nacq265
    filename = strcat('meas_TINT_0265.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_265(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,15) = mean(Imread_265(i,j,:));
    end
end
%% Time integration 295
Nacq295 = 2;
for i=1:Nacq295
    filename = strcat('meas_TINT_0295.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_295(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,16) = mean(Imread_295(i,j,:));
    end
end
%% Time integration 325
Nacq325 = 2;
for i=1:Nacq325
    filename = strcat('meas_TINT_0325.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_325(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,17) = mean(Imread_325(i,j,:));
    end
end
%% Time integration 355
Nacq355 = 2;
for i=1:Nacq355
    filename = strcat('meas_TINT_0355.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_355(:,:,i)= Image ;
end
for i=1:520
```

```matlab
    for j=1:520
        DN(i,j,18) = mean(Imread_355(i,j,:));
    end
end
%% Time integration 385
Nacq385 = 2;
for i=1:Nacq385
    filename = strcat('meas_TINT_0385.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_385(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,19) = mean(Imread_385(i,j,:));
    end
end
%% Time integration 415
Nacq415 = 2;
for i=1:Nacq415
    filename = strcat('meas_TINT_0415.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_415(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,20) = mean(Imread_415(i,j,:));
    end
end
%% Time integration 430
Nacq430 = 2;
for i=1:Nacq430
    filename = strcat('meas_TINT_0430.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_430(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,21) = mean(Imread_430(i,j,:));
    end
end
%% Time integration 450
Nacq450 = 2;
for i=1:Nacq450
    filename = strcat('meas_TINT_0450.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_450(:,:,i)= Image ;
end
for i=1:520
```

```matlab
    for j=1:520
        DN(i,j,22) = mean(Imread_450(i,j,:));
    end
end
%% Time integration 465
Nacq465 = 2;
for i=1:Nacq465
    filename = strcat('meas_TINT_0465.0_',num2str(i),'.nc');
    Image = ncread( filename, 'NON_LINEARITY_CALIB/VNIR/MEASUREMENT');
    Imread_465(:,:,i)= Image ;
end
for i=1:520
    for j=1:520
        DN(i,j,23) = mean(Imread_465(i,j,:));
    end
end
```