Kjell Henriksen Observatory

## Internship Report
# Silver Bullet Calibration


| | |
|---|---|
| **Name** | Raphael Deirmendjian |
| **Date** | August 2022 |
| **Supervisor** | Noora Partamies |

## Contents

## List of acronyms

**KHO** *Kjell Henriksen Observatory*

**BG** Background calibration scan (lamp OFF)

**ABS** Absolute calibration scan (lamp ON)

**CAL** Wavelength calibration file

**NIST** *National Institute of Standards and Technology*

# 1 Abstract

This report gives an update on the stability of the 1 m "silver" Ebert-Fastie spectrometer (located at the Kjell Henriksen Observatory) used to retrieve mesospheric temperature from the OH* (6-2) airglow. The analysis is done by analysing data from the calibration conducted each year. The last analysis covered the 2007-2013 period. Data from 2011 and 2013 have been analysed to check the method.

The analysis showed a bigger wavelength drift from 2021, which is linked to the motor change that happened during the summer of 2020, but this is not deteriorating the measurements. The analysis led to the conclusion that the spectrometer has been operating stably during the studied period. Analysis of all-sky camera images taken during the previous calibration led to an experimental protocol to ensure quality calibration in the incoming years.

This report provides a script as well as instructions for the next analysis to be done automatically.

## 2 Introduction

The instrument focused on in this report is a spectrometer used to observe a night sky phenomenon: airglow. This section gives an overview of the phenomenon itself and presents the instrument.

### 2.1 Airglow

Airglow is a phenomenon describing a spontaneous emission. Unlike auroras, it is not restricted to high latitudes. We can observe airglow anywhere on earth. However, the airglow within the visible light is most of the time not bright enough to be observed with the naked eye. When excited atoms and molecules in the middle and upper atmosphere go to a lower energetic level, they emit light, producing airglow. The wavelength and intensity depend on the energetic level and the particle.
The airglow we are interested in is the hydroxyl (OH*) airglow, producing intense infrared light at 87 km. Figure 1 shows a typical OH* spectrum. From measured airglow intensity, it is possible to retrieve temperatures of the mesopause region.

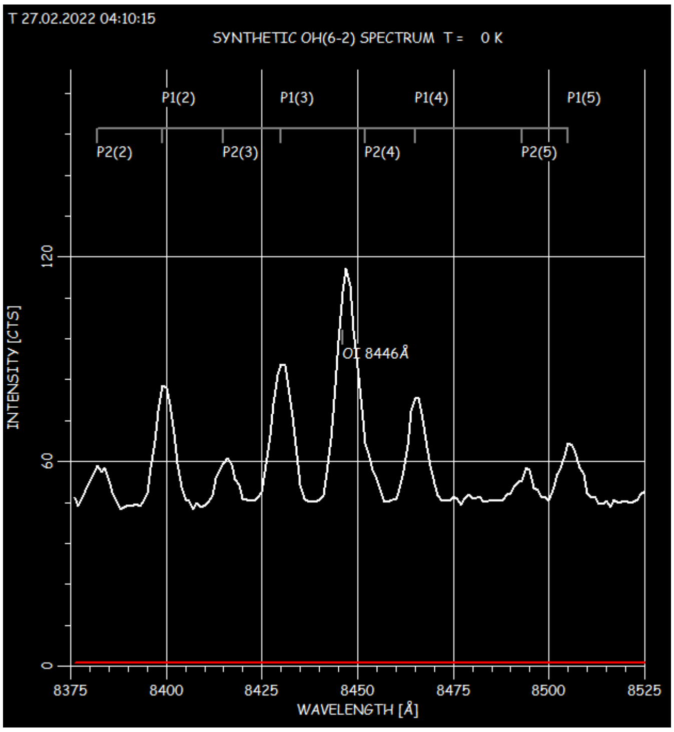



Figure 1: 27.02.2022 OH* spectrum measured by the Silver Bullet

Temperatures are derived from the fit of a linear fit to a Boltzmann plot, looking at the $P_1(2)$, $P_1(3)$, $P_1(4)$, $P_1(5)$ rotational line intensities of the OH*(6-2) vibrational bands [1].

## 2.2 The Silver Bullet spectrometer

The silver bullet is an Ebert-Fastie spectrometer that provided OH* airglow measurements in the polar region since 1983. It was first located at the Auroral Station in Adventdalen (6 m.a.s., 78.202 N, 15.829 E), but in 2007, it was moved to the newly built Kjell Henriksen Observatory (KHO) (520 m.a.s., 78.148 N, 16.043 E). Like other optical instruments, it is working during the dark season, approximately from early November to late February.

The spectrometer has a 1 m focal length and is able to scan between 7250 Å and 8650 Å. This range allows covering the P branch of the OH*(6-2) vibrational band. It is looking straight at zenith and has a 5 degrees field of view. One scan takes approximately 25 s.

Figure 2: The 1m Ebert-Fastie spectrometer, "Silver Bullet"

# 3 Calibration process

The Silver Bullet is used for long-term analysis. Therefore, it is fundamental to check its stability regularly, through calibration. Indeed, because the spectrometer is regularly updated (new components, maintenance), it is fundamental to check that it is still reacting the same way, in order to compare the results through the years.

## 3.1 General method and instrumentation

During the calibration, the spectrometer measures the spectra of a calibration lamp. The measured spectrum is then compared to the known spectrum of the lamp. The yearly calibrations done between 2008 and 2022 are secondary calibrations. That means that the calibration has been done with a 200 W tungsten lamp, certified by comparing it to the NIST (National Institute of Standards and Technology) lamp from the lab at UNIS. A primary calibration would be a calibration done with the traceable lamp.
The 200 W lamp certificate data is shown in table 4 and represented in figure 3.

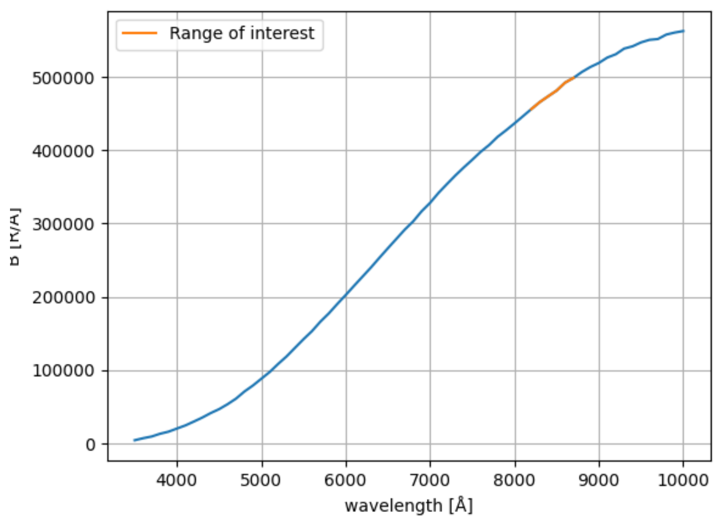



Figure 3: B 200 W lamp data from certificate

For the purpose of the spectrometer, we will only be interested in the range shown in orange in figure 3.
The lamp is not directly the source. A Lambertian diffusive screen (SRT-99-180, Spectralon®, Labsphere Inc.) screen is used as a source to ensure the slit of the

spectrometer is uniformly enlightened.
The brightness of the screen is given by:

$$B(\lambda) = \rho(\lambda) M_0(\lambda) \left(\frac{z_0}{z}\right)^2 \cos\alpha \tag{1}$$

with :

- $\rho(\lambda)$ the reflectance factor of the screen. Within the wavelength range of interest, it is considered as constant and we take $\rho \approx 0.98$.
- $M_0(\lambda)$ is the known brightness of the lamp, obtained at distance $z_0$. This value is found on the lamp's certificate.
- $z$ is the distance from the screen to the lamp.
- $\alpha$ is the angle between the screen and the lamp's optical axis.

The calibration factor of the instrument is given by :

$$K(\lambda) = \frac{B(\lambda)}{C(\lambda)} \tag{2}$$

With $C(\lambda)$ the raw count of the spectrometer. The final objective of the calibration is to give an expression of $K(\lambda)$. Then, we can get the absolute brightness from the raw count.

### 3.2 Calibration protocol: experiment

The calibration setup is shown in figure 4.

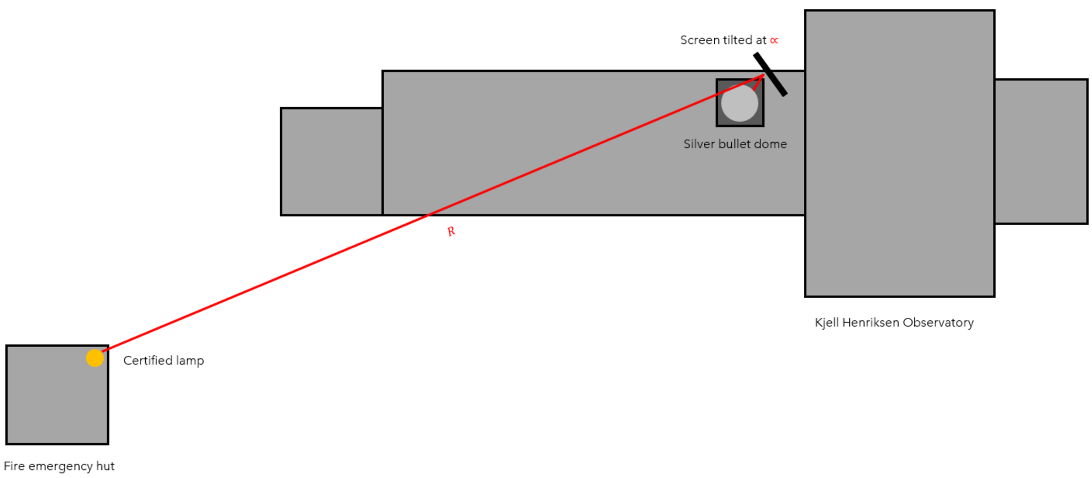



Figure 4: Calibration setup. Only the Silver Bullet dome is shown, the figure is not to scale.

The lamp illuminates the screen, which is acting as a source for the spectrometer. The calibration is done in 2 steps. First, the lamp is turned on and a certain amount of scans of the lamp is performed, giving the *Absolute calibration* data. Then, the lamp is turned off and a certain amount of dark scans is performed, giving the *Background* data.

### 3.3 Calibration protocol: data processing

This subsection briefly describes the process to get the calibration factor from the raw data from the spectrometer.

- Average the measurement for the calibration and the background for each wavelength over the scans.
- Adjust the wavelength to cancel the wavelength drift. The instrument is set up to scan within a pre-defined range. However, there is a wavelength drift between the set-up range and the measured range. This is done within the *Synthetic OH* software [2] using the *wavelength calibration* module.
- Subtract the background measurement from the calibration measurement.
- For each wavelength, get B from the interpolated plot from the certificate
- Calculate the calibration factor

The whole process is summarized in figure 5.

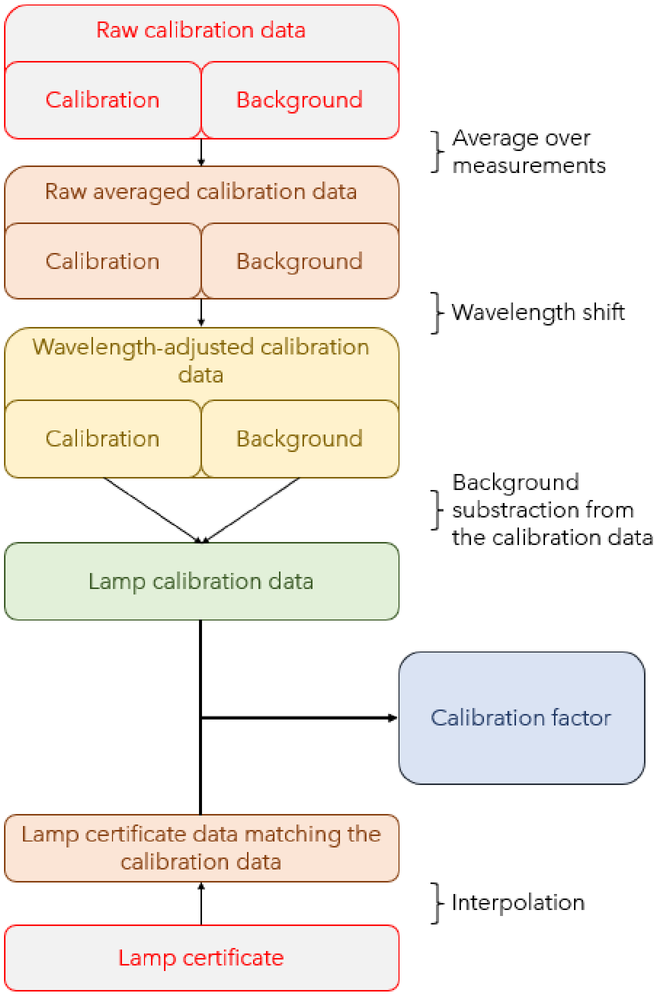



Figure 5: Calibration process: data analysis

### 3.4 Long-term calibration data analysis

The silver bullet has been used for more than 40 years. Within this time, it has been constantly upgraded. Some components have been changed. Since the instrument has been active for quite a long time, long-term trend analysis is possible and very valuable for scientists. Thus, it is necessary for the long-term trend analysis to check that the instrument has been operating stably during these years.

# 4 Overview of the previous work

## 4.1 The early days

The first calibration has been done in 1980, and the second one in 2002. The first calibration data analysis stated that the spectrometer has been operating stably during this period (Sigernes et al, 2003 [3]). Another calibration has been done in 2004 and the same conclusion has been drawn (Dyrland and Sigernes, 2007 [3]).

## 4.2 2007 to 2013

In 2007, the spectrometer moved from the auroral station in Adventdalen and an indoor calibration has been done, using a 45 W lamp. Since 2008, an outdoor calibration has been done yearly at the KHO, using a 200 W lamp.
The calibration factor against the wavelength plot is shown in figure 6.

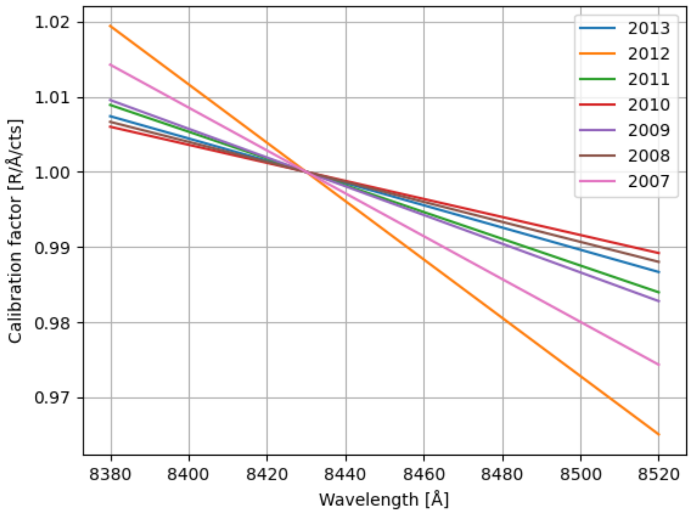



Figure 6: Calibration factor versus wavelength from 2007 to 2013. Plot obtained from the processed data available on the KHO server.

The conclusion that the spectrometer has been operating stably from 2007 to 2013 (without any significant impact of the change of location of the spectrometer) has been drawn (Holmen et al, 2014 [5]).

# 5 Calibration Data Analysis

The work presented in this report is focused on the 2014 to 2022 period. Therefore, an updated analysis of the 9 last calibrations is provided. This analysis will allow giving a conclusion regarding the instrument stability over the 2014-2022 period. An analysis of the 2011 and 2013 data is also done in this report to compare the method with the last analysis done (Holmen et al, 2014 [5]).
The data have been processed with Python. Therefore, the stability analysis will easily be performed in the next years. It will only be necessary to provide some inputs to the Python script. The inputs are the BG scan, the ABS scan, the CAL file, and the distance and angle. The script can be found in appendix A.4. In appendix A.3, a tutorial describes the steps to perform this analysis in the future.

## 5.1 Wavelength drift correction

The spectrometer is in theory set up to scan the 8285–8706 Å range. However, in real life, there is a wavelength drift. Thus, we also need a wavelength calibration.

To wavelength calibrate the spectrometer, we use a "good" spectrum measured by the instrument at a date close to the calibration. Then, we compare the peaks of the measured spectrum to the known values of each peak of the HO*(6-2) band of airglow. This is done each season as it is essential when it comes to temperature calculation; it is called wavelength calibration. After the wavelength calibration, we can get a calibration file from the software, named *Wavelength.cal*. For this calibration data analysis, the *.Cal* files have been used as an input to get the real wavelength from the measured wavelength.

The wavelength drift for 2013 is shown in figure 7 as an example.

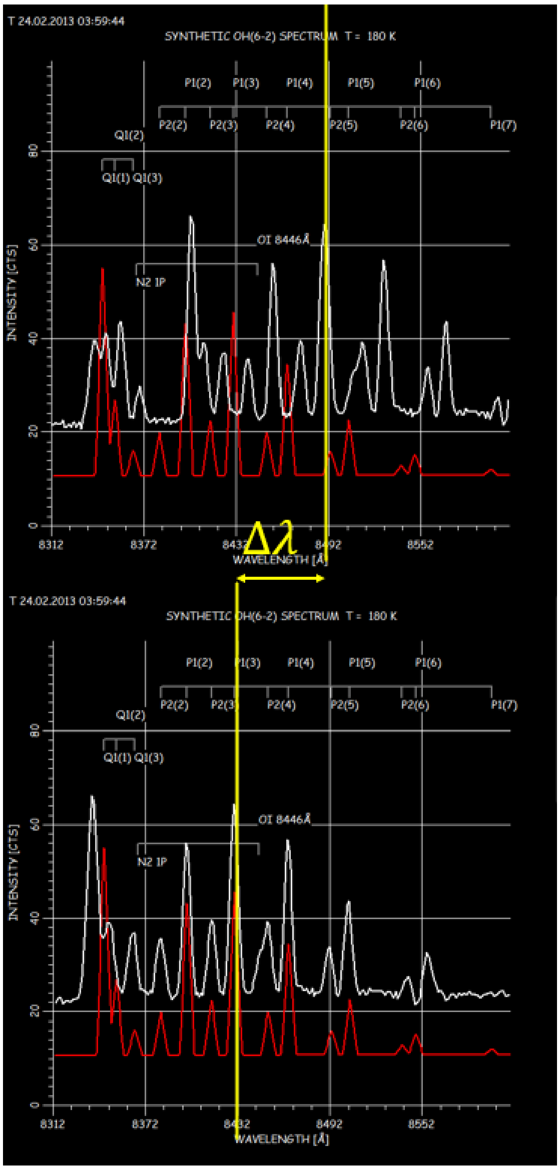



Figure 7: Wavelength drift from 2013, shown in the *SyntheticOH* software. The white curve corresponds to the measured spectrum, before adjusting the wavelength (up) and after (down).

**<u>Structure of a *.Cal* file:</u>**

A *.Cal* file is a text file containing four rows. On the first row is the number of recorded points (381). In the other lines, the numbers are polynomial coefficients $x_0$, $x_1$, $x_2$ and $x_3$.

Let $i \in [0, 380]$. $i = 0$ corresponds to the theoretical starting wavelength (8285 ) and $i = 380$ corresponds to the theoretical stopping wavelength (8706 ). We have :

$$\lambda_{real} = x_0 + x_1 i + x_2 i^2 + x_3 i^3 \quad (3)$$

Figure 8 shows the difference between the measured wavelength and the actual wavelength, and figure 9 shows the wavelength drift in the 2011 to 2022 period.

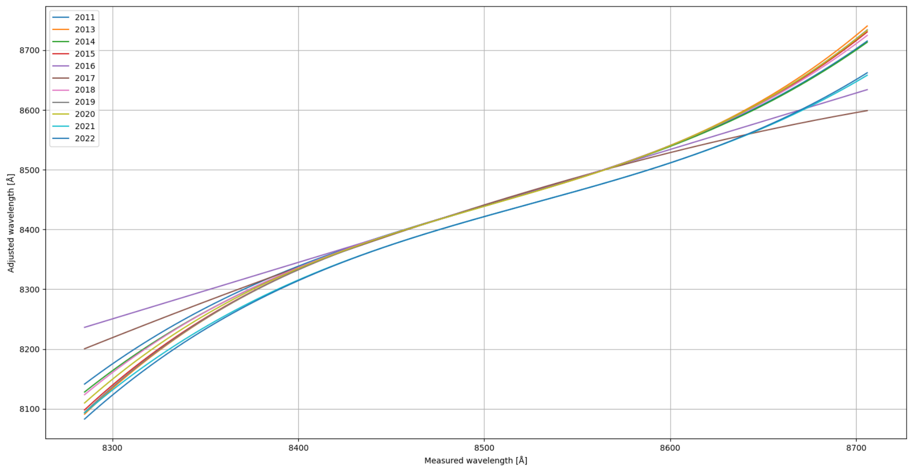



Figure 8: Measured wavelength VS real wavelength

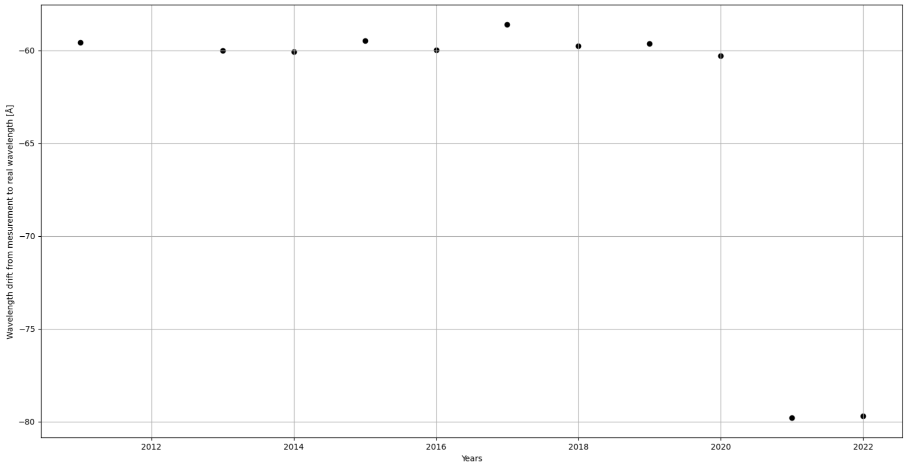



Figure 9: Wavelength drift at 8430 Å

In figure 8, we can see that between 8300 and 8500 Å (real wavelength), the curves are almost straight lines, meaning that the wavelength difference within this range is linear. Figure 9, shows a stable offset from 2013 to 2020, corresponding to approximately −60 Å. From 2021, the drift increased a lot, to approximately −80 Å. During the summer of 2020, the motor of the silver bullet has been changed [6]. This might be the explanation for the wavelength drift difference observed.

### 5.2 Interpolation from the lamp Certificate

The lamp's certificate gives a discrete amount of values. We have a value every 100 Å. To have an accurate result it is thus necessary to interpolate some data. Linear regression is used over the 8200-8700 Å range. This wavelength range has been chosen in order to be consistent with the last analysis done. The linear regression is shown in figure 10.

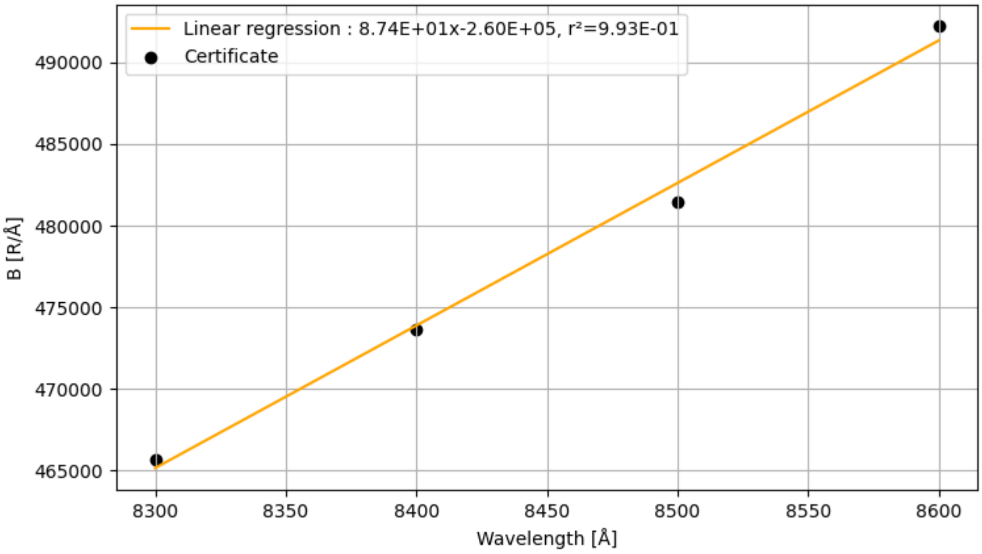



Figure 10: Interpolated data (orange curve) from the lamp's certificate (black dots)

## 5.3 Absolute calibration

The purpose of the calibration data analysis is to tell how accurate the measurements are, and how steady the instrument is. The first thing that has been done was to plot the calibration factor versus the wavelength, accordingly to the method developed in figure 5. In order to be consistent with the last analysis, the 8374-8525 Å range has been chosen.

### 5.3.1 First result

The result for the analysed years is shown in figure 11

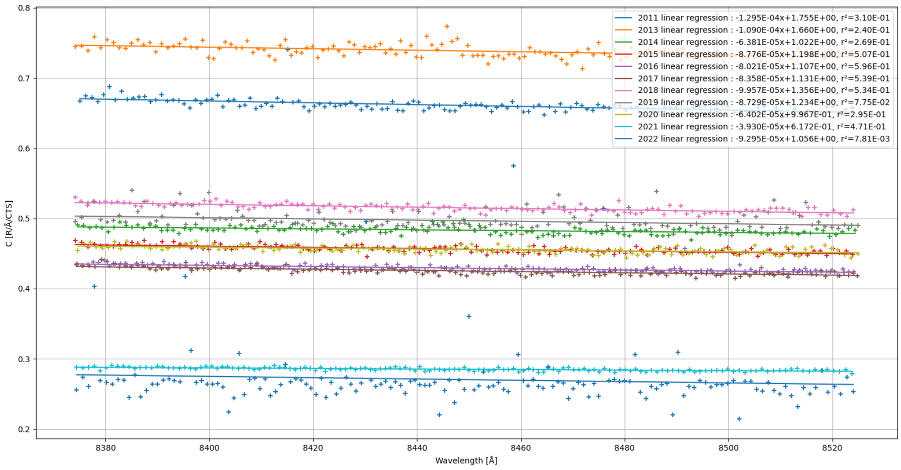



Figure 11: Raw calibration factor vs wavelength plot

A lot of values seem wrong in this plot. Looking back at the raw data showed that some values were not making sense. Some scans showed sporadic values that are more than three standard deviations away from the mean. For this reason, an outlier treatment has been chosen.

#### 5.3.2 Outliers treatment

Deleting an outlier is not a scientifically approved method in many cases. Deleting the "wrong" data points would cause a loss of information. It is important to remind that the data are averaged over the number of scans. For this reason, if there is one scan which shows an outlier, it will distort the whole data set. For this reason, it is acceptable to delete the outliers, before averaging the scans. This way, only the problematic values are lost. The outliers can be caused by many different things, including unwanted light pollution. Indeed, since the calibration is conducted outside, we get light pollution from Longyearbyen for instance.

#### 5.3.3 Outliers detection

For each wavelength, an outlier detection has been performed. Then, based on a criterion, the few biased values have been dropped. Peirce criterion ([7], [8]) has been chosen to detect and delete the outliers. This criterion has been chosen because it is likely that there is more than one outlier per scan.
The calibration factor after the outlier deletion is shown in figure 12. This result will be the one retained for the rest of the analysis.

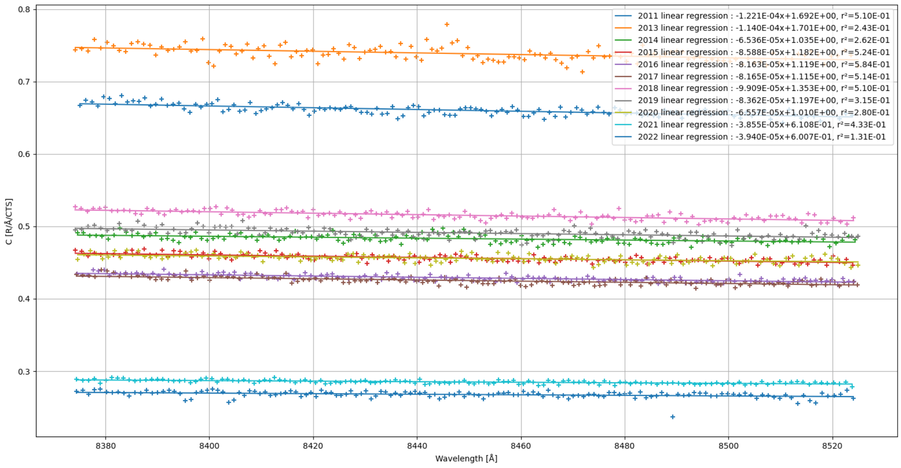



Figure 12: Calibration factor vs wavelength plot, after outlier deletion

The outliers deleted are mostly more than 2 standard deviations away from the mean. Between 0 and 4 values are dropped per wavelength, and less than 4% of the values are dropped. The goal of this outlier treatment is to eliminate light pollution. However, when too many outliers are deleted, then the conditions were not good for calibration and proceeding to another calibration should be the right decision.

### 5.3.4 Comparison with 2011 and 2013

In order to state the validity of this method, the results have been compared to the previous analysis, both for 2011 and 2013. 2012 has been dropped because of a lack of reliable calibration data this year. The last analysis has been done using degrees instead of radians. For this reason, in this section (and only in this section), the analysis has been done using degrees. This is the reason why the results are different from the one exposed in figure 12. The slopes have been compared, between this analysis and the previous one. The result is summed up in table 1.

Table 1: Comparison with the previous analysis for 2011 and 2013

| **Year** | **2011** | **2013** |
|---|---|---|
| **Slope (last analysis)** | $-1.778e-4$ | $-1.485e-4$ |
| **Slope** | $-1.698e-4$ | $-1.392e-4$ |
| **Relative gap (%)** | 5.0 | 6.2 |

The differences between the two analyses are small. Looking back at the data, the difference is mainly due to the outliers treatments. No information has been found in the last analysis about outliers. It seems that some values have been dropped when absurd. For this reason, this analysis might be more accurate, since fewer data has been lost. The overall small differences between the last analysis and this one validate the methods.

#### 5.3.5 Weather and sky condition influence

In figure 12, the $r^2$ value is quite variable from one year to another. In 2022 the data looks particularly distorted.
This section gives an overview of the weather conditions during the last calibrations and their influence on the quality of the calibration. This analysis is based on all-sky camera images. The all-sky camera is located at KHO.

In figure 13, 14 and 15, the pictures are showing the difference with the lamp ON (left) and OFF (right). Thus, we can see the difference between the absolute calibration measurement and the background measurement. It is clear that in 2022 the difference between the background measurement (15d) and the absolute calibration (15c) measurement is very little. This is due to the moon which was up in the sky, with a 79 % moon phase.

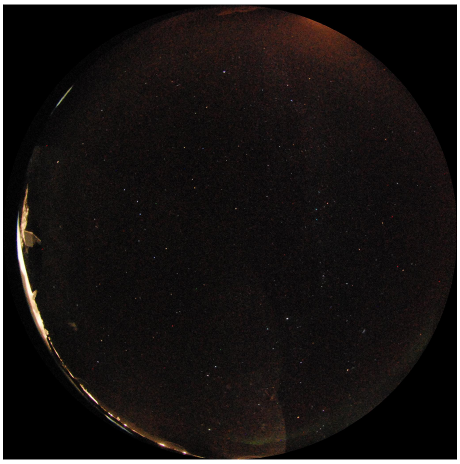
Figure 17 shows the absolute calibration and the background measurement. Due to auroras, the is a big variability in the measurement.

Figure 16 shows the variability of the intensity of the source seen by the all-sky camera.
Depending on the sky condition, the calibration quality can change. A poor quality calibration is shown by a low $r^2$ value, and by a high standard deviation in the measurements. The following characteristics are related to poor calibration.

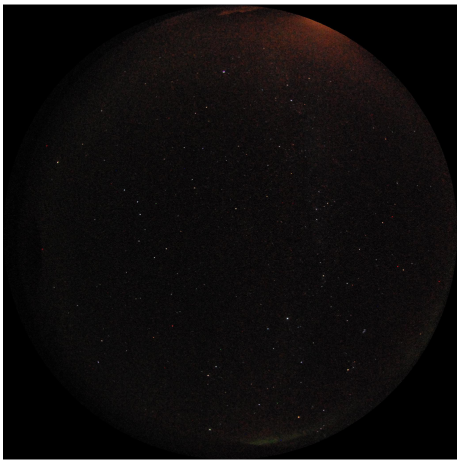
- Absolute calibration and background measurements close to each other, making the SNR poor.
- Variable background measurement
- Variable Absolute calibration

To ensure a good calibration, the following conditions need to be avoided.

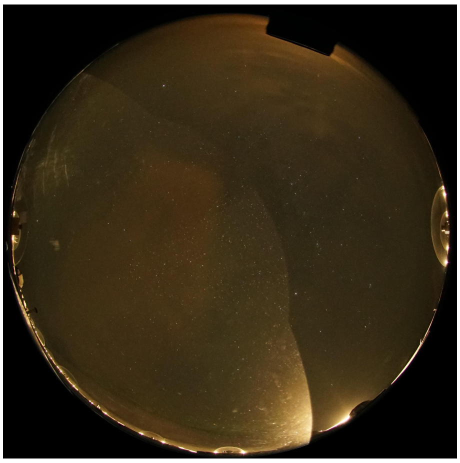
- Moonlight
- Moving clouds with bright background (light pollution, moonlight, auroras)



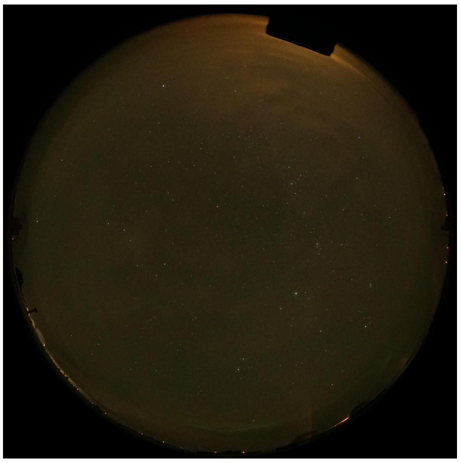
(a) 2011, ABS



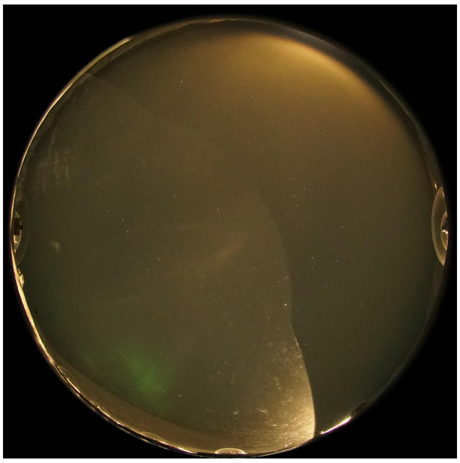
(b) 2011, BG



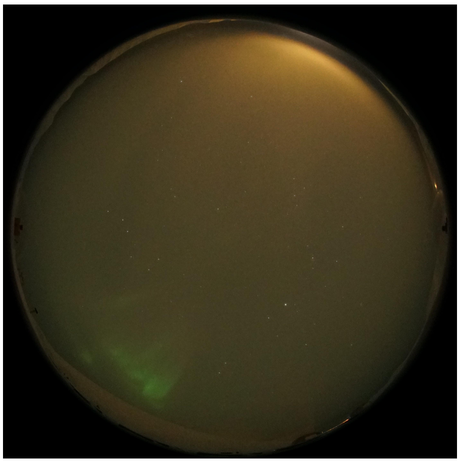
(c) 2016, ABS



(d) 2016, BG



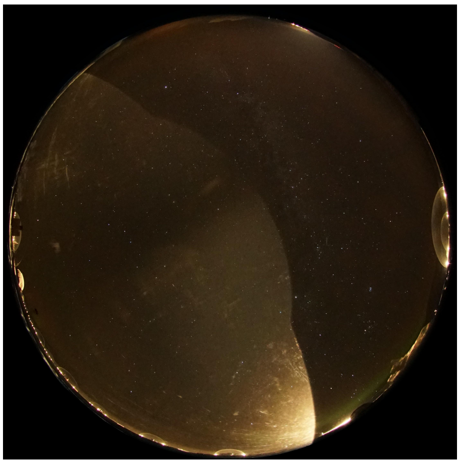
(e) 2017, ABS



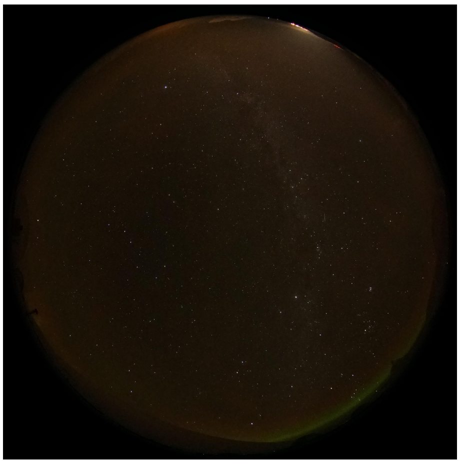
(f) 2017, BG



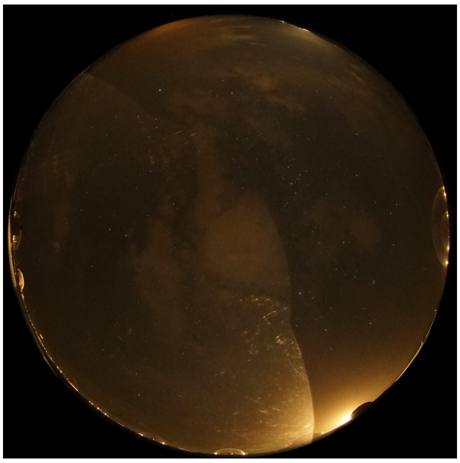
Figure 13: All-sky camera: lamp ON (left) and OFF (right) (1/3)



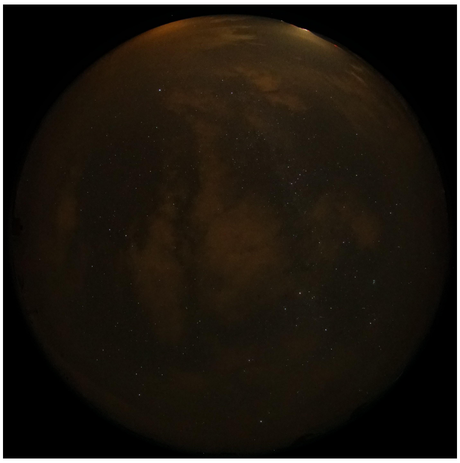
(a) 2018, ABS



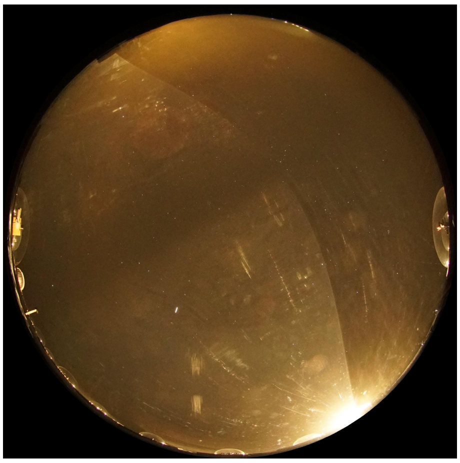
(b) 2018, BG



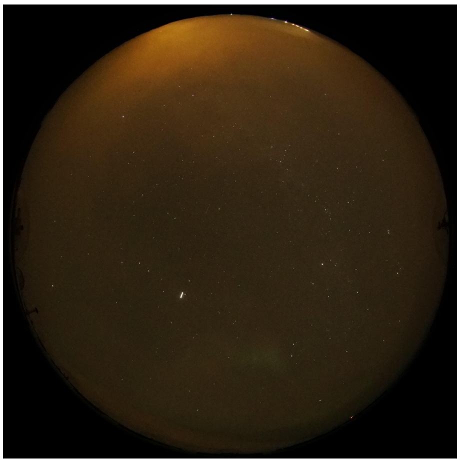
(c) 2019, ABS



(d) 2019, BG



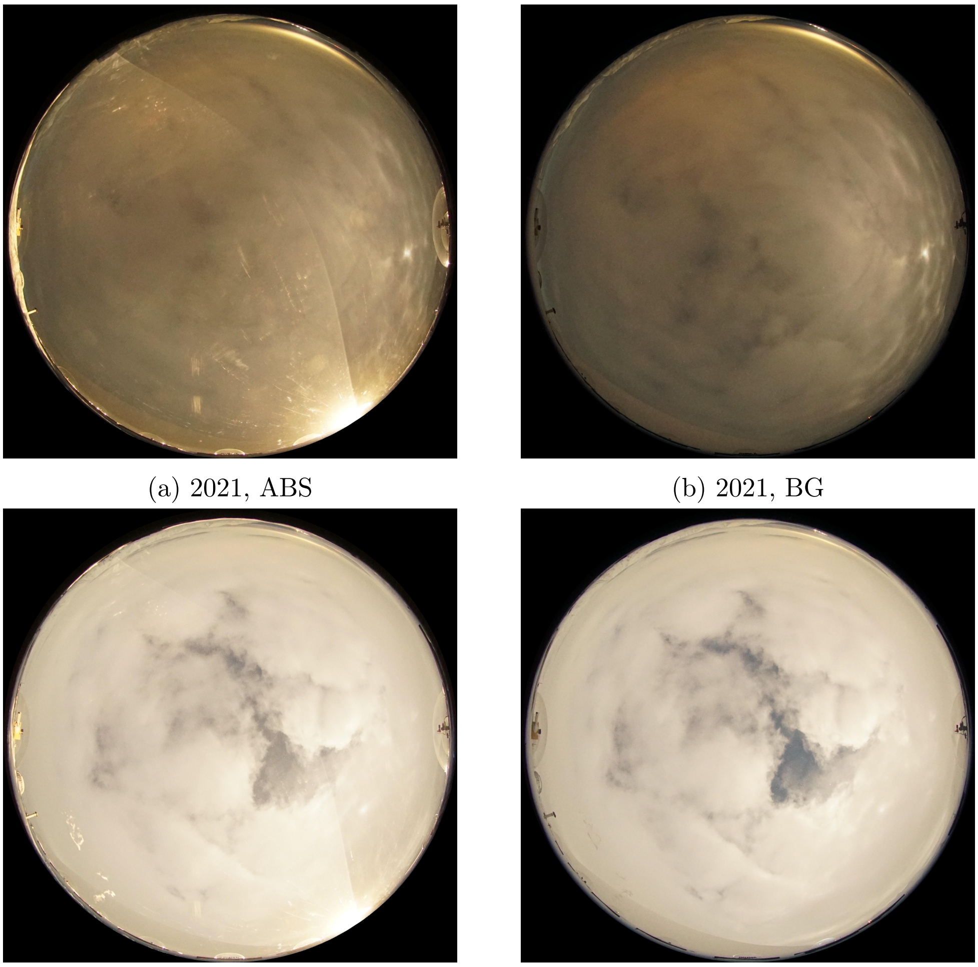
(e) 2020, ABS



(f) 2020, BG



Figure 14: All-sky camera: lamp ON (left) and OFF (right) (2/3)



(a) 2021, ABS

(b) 2021, BG

(c) 2022, ABS

(d) 2022, BG

Figure 15: All-sky camera: lamp ON (left) and OFF (right) (3/3)

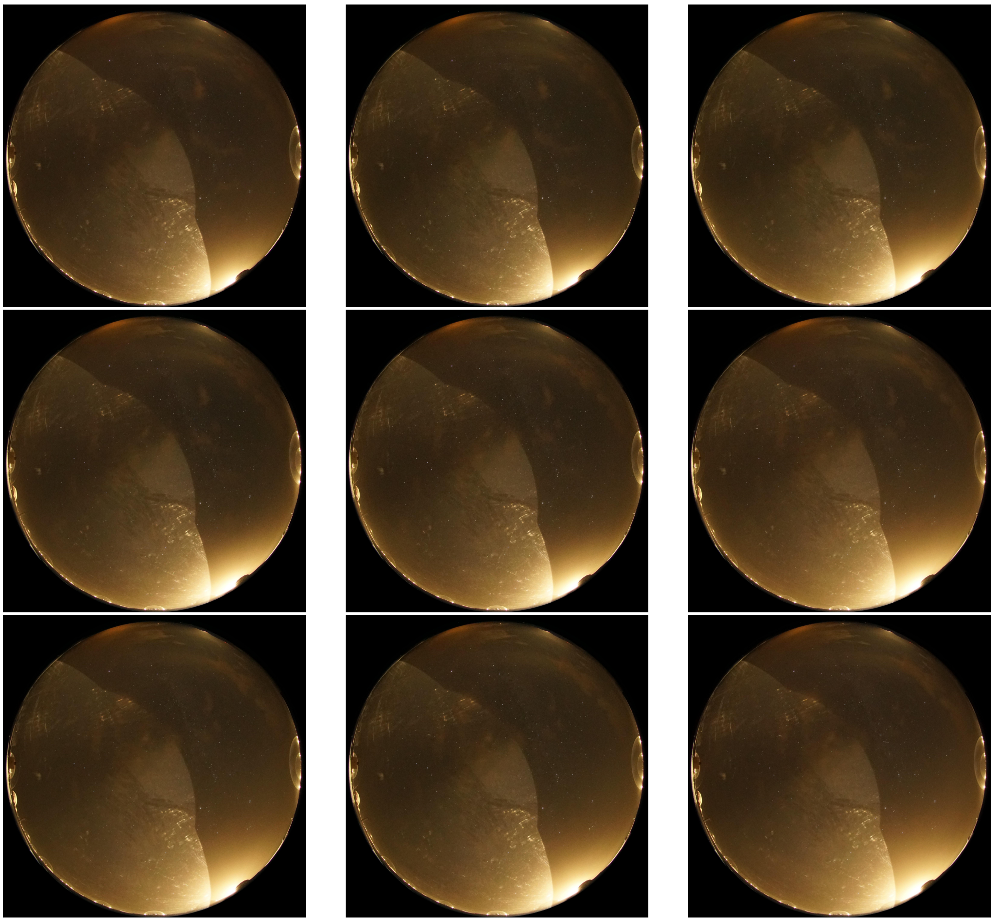


Figure 16: 2019 ABS

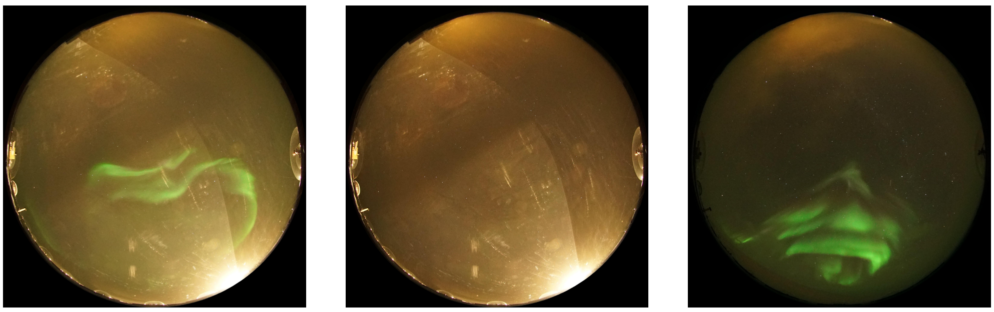


Figure 17: 2020 calibration, ABS and BG

- Auroras
- High humidity or snow/ice crystals

It is recommended to check the all-sky camera before proceeding to the calibration. The following experimental protocol is recommended:

1. Choose a time with the less moonlight as possible, the fewer clouds as possible, and the less wind as possible. The presence of ice crystals in the air should be avoided.
2. Before turning on the lamp, look at the all-sky camera for the last 10 minutes. If there are changes in the sky (auroras, clouds moving in front of the moon...), postpone the calibration.
3. Turn on the lamp and wait 5 minutes.
4. Look at the all-sky camera. If there is no big change in the all-sky camera from one picture to another, proceed to the calibration. Otherwise, postpone the calibration

### 5.4 Influence of the setup measurement uncertainties on calibration factor

The calibration factor is a function of 2 measurements, conducted each year during the calibration: $\alpha$ and $z$. Since these values are measured, it is necessary to know the influence of their uncertainty. From the equations 1 and 2, we have :

$$K(\lambda) = \frac{\rho(\lambda) M_0(\lambda) \left(\frac{z_0}{z}\right)^2 \cos\alpha}{C(\lambda)} \tag{4}$$

Since there is an uncertainty only on the $\alpha$ and on the $z$, the uncertainty $\Delta K$ on the calibration factor $K$ can be expressed as a function of the uncertainties $\Delta\alpha$ and $\Delta z$ on $\alpha$ and $z$ as shown in equation 5 (derivation in appendix A.2).

$$\frac{\Delta K}{K} = \frac{\tan(\alpha)}{\alpha}\Delta\alpha + 2\frac{\Delta z}{z} \tag{5}$$

As a first estimate, the error on the angle measurement can be taken at 1° since it is read on a protractor and the error on the distance measurement can be taken at 1 cm. Using the values from 2022 calibration leads to a relative gap of less than 2%. This is minor and would not have a major impact on the result.

# 6 Conclusions

## 6.1 Wavelength drift

The instrument has shown an increased wavelength drift since the step motor has been replaced. Since wavelength calibration is performed during temperature calculation, this does not influence the measurement. The drift is considered linear within the range of interest for the OH*(6-2) emission band. However, if we were interested in another wavelength range, the linear drift approximation would not be acceptable.

## 6.2 Sensitivity stability

The calibration factor slopes have an impact on temperature calculation. Table 2 shows the conclusion on the temperature uncertainty depending on the spectral close of the calibration factor. It is important to mention that the slopes can not be directly compared to the current analysis. Indeed, the last analysis used degrees instead of radians. However, this did not impact the final stability conclusion.

Table 2: Temperature uncertainty and spectral slope from the last analysis [5] (using degrees)

| **Year** | **Slope** | **Temperature Uncertainty** |
|---|---|---|
| 2007 | -2.85e-4 | 1-3 |
| 2008 | -1.33e-4 | 1-2 |
| 2009 | -1.91e-4 | 1-2 |
| 2010 | -1.20e-4 | 1-2 |
| 2011 | -1.78e-4 | 2-4 |
| 2012 | -3.88e-4 | 3-6 |
| 2013 | -1.48e-4 | 1-3 |

Figure 18 shows the calibration factor plot, normalized at 8430 Å, and table 3 shows the spectral slope for the 2014-2022 period. It is including 2011 and 2013, and these 2 years show the steepest slope. Table 1 shows that the current analysis is similar to the last one, which validated the method. Since 2011 has the steepest slope, it is possible to conclude that the temperature uncertainty within the 2014-2022 period is smallest than 2-4 K.

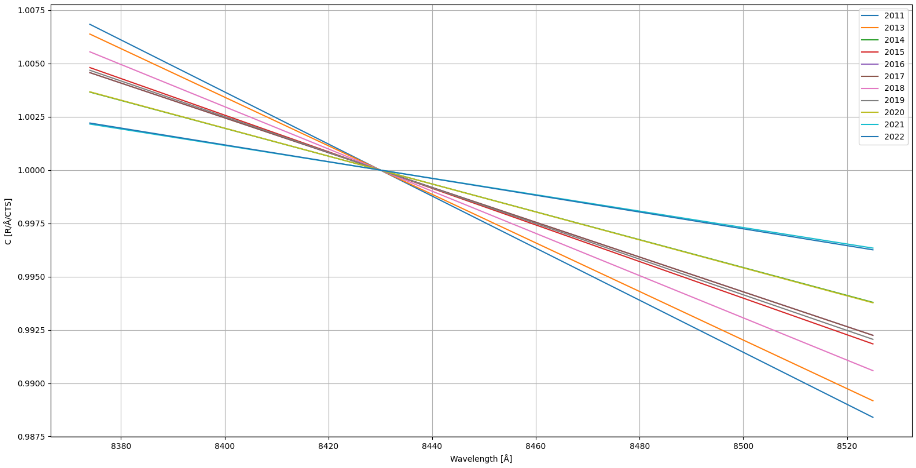



Figure 18: Calibration factor plot, normalized at 8430 Å

Table 3: Spectral slope for the 2014-2022 period (using radians)

| **Year** | **Slope** |
|---|---|
| 2014 | -6.536e-5 |
| 2015 | -8.588e-5 |
| 2016 | -8.163e-5 |
| 2017 | -8.165e-5 |
| 2018 | -9.909e-5 |
| 2019 | -8.362e-5 |
| 2020 | -6.557e-5 |
| 2021 | -3.855e-5 |
| 2022 | -3.940e-5 |

Relating the slopes found to the previous shows that the instrument has been operating stably between 2014 and 2022. However, some calibrations can not be trusted. 2022 was a particularly bad calibration, due to weather conditions.

### 6.3 Calibration quality assessment

As mentioned before, some calibrations are not reliable. A few tools can help to discriminate a good calibration from a bad calibration. The residual to a linear fit of the calibration factor is a good indicator. A very spread-out data-set is also showing that calibration conditions were not optimal. Looking at the standard deviation of the scans is a good way to assess how spread out the values are. Finally, looking at the signal-to-noise ratio is a good starting point to assess the quality of the calibration.

# A Appendix

## A.1 Lamp certificate

Table 4: 200 W calibration lamp data from certificate

| **Wavelength [Å]** | **R/Å** |
|---|---|
| 3500 | 4448.72 |
| 3600 | 7219.88 |
| 3700 | 9511.33 |
| 3800 | 13281.4 |
| 3900 | 16120.3 |
| 4000 | 20387.1 |
| 4100 | 24631.2 |
| 4200 | 29902.4 |
| 4300 | 35354.9 |
| 4400 | 41494.4 |
| 4500 | 46863.6 |
| 4600 | 53575.1 |
| 4700 | 61173.8 |
| 4800 | 70900.6 |
| 4900 | 79031.3 |
| 5000 | 88298 |
| 5100 | 97581.5 |
| 5200 | 108563 |
| 5300 | 118854 |
| 5400 | 130603 |
| 5500 | 142257 |
| 5600 | 153216 |
| 5700 | 166162 |
| 5800 | 177443 |
| 5900 | 190263 |
| 6000 | 202493 |
| 6100 | 215192 |
| 6200 | 227686 |
| 6300 | 240157 |
| 6400 | 253386 |
| 6500 | 266310 |
| 6600 | 279024 |
| 6700 | 291707 |

| **Wavelength [Å]** | **R/Å** |
|---|---|
| 6800 | 303013 |
| 6900 | 316775 |
| 7000 | 328223 |
| 7100 | 341842 |
| 7200 | 353857 |
| 7300 | 365610 |
| 7400 | 376588 |
| 7500 | 387016 |
| 7600 | 397864 |
| 7700 | 407171 |
| 7800 | 418275 |
| 7900 | 427179 |
| 8000 | 436601 |
| 8100 | 446335 |
| 8200 | 456129 |
| 8300 | 465687 |
| 8400 | 473653 |
| 8500 | 481425 |
| 8600 | 492220 |
| 8700 | 498682 |
| 8800 | 506821 |
| 8900 | 513554 |
| 9000 | 519025 |
| 9100 | 526424 |
| 9200 | 530882 |
| 9300 | 538656 |
| 9400 | 541776 |
| 9500 | 547098 |
| 9600 | 550620 |
| 9700 | 551713 |
| 9800 | 557813 |
| 9900 | 560542 |
| 10000 | 562588 |

## A.2 Uncertainty calculation

The starting point is the calibration factor equation :

$$K(\lambda) = \frac{\rho(\lambda) M_0(\lambda) \left(\frac{z_0}{z}\right)^2 \cos\alpha}{C(\lambda)}$$

$$\implies \ln K = \ln\rho + \ln M_0 + 2\ln z_0 - \ln C + \ln(\cos(\alpha)) - 2\ln z$$

$$\implies \frac{\mathrm{d}K}{K} = \frac{\mathrm{d}\rho}{\rho} + \frac{\mathrm{d}M_0}{M_0} + 2\frac{\mathrm{d}z_0}{z_0} - \frac{\mathrm{d}C}{C} + \tan(\alpha)\frac{\mathrm{d}\alpha}{\alpha} - 2\frac{\mathrm{d}z}{z}$$

$$\implies \frac{\Delta K}{K} = \frac{\Delta\rho}{\rho} + \frac{\Delta M_0}{M_0} + 2\frac{\Delta z_0}{z_0} + \frac{\Delta C}{C} + \tan(\alpha)\frac{\Delta\alpha}{\alpha} + 2\frac{\Delta z}{z}$$

Since there is no uncertainty on $\rho$, $M_0$, $z_0$, $C$, we have :

$$\frac{\Delta K}{K} = \tan(\alpha)\frac{\Delta\alpha}{\alpha} + 2\frac{\Delta z}{z}$$

## A.3 Tutorial for upcoming analysis

**Important:** Directory files need to be updated for this script to work on any laptop.

### A.3.1 Run an analysis

To plot the results, run the script. Then, call one of the functions listed below, depending on the information wanted.

- `Plot_raw(year)`: returns the calibration factor plot against the wavelength and the signal-to-noise ratio estimate.
- `Outlier_report_peirce(year)`: returns the amount of deleted outliers
- `Outlier_report_peirce_std(year)`: returns the standard deviation of deleted outliers
- `Plot_norm(year)`: returns the calibration factor plot, normalized at 8430 Å, with linear regression
- `Plot_std(year)`: returns the standard deviation of the calibration factor measurements
- `Plot_std_variation(year)`: returns the relative standard deviation of the calibration factor measurements

As an argument, input a list of years. For example, call a function with the argument `[2011,2016,2022]` will return calibration information from 2011, 2016 and 2022. Giving the argument `all_years` will return calibration information from all the available years.

### A.3.2 Treating new data

This section gives a tutorial to proceed to future analyses. Line numbers correspond to the lines in appendix A.4. To run this analysis, 3 files are required:

- BG scan
- ABS scan
- CAL file

and 2 parameters:

- R: distance from the lamp to the Silver Bullet dome

- $\alpha$ angle between the screen and the lamp's optical axis

These data are stored in the *INPUTS AND VARIABLES* part of the script (line 529). To add new data from a year:

1. Declare the variable `path_data_XXXX_bg` equal to the directory of the BG scan. Write the year instead of `XXXX`
2. Declare the variable `path_data_XXXX_abs` equal to the directory of the ABS scan. Write the year instead of `XXXX`
3. Declare the variable `path_cal_XXXX` equal to the directory of the CAL file. Write the year instead of `XXXX`
4. Declare the variable `R_XXX` equal to the distance (in meters) from the lamp to the Silver Bullet dome. Write the year instead of `XXXX`
5. Declare the variable `a_XXX` equals to the angle (in degrees) between the screen and the lamp's optical axis. Write the year instead of `XXXX`
6. Add to `dico` (line 604):

   `aXXXX=[path_data_XXXX_bg,path_data_XXXX_abs,path_cal_XXXX]`. Write the year instead of `XXXX`
7. Add to `dicop` (line 616): `p2022=[R_XXXX,a_XXXX]`. Write the year instead of `XXXX`
8. Add the year to the list `all_years`

## A.4 Python script

```python
import matplotlib.pyplot as plt
import numpy as np
import statistics
import scipy.special
from sklearn.metrics import r2_score
import time

##FUNCTIONS

## Read_Sorted returns a 381 wl containing X measurements of the same wavelength
def Read_Sorted(path,year):
    path_final=path
    f=open(path_final,"r")
    data1=[]
    data2=[]
    for ligne in f:
        data1.append(ligne.strip().split())
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    data1.pop(0)
    realdata=[]
    data2=[]
    for i in data1:
        if i[0]!=str(year):
            for k in i:
                data2.append(k)
        if i[0]==str(year):
            realdata.append(data2)
            data2=[]
    realdata.append(data2)
    realdata.pop(0)
    wl=np.linspace(8285,8706,381,True)
    values_sorted=[]
    for i in range (0,len(realdata[0])):
        values_sorted.append([int(realdata[0][i])])
    for i in range(1,len(realdata)):
        for k in range (0,len(realdata[i])):
            values_sorted[k].append(int(realdata[i][k]))
    return(values_sorted)

#Returns the adjusted wavelength tab depending on the .Cal file
def Adjustment(path):
    file=open(path)
    data_loc=[]
    for k in file :
        data_loc.append(k.strip())
    x0=float(data_loc[1])
    x1=float(data_loc[2])
    x2=float(data_loc[3])
    x3=float(data_loc[4])
    wl=np.linspace(0,381,381,True)
    adjusted_wl=[]
    for (i) in wl:
        adjusted_wl.append((x0+float(i)*x1+float(i)**2*x2+float(i)**3*x3))
```

```
    return(adjusted_wl)

##Calculate Peirce's coef (related to Peirce's criterion outlier treatment)
def peirce_dev(N: int, n: int, m: int) -> float:
    """Peirce's criterion

    Returns the squared threshold error deviation for outlier identification
    using Peirce's criterion based on Gould's methodology.

    Arguments:
        - int, total number of observations (N)
        - int, number of outliers to be removed (n)
        - int, number of model unknowns (m)
    Returns:
        float, squared error threshold (x2)
    """
    # Assign floats to input variables:
    N = float(N)
    n = float(n)
    m = float(m)

    # Check number of observations:
    if N > 1:
        # Calculate Q (Nth root of Gould's equation B):
        Q = (n ** (n / N) * (N - n) ** ((N - n) / N)) / N
        #
        # Initialize R values (as floats)
        r_new = 1.0
        r_old = 0.0  # <- Necessary to prompt while loop
        #
        # Start iteration to converge on R:
        while abs(r_new - r_old) > (N * 2.0e-16):
            # Calculate Lamda
            # (1/(N-n)th root of Gould's equation A'):
            ldiv = r_new ** n
            if ldiv == 0:
                ldiv = 1.0e-6
            Lamda = ((Q ** N) / (ldiv)) ** (1.0 / (N - n))
            # Calculate x-squared (Gould's equation C):
            x2 = 1.0 + (N - m - n) / n * (1.0 - Lamda ** 2.0)
            # If x2 goes negative, return 0:
            if x2 < 0:
                x2 = 0.0
                r_old = r_new
            else:
                # Use x-squared to update R (Gould's equation D):
                r_old = r_new
                r_new = np.exp((x2 - 1) / 2.0) * scipy.special.erfc(
                    np.sqrt(x2) / np.sqrt(2.0)
                )
    else:
        x2 = 0.0
    return np.sqrt(x2)

#Eliminates outliers using Peirce's criterion and returns additionnaly the amount
#of outliers and their standard deviation
def Peirce_crit(dataset):
    mean=statistics.mean(dataset)
    std=statistics.stdev(dataset)
    rang=1
    count=0
```

```
    std_count=[]
    newdata=dataset[:]
    while rang !=0:
        to_delete=[]
        R=peirce_dev(len(newdata),rang,1)
        dev_max=R*std
        for i in range (0,len(newdata)):
            dev=abs(newdata[i]-mean)
            if dev>dev_max:
                to_delete.append(i)
                std_count.append(abs((newdata[i]-mean)/std))
        if len(to_delete)==0:
            rang=0
        else:
            for k in reversed(to_delete):
                del newdata[k]
            rang=len(to_delete)+1
            count+=len(to_delete)
    return(newdata,count,std_count)

#Return an average tab
def Average(dataset):
    average=[]
    for i in range (0,len(dataset)):
        average.append(statistics.mean(dataset[i]))
    return(average)

#Substracts the bg from the abs cal
def Bg_substraction(bg,abscal):
    result=[]
    for i in range(0,len(bg)):
        result.append(abscal[i]-bg[i])
    return(result)
#Calculates the known brightness
def Cal_fact(r,R,rho,wl,alpha,cert):
        alpha_rad=(alpha*np.pi)/180
        return(rho*cert*((r/R)**2)*np.cos(alpha_rad))

#Plots the calibration factor
def Plot_raw(year):
    count=0
    snr=[]
    for i in year:
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        for ki in range(0,len(wl)):
            bg_out.append(Peirce_crit(bg[ki])[0])
            abscal_out.append(Peirce_crit(abscal[ki])[0])
        #averaging
        bg_m=[]
        abscal_m=[]
        for k in range(0,len(wl)):
            bg_m.append(statistics.mean(bg_out[k]))
            abscal_m.append(statistics.mean(abscal_out[k]))
        #truncate
        bot=8374
        top=8525
```

```
        bg_c=[]
        abscal_c=[]
        wl_c=[]
        for k in range(0,len(wl)):
            if wl[k]>=bot and wl[k]<=top:
                bg_c.append(bg_m[k])
                abscal_c.append(abscal_m[k])
                wl_c.append(wl[k])
        #Calculating the known brightness
        B=[]
        for k in range (0,len(wl_c)):
            B.append(Cal_fact(r,dicop["p"+str(i)][0],rho,wl[k],
                                  dicop["p"+str(i)][1],synthetic_fit(wl_c[k])))
        k=[] #k=calibration factor
        for j in range (0,len(B)):
            k.append(B[j]/(abscal_c[j]-bg_c[j]))

        plt.scatter(wl_c,k,marker='+') #real plot
        #Interpolates
        fit=np.polyfit(wl_c,k,1,rcond=None,full=True)
        k_int=[]
        for l in wl_c:
            k_int.append(interpol(l,fit))
        r_2=r2_score(k,k_int)
        plt.plot(wl_c,k_int,
                 label=str(i)+' linear regression : '+str('%.3E' %fit[0][0])+
                 'x+' +str('%.3E' %fit[0][1] )+
                 ', rÂš='+str('%.2E' %r_2))
        count+=1
        snr.append(statistics.mean(abscal_c)/statistics.mean(bg_c))
    plt.grid()
    plt.legend()
    #plt.title("Values filtered with Peirce's criterion")
    plt.xlabel('Wavelength [ÃĚ]')
    plt.ylabel('C [R/ÃĚ/CTS]')
    plt.figure(2)
    plt.scatter(year,snr,color='black')
    plt.xlabel("years")
    plt.ylabel('SNR estimate')
    plt.show()
    return(snr)

#Gives a report on the amount of outliers deleted
def Outlier_report_peirce(year):
    for i in year:
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        nb_bg=[]
        nb_abs=[]
        for ki in range(0,len(wl)):
            bg_out.append(Peirce_crit(bg[ki])[0])
            abscal_out.append(Peirce_crit(abscal[ki])[0])
            nb_bg.append(Peirce_crit(bg[ki])[1])
            nb_abs.append(Peirce_crit(abscal[ki])[1])
        plt.scatter(wl,nb_bg,label=str(i)+', background. '+
                    str(sum(nb_bg))+' datapoints removed out of '+
                    str(len(bg[0] )*381))
```

```
        plt.scatter(wl,nb_abs,label=str(i)+', absolute calibration. '+
                    str(sum(nb_abs))+' datapoints removed out of '+
                    str(len(abscal[0])*381))
    plt.grid()
    plt.legend()
    plt.title("Outlier deletion report")
    plt.xlabel('Wavelength [ÃĔ]')
    plt.ylabel('Deleted outliers [CTS]')
    plt.show()

#Gives a report on the standard deviation of the deleted outliers
def Outlier_report_peirce_std(year):
    for i in year:
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        dev_bg=[]
        dev_abs=[]
        for ki in range(0,len(wl)):
            for w in (Peirce_crit(bg[ki])[2]):
                dev_bg.append(w)
            for w in (Peirce_crit(abscal[ki])[2]):
                dev_abs.append(w)
        plt.hist(dev_bg,label=str(i)+', background ')
        plt.hist(dev_abs,label=str(i)+', absolute calibration ')
    plt.grid()
    plt.legend()
    plt.title("Outlier deletion report")
    plt.xlabel('Number of STD from mean [CTS]')
    plt.ylabel('Deleted outliers [CTS]')
    plt.show()

#Plots the calibration factor, normalized at 8430
def Plot_norm(year):
    norm=8430
    for i in year:
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        for ki in range(0,len(wl)):
            bg_out.append(Peirce_crit(bg[ki])[0])
            abscal_out.append(Peirce_crit(abscal[ki])[0])
        #averaging
        bg_m=[]
        abscal_m=[]
        for k in range(0,len(wl)):
            bg_m.append(statistics.mean(bg_out[k]))
            abscal_m.append(statistics.mean(abscal_out[k]))
        #truncate
        bot=8374
        top=8525
        bg_c=[]
        abscal_c=[]
        wl_c=[]
```

```
        for k in range(0,len(wl)):
            if wl[k]>=bot and wl[k]<=top:
                bg_c.append(bg_m[k])
                abscal_c.append(abscal_m[k])
                wl_c.append(wl[k])
        #Calculating the known brightness
        B=[]
        for k in range (0,len(wl_c)):
            B.append(Cal_fact(r,dicop["p"+str(i)][0],rho,wl[k],
                                  dicop["p"+str(i)][1],synthetic_fit(wl_c[k])))
        k=[] #k=calibration factor
        for j in range (0,len(B)):
            k.append(B[j]/(abscal_c[j]-bg_c[j]))
        #Interpolates
        fit=np.polyfit(wl_c,k,1,rcond=None,full=True)
        k_int=[]
        wl_b=np.linspace(bot,top,2)
        for l in wl_b:
            k_int.append(l*fit[0][0]+(1-(norm*fit[0][0])))
        plt.plot(wl_b,k_int,label=i)

    plt.grid()
    plt.legend()
    plt.xlabel('Wavelength [ÃĚ]')
    plt.ylabel('C [R/ÃĚ/CTS]')
    plt.show()

#Plots the std
def Plot_std(year):
    bg_std=[]
    abs_std=[]
    count=0
    fig,axs=plt.subplots(2,2)
    avg_std_bg=[]
    avg_std_abs=[]
    for i in year:
        bg_std=[]
        abs_std=[]
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        bg_std_out=[]
        abs_std_out=[]
        for ki in range(0,len(wl)):
            bg_out.append(Peirce_crit(bg[ki])[0])
            abscal_out.append(Peirce_crit(abscal[ki])[0])
        #averaging
        bg_m=[]
        abscal_m=[]
        bg_out_m=[]
        abscal_out_m=[]
        for k in range(0,len(wl)):
            bg_m.append(statistics.mean(bg[k]))
            abscal_m.append(statistics.mean(abscal[k]))
            bg_std.append(statistics.stdev(bg[k]))
            abs_std.append(statistics.stdev(abscal[k]))
            bg_std_out.append(statistics.stdev(bg_out[k]))
            abs_std_out.append(statistics.stdev(abscal_out[k]))
```

```
        #truncate
        bot=8374
        top=8525
        bg_std_c=[]
        abs_std_c=[]
        wl_c=[]
        bg_std_out_c=[]
        abs_std_out_c=[]
        for k in range(0,len(wl)):
            if wl[k]>=bot and wl[k]<=top:
                bg_std_c.append(bg_std[k])
                abs_std_c.append(abs_std[k])
                abs_std_out_c.append(abs_std_out[k])
                bg_std_out_c.append(bg_std_out[k])
                wl_c.append(wl[k])

        avg_std_bg.append(statistics.mean(bg_std_c))
        avg_std_abs.append(statistics.mean(abs_std_c))

        axs[0,0].plot(wl_c,bg_std_c,label=str(i))
        axs[0,0].set_title('Background, outliers not treated')

        axs[1,0].plot(wl_c,bg_std_out_c,label=str(i))
        axs[1,0].set_title('Background, outliers treated')

        axs[0,1].plot(wl_c,abs_std_c,label=str(i))
        axs[0,1].set_title('Absolute calibration, outliers not treated')

        axs[1,1].plot(wl_c,abs_std_out_c,label=str(i))
        axs[1,1].set_title('Absolute calibration, outliers treated')

        count+=1
    for ax in axs.flat:
        ax.set(xlabel='Wavelength [ÃĚ]', ylabel='Standard deviation')
        ax.legend()

    plt.figure(2)
    plt.plot(year,avg_std_bg,label='Background')
    plt.plot(year,avg_std_abs,label='Absolute calibration')
    plt.ylabel('Standard deviation')
    lt.title('Outliers not treated')
    plt.xlabel("Years")
    plt.legend()
    plt.show()

#Plots the relative standard deviation
def Plot_std_variation(year):
    measurements_bg=[35,20,32,34,42,38,34,26,21,29,33]
    measurements_abs=[30,23,32,41,47,38,39,24,18,30,38]
    bg_std=[]
    abs_std=[]
    count=0
    fig,axs=plt.subplots(2,2)
    avg_std_bg=[]
    avg_std_abs=[]
    avg_std_out_bg=[]
    avg_std_out_abs=[]
    for i in year:
        bg_std=[]
        abs_std=[]
        path=dico["a"+str(i)]
        wl=Adjustment(path[2])
```

```
        bg=Read_Sorted(dico["a"+str(i)][0],int(i)) #raw file
        abscal=Read_Sorted(dico["a"+str(i)][1],int(i)) #raw file
        #outliers sorting with Peirce's crit
        bg_out=[]
        abscal_out=[]
        bg_std_out=[]
        abs_std_out=[]
        for ki in range(0,len(wl)):
            bg_out.append(Peirce_crit(bg[ki])[0])
            abscal_out.append(Peirce_crit(abscal[ki])[0])
        #averaging
        bg_m=[]
        abscal_m=[]
        bg_out_m=[]
        abscal_out_m=[]
        for k in range(0,len(wl)):
            bg_m.append(statistics.mean(bg[k]))
            abscal_m.append(statistics.mean(abscal[k]))
            bg_out_m.append(statistics.mean(bg_out[k]))
            abscal_out_m.append(statistics.mean(abscal_out[k]))
            bg_std.append(statistics.stdev(bg[k]))
            abs_std.append(statistics.stdev(abscal[k]))
            bg_std_out.append(statistics.stdev(bg_out[k]))
            abs_std_out.append(statistics.stdev(abscal_out[k]))

        #truncate
        bot=8374
        top=8525
        bg_std_c=[]
        abs_std_c=[]
        wl_c=[]
        bg_std_out_c=[]
        abs_std_out_c=[]
        for k in range(0,len(wl)):
            if wl[k]>=bot and wl[k]<=top:
                bg_std_c.append(bg_std[k]/bg_m[k])
                abs_std_c.append(abs_std[k]/abscal_m[k])
                abs_std_out_c.append(abs_std_out[k]/abscal_out_m[k])
                bg_std_out_c.append(bg_std_out[k]/bg_out_m[k])
                wl_c.append(wl[k])

        avg_std_bg.append(statistics.mean(bg_std_c))
        avg_std_abs.append(statistics.mean(abs_std_c))

        avg_std_out_bg.append(statistics.mean(bg_std_out_c))
        avg_std_out_abs.append(statistics.mean(abs_std_out_c))

        axs[0,0].plot(wl_c,bg_std_c,label=str(i))
        axs[0,0].set_title('Background, outliers not treated')

        axs[1,0].plot(wl_c,bg_std_out_c,label=str(i))
        axs[1,0].set_title('Background, outliers treated')

        axs[0,1].plot(wl_c,abs_std_c,label=str(i))
        axs[0,1].set_title('Absolute calibration, outliers not treated')

        axs[1,1].plot(wl_c,abs_std_out_c,label=str(i))
        axs[1,1].set_title('Absolute calibration, outliers treated')

        count+=1
    for ax in axs.flat:
        ax.set(xlabel='Wavelength [ÃĚ]', ylabel='Relative standard deviation')
```

```
        ax.legend()

    plt.figure(2)
    plt.plot(year,avg_std_bg,label='Background')
    plt.plot(year,avg_std_abs,label='Absolute calibration')
    plt.ylabel('Relative standard deviation')
    plt.title('Ouliers not removed')
    plt.xlabel("Years")
    plt.legend()
    plt.figure(3)
    plt.plot(year,avg_std_out_bg,label='Background')
    plt.plot(year,avg_std_out_abs,label='Absolute calibration')
    plt.title("Outliers removed")
    plt.ylabel('Relative standard deviation')
    plt.xlabel("Years")
    plt.legend()
    plt.show()
    return(bg_std_c,abs_std_c)

#Interpolation fuction to plot the linear interpolation of the calibration factor
def interpol(x,FIT):
        return(FIT[0][0]*x+FIT[0][1])

##Lamp
Lamp_200W_B_range_of_interest=[456129,465687,473653,481425,492220,498682]
Lamp_200W_wavelength_range_of_interest=np.arange(8200,8800,100)
fiti=np.polyfit(Lamp_200W_wavelength_range_of_interest,Lamp_200W_B_range_of_interest,1,rcond=None, f
def synthetic_fit(x):
  return(fiti[0][0]*x+fiti[0][1])
score=r2_score(Lamp_200W_B_range_of_interest,
               synthetic_fit(Lamp_200W_wavelength_range_of_interest))

##INPUTS AND VARIABLES
##In this section, input the files and parameters
scan_start=8285
scan_stop=8705
nb_scan=381
r=2
rho=0.98
scanning_wl=np.linspace(scan_start,scan_stop,nb_scan,True)

path_data_2011_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2011\\s0202112_b
path_data_2011_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2011\\s0202112_
path_cal_2011="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2011\\Wavelength.Cal"
R_2011=51.85 #m
a_2011=44 #deg

path_data_2013_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2013\\s0602132Si
path_data_2013_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2013\\s0602132S
path_cal_2013="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2013\\Wavelength.Cal"
R_2013=52.05 #m
a_2013=43.5 #deg

path_data_2014_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2014\\silver_Bac
path_data_2014_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2014\\silver_AB
path_cal_2014="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2014\\Wavelength.Cal"
R_2014=51.87 #m
a_2014=45 #deg

path_data_2015_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2015\\s1202152Ba
path_data_2015_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2015\\s1202152A
path_cal_2015="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2015\\Wavelength.Cal"
```

```
R_2015=51.88 #m
a_2015=46 #deg

path_data_2016_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2016\\Back_10022
path_data_2016_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2016\\ABS_CAL_1
path_cal_2016="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2016\\Wavelength.Cal"
R_2016=51.93 #m
a_2016=45 #deg

path_data_2017_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2017\\s0102172_B
path_data_2017_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2017\\s0102172_
path_cal_2017="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2017\\Wavelength.Cal"
R_2017=51.974 #m
a_2017=44 #deg

path_data_2018_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2018\\s0802182_B
path_data_2018_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2018\\s0802182_
path_cal_2018="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2018\\Wavelength.Cal"
R_2018=51.92 #m
a_2018=44 #deg

path_data_2019_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2019\\DARK_SCAN_
path_data_2019_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2019\\ABS_CALsC
path_cal_2019="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2019\\Wavelength.Cal"
R_2019=52.02 #m
a_2019=45.2 #deg

path_data_2020_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2020\\s2002202_C
path_data_2020_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2020\\s2002202_
path_cal_2020="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2020\\Wavelength.Cal"
R_2020=51.98 #m
a_2020=45 #deg

path_data_2021_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2021\\s1802212_B
path_data_2021_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2021\\s1802212_
path_cal_2021="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2021\\Wavelength.Cal"
R_2021=51.95 #m
a_2021=45.5 #deg

path_data_2022_bg=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2022\\s1002222_B
path_data_2022_abs=("C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2022\\s1002222_
path_cal_2022="C:\\Users\\rapha\\iCloudDrive\\Svalbard 2022\\Internship\\Data\\2022\\Wavelength.Cal"
R_2022=51.96 #m
a_2022=45  #deg

dico=dict(a2011=[path_data_2011_bg,path_data_2011_abs,path_cal_2011],
          a2013=[path_data_2013_bg,path_data_2013_abs,path_cal_2013],
          a2014=[path_data_2014_bg,path_data_2014_abs,path_cal_2014],
          a2015=[path_data_2015_bg,path_data_2015_abs,path_cal_2015],
          a2016=[path_data_2016_bg,path_data_2016_abs,path_cal_2016],
          a2017=[path_data_2017_bg,path_data_2017_abs,path_cal_2017],
          a2018=[path_data_2018_bg,path_data_2018_abs,path_cal_2018],
          a2019=[path_data_2019_bg,path_data_2019_abs,path_cal_2019],
          a2020=[path_data_2020_bg,path_data_2020_abs,path_cal_2020],
          a2021=[path_data_2021_bg,path_data_2021_abs,path_cal_2021],
          a2022=[path_data_2022_bg,path_data_2022_abs,path_cal_2022],
          )
dicop=dict(p2011=[R_2011,a_2011],p2013=[R_2013,a_2013],p2014=[R_2014,a_2014],
           p2015=[R_2015,a_2015],p2016=[R_2016,a_2016],p2017=[R_2017,a_2017],
           p2018=[R_2018,a_2018],p2019=[R_2019,a_2019],p2020=[R_2020,a_2020],
           p2021=[R_2021,a_2021],p2022=[R_2022,a_2022]
           )
```

```
all_years=[2011,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022]
```

## References


[1] Trends and variability of polar mesopause region temperatures attributed to atmospheric dynamics and solar activity, Holmen, S.E., PhD thesis

[2] The Synthetic Hydroxyl Spectrum Generator, F. Sigernes, Download page UNIS, 2012.
http://kho.unis.no/Software/SyntheticOH/

[3] Sigernes, F. Shumilov, N. Deehr, C. Nielsen, Kristian SvenÃÿe, T. & Havnes, O.. (2003). Hydroxyl rotational temperature record from the auroral station in Adventdalen, Svalbard (78ÂřN, 15ÂřE). Journal of Geophysical Research. 108. 10.1029/2001JA009023.

[4] Dyrland, M.E. Sigernes, F. (2007). An update on the hydroxyl airglow temperature record from the Auroral Station in Adventdalen, Svalbard (1980-2005). Canadian Journal of Physics. 85. 143-151. 10.1139/P07-040.

[5] Holmen, S. E., M. E. Dyrland, and F. Sigernes (2014), Long-term trends and the effect of solar cycle variations on mesospheric winter temperatures over Longyearbyen, Svalbard (78ÂřN), J. Geophys. Res. Atmos., 119, 6596âĂŞ6608, doi:10.1002/2013JD021195.

[6] Fred Sigernes and Mikko SyrjÃďsuo, Silver Bullet upgrade, New motor drive system for the Silver Bullet spectrometer, University Courses on Svalbard (UNIS), Box 156, 9171 Longyearbyen, Norway.

[7] Ross, Stephen. (2003). Peirce's criterion for the elimination of suspect experimental data. J. Eng. Technol.. 20.

[8] Gould, Benjamin Apthorp. âĂIJOn Peirce's Criterion for the Rejection of Doubtful Observations, with tables for facilitating its application.âĂİ The Astronomical Journal 4: 81-87.