Reconstructing Signals from Astronomical Observations

Bachelor thesis

EE3L11: Bachelor End Project

A.A. Bacha
E. van den Burg

Delft University of Technology

TU Delft

Reconstructing Signals from Astronomical Observations

Bachelor thesis

by

A.A. Bacha
E. van den Burg

to obtain the degree of Bachelor of Science at the Delft University of Technology,
to be defended on June 22, 2026

Authors:	A.A. Bacha	E. van den Burg
Project duration:	20-04-2026	26-06-2026
Thesis committee:	Dr. A. Endo	TU Delft, chair
	Dr.ir. S. Brackenhoff	TU Delft, supervisor
	Prof.dr.ir. A.J. van der Veen	TU Delft, neutral assessor

Cover illustration: *Messier 77 by ESA/Hubble & NASA, L. C. Ho, D. Thilker*

TU Delft

Abstract

This thesis addresses the challenge of astronomical signal recovery by developing and testing a reconstruction algorithm for the signals emitted by Dusty Star-Forming Galaxies using simulated data from the spectrometer DESHIMA 2.0. To be able to achieve this, first, a noise estimation was constructed in the frequency-domain using Weighted Least Squares. Then a reconstruction method in the time-domain using a Generalized Least Squares framework was developed. For the reconstruction two different modes are evaluated (1) Continuously Staring and (2) Position Switching.
Furthermore, an analysis of the noise model and performance of the reconstruction algorithm is done. For the noise model the estimated values is compared to the actual value that was simulated. Here it is demonstrated that a good estimation was made.
Of the algorithm itself first the noise levels present are analysed. Furthermore, the reconstruction algorithm was evaluated across varying percentages of input data. Where both of which resulted in favourable results for position switching mode. It was found that the system performed well; even when utilizing only 4% of the dataset (representing $\sim$2 minutes of data), the signal could be reconstructed with an RMSE of 0.0016 K, for a galaxy with continuum emission ranging between $\sim$0.001 and 0.01 K.

Preface

This thesis presents the research of our Bachelor End Project in Electrical Engineering at the Faculty of Electrical Engineering, Mathematics and Computer Science (EEMCS), Delft University of Technology. During this project, we had the opportunity to apply our knowledge to the, for us, new research field of astronomy, by developing a data-scientific noise removal method to separate detector noise and reconstruct the emitted source signal from astronomical observations for the DESHIMA 2.0 spectrometer.
This project was an incredible learning experience that will undoubtedly influence our future careers. Beyond the technical challenges, we have learned how to think as scientists and how to explain our research in a manner that it satisfies both physicists and engineers, reminding us of the bigger picture of our project.

We would like to thank our supervisors, whose support made this project possible. First, we wish to thank Dr. Akira Endo for introducing us to the world of DESHIMA 2.0 and welcoming us into the research group.
Most importantly, we want to sincerely thank Dr.ir. Stefanie Brackenhoff for her great day-to-day guidance. Her encouragement, insights, and meaningful feedback were essential to our project. Moreover, we would like to thank her for the fun, engaging, and motivating atmosphere she brought to every meeting.
Finally, we would like to thank Prof.dr.ir. Alle-Jan van der Veen for taking the time to be part of our Green Light Assessment and our thesis committee.

A.A. Bacha
E. van den Burg
Delft, July 2026

Contents

Nomenclature

Abbreviations

Abbreviation	Description
DESHIMA	DEep Spectroscopic HIgh-redshift MApper
DSFG	Dusty Star-Forming Galaxy
GLS	Generalized Least Squares
HDF5	Hierarchical Data Format version 5
ISS	Integrated Superconducting Spectrometer
MKID	Microwave Kinetic Inductance Detector
PSD	Power Spectral Density
PSW	Position SWitching
RMSE	Root Mean Squared Error
SD	Standard Deviation
SNR	Signal-to-Noise Ratio
TLS	Two-Level System
WGN	White Gaussian Noise
WLS	Weighted Least Squares

Mathematical Notation

Notation	Description
$\vec{x}$	Column vector (lowercase with vector arrow)
$\mathbf{M}$	Matrix (uppercase, bold)
$\vec{x}^T$, $\mathbf{M}^T$	Transpose of a vector or matrix
$\mathbf{M}^{-1}$	Inverse of a square, non-singular matrix
$\mathbf{M}^{1/2}$	Matrix square root
$\mathbf{M}^{-1/2}$	Inverse matrix square root
$\odot$	Hadamard (element-wise) product
$\|\cdot\|_W^2$	Weighted squared Euclidean norm
$\mathbb{E}\{\cdot\}$	Statistical expectation operator
$\hat{.}$	Estimated of

Symbols

Symbol	Description	Unit
$\vec{a}$	Atmospheric transmission efficiency vector over time	
C	Estimated photon noise floor power level	K^2/Hz
f	Frequency in the Fourier analysis domain	Hz
$\mathbf{H}$	Design / observation matrix for linear regression	
N	Total number of frequency bins	
N_c	Total number of spectral filterbank channels	
N_t	Length of time-series dataset / number of samples	
$\vec{n}$	Noise contribution vector over time	K
$\mathbf{R}_n$	Time-domain noise covariance matrix	K^2
$S_{nn}(f)$	Power Spectral Density of the noise process	K^2/Hz
t	Time parameter	s
T_{measured}	Total measured raw brightness temperature	K
T_{sky}	Total sky brightness temperature entering the instrument	K
T_{ph}	Brightness temperature contribution from photon noise	K
T_{TLS}	Brightness temperature contribution from TLS noise	K
T_A^*	True, unattenuated target source brightness temperature	K
$T_{p,\text{atm}}$	Physical temperature of the atmosphere	K
$\mathbf{W}$	Weighting matrix for Weighted Least Squares estimation	
x	Target source brightness temperature scalar	K
$\vec{y}$	Vector of time-series observed signal data	K
$\tilde{\vec{a}}$	Whitened atmospheric transmission coefficient vector	
$\tilde{\vec{y}}$	Whitened observed measurement data vector	
α	Power level scaling coefficient for the TLS noise model	K^2
β	Dimensionless exponent / slope parameter for TLS noise	
Δt	Time interval step / chopper wheel switching frame period	s
$\boldsymbol{\epsilon}$	Statistical error vector / residual column vector	
η_{atm}	Atmospheric transmission admittance efficiency factor	
$\boldsymbol{\theta}$	Parameter column vector for linear noise estimation	
μ	Expected value / statistical mean of the reconstructed signal	K
ν	Central reading frequency of a spectral filter channel	GHz
σ	Standard deviation / uncertainty noise level parameter	K

1

Introduction

Galaxies far, far away hold immeasurable knowledge and there is still a lot to uncover about the early universe, such as the physical conditions, gas density, and composition of the medium surrounding the earliest and furthest galaxies. But the further we look, the fainter the signal, and the harder it becomes to separate the truly astronomical signal from noise. To discover the history of these galaxies, the observed data must first be filtered to uncover the truly emitted signal of the Dusty Star-Forming Galaxy (DSFG), making noise elimination a crucial step for astronomical observations.

The DEep Spectroscopic HIgh-redshift MApper, DESHIMA 2.0, is designed to detect the redshifted emission from DSFGs, making it possible to study the history of star formation and the evolution of galaxies [1]. DESHIMA 2.0 has a frequency range of approximately 200-400 GHz, and observed signals in this range are sorted into hundreds of spectral channels [2]. Asides from the targeted astronomical source signal, the data in each spectral channel has the following noise sources present: (1) photon noise (white Gaussian noise), (2) atmospheric noise and (3) Two-Level System (TLS) noise [3], which originates from the Microwave Kinetic Inductance Detectors (MKIDs) used. Figure 1.1 depicts an overview of the noise sources present in channel 14 of DESHIMA, which reads the spectral frequency of 205.7 GHz. There are already data-scientific noise-removal methods, and the atmospheric emission can be characterized by a distinct spectral profile [4], since emission from the atmosphere dominanites the data, removing it is necessary to extract the astronomical signal from the data. In this thesis, we assume the atmospheric emission is well-characterized and can be perfectly subtracted prior to our analysis using independent atmospheric measurements. This is an approximation that is difficult to achieve in practice, but allows us to focus on the less-studied effects of TLS noise in post-processing.

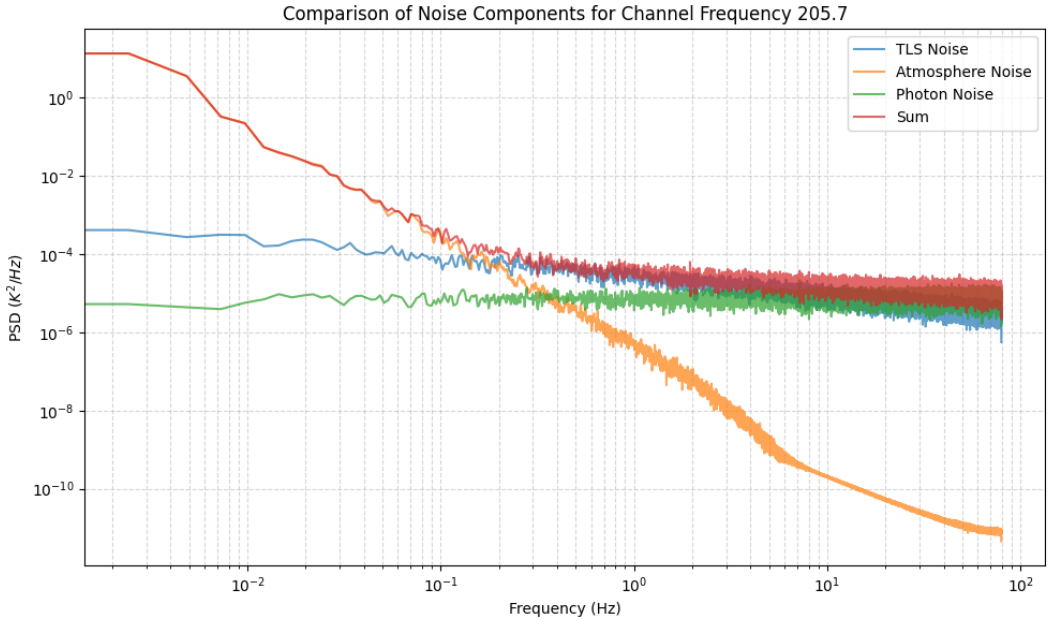

Figure 1.1: The PSD, acquired from simulated data of DESHIMA 2.0, of all the noise sources and the total noise present in channel 14 of DESHIMA 2.0.

Thus far, there have been several studies done to decrease the TLS noise, Noroozian et al. [5] have shown that TLS noise can be reduced in MKIDs by increasing the size of the capacitive section and narrowing down the inductive section of the MKID. Pan et al. [6] have demonstrated it is possible to decrease TLS noise by (1) lowering the capacitor filling factor, (2) increasing the operating temperature, and (3) increasing the driving power.
However, these approaches require hardware modifications and cannot be applied to already collected data. Because collecting data of DSFGs requires a large observation time and are difficult to acquire, already existing data is valuable. Therefore, it is essential to be able to use every sample of data. This report addresses that gap, by proposing a data-scientific method to model and eliminate TLS noise from the simulated DESHIMA 2.0 data, allowing for reconstruction of the emitted signal of the astronomical source.

The objective of this project is to create a system that is capable of reconstructing the true brightness temperature of the target DSFG by separating the detector noise from the observed data for DESHIMA 2.0 simulations. To achieve this objective, the TLS and photon noise ought to be accurately estimated. Therefore, a Weighted Least Squares (WLS) estimation framework is implemented to acquire the most optimal parameters for the TLS noise modelling. Finally, a data-whitening filter method is proposed alongside a Generalized Least Squares (GLS) algorithm to reconstruct the faint astronomical source signal for two observation modes of the spectrometer, (1) Continuously Staring, where the spectrometer is constantly pointed at the source, resulting in a long observation of the astronomical source. And (2) Position SWitching (PSW), where the spectrometer points at the source and right besides it in one switching cycle, this is acquired via the chopper wheel, where it is assumed that the conditions of the atmosphere are the same, resulting in two measurements per chopper wheel cycle. (1) On-source, where the light emitted by the source and the noise are measured in the data. And (2) Off-source, where only the noise sources are measured.

2

Program of Requirements

Below the requirement of the project are explained. For clarity, they are split into three sections: functionality, performance and additional.

2.1. List of requirements

2.1.1. System Functionality

1. The system shall be able to reconstruct at least 95% of the emitted signal by DSFGs.
2. The system shall be able to read HDF5 files.
3. The system shall not modify the original HDF5 files.
4. The system shall use simulated DESHIMA 2.0 data.
5. The system shall preprocess input data to check for completeness (NaN values, unexplained zero values).
6. The system shall be able to transform the data to the frequency domain for analysis.
7. The system shall be able to create an estimation of the TLS noise for each channel of DESHIMA 2.0 independently.
8. The system shall be able to create an estimation of the photon noise for each channel of DESHIMA 2.0 independently.
9. The system shall reconstruct the DSFG signal independently for each of the channels of the filterbank of DESHIMA 2.0.
10. The system shall store the reconstructed signal of the DSFG.
11. The system shall be able to visualize the results in frequency domain.
12. The system shall quantify the difference between the reference observation (ground truth) and the reconstructed DSFG signal.
13. The input data shall be read and results shall be written only once to avoid unnecessary multiple passes through the data.

2.1.2. System Performance

1. The system shall produce a constant result when executed multiple times on the same data.
2. The reconstructed signal shall have a signal-to-noise ration (SNR) that is the same or higher than the reference observation (ground truth).
3. The system shall have a runtime lower than the observation time (1 hour).
4. The mean of the reconstructed DSFG signal shall be approximately identical to the mean of the simulated DSFG signal, demonstrating for both observation modes (continuously staring and position switching) that the reconstruction algorithms function as unbiased estimators.

5. The variance of the reconstructed DSFG signal shall decrease as the total duration of the input observations increases.

2.1.3. System Additional Requirements

1. The system shall support data analysis of multiple channels of the DESHIMA 2.0 filterbank.
2. The system shall be implemented in Python 3.12.
3. The system shall allow processing modules to be executed independently.
4. The system shall be adaptable to different channels without changing the core code; only changing the parameters.
5. The system shall be able to handle datasets of different sizes without changing the core code.
6. The code shall be maintained in a version-controlled repository hosted on GitHub.
7. All functions shall include comments describing their purpose, input parameters, and outputs.
8. All physical quantities in the code shall include their SI unit in comments or documentation.

3

Background

3.1. The DESHIMA 2.0 Spectrometer

DESHIMA 2.0 is an Integrated Superconducting Spectrometer (ISS) designed to enable ultra-wideband spectroscopy for submillimeter wave astronomy [7]. By using an MKID filterbank chip, the spectrometer can measure DSFGs at high redshifts. DESHIMA 2.0 has a spectral bandwidth of 200 GHz, which ranges from 200-400 GHz, distributed over 339 spectral channels where each of these 339 have a part of this 200-400 GHz frequencies. And a sampling rate of $\approx$160 Hz, meaning that DESHIMA 2.0 registers approximately 160 data samples per second per channel. Each channel covers a slice of this spectral frequency range, which allows for simultaneous measurements of multiple emission lines [8].

3.2. Data Simulations and Observation Strategies

This project uses simulated time-series observations of the DSFG Messier 77, acquired using gateau (Gpu-Accelerated Time-dEpendent observAtion simUlator) [9]. However, the datasets used in this project contained valid data up to channel 300, corresponding to a specrtal frequency of 364.2 GHz. The Gateau framework simulates observations by using user defined parametes, such as, astronomical source properties, a set of atmospheric screens, telescope scan patterns, and instrument parameters. The data generated by Gateau has been validated against data from the DESHIMA 2.0 spectrometer [10].

For this project, two observation strategies of DESHIMA 2.0 are considered:

1. Continuous Staring Mode: In this observation strategy, the spectrometer continuously stares at the astronomical source. In this configuration, the signal of the source is constantly present in the recorded data.
2. Position Switching Mode: In this method, the spectrometer switches between observing the astronomical source (on-source) and observing next to the source (off-source), as illustrated in Figure 3.1, under the assumption that the properties of the atmosphere in both scenarios are equal. Subsequently, the off-source spectrum is subtracted from the on-source spectrum [7]. The rapid switching between on-source and off-source measurements can be acquired with the chopper wheel which is incorporated in the hardware of DESHIMA 2.0 [11]. The switching frequency of the chopper wheel is set at 10 Hz with a duty cycle of 0.5 [2], meaning that in 0.1 seconds an on-source and off-source observation is made.
   Due to observational overhead, some samples are typically lost. In this project, the chopper method is implemented with 6 data samples for each on- and off-source observation. This results in a switching frequency of the chopper wheel of $\sim$13 Hz, meaning that the full chopper cycle in this project has a duration of $\sim$0.08 seconds, resulting in both an on- and off-source phase of $\sim$ 0.04 seconds [1].

[1] The DESHIMA sampling rate of 160 Hz and a 10 Hz chopper frequency result in 16 samples per complete cycle. In reality, 4 of these samples are lost, leaving 12 usable data samples per chopper cycle. This project erroneously modelled the entire cycle

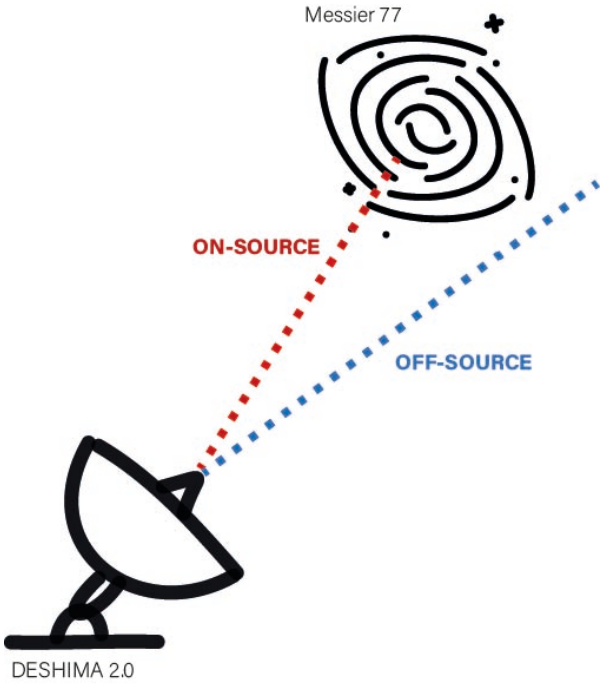

Figure 3.1: Schematic of different optical paths of the spectrometer in PSW mode, demonstrating the on-source (red) and off-source (blue) paths, note that this schematic is not to scale.

3.3. Measured Data

In both observation modes, the recorded data contains a lot of noise. There are three noise sources present in the acquired data, namely, (1) atmospheric noise, (2) Two-Level System noise, and (3) photon noise. The total measured brightness temperature when the spectrometer is pointed on-source, $T_{\text{measured, on-source}}(\nu, t)$ in K, across a given frequency channel ν as a function of time t, can be modelled as follows:

$$T_{\text{measured, on-source}}(\nu, t) = T_{\text{sky}}(\nu, t) + T_{\text{ph}}(\nu, t) + T_{\text{TLS}}(\nu, t) \tag{3.1}$$

Where $T_{\text{sky}}(\nu, t)$ in K, is the sky brightness temperature entering DESHIMA 2.0, this contains the brightness temperature from the source and the atmospheric emission. $T_{\text{ph}}(\nu, t)$ in K, represents the photon noise, and $T_{\text{TLS}}(\nu, t)$ also in K, represents the Two-Level System noise [12]. The sky brightness temperature $T_{\text{sky}}(\nu, t)$ is represented by the radiative transfer function [13]:

$$T_{\text{sky}}(\nu, t) = (1 - \eta_{\text{atm}}(\nu, t))\, T_{\text{p,atm}} + \eta_{\text{atm}}(\nu, t) T_{\text{A}}^{*}(\nu) \tag{3.2}$$

Where:

- $T_{\text{p,atm}}$ is the physical temperature of the atmosphere in K (set to 273 K in these simulations).
- $\eta_{\text{atm}}(\nu, t) \in [0, 1]$ is the atmospheric transmission efficiency (or admittance).
- $(1 - \eta_{\text{atm}}(\nu, t))\, T_{\text{p,atm}}$ represents the emission from the atmosphere in K.
- $T_{\text{A}}^{*}(\nu)$ is the true, unattenuated source brightness temperature above the atmosphere in K. The goal of this project is to reconstruct $T_{\text{A}}^{*}(\nu)$ as accurately as possible from the observed data.

When the spectrometer is pointed off-source, the following temperature in K is measured:

$$T_{\text{measured, off-source}}(\nu, t) = (1 - \eta_{\text{atm}}(\nu, t))\, T_{\text{p,atm}} + T_{\text{ph}}(\nu, t) + T_{\text{TLS}}(\nu, t) \tag{3.3}$$

using only these 12 usable samples, implicitly accelerating the simulated chopper frequency to $\approx$ 13.3 Hz.

For this project, it can be assumed that the atmospheric emission $(1 - \eta_{\text{atm}}(\nu, t))\, T_{\text{p,atm}}$ can be perfectly subtracted from the observed data in post-processing, allowing for an ideal situation for the reconstruction of the source signal. However, this perfect subtraction is not possible in practice. In real observations, the atmospheric emission has to be estimated using a reference measurement, for example from the off-source pointing or from a highly sensitive radiometer which measures the Precipitable Water Vapour (PWV). With this radiometer measurement, models such as that presented by Pardo et al. [4] can be used to determine the transmission coefficient of the atmosphere from which the measured atmospheric emission can be computed.

3.4. Noise Sources

3.4.1. Atmospheric noise

Emission from the atmosphere fluctuates at low frequencies and is the most dominant noise source present in the observed data. It is caused by the Earth's atmosphere absorbing and re-emitting incoming light originating from the astronomical source.
Since simulations of the atmospheric noise only were provided, where the spectrometer is pointed off-source, meaning that $T_{\text{sky}}(\nu, t) = (1 - \eta_{\text{atm}}(\nu, t))\, T_{\text{p,atm}}$, the atmospheric transmission efficiency $\eta_{\text{atm}}(\nu, t) \in [0, 1]$ can be obtained as follows:

$$\eta_{\text{atm}} = 1 - \frac{T_{\text{sky}}(\nu, t)}{T_{\text{p,atm}}} \tag{3.4}$$

The atmospheric transmission efficiency η_{atm} will be used in the data model presented in Chapter 4 to account for the dilution of the source signal due to the atmosphere.

The Python code that computes the η_{atm} for each channel at each time step is shown in Appendix B.4.

3.4.2. Two-Level System noise

Two-Level System noise is an instrumental noise source that occurs due to the dielectric substrate of the MKID detectors. In the surface substrate of the MKID, electrical fluctuations occur between the ground state and an excited energy state of charge carriers, which causes the TLS noise in the observed data [14]. Since TLS noise is created in each spectral channel of DESHIMA 2.0 separately, there is no frequency-domain correlation. However, the TLS noise is time correlated. Figure 3.2 illustrates how the MKID is incorporated in the chip.

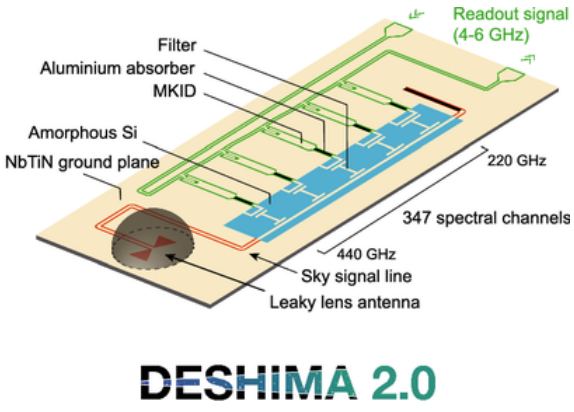

Figure 3.2: Chip design of DESHIMA 2.0. Illustrating how the sky signal is guided to the filterbank. Each filter is connected to an MKID, where the TLS noise originates from. Figure adapted from [7]

3.4.3. Photon noise

Another noise source present in the observed data is the photon noise. Photon noise is caused by the amount of light hitting the detector. Therefore, it scales directly with the loading power hitting the

MKID, if there are more photons, there is more photon noise. Because the atmospheric transmission η_{atm} changes per channel, which influences how much light hits the detector, different channels see different amounts of power, meaning that photon noise varies per channel.

4

Data Model

To reconstruct the astronomical source signal from the time-series data provided by the DESHIMA 2.0 simulations, the theory discussed in Chapter 3 must be translated into a mathematical framework. This chapter proposes (1) a generalized noise model, which is used to estimate statistical power distribution of the noise in each spectral channel. (2) a linear data model that can be used to isolate the source temperature from the TLS and photon noise.

4.1. Noise Model

Before the astronomical source can be reconstructed, the noise present in the data, (after the subtraction of the atmospherical noise), TLS and photon noise, must be characterized for each channel. Because these are stochastic processes, a time-domain realization is unpredictable and unable to be modelled directly. Therefore, a general frequency-domain model that characterizes the average noise power distribution is proposed.
In general, TLS noise is 1/f-type noise [15], the TLS noise occurring in DESHIMA 2.0 also shows this behaviour. Therefore, the PSD of the TLS noise can be modelled with the power law as follows [6]:

$$S_{\text{TLS}}(f) = \frac{\alpha}{f^{\beta}} \quad [\text{K}^2/\text{Hz}] \tag{4.1}$$

Where f is the frequency in Hz, α determines the overall power level of the noise in K^2, and β is a dimensionless parameter that determines the slope of the noise power. One of the methods to find the α parameter, is by dividing the PSD of the actual TLS noise by $f^{-\beta}$. For the β parameter, Pan et al. [6] concluded that the fit values for β are $= 0.8 \pm 0.2$, when using their phenomenological TLS model. However, it was found that a β of ~ 0.5 corresponds better to the provided simulations of the TLS noise in DESHIMA 2.0.
However, a more accurate value for α and β for each channel can also be found with Weighted Least Squares (WLS), which will be further discussed in Section 5.1.

To account for the photon noise in each spectral channel, the mean of the PSD of the photon noise can be taken, and added to the noise model. This can be done as in the provided simulated data, photon noise is mathematically modelled as an additive White Gaussian Noise (WGN) process [10]. The PSD of the TLS and photon noise together then becomes as follows:

$$S_{nn}(f) = \frac{\alpha}{f^{\beta}} + C \quad [\text{K}^2/\text{Hz}] \tag{4.2}$$

Where C is the estimated photon noise in K^2/Hz.

4.2. Data Model

The signal measured by the spectrometer is divided into a vector for each of the N_c spectral channels of DESHIMA 2.0. For a single channel, the measured brightness of the sky (T_{measured}), as mentioned in Equation 3.1, can be described with the general time-domain linear data model, as shown in Equation 4.3, this model allows the noise to have a covariance matrix [16], which is needed since the TLS noise present in the measured signal varies over time.

$$\vec{y}_t = \vec{a}_t x + \vec{n}_t \tag{4.3}$$

Where:

- $\vec{y} \in \mathbb{R}^{N_t \times 1}$ is a column vector containing the observed data (T_{measured}) for a single spectral channel over a total of N_t samples.
- $\vec{a} \in \mathbb{R}^{N_t \times 1}$ is a column vector of length N_t containing the atmospheric transmission coefficient (η_{atm}) over time.
- $x \in \mathbb{R}$ is a 1-dimensional scalar representing the source temperature (T_A^*) for that single channel.
- $\vec{n} \in \mathbb{R}^{N_t \times 1}$ is a column vector of length N_t representing the TLS and photon noise over time present in the observed data.

Here N_t denotes the total number of time samples for a single channel.

4.3. Observation Modes

The values of the elements contained in $\vec{y}$ and $\vec{a}$ are dependent on the observation strategy of the spectrometer. As mentioned in Section 3.2, the different type of observation strategies are: (1) Continuous Staring mode and (2) Position Switching mode.
In Continuous Staring mode the source signal is constantly present and, therefore, the vector $\vec{a}$ maps directly to the atmospheric transmission coefficient η_{atm}. The data model can therefore also be written as:

$$\vec{y} = \vec{\eta}_{\text{atm}} x + \vec{n} \tag{4.4}$$

In Position Switching mode, the spectrometer switches between observing the astronomical source and observing besides the source. Here an off-source spectrum is subtracted from an on-source source spectrum in time-domain [7]. To account for the different pointings of the spectrometer the data model, the vector $\vec{a}$ is defined as follows:

$$\vec{a}[t] = \begin{cases} \vec{\eta}_{\text{atm}}[t] & \text{for on-source} \\ 0 & \text{for off-source} \end{cases} \tag{4.5}$$

Hence, the data model for PSW becomes:

$$\vec{y}[t] = \begin{cases} \vec{a}[t]x + \vec{n}[t] & \text{for on-source} \\ \vec{n}[t] & \text{for off-source} \end{cases} \tag{4.6}$$

This ensures that the effects of the TLS and photon noise $n[t]$ to the measured data remains the same, whilst, the contribution of the source's signal will only be considered when the spectrometer is aimed at the astronomical source.

5 Algorithms

5.1. Implementing Weighted Least Squares for TLS Noise Estimation

Weighted Least Squares is an estimation method that can be used when not all observations have the same precision or importance, the weights express the importance of each observation [17].
Since the TLS noise is dominant in the frequency band between the knee frequency with the atmospheric noise and the intersection with the photon noise floor, WLS is a suitable regression method to find the α and β parameters for the TLS noise model, as WLS can prioritize that frequency band.
The WLS estimator is applied in the frequency-domain, after the natural logarithm has been taken of both the frequency bins and the PSD of the TLS noise, to obtain linearity. For a dataset containing N frequency bins, the linear system components are defined as follows:

- $\mathbf{W} \in \mathbb{R}^{N \times N}$ is a square, diagonal matrix containing the relative weights assigned to each frequency bin. The entries of the weight matrix are initialized as the inverse of the PSD (P_{xx}) of the actual TLS noise at each corresponding frequency bin. This prevents the larger noise power at lower frequencies from dominating the fit.

$$\mathbf{W} = \begin{bmatrix} \frac{1}{P_{xx}(f_1)} & 0 & \dots & 0 \\ 0 & \frac{1}{P_{xx}(f_2)} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{P_{xx}(f_N)} \end{bmatrix} \tag{5.1}$$

- $\mathbf{H} \in \mathbb{R}^{N \times 2}$ is the design matrix and contains the linear intercept and slope:

$$\mathbf{H} = \begin{bmatrix} 1 & \ln(f_1) \\ 1 & \ln(f_2) \\ \vdots & \vdots \\ 1 & \ln(f_N) \end{bmatrix} \tag{5.2}$$

- $\vec{u} \in \mathbb{R}^{N \times 1}$ is the observation vector containing the logarithm of the measured PSD from the simulated data:

$$\vec{u} = \begin{bmatrix} \ln(P_{xx}(f_1)) \\ \ln(P_{xx}(f_2)) \\ \vdots \\ \ln(P_{xx}(f_N)) \end{bmatrix} \tag{5.3}$$

To solve for the noise parameters, the model is first framed as a standard linear model:

$$\vec{u} = \mathbf{H}\hat{\theta} + \vec{\epsilon} \tag{5.4}$$

Where $\vec{u}$ is the observation vector, $\mathbf{H}$ is the design matrix, $\hat{\theta}$ contains the noise parameters to be estimated, and $\vec{\epsilon}$ represents the errors. The objective of WLS is to find the parameter vector $\hat{\theta}$ that minimizes the weighted sum of the squared residuals, this can be expressed as follows:

$$\hat{\theta} = \underset{\boldsymbol{\theta}}{\text{argmin}} \, \|\vec{u} - \mathbf{H}\hat{\theta}\|^2_{\mathbf{W}} \tag{5.5}$$

Using these defined variables, the general form of the Weighted Least Squares Estimator is as follows [16]:

$$\hat{\theta} = (\mathbf{H^T W H})^{-1}\mathbf{H^T W}\vec{u} \tag{5.6}$$

Where the resulting parameter $\hat{\theta} \in \mathbb{R}^{2\times 1}$ vector yields the solved intercept and slope values:

$$\hat{\theta} = \begin{bmatrix} \ln(\alpha) \\ -\beta \end{bmatrix} \tag{5.7}$$

Once $\hat{\theta}$ is solved via Equation 5.6, the scaling parameter is recovered via $\alpha = e^{\hat{\theta}_1}$, providing the parameters required for the noise model of each spectral channel.

The Python code that implements the WLS algorithms to find the noise parameters is shown in Appendix B.1.

5.2. Source Reconstruction

By using the data model defined in Equation 4.3 ($\vec{y} = \vec{a}x + \vec{n}$), alongside the model for TLS and photon noise PSD $S_{nn}(f)$, as defined in Equation 4.2, an optimal estimator for the astronomical source can be deduced to obtain the scalar x in K, which contains the reconstructed source temperature for a single channel.

The previously used WLS framework assumed that the noise samples in frequency-domain are independent, this resulted in a diagonal weight matrix. However, because TLS noise is an $1/f$-type noise process, subsequent samples in the time-domain noise vector $\vec{n}$ are correlated with one another over time. To account for this the Generalized Least Squares (GLS) [16] framework must be applied to Equation 5.5 When the variables are updated to the variables of the data model ($\vec{u} \to \vec{y}$, $\mathbf{H} \to \vec{a}$, $\boldsymbol{\theta} \to x$), the following holds:

$$x_{\text{opt}} = \underset{x}{\text{argmin}} \, \|\vec{y} - \vec{a}x\|^2_{\mathbf{W_n}} \tag{5.8}$$

Where the weight matrix is defined as the inverse of the time-domain covariance matrix $\mathbf{W_n} = \mathbf{R}_n^{-1}$. Since the noise has time correlation, the noise covariance matrix relates to the expected value of the noise vector, $\mathbf{R}_n = \mathbb{E}\{\vec{n}\vec{n}^T\}$. Solving the optimization problem in Equation 5.8 results in the following expression for the reconstructed source:

$$x_{\text{opt}} = \frac{\vec{a}^T\mathbf{R}_n^{-1}\vec{y}}{\vec{a}^T\mathbf{R}_n^{-1}\vec{a}} \tag{5.9}$$

The complete derivation of Equation 5.9 can be found in Appendix D.1.

5.2.1. Frequency-Domain Data Whitening

Computing Equation 5.9 directly in the time domain is computationally expensive, as this requires the inversion of a large $\mathbb{R}^{N_t \times N_t}$ covariance matrix. To avoid this, the inverse noise covariance matrix can be rewritten as follows: $\mathbf{R}_n^{-1} = \mathbf{R}_n^{-T/2}\mathbf{R}_n^{-1/2}$. Equation 5.9 can then be redefined as:

$$x_{\text{opt}} = \frac{(\mathbf{R}_n^{-1/2}\vec{a})^T(\mathbf{R}_n^{-1/2}\vec{y})}{(\mathbf{R}_n^{-1/2}\vec{a})^T(\mathbf{R}_n^{-1/2}\vec{a})} = \frac{\tilde{\vec{a}}^T\tilde{\vec{y}}}{\tilde{\vec{a}}^T\tilde{\vec{a}}} \tag{5.10}$$

Where $\tilde{\vec{y}} = \mathbf{R_n}^{-1/2}\vec{y}$ and $\tilde{\vec{a}} = \mathbf{R_n}^{-1/2}\vec{a}$ represent the whitened measurement and atmospheric transmission coefficient vector, respectively. The full derivation of this equation can be found in Appendix D.2.

By using the Wiener-Khinchin theorem, which states that the autocorrelation function of a wide-sense stationary process and its power spectral density are a Fourier transform pair [18], the computation of x_{opt} in K can be done in the frequency domain.

Strictly speaking, photon noise in MKID detectors is non-stationary, as its variance directly scales with the loading power, which is dominated by atmospheric noise. However, when analysing the photon noise in time domain, it can be seen that the standard deviation (SD) remains stable, as it has a mean value of 2.42×10^{-2} K with a maximum deviation of 12.98%. The SD of the photon noise is shown in Figure 5.1. Moreover, since the chopper wheel has a phase duration of Δt = 0.05 seconds[1], the atmosphere has not changed significantly enough between a subsequent on-source and off-source observation. Therefore, the loading power of the atmosphere and subsequently the variance of the photon noise remain constant during a cycle of the chopper wheel. Hence, for this project the photon noise can be considered WSS by approximation.

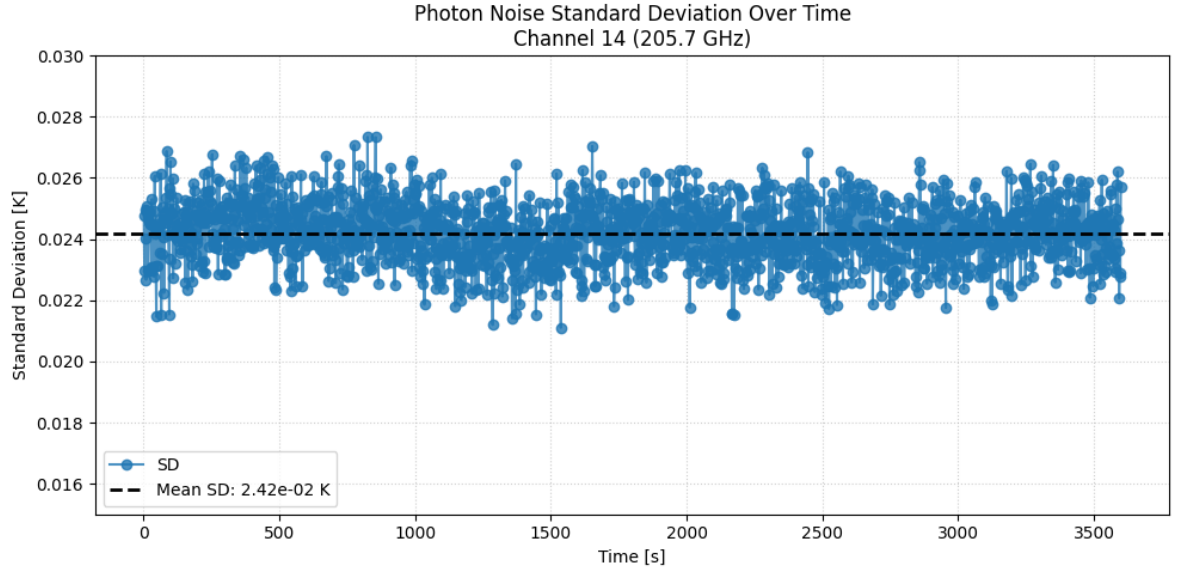

Figure 5.1: Visualisation of the SD of the photon noise in time domain, demonstrating that the photon noise can be taken as WSS by approximation.

Since TLS noise is theorized to be dependent on the defects in the MKID detectors, it can be assumed that, by approximation, the noise model is WSS. Under these assumptions, the operant $R_n^{-1/2}$ acts as a whitening filter ($W_{z,y}(f) = 1/\sqrt{S_{nn}}$). The time-series data is transformed via the Fast Fourier Transform ($\mathcal{F}$), multiplied with the whitening filter, and transformed back to the time domain via the Inverse Fast Fourier Transform ($\mathcal{F}^{-1}$):

[1]This is based on the actual chopper wheel frequency of 10 Hz rather than the simulated ∼13 Hz, this choice is made because the resulting cut-off frequency is intended for theoretical justification, and will not be used for any further computations.

$$\tilde{\vec{y}} = \mathcal{F}^{-1}\{W_{z,y}(f) \odot \mathcal{F}\{\vec{y}\}\} \tag{5.11}$$

$$\tilde{\vec{a}} = \mathcal{F}^{-1}\{W_{z,y}(f) \odot \mathcal{F}\{\vec{a}\}\} \tag{5.12}$$

Where $\odot$ denotes the Hadamard product. By obtaining the operants in this manner, the reconstructed source x_{opt} can then be computed with a simple dot-product, according to Equation 5.10, this reduces the processing time significantly when making long observations with DESHIMA 2.0.

5.2.2. Observation Strategies and Whitening Filter Realizations

The implementation of the whitening filter $W(f)$ depends on the chosen observation strategy of the spectrometer. The whitening filter $W_y(f)$ is defined as follows:

$$W_{z,y}(f) = \frac{1}{\sqrt{S_{nn}(f)}} \quad \left[\frac{\sqrt{\text{Hz}}}{\text{K}}\right] \tag{5.13}$$

Where the estimated noise $S_{nn}(f)$ has two possible realisations based on the observation strategy of the spectrometer.

Continuous Staring Mode

When the spectrometer continuously stares at the astronomical source, no signal modulation is performed on the observed data, therefore the estimated noise is equal to the TLS and photon noise ($S_{nn}(f) = \frac{\alpha}{f^\beta} + C$), as discussed in Section 4.1.

The Python code that implements the proposed algorithms for continuously staring is shown in Appendix B.3.

Chopper Wheel Implementation

The chopper wheel can be implemented in the algorithm by subtracting the on-source from the off-source observation in the time domain for each chopper cycle. If k denotes the discrete time index of the k-th chopper cycle, the resulting signal $z[k]$ in K is then defined as:

$$z[k] = y_{\text{on}}[k] - y_{\text{off}}[k] \tag{5.14}$$

Where $y_{\text{on}}[k]$ and $y_{\text{off}}[k]$, both in K, are the means of 6 samples taken during the on-source measurement and the mean of 6 samples taken during the off-source measurement of cycle k, respectively, according to the model proposed in Equation 4.6. The computation of $y_{\text{on}}[k]$ and $y_{\text{off}}[k]$ can mathematically be described as follows:

$$y_{\text{on}}[k] = \frac{1}{6}\sum_{n=12k}^{12k+5} x[n] \tag{5.15}$$

$$y_{\text{off}}[k] = \frac{1}{6}\sum_{n=12k+6}^{12k+11} x[n] \tag{5.16}$$

For the chopper wheel implementation, Equation 5.10 becomes:

$$x_{\text{opt}} = \frac{\tilde{\vec{a}}^T \tilde{\vec{z}}}{\tilde{\vec{a}}^T \tilde{\vec{a}}} \tag{5.17}$$

Where $\tilde{\vec{z}} = {\mathbf{R}_\mathbf{n}}^{-1/2}\vec{z}$ represents whitened measurement vector when using chopper modulation. Which can be acquired with $\tilde{\vec{z}} = \mathcal{F}^{-1}\{W_z(f) \odot \mathcal{F}\{\vec{z}\}\}$.

To isolate and evaluate how the chopper modulation effects the underlying noise properties, the subtraction can be modeled in the continuous time-domain. If $n(t)$ represents the mean of the noise in the on-source measurement, and $n(t - \Delta t)$ the mean of the noise in the off-source measurement, the effect of the subtraction can be simplified with the following noise term:

$$n_{\text{diff}}(t) = n(t) - n(t - \Delta t) \tag{5.18}$$

Where, Δt is the time difference in seconds between the midpoints of consecutive on-source and off-source observation. The transfer function $H(f)$ of Equation 5.18 is as follows:

$$H(f) = 1 - e^{-j2\pi f \Delta t} \tag{5.19}$$

Since $H(f)$ acts as a linear high-pass filter, it alters the PSD of the estimated noise as follows [19]:

$$S_{\text{nn, diff}} = |H(f)|^2 S_{nn}(f) = 4\sin^2(\pi f \Delta t) S_{nn}(f) \quad [K^2/Hz] \tag{5.20}$$

The full derivation of $H(f)$ and $|H(f)|^2$ can be found in Appendix D.3.
For a Δt of 0.05 seconds, which is the full duration of one on- or off-source observation, the chopper modulation has the following cut-off frequency:

$$f_{\text{-3dB}} = \frac{1}{4\Delta t} = 5 \text{ Hz} \tag{5.21}$$

Therefore, most of the low-frequency atmospheric and TLS noise will be suppressed due to the chopper modulation, since the power of these noise sources is much larger at lower frequencies, as depicted in Figure 1.1. Meaning that the values of the TLS and atmospheric noise could not have changed significantly during a chopper cycle. The full derivation of Equation 5.21 can be found in Appendix D.4.

Because the operation in Equation 5.14 combines two observations into a single discrete output ($z[k]$), it downsamples the data by with $D = 2$. Therefore, it is needed to match the altered spectral shape of the estimated noise $S_{\text{nn}}(f)$ to the obtained measurement vector $\vec{z}$. Downsampling with $D = 2$ has the following effect on the noise spectrum [19]:

$$S_{\text{nn, downsampled}} = \frac{1}{2}\left[S_{\text{nn, diff}}(\tfrac{f}{2}) + S_{\text{nn, diff}}(\tfrac{f - f_{\text{downsampled}}}{2})\right] \quad [\text{K}^2/\text{Hz}] \tag{5.22}$$

Where $f_{\text{downsampled}} = \frac{f_s}{6} = \frac{160}{6} \approx 27$ Hz is the downsampled frequency due to the averaging over 6 samples when computing the on-source and off-source observation vectors. The complete derivation of Equation 5.22 can be found in Appendix D.5.
When subtraction two independent white noise sources, the variance contained in the final signal adds together [19]. Since the chopper modulation has a duty cycle of 0.5, it holds that $t_{\text{on-source}} = t_{\text{off-source}} = \Delta t$, hence $T_{\text{cycle}} = 2\Delta t$, both in seconds. The variance of the subtracted signal $z[k]$ will then be:

$$\sigma_z^2 = \sigma_{\text{on-source}}^2 + \sigma_{\text{off-source}}^2 = \frac{C}{2\Delta t} + \frac{C}{2\Delta t} = \frac{C}{\Delta t} \tag{5.23}$$

Since the downsampling uses a sampling period of $T_{\text{cycle}} = 2\Delta t$ in seconds, corresponding to a $f_{\text{Nyquist}} = \frac{1}{4\Delta t}$, it is expected that the noise floor of $S_{\text{nn, downsampled}}$ is the total variance contained in $z[k]$ distributed over the new bandwidth:

$$S_{\text{nn, downsampled, floor}}(f) = \frac{\sigma_z^2}{f_{\text{Nyquist}}} = \frac{\frac{C}{\Delta t}}{\frac{1}{4\Delta t}} = 4C \quad [\text{K}^2/\text{Hz}] \tag{5.24}$$

Note that, since TLS noise varies slowly over time compared to the chopper switching period, it is assumed that during subsequent on- and off cycles, the TLS noise has similar values. Meaning that

the chopper method cancels out most of the TLS noise contained in the signal $z[k]$. On top of that, $S_{\text{nn, downsampled, floor}}(f)$ will not be used directly in the whitening filter, as it only represents the expected noise floor. Instead, it serves as a reference to verify the results of $S_{\text{nn, downsampled}}(f)$ in Section 6.1.1.

When using the chopper modulation the whitening filter becomes:

$$W_z(f) = \frac{1}{\sqrt{S_{\text{nn, downsampled}}(f)}} \tag{5.25}$$

With these algorithms, the astronomical source x_{opt} can be reconstructed when the spectrometer acquires its data in both Continuously Staring mode or Position Switching mode.

The Python code that implements the proposed algorithms for PSW is shown in Appendix B.4.

6

Results and Analysis

Having determined the noise en data models for each spectral channel of DESHIMA 2.0 in Chapter 4, and the three algorithms for (1) estimating the TLS and photon noise, (2) reconstructing the astronomical source and (3) whitening of the data in Chapter 5, this chapter presents the evaluation of the proposed methods. The objective of this chapter is to determine the accuracy in which the astronomical source brightness temperature can be reconstructed from the noise containing time-series data acquired from the Gateau simulations.
First, the proposed noise model and its transformations due to the chopper method and downsampling are evaluated. Then, the noise levels are found, after which the accuracy of the system with different amounts of data are considered. Finally, the reconstruction of the source when atmospheric emission is contained in the observed date is evaluated.

6.1. Noise Model Evaluation

To validate the results from the WLS algorithm, described in Section 5.1, a comparison of the PSD of the acquired noise estimation versus the actual noise present in channel 14 (which has a channel frequency of 205.7 GHz) of the spectrometer when looking off-source, is shown in Figure 6.1.

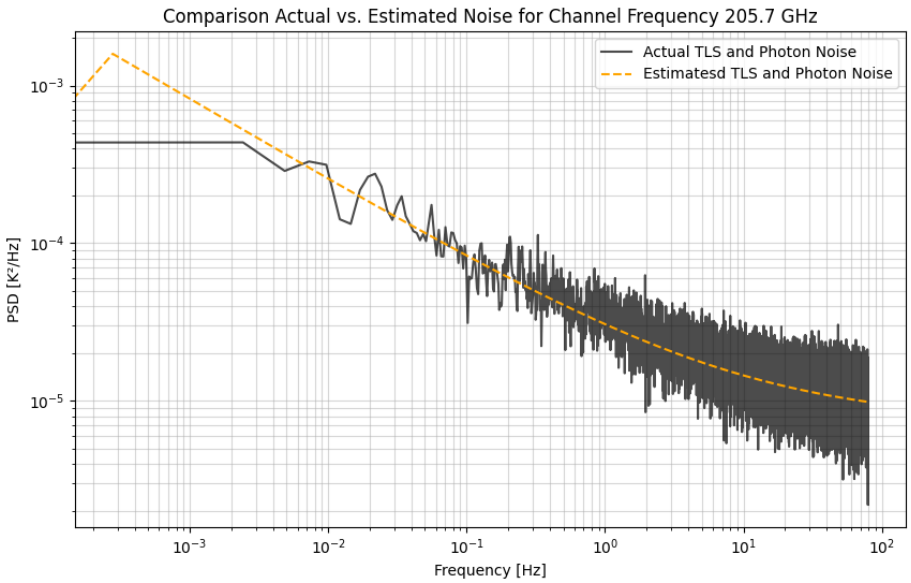

Figure 6.1: Comparison of the actual TLS and Photon noise present in channel 14 of DESHIMA 2.0 with the noise estimation from Section 4.1.

Where the frequency-axis spans from 2×10^{-4} Hz up to the Nyquist frequency of $\sim$80 Hz (as the sampling frequency of DESHIMA 2.0 $f_s \approx 160$ Hz). It can be seen that at higher frequencies (f$>$1 Hz), the estimated noise flattens out, this is due to the dominating uncorrelated photon noise. At lower frequencies (f$<$1 Hz), there is a slope in the estimated noise, this is due to the TLS noise being dominant, which acts as correlated 1/f type noise.
Because the estimated PSD of the noise accurately tracks the actual noise data, it can be concluded that the proposed noise model, optimized with WLS, successfully captures the behaviour of the TLS and photon noise present in the channels of DESHIMA 2.0.

6.1.1. Influence of the Chopper Method

When incorporating the chopper method into the algorithm, the PSD of the noise signal will be altered according to Equation 5.22, the noise spectrum will first be averaged, than subtracted, and finally downsampled. As discussed in the section above, at low frequencies, the noise is correlated. However, the use of the time-domain subtraction, as discussed in Equation 5.14, removes this correlated noise almost completely. Figure 6.2 shows the effects of subtracting the two spectra on the resulting estimated noise contained in the observed data.

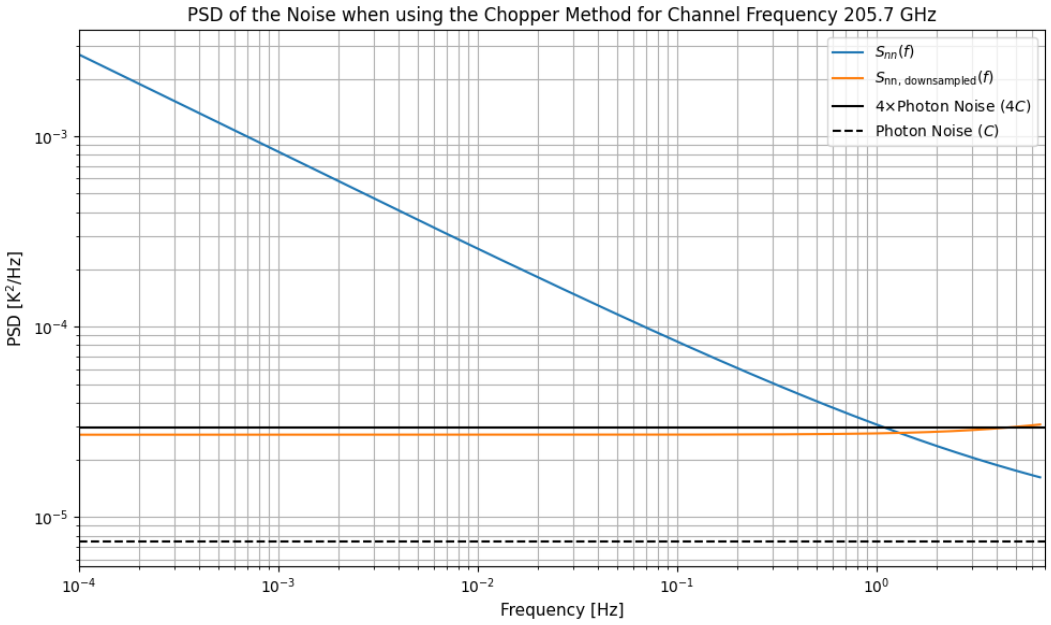

Figure 6.2: PSD of the resulting noise $S_{\text{nn, downsampled}}$ included in the data after performing the chopper modulation for Channel 14, demonstrating that the resulting noise after chopper modulation is white.

The frequency-axis spans from 2×10^{-4} Hz up to the Nyquist frequency of $\sim$7 Hz (as the mean is taken from 6 samples and then downsampled by a factor D =2, hence $f_{\text{Nyquist}} = \frac{160}{6 \times 2 \times 2} \approx$7 Hz). On top of that, it can be seen that the resulting noise $S_{\text{nn, downsampled}}$ is nearly white, and has a noise floor of approximately $4C$, which is expected and discussed in Section 5.2.2.

The resulting PSD when using the whitening filter on the data vector $\vec{z}$ is illustrated in Figure 6.3. Note that the PSD is normalized as the unwhitened vector $\vec{z}$ has the unit K, and the whitened vector $\vec{\tilde{z}}$ is dimensionless. It can be seen that using the whitening filter has no influence on the PSD of the data vector, as the noise contained in the data is already white, due to the chopper modulation.

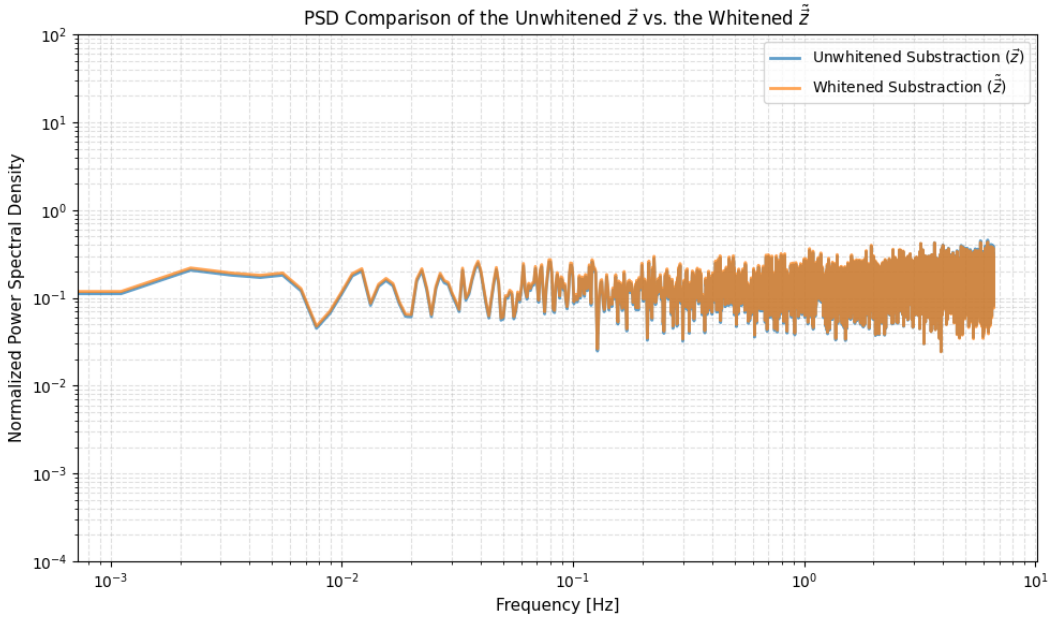

Figure 6.3: Comparison between the PSDs of the unwhitened observation vector $\vec{z}$ and the whitened observation vector $\vec{\tilde{z}}$, demonstrating that they are almost identical.

From this, it can be concluded that frequency-domain whitening is not necessary for reconstructing the astronomical source temperature when using the time-domain chopper modulation method, as the resulting PSD is identical to the PSD of the unwhitened data vector.

6.2. Noise analysis

In the context of this section, noise represents the stochastic uncertainty that prevents the model from a perfect observation when comparing it to the ground truth. For the analysis of this noise the Standard Deviation of the signal is considered. Because the data is described as a linear model $\vec{y} = \vec{a}x + \vec{n}$, it is important to note that the noise is modeled as a linear system as well.

6.2.1. Standard Deviation and Variance

The standard deviation of the signal is the square root of the variance. Here the standard deviation is a measurement for how far the signal fluctuates from the mean of the signal. The variance represents the power of this fluctuation [20].
The variance of the linear system can be defined as 6.1. The full derivation of this formula can be found in D.6.

$$\text{var}(x) = \sigma^2 \frac{1}{\tilde{\vec{a}}^T \tilde{\vec{a}}} \tag{6.1}$$

Here, $\tilde{\vec{a}} = \mathbf{R}_n^{-1/2}\vec{a}$ is the whitened atmospheric transmission coefficient vector, as defined in Equation 5.10. For continuously staring mode, $\vec{a}$ maps directly to $\vec{\eta}_{\text{atm}}$, whereas for PSW it maps to $\vec{\eta}_{\text{atm}}[t]$ for on-source observations and to 0 for off-source observations, as defined in Section 4.3. The scaling factor σ^2 represents the remaining noise power after whitening the data. Since the whitening filter is defined as $W_{z,y}(f) = 1/\sqrt{S_{nn}(f)}$, it normalizes the noise covariance such that $\mathbf{E}[\tilde{\vec{n}}\tilde{\vec{n}}^T] = \sigma^2\mathbf{I}$, when the estimated noise perfectly matches the actual noise σ^2 = 1, as explained in Appendix D.7.

With the variance computed, only the square root of it needs to be taken to find the SD, the results of this are shown in Figure 6.4.

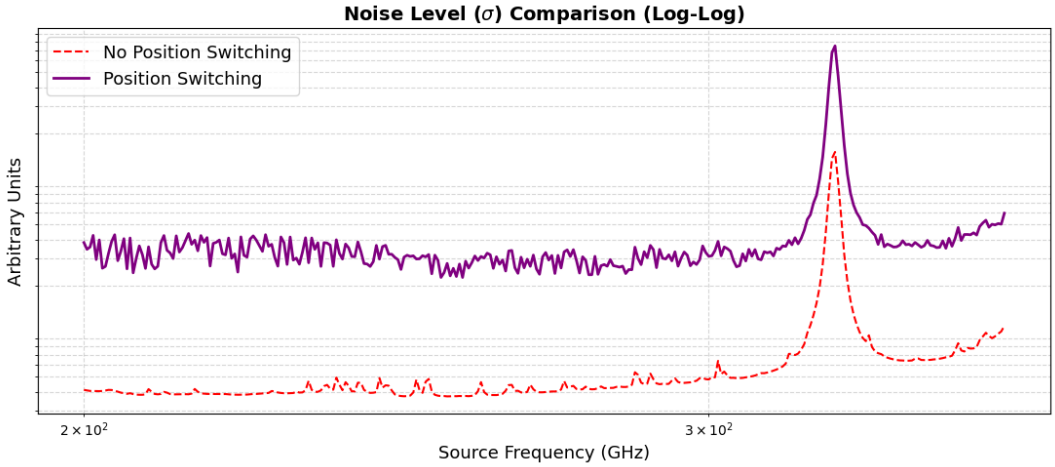

Figure 6.4: Noise Level comparison, **Noise Level is not correct**

In Figure 6.4, the SD or noise level σ of the source is plotted against the frequencies of the astronomical source. It can be seen that the noise level of the continuously staring reconstruction is lower than the PSW reconstruction.
It is important to note that the **level of the noise of Figure 6.4 is not correct** because of the usage of a wrong scaling factor.
As explained by Taniguchi et al. [7] a higher noise level of factor $\sqrt{2}$ would be expected when using PSW. Additionally, because only half the time is spent on-source during PSW, an extra factor $\sqrt{2}$ is expected, bringing the total ratio in noise level to 2.
However, as can be seen in Figure 6.4, this is not the case here. The figure indicates that the noise level for the continuous staring mode is actually about 6 times lower than that of the position-switching mode [1].

Furthermore, a very high peak can be seen around 325 GHz. To explain the cause of this peak three different channel frequencies and their noise components are plotted in Figure 6.5. Here it can be seen that the channel where the spike is present (the 325 GHz channel) has a much higher photon noise than the other two channels plotted. This could be because photon noise is a property of the amount of light that is seen by the detector. This means that at this frequency, the amount of light reaching the detector is very high. Around 325 GHz, this results in a big drop in η_{atm}, which leads to a higher $1 - \eta_{\text{atm}}$. This means that the atmospheric emission is a lot stronger, causing a lot more light to hit the detector and directly driving up the photon noise.

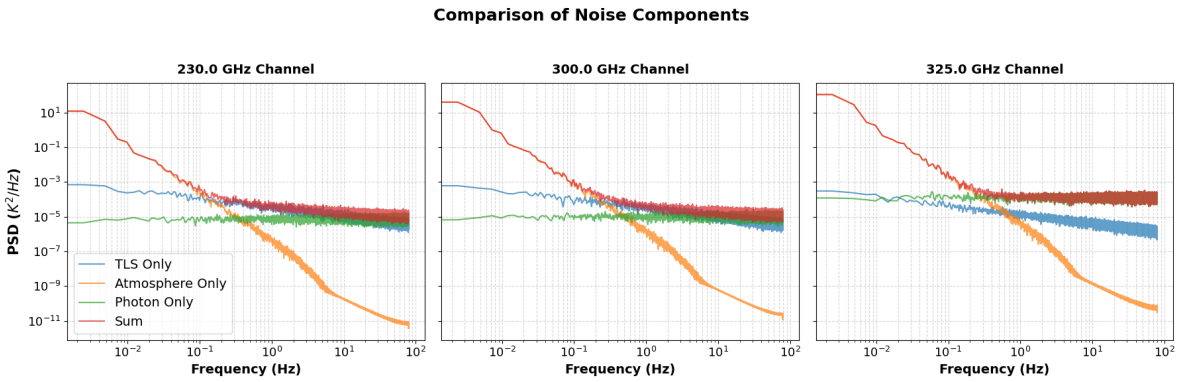

Figure 6.5: Comparison PSDs of Different Channel Frequencies

[1] Due to time constraints it was not possible to investigate this discrepancy.

To be able to verify this the noise model parameters are shown in Figure 6.6. Here the C value corresponds to the photon noise. Since photon noise is a form of white noise and is modelled as a frequency independent variable in the noise estimation, this increase does not change the frequency dependent slopes of α and β as can be observed in the Figure 6.6, as these parameters are not dependent on the photon noise. Instead, it only elevates the constant C term in Equation 5.6, which explains the large spike of the C Value.

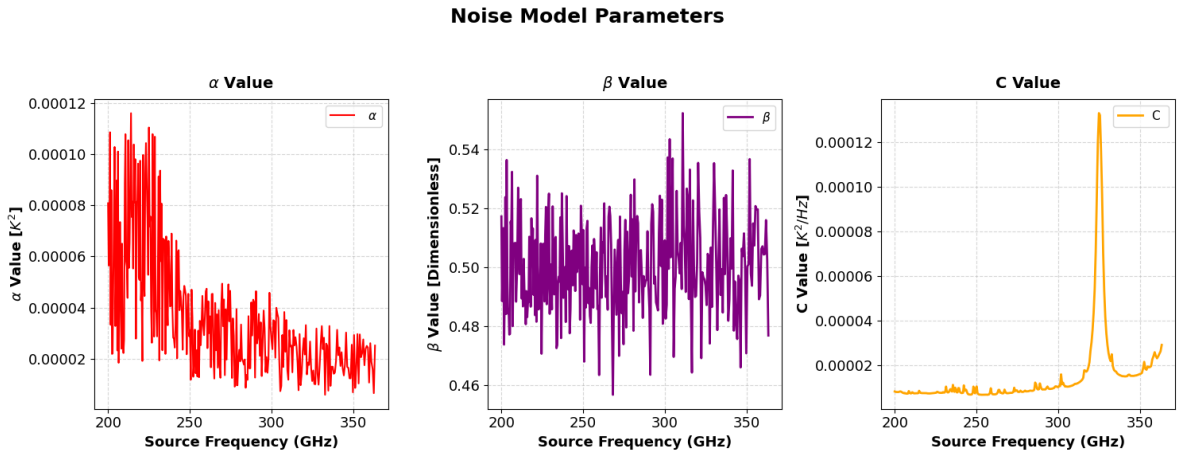

Figure 6.6: Noise parameters for all channel frequencies

6.3. Reconstruction Analysis

Until now, the full dataset was used, consisting of 572,204 data points per channel frequency. For this section, percentages of this data are considered to see how well the reconstruction would perform when only a fraction of the data is used. For both continuously staring and position switching, the following percentages of the data were used: 1% to 10%, 50%, 75%, and 100%. Here 100% provides a baseline for comparison, allowing the other percentages to be evaluated against it.

6.3.1. Reconstruction Continuously Staring

Figure 6.7 displays the reconstructions for four different data percentages (10%, 50%, 75%, and 100%). To analyse all the results, the noise level of these signal are plotted in Figure C.2 as well as two other performance metrics.

Firstly, the Root Mean Squared Error (RMSE) [21], which measures the average deviation from the ground truth in Kelvin, which here is the T_A^* defined in Section 3.2, can be found in Equation 6.2, in this equation, the ground truth is subtracted from the reconstructed signal, also in Kelvin, for each of the different channels. For RMSE a low value is desired as a low RMSE corresponds to a small difference between the ground truth and the reconstructed signal.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_{\text{source, i}} - x_{\text{reconstruction , i}})^2} \tag{6.2}$$

Secondly, the R^2 score, which quantifies the proportion of variance in the ground truth explained by the reconstruction, this can be found in Equation 6.3 [21]. Here, the numerator features the same quantifiers as in the previous formula, while the denominator represents the ground truth, as defined above, subtracted by the average value of the ground truth. Subtracting this by 1 yields the R^2 score of the signal. For the R^2 score, the value corresponds to the amount of signal that is successfully reconstructed, where 1 is totally accurate. Therefore, a high value is considered a good reconstruction.

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(x_{\text{source}}(f_i) - x_{\text{reconstruction}}(f_i)\right)^2}{\sum_{i=1}^{N}\left(x_{\text{source}}(f_i) - \bar{x}_{\text{source}}\right)^2} \tag{6.3}$$

Both of these metrics were found for 1% to 10%, 50%, 75%, and 100% resulting in Table 6.1.

Table 6.1: Reconstruction Performance Metrics Continuously Staring

Dataset Percentage	RMSE (K)	R^2 Score
1%	0.0042	0.8318
2%	0.0030	0.9128
3%	0.0026	0.9375
4%	0.0023	0.9511
5%	0.0022	0.9553
6%	0.0021	0.9559
7%	0.0020	0.9601
8%	0.0019	0.9649
9%	0.0018	0.9700
10%	0.0018	0.9632
50%	0.0007	0.9939
75%	0.0004	0.9979
100%	0.0001	0.9999

Table 6.1 shows that both the RMSE becomes lower, and the R^2 score becomes higher which would indicate better performance when the amount of data used is higher, which is in line with expectations. Interestingly, even when utilizing only 1% of the data, 83% of the ground truth is still reconstructed. When increasing the data fraction to 4%, this metric rises to 95%, which is remarkable considering that 96% of the original observation is omitted.

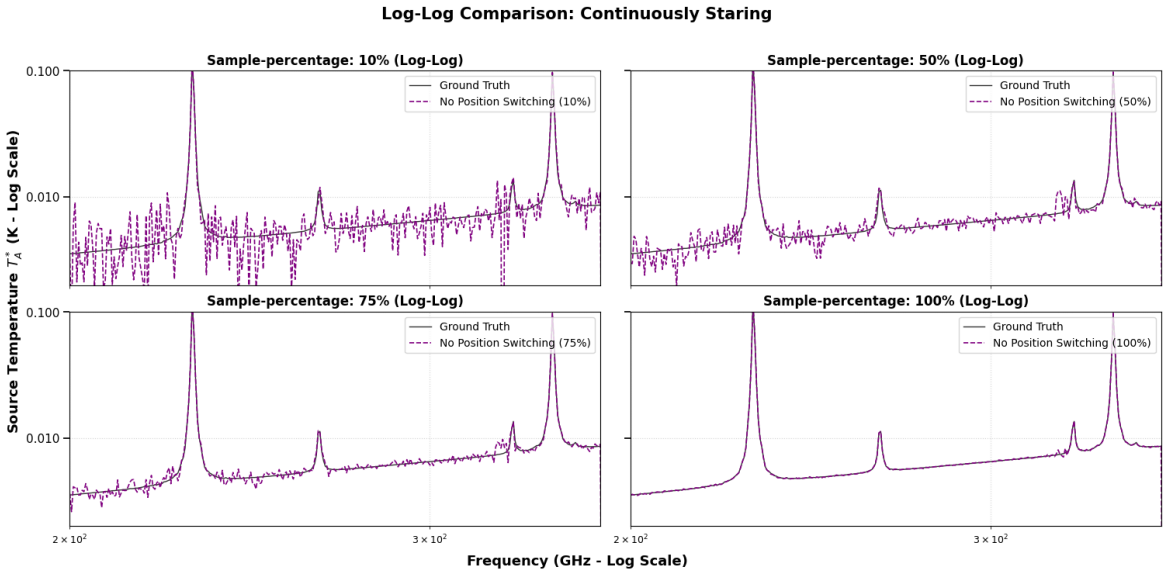

Figure 6.7: Reconstruction Continuously Staring

6.3.2. Reconstruction Positions Switching

Similar trends and metrics were observed for position switching compared to continuously staring. In Figure 6.8, the reconstructed signal is displayed across four different data percentages, and Table 6.2 presents the performance metrics. Here, it becomes apparent that position switching is significantly better at recovering the signal than continuously staring, especially when looking at the lower percentages. The R2 score for 1% is almost 96%, meaning that 96% of the ground truth can be reconstructed when only 1% of the signal is known. Interestingly, this number increases until 4% of the data, where it decreases slightly, before increasing again.

Furthermore, it becomes apparent that the reconstruction of the source signal does not become significantly better when analysing 75% or 100% of the data. However, Figure C.3 shows that when the full dataset is used, the noise level is slightly smaller than when 75% is used. Regardless, this difference is so small that it is not visible due to the rounding of the performance metrics

Table 6.2: Reconstruction Performance Metrics Position Switching

Dataset Percentage	RMSE (K)	R^2 Score
1%	0.0020	0.9598
2%	0.0012	0.9871
3%	0.0013	0.9839
4%	0.0016	0.9775
5%	0.0013	0.9843
6%	0.0016	0.9746
7%	0.0013	0.9848
8%	0.0008	0.9936
9%	0.0008	0.9941
10%	0.0009	0.9918
50%	0.0005	0.9976
75%	0.0004	0.9984
100%	0.0004	0.9984

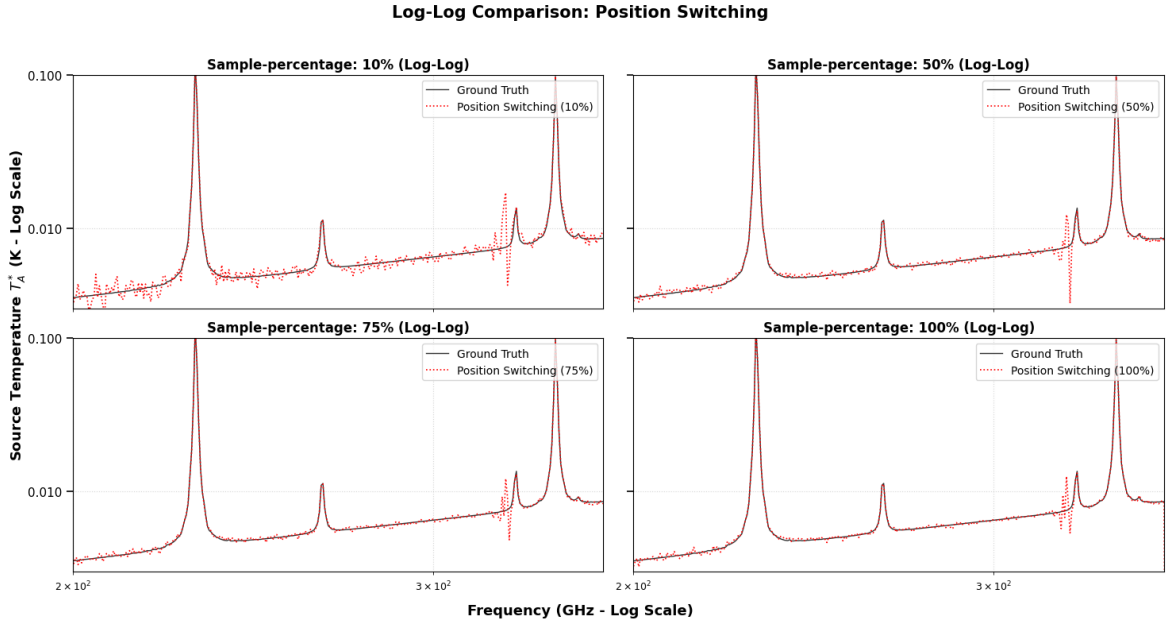

Figure 6.8: Reconstruction Position Switching

Finally, with the reconstructions for both continuously staring and position switching known, they were plotted against each other on a log-log scale for clarity, as shown in Figure 6.9. This figure clearly demonstrates that position switching achieves a better reconstruction across all data percentages.

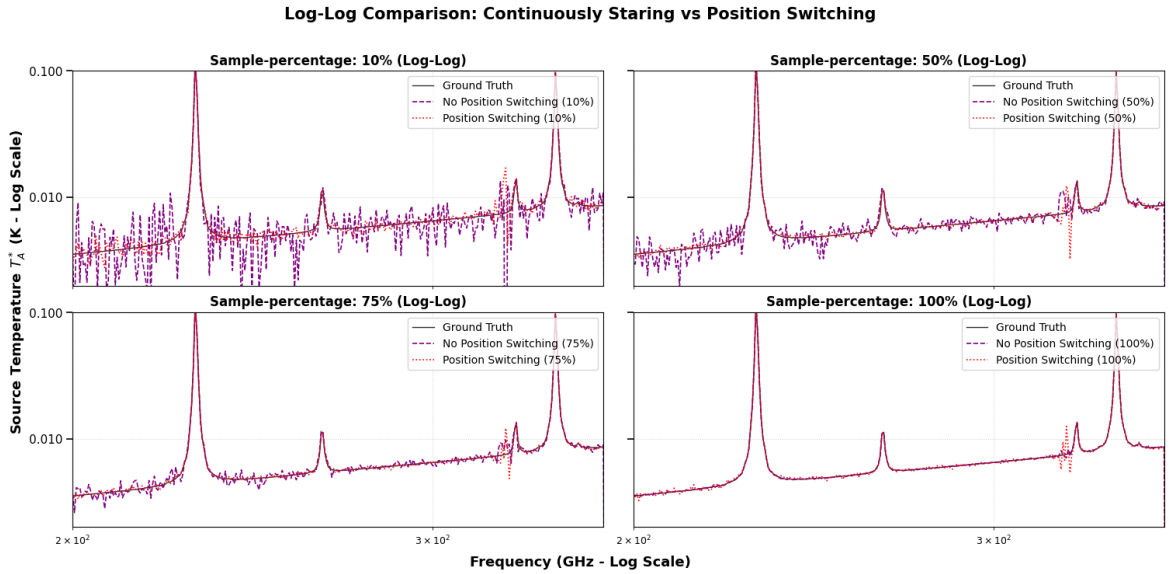

Figure 6.9: Log-Log Comparison of data reconstruction

6.3.3. Reconstruction with Atmosphere Noise in Observed Data

Until now, it was assumed that the noise from the atmosphere was perfectly subtracted from the observed data. However, as computed and discussed in Section 5.2.2, the high-pass filter cut-off frequency due to chopper modulation is 5 Hz. Therefore, with the chopper method, the proposed algorithm should be able to reconstruct the brightness temperature of the source x_{opt} even when the emission from the atmosphere is present in the observed data vector $\vec{y}$. The reconstructed brightness temperature of the source when the noise of the atmosphere is included in the observed data is shown in Figure 6.10.

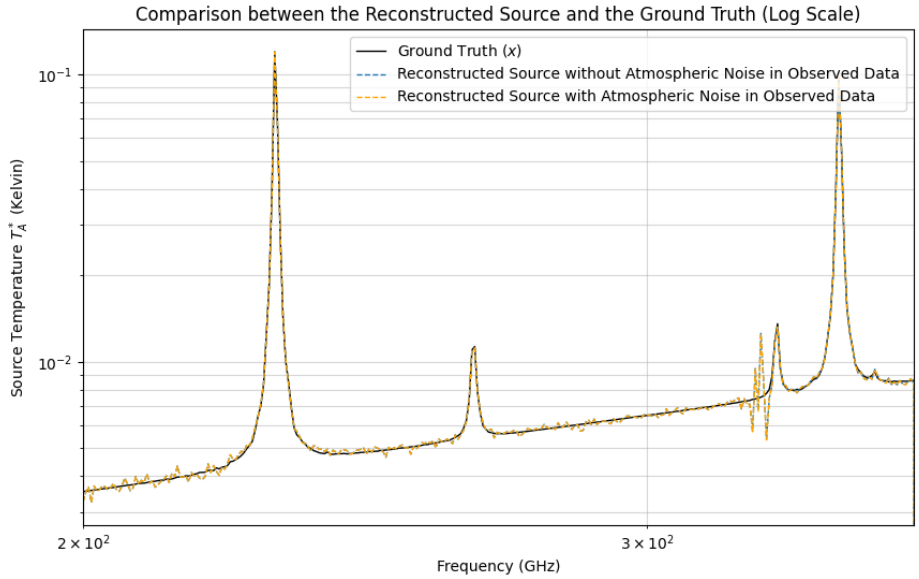

Figure 6.10: Comparison between the reconstructed brightness temperature of the source T_A^* in K and the actual brightness temperature of the source, i.e. the ground truth, on log scale. Demonstrating that the chopper method suppresses the atmospheric noise in the observed data.

Figure 6.10 shows that the proposed algorithm can successfully reconstruct the signal emitted by the

source, as it accurately tracks the ground truth. However, strong claims of the performance cannot be made, as a performance analysis, such as the RMSE and R^2 score have not yet been computed for this case. It is important to note that there are some fluctuations present in the reconstructed signal. However, these are the same as when the atmospheric noise is not included in the observed data. Nevertheless, the alignment of the obtained reconstruction and ground truth indicates that the underlying reasoning is valid. As the 5 Hz high-pass filter successfully filtered out the atmospheric noise, since there are no fluctuations in the reconstructed signal. From this, it can be concluded that the chopper wheel, in combination with the proposed data-scientific noise removal method, also performs sufficiently when it cannot be assumed that the atmospheric noise can be perfectly subtracted.

7

Conclusion and Discussion

7.1. Conclusion

This thesis proposes a system capable of reconstructing the signal emitted by a DSFG across two different observation modes of the DESHIMA 2.0 spectrometer: (1) continuously staring and (2) position switching. To achieve this, a noise model was developed that captures the behaviour of the PSD of the TLS and photon noise, which was optimized with WLS. For both observation modes, an algorithm was proposed that enables the reconstruction of the emitted signal of the astronomical source.

Once complete, the model was verified. First the noise model was evaluated after which the residual noise present in the system of the two different observation methods were analysed. Finally, the system's performance was evaluated when using only subsets of the dataset. This process yielded the following results:

- A noise model that successfully captures the behaviour of the TLS and photon noise in the channels of DESHIMA 2.0.
- A comparison of the noise levels in the channels of DESHIMA 2.0 when continuously staring mode is used versus position switching.
- An analysis of how well the system performs when only parts of the dataset are used, finding the point where the reconstruction algorithm fails.

7.2. Discussion

Even though the results of the algorithm are promising there is some room for discussion of these results.

7.2.1. Limitations

In this thesis, it is assumed that the atmospheric emission can be perfectly subtracted from the data, from which a simplified version of the measured brightness equation was obtained. This assumption was made to be able to focus on TLS noise mitigation, whereas in reality, residual atmospheric fluctuations persist. The performance of the system when atmospheric noise is present in the observed data requires further investigation.

Moreover, the value for η_{atm} was acquired using simulation of the atmospheric emission only, in reality this data is difficult to isolate.

Additionally, the assumption was made that photon noise can be considered wide-sense stationary, although this assumption can be justified theoretically, in reality, the photon noise deviates from WSS behaviour over time.

7.2.2. Suggestions for Future Validation

The validation of the algorithm relied entirely on simulated DESHIMA 2.0 data, which gives a good indication of performance, however, the model is not yet verified with actual observations of DESHIMA 2.0 data. Therefore, it remains to be demonstrated how well the system would perform on actual data. Additionally, the system was only evaluated using simulated data of one extremely bright DSFG, results might be different when other DSFGs are tested.

Besides, the reconstruction performance was evaluated using a single baseline dataset. To obtain more robust and statistically sound results, multiple distinct slices of the dataset should be tested, and their outcomes averaged.

Moreover, while the Python code for the reconstruction algorithm performs well, it has not yet been optimized for execution time.

7.2.3. Suggestions for Future Work

The environment in which the observation is made also determines the most suitable observation method. For example, when the observation is made from future satellites or space missions, there is no atmospheric emission or absorption present in the observed data. Therefore, the continuously staring method becomes more favourable, as the chopper wheel is heavy and expensive hardware to implement, and the data-scientific noise removal method of this project is proven to be able of reconstructing the signal of the astronomical source when continuously staring at the source. However, this cannot be done with the same accuracy, so there is an expense-accuracy trade-off that needs to be made.

Besides, with further developments in MKID technology, the resulting TLS noise could be decreased. Meaning that the TLS noise knee frequency (f_{knee}) will shift to lower frequencies, this results in the photon noise dominating over the TLS noise until lower frequencies. And since the TLS noise varies slower over time, there is less need for a chopper wheel to eliminate the TLS noise. Meaning that a continuously staring observation might be more favourable as this allows for a longer total integration time of the astronomical source. However, the proposed system to reconstruct the source signal, when the spectrometer is continuously staring at the source, needs improving, as it currently cannot reconstruct the signal when atmospheric emission is present in the observed data.

A

Ethical Implications

This appendix discusses the ethical implications of the project.

A.1. Societal Impact

Developing algorithms that analyse data related to the DESHIMA 2.0 spectrometer significantly enhances research efficiency by making it easy to analyse the data quickly. Analysing this data quickly is beneficial, of course, but it is more important what this data can achieve. Society as a whole wants to know how the world works. The research being done with, among other things, makes it possible for humanity to better understand the universe as a whole. Furthermore, quick data analysis is beneficial not just for the astronomy sector, but for almost all other data analysing sectors.

A.2. Ethical Aspect of Development

During the engineering process, it was discussed that simulated data would be used for this project. This decision was driven by several factors, but one of these factors is that DESHIMA 2.0 data is costly. Not just because it is financially expensive to build the DESHIMA 2.0 spectrometer, but also because the spectrometer is located in Chile, and therefore it would be costly for the environment to travel there. Therefore, using simulated data for the project makes the development a lot less expensive. Moreover, the validation of the project becomes easier when simulated data is used instead of actual recorded data. This is also beneficial for the environment, as testing with simulated data makes it more likely that the test will be successful during the real test, which makes the impact on the environment considerably smaller.

A.3. Ethical Evaluation

Some parts of the DESHIMA 2.0 project could raise some genuine ethical concerns. One of these is the fact that the spectrometer is located in the Atacama Desert in Chile, however, the researchers that work on the project are mostly from the University of Delft and Japan, which are quite far away from Chile. This results in the research group trying to do most of their observations remotely, as to not put unnecessary strain on the environment. // Furthermore, the question might be raised if the research of DSFGs is worth the funding that goes towards it, if it were to be compared to research into more urgent fields, such as climate change. However, this question is not as easily answered as it is difficult to define what makes one type of research 'worth' more than other types of research.

A.4. Reflection on Design Choices

Most of the ethical choices regarding this project were already made before the project started. As mentioned above, in this project, simulated data is used instead of 'real' data. But this does show that the proposers of this project have thought about the impact of the research that is being done. Furthermore, the usage of simulated data in general is ethically good because using the simulated data

to test on makes the tests that are done on the actual spectrometer more likely to succeed. Moreover, all the choices made in this thesis will be readily available on the internet, as is the case for most of the DESHIMA 2.0 data; this is a good thing when thinking about the usage of public resources and transparency.
When designing the system one of the main goals was efficiency. This does not only benefit the system itself but also for the computational cost. Keeping this low is good practice as it reduces both energy consumption and processing time.

A.5. Forward-Looking Responsibility

If this project were to be developed into a real world application, it would be important to keep in mind that the project is developed to help researchers interpret their data. Therefore, it should not be used to replace the individuals currently interpreting the astronomical signals. This is because even though the model works, there are some parts of the system that needs to be checked because the system is not 100% accurate.

B

Algorithm Code

GitHub Link: https://github.com/EndearingUnion/BAP.git

B.1. Weigthed Least Squares

```python
import numpy as np
from scipy.signal import welch
from scipy.optimize import curve_fit
import h5py

from BAP_functions import *


input_file = "blank_tls_only.h5"
output_file_alg = "tls_wls_params_alg.npy"
nperseg = 2**16

# Reading data with tls noise
with h5py.File(input_file, "r") as f:
    data_tls_blank = f["SPAXEL0"]["data"][...]
    times_tls_blank = f["OBSATTRS"]["times"][...]
f.close()

# Reading data with atmospheric noise, photon noise and source
with h5py.File("source_atm_plus_photon.h5", "r") as f:
    data_atm_plus_photon_source = f["SPAXEL0"]["data"][...]
    frequencies_atm_plus_photon_source = f["OBSATTRS"]["frequencies"][...]
    times_atm_plus_phton_source = f["OBSATTRS"]["times"][...]
f.close()

# Reading data with atmospheric noise and source
with h5py.File("source_atm_only.h5", "r") as f:
    data_atm_source = f["SPAXEL0"]["data"][...]
    frequencies_atm_source = f["OBSATTRS"]["frequencies"][...]
    times_atm_source = f["OBSATTRS"]["times"][...]

f.close()

dt = np.mean(np.diff(times_tls_blank))      #Calculates the average time between samples
fs = 1.0 / dt                               #Calculates the sample frequency of DESHIMA 2.0 simulations
num_channels = data_tls_blank.shape[0]

print("sampling␣frequency␣=␣", fs, "Hz")

wls_params_alg = np.zeros((num_channels, 3))                            #Row = channel, col0 = alpha, col1 = beta, col2 = C
data_photon_blank = data_atm_plus_photon_source - data_atm_source       #Creates photon noise only data set

for channel in range(num_channels):

    f_welch, Pxx_ch = welch(data_tls_blank[channel, :], fs=fs, nperseg=nperseg, detrend='constant')         #PSD of TLS noise
    _, Pxx_photon = welch(data_photon_blank[channel, :], fs= fs, nperseg=nperseg, detrend='constant')       #PSD of photon noise

    C_estimate = np.mean(Pxx_photon)            #Calculates the mean of the photon noise and stores in C

    if np.max(Pxx_ch) <= 0:
        wls_params_alg[channel, :] = [0, 0,0]           #For the empty channels
        continue

    intersection_tls_photon = np.where(Pxx_ch < (1.5 * C_estimate))[0]      #Calculates where the photon noise and TLS intersect
    f_upper_bound = f_welch[intersection_tls_photon[0]] * 0.8 if len(intersection_tls_photon) > 0 else 0.18 #Multiplies intersection frequency with 0.8 due to fluctuations


    seg = (f_welch > f_knee_atmos) & (f_welch < f_knee_photon)          #Range where TLS noise is dominant

    f_fit = f_welch[seg]
```

```
    Pxx_fit = Pxx_ch[seg]


    ln_f = np.log(f_fit)        #Makes noise model linear
    x = ln_S = np.log(Pxx_fit)


    w = 1.0 / (Pxx_fit**2 + 1e-30)  #initializes the weight matrix



    HTWH = np.array([                               #Computes H.T @ W @ H
        [np.sum(w), np.sum(w * ln_f)],
        [np.sum(w * ln_f), np.sum(w * ln_f**2)]
    ])


    HTWx = np.array([                               #Compute H.T @ W @ x  (2x1 vector)
        np.sum(w * x),
        np.sum(w * ln_f * x)
    ])


    try:
        theta_hat = np.linalg.inv(HTWH) @ HTWx

        wls_params_alg[channel, 0] = np.exp(theta_hat[0])  # alpha_wls
        wls_params_alg[channel, 1] = -theta_hat[1]         # beta_wls
        wls_params_alg[channel, 2] = C_estimate            #C
    except np.linalg.LinAlgError:
        wls_params_alg[channel, :] = [0, 0, 0]
        continue




np.save(output_file_alg, wls_params_alg)                   #Saves parameters in output file
```

B.2. η_{atm}

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, fftshift
from scipy.signal import butter, buttord, convolve, firwin, freqz, lfilter, kaiserord, iirdesign, welch, wiener
from scipy.signal import spectrogram, impulse, correlate

import h5py

#Reads atmospheric noise only dataset
with h5py.File("blank_atm_only.h5", "r") as f:
    data_atm_blank = f["SPAXEL0"]["data"][...]
    frequencies_atm_blank = f["OBSATTRS"]["frequencies"][...]
    times_atm_blank = f["OBSATTRS"]["times"][...]

f.close()

T_P_ATM = 273                           #T_p,atm Kelvin


num_channels = data_atm_blank.shape[0]
num_times = len(times_atm_blank)

print("channels number=", num_channels)

eta_atm = np.zeros((num_channels, num_times)) #Row = channel, collumn = time




for channel in range(num_channels):
    eta_atm[channel, :] = 1 - (data_atm_blank[channel, :] / T_P_ATM)      #eta_atm = 1-T_sky/T_p,atm


np.save("eta_atmosphere.npy", eta_atm )                                     #Saves eta_atm for each point in time
```

B.3. Continuously Staring

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, ifft, fftshift, fftfreq
from scipy.signal import butter, buttord, convolve, firwin, freqz, lfilter, kaiserord,
    iirdesign, welch, wiener
from scipy.signal import spectrogram, impulse, correlate

import h5py
import pickle


with open('source.pkl', 'rb') as f:
    source_dict = pickle.load(f)

f.close()


print(source_dict.keys())

frequencies_source = source_dict['frequencies␣(Hz)']
temp_source = source_dict['source␣temperature␣(K)']


#Loading data
with h5py.File("blank_atm_only.h5", "r") as f:
    data_atm_blank = f["SPAXEL0"]["data"][...]
    frequencies_atm_blank = f["OBSATTRS"]["frequencies"][...]
    times_atm_blank = f["OBSATTRS"]["times"][...]

f.close()


with h5py.File("blank_tls_only.h5", "r") as f:
    data_tls_blank = f["SPAXEL0"]["data"][...]
    frequencies_tls_blank = f["OBSATTRS"]["frequencies"][...]
    times_tls_blank = f["OBSATTRS"]["times"][...]
f.close()

# Reading data with atmospheric and photon noise
with h5py.File("source_atm_plus_photon.h5", "r") as f:
    data_atm_plus_photon_source = f["SPAXEL0"]["data"][...]
    frequencies_atm_plus_photon_source = f["OBSATTRS"]["frequencies"][...]
    times_atm_plus_phton_source = f["OBSATTRS"]["times"][...]
f.close()


with h5py.File("blank_atm_plus_photon.h5", "r") as f:
    data_atm_plus_photon_blank = f["SPAXEL0"]["data"][...]
    frequencies_atm_plus_photon_blank = f["OBSATTRS"]["frequencies"][...]
    times_atm_plus_phton_blank = f["OBSATTRS"]["times"][...]
f.close()



eta_atm = np.load("eta_atmosphere.npy") #Load eta_atm
tls_noise_parameters = np.load("tls_wls_params_alg.npy")
atm_noise_parameters = np.load("atm_noise_param.npy")



T_P_ATM = 273 #T_p,atm Kelvin


num_channels = data_atm_blank.shape[0]
num_times = len(times_atm_blank)
dt = np.mean(np.diff(times_atm_plus_phton_source))


```

```
69
70 #Arrays to store values in
71 x_opt_channels = np.zeros(num_channels)
72 variance_channels = np.zeros(num_channels)
73 noise_level_channels = np.zeros(num_channels)
74
75
76
77 print("Shape␣eta_atm",eta_atm.shape)
78 print("Shape␣freq␣source",frequencies_source.shape)
79 print("Shape␣temp␣source",temp_source.shape)
80
81 data_photon_tls_noise = data_tls_blank + data_atm_plus_photon_blank - data_atm_blank        #
       Results in TLS and photon noise
82
83
84 for channel in range(300):
85
86     #Loads parameters for noise model
87     alpha = tls_noise_parameters[channel, 0]
88     beta = tls_noise_parameters[channel, 1]
89     C = tls_noise_parameters[channel, 2]
90
91
92
93     #Data model \vec{y} = \vec{a}x + \mathbf{n}
94     a_vec = eta_atm[channel, :]             #Atmoshpere transmission coefficient
95
96     y_vec = (a_vec * temp_source[channel]) + data_photon_tls_noise[channel]     #Observed
           data vector y
97
98     freqs = fftfreq(num_times, d=dt)
99     S_nn = np.zeros_like(freqs)
100
101    nonzero_mask = freqs != 0
102
103
104    S_nn[nonzero_mask] = alpha * np.abs(freqs[nonzero_mask])**-beta + C         #Creates the
           noise esitmation
105
106
107
108
109    W = 1.0 / np.sqrt(S_nn)                     #Computes the whitening filter
110    y_vec_tilde = np.real(ifft(W * fft(y_vec))) #Whitening of vector y
111    a_vec_tilde = np.real(ifft(W * fft(a_vec))) #Whitening of vector a
112
113
114    #Calculated the reconstructed emitted signal of the astronomical source
115    x_opt = np.dot(a_vec_tilde, y_vec_tilde) / np.dot(a_vec_tilde, a_vec_tilde) #x optimal
116
117    x_opt_channels[channel] = x_opt
118
119
120    #Performence metrics
121    sum_a_vec_tilde = np.sum(a_vec_tilde**2)
122    sum_a_vec       = np.sum(a_vec**2)
123    variance_channels[channel] = (sum_a_vec)/ (sum_a_vec_tilde**2 )
124    noise_level_channels[channel] = np.sqrt(variance_channels[channel])
125
126 np.save("x_optimal_cs.npy", x_opt_channels )
```

B.4. Position Switching

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.fft import fft, ifft, fftshift, fftfreq
import h5py, pickle

with open('source.pkl', 'rb') as f:
    source_dict = pickle.load(f)

frequencies_source = source_dict['frequencies␣(Hz)']
temp_source = source_dict['source␣temperature␣(K)']


#Loading data
with h5py.File("blank_atm_only.h5", "r") as f:
    data_atm_blank = f["SPAXEL0"]["data"][...]
    times_atm_blank = f["OBSATTRS"]["times"][...]

with h5py.File("blank_tls_only.h5", "r") as f:
    data_tls_blank = f["SPAXEL0"]["data"][...]

with h5py.File("blank_atm_plus_photon.h5", "r") as f:
    data_atm_plus_photon_blank = f["SPAXEL0"]["data"][...]
    times_atm_plus_phton_blank = f["OBSATTRS"]["times"][...]
    az_switched = f["OBSATTRS"]["az"][...]

eta_atm = np.load("eta_atmosphere.npy")         #Loads calculated eta_atm
tls_noise_parameters = np.load("tls_wls_params_alg.npy")

num_channels = data_atm_blank.shape[0]
dt = np.mean(np.diff(times_atm_plus_phton_blank))

data_photon_tls_noise = data_tls_blank + data_atm_plus_photon_blank - data_atm_blank        #Results in TLS and photon noise

samples_per_cycle = 12                                              #Amount of DESHIMA 2.0 samples in a chopper cycle
num_cycles = data_photon_tls_noise.shape[1] // samples_per_cycle    #Amount of chopper cycles made during the full observation


#Arrays to store values in
x_opt_channels = np.zeros(num_channels)
variance_channels = np.zeros(num_channels)
noise_level_channels = np.zeros(num_channels)
snr_channels = np.zeros(num_channels)

t0 = 6 * dt             #Time of one observation phase (on- or off-source)
scaling_factor = 1      #To add atmospheric noise to the observed data


z_store = np.zeros((300, num_cycles))
a_store = np.zeros((300, num_cycles))

for channel in range(300):

    #Loads parameters for noise model
    alpha = tls_noise_parameters[channel, 0]
    beta = tls_noise_parameters[channel, 1]
    C = tls_noise_parameters[channel, 2]

    a_vec = eta_atm[channel, :].copy()
    n_vec = data_photon_tls_noise[channel]              #The TLS and Photon noise contained in the observed data
    atm_vec = data_atm_blank[channel]                   #Used to add atmospheric noise in observed data


    #Creates empty arrays to store values in
    y_on_series = np.zeros(num_cycles)
```

```
    y_off_series = np.zeros(num_cycles)
    a_on_series = np.zeros(num_cycles)

    for i in range(num_cycles):
        start_on_source = i * samples_per_cycle         #Begin on-source observation
        end_on_source = start_on_source + 6             #End on-source observation
        start_off_source = end_on_source                #Begin off-source observation
        end_off_source = start_off_source + 6           #End off-source observation

        atm_on_mean  = np.mean(atm_vec[start_on_source:end_on_source])
        atm_off_mean = np.mean(atm_vec[start_off_source:end_off_source])

        #Computes the mean of on-source observation of 6 samples
        y_on_series[i] = np.mean((a_vec[start_on_source:end_on_source] * temp_source[channel
            ]) + n_vec[start_on_source:end_on_source]) + scaling_factor * atm_on_mean

        #Computes the mean of off-source observation of 6 samples
        y_off_series[i] = np.mean(n_vec[start_off_source:end_off_source]) + scaling_factor *
            atm_off_mean

        a_on_series[i] = np.mean(a_vec[start_on_source:end_on_source])


    y_stream = y_on_series - y_off_series           #Performs chopper modulation
    a_stream = a_on_series

    fs = 1 / dt
    fs_downsampled = 1 / (dt * samples_per_cycle)       #Downsampled frequency
    fs_phase = 1 / (dt * 6)                             #Frequency of on- and off-phase

    N_z = len(y_stream)
    f_bins = fftfreq(N_z, d = 1 / fs_downsampled)


    #Creates the noise esitmation
    def s_nn_mod(f_bin):
        f_new = np.where(f_bin == 0, 1e-6, f_bin)
        estimated_noise = (alpha / (np.abs(f_new)**beta)) + C       #Estimates the TLS and
            photon noise
        transfer_function = 4 * (np.sin(np.pi * f_new * t0)**2)     #Altering in noise PSD
            due to chopper modulation
        return transfer_function * estimated_noise

    s_nn_downsampled = 0.5 * (s_nn_mod(f_bins / 2.0) + s_nn_mod((f_bins / 2.0) - (fs_phase /
        2.0)))     #Downsampling with D =2



    W = (1 / np.sqrt(s_nn_downsampled))                 #Computes the whitening filter

    z_vec_tilde = np.real(ifft(W * fft(y_stream)))      #Whitened vector z
    a_vec_tilde = np.real(ifft(W * fft(a_stream)))      #Whitened vector a

    x_opt = np.mean(z_vec_tilde / a_vec_tilde)          #Calculated the reconstructed emitted
         signal of the astronomical source
    x_opt_channels[channel] = x_opt

    z_store[channel] = z_vec_tilde
    a_store[channel] = a_vec_tilde
    sum_a_vec_tilde = np.sum(a_vec_tilde**2)
    sum_a_vec       = np.sum(a_stream**2)
    variance_channels[channel] = (sum_a_vec)/ (sum_a_vec_tilde**2 )
    noise_level_channels[channel] = np.sqrt(variance_channels[channel])

np.save("z_whitened_chopper.npy", z_store)
np.save("x_optimal_chopper_whitened_atmosphere_included.npy",   x_opt_channels)
np.save("a_vec_tilde_chopper_whitened.npy", a_store)
np.save("chopper_variance_channels_whitened.npy",    variance_channels)
np.save("chopper_noise_level_channels_whitened.npy", noise_level_channels)
np.save("chopper_snr_channels_whitened.npy",         snr_channels)
```

C

Figures

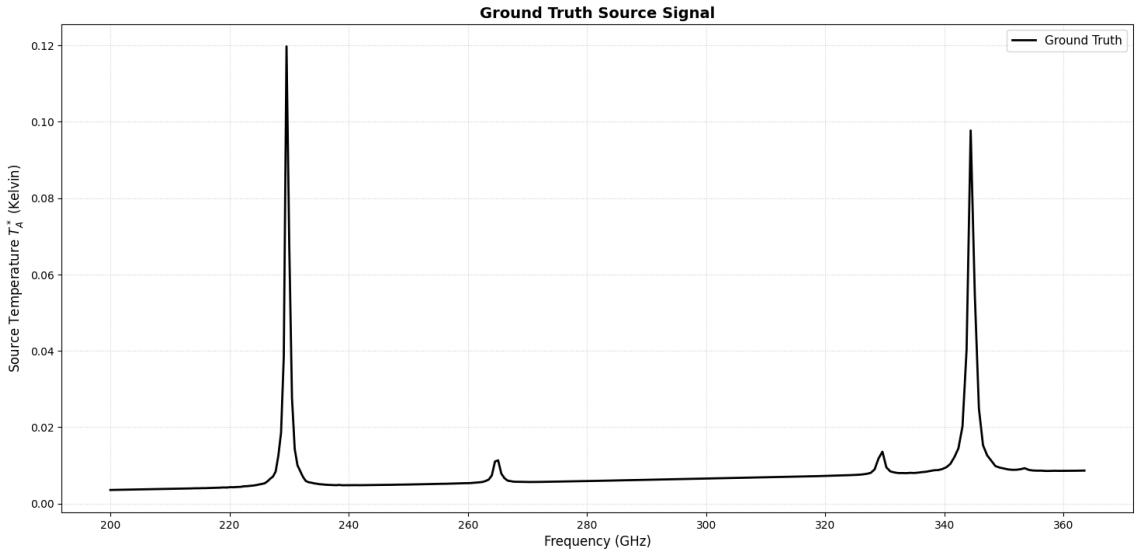

Figure C.1: Ground Truth

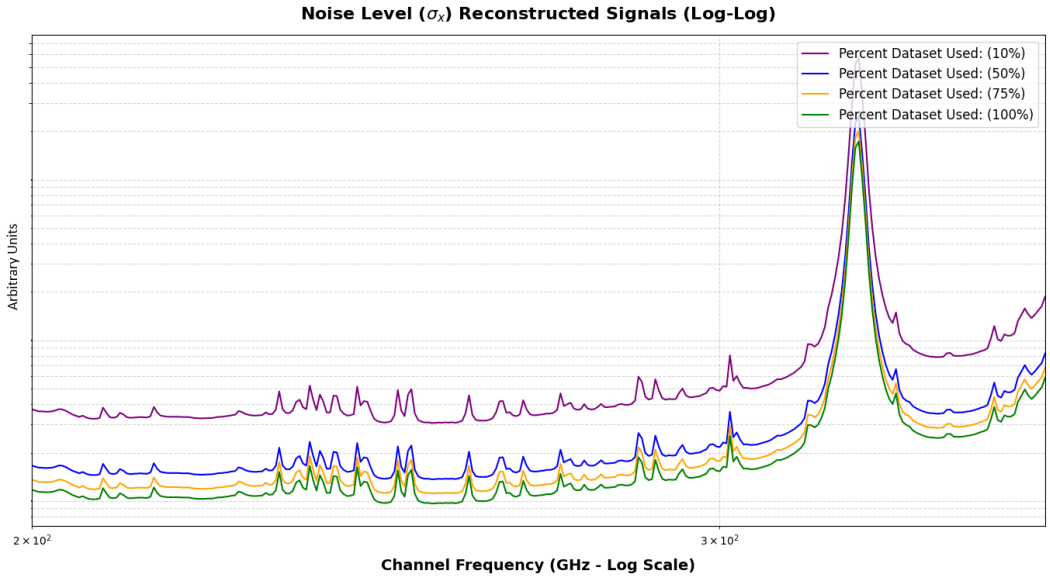

Figure C.2: Noise level Continuously Staring, **Noise Level Not Correct**

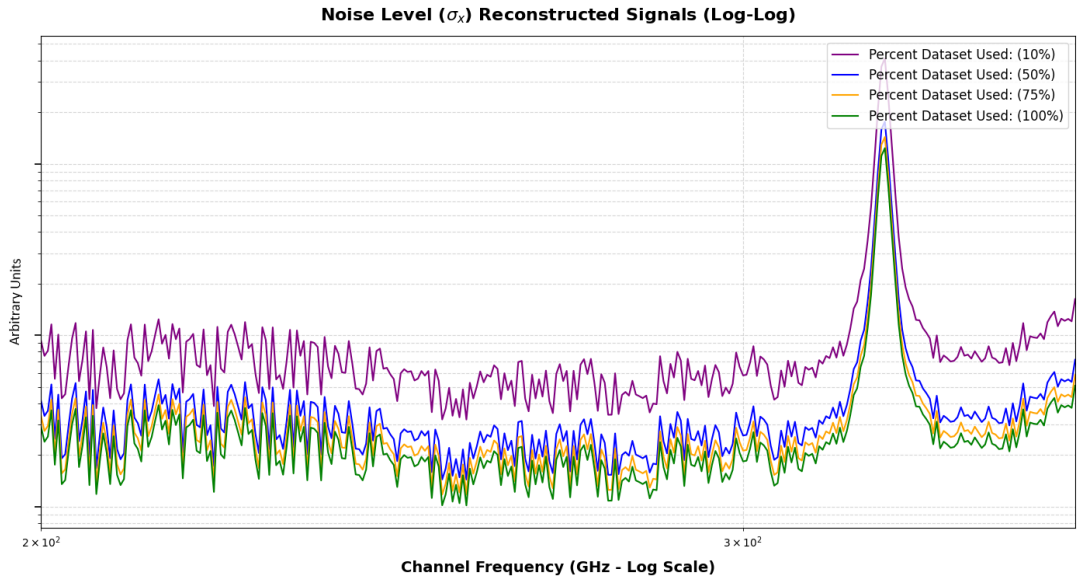

Figure C.3: Noise level Position Switching, **Noise Level Not Correct**

D

Derivations

D.1. Derivation of Equation 5.9

Equation 5.8 minimizes the total error, which is defined as:

$$J = \|\vec{y} - \vec{a}x\|^2_{\mathbf{R_n}^{-1}}$$

Which is equivalent to:

$$J = (\vec{y} - \vec{a}x)^T \mathbf{R_n}^{-1}(\vec{y} - \vec{a}x)$$

Multiplying yields the following:

$$J = \vec{y}^T \mathbf{R_n}^{-1}\vec{y} - \vec{y}^T \mathbf{R_n}^{-1}\vec{a}x - \vec{a}^T \mathbf{R_n}^{-1}\vec{y}x + \vec{a}^T \mathbf{R_n}^{-1}\vec{a}x^2$$

Since $\vec{y}^T \mathbf{R_n}^{-1}\vec{a}x = \vec{a}^T \mathbf{R_n}^{-1}\vec{y}x$, the equation becomes:

$$J = \vec{y}^T \mathbf{R_n}^{-1}\vec{y} - 2\vec{a}^T \mathbf{R_n}^{-1}\vec{y}x + \vec{a}^T \mathbf{R_n}^{-1}\vec{a}x^2$$

To minimize the error the derivative has to be taken with respect to x:

$$\frac{dJ}{dx} = -2\vec{a}^T \mathbf{R_n}^{-1}\vec{y} + 2\vec{a}^T \mathbf{R_n}^{-1}\vec{a}x$$

Solving for x when $\frac{dJ}{dx} = 0$ yields:

$$2\vec{a}^T \mathbf{R_n}^{-1}\vec{y} = 2\vec{a}^T \mathbf{R_n}^{-1}\vec{a}x$$

Therefore:

$$x_{\text{opt}} = \frac{\vec{a}^T \mathbf{R}_n^{-1}\vec{y}}{\vec{a}^T \mathbf{R}_n^{-1}\vec{a}}$$

D.2. Derivation of Equation 5.10

Building from Equation 5.10, since $\mathbf{R_n}^{-1} = \mathbf{R_n}^{-1/2}\mathbf{R_n}^{-1/2}$, as $\mathbf{R_n}^{-1}$ it is a covariance matrix and therefore symmetric, the following holds:

$$x_{\text{opt}} = \frac{\vec{a}^T\mathbf{R_n}^{-1/2}\mathbf{R_n}^{-1/2}\vec{y}}{\vec{a}^T\mathbf{R_n}^{-1/2}\mathbf{R_n}^{-1/2}\vec{a}}$$

Due to symmetry, $\mathbf{R_n}^{-1/2} = \mathbf{R_n}^{-T/2}$, therefore the equation becomes as follows:

$$x_{\text{opt}} = \frac{\vec{a}^T\mathbf{R_n}^{-T/2}\mathbf{R_n}^{-1/2}\vec{y}}{\vec{a}^T\mathbf{R_n}^{-T/2}\mathbf{R_n}^{-1/2}\vec{a}}$$

Rearranging the terms leads to the following:

$$x_{\text{opt}} = \frac{(\mathbf{R_n}^{-1/2}\vec{a})^T(\mathbf{R_n}^{-1/2}\vec{y})}{(\mathbf{R_n}^{-1/2}\vec{a})^T(\mathbf{R_n}^{-1/2}\vec{a})}$$

When defining $\tilde{\vec{y}} = \mathbf{R_n}^{-1/2}\vec{y}$ and $\tilde{\vec{a}} = \mathbf{R_n}^{-1/2}\vec{a}$, the following equation is achieved:

$$x_{\text{opt}} = \frac{\tilde{\vec{a}}^T\tilde{\vec{y}}}{\tilde{\vec{a}}^T\tilde{\vec{a}}}$$

D.3. Derivation of Equations 5.19 and 5.20

The difference between two subsequent on-source and off-source observations is as follows:

$$n_{\text{diff}}(t) = n(t) - n(t - \Delta t)$$

When taking the Fourier Transform ($\mathcal{F}$) the following is obtained:

$$N_{\text{diff}}(f) = \mathcal{F}\{n(t)\} - \mathcal{F}\{n(t - \Delta t)\}$$

Applying the Fourier Time-Shifting Theorem [19] results in:

$$\mathcal{F}\{n(t - \Delta t)\} = N(f) \cdot e^{-j2\pi f \Delta t}$$

By substituting and rearranging the following equation is obtained:

$$N_{\text{diff}}(f) = N(f) - N(f) \cdot e^{-j2\pi f \Delta t} = N(f) \cdot \left(1 - e^{-j2\pi f \Delta t}\right)$$

From this expression, the linear filter transfer function, $H(f)$, mapping the input noise spectrum to the modulated output spectrum is as follows:

$$H(f) = 1 - e^{-j2\pi f \Delta t}$$

To obtain the transformation of the noise PSD ($S_{nn}(f)$), the squared magnitude of the transfer function has to be found, which is as follows:

$$\begin{aligned} |H(f)|^2 &= H(f) \cdot H^*(f) \\ &= \left(1 - e^{-j2\pi f \Delta t}\right)\left(1 - e^{j2\pi f \Delta t}\right) \\ &= 1 - e^{j2\pi f \Delta t} - e^{-j2\pi f \Delta t} + e^0 \\ &= 2 - \left(e^{j2\pi f \Delta t} + e^{-j2\pi f \Delta t}\right) \end{aligned}$$

By applying Euler's identity, which states that $\cos(\theta) = \frac{e^{j\theta} + e^{-j\theta}}{2}$ the equation can be simplified to:

$$|H(f)|^2 = 4\sin^2(\pi f \Delta t)$$

D.4. Derivation of Equation 5.21

The maximum power gain of the noise due to the chopper modulation is:

$$|H(f)|^2 = 4\sin^2(\pi f \Delta t) = 4$$

The cut-off frequency is where the maximum power gain is halved:

$$4\sin^2(\pi f_{\text{-3dB}} \Delta t) = 2$$

Which can be rewritten as:

$$\sin(\pi f_{\text{-3dB}} \Delta t) = \frac{1}{\sqrt{2}}$$

Which is equals to:

$$\pi f_{\text{-3dB}} \Delta t = \frac{\pi}{4}$$

Therefore, the cut-off frequency can be found as follows:

$$f_{\text{-3dB}} = \frac{1}{4\Delta t}$$

D.5. Derivation of Equation 5.22

The following equation is used for downsampling by a factor D [19]:

$$S_{\text{downsampled}}(\omega) = \frac{1}{D}\sum_{k=0}^{1} S_{\text{diff}}\left(\frac{\omega - 2\pi k}{D}\right)$$

Expanding the summation for $D = 2$, $k = 0$ and $k = 1$ yields:

$$S_{\text{downsampled}}(\omega) = \frac{1}{2}\left[S_{\text{diff}}\left(\frac{\omega}{2}\right) + S_{\text{diff}}\left(\frac{\omega - 2\pi}{2}\right)\right]$$

Where ω (radians/sample):

$$\omega = 2\pi\frac{f}{f_s}$$

For $k = 0$:

$$\frac{\omega}{2} = \frac{1}{2}\left(2\pi\frac{f}{f_s}\right) = 2\pi\frac{\left(\frac{f}{2}\right)}{f_s}$$

Resulting in $\frac{f}{2}$
For $k = 1$:

$$\frac{\omega - 2\pi}{2} = \frac{\omega}{2} - \pi$$

Using ($\frac{\omega}{2} = 2\pi\frac{f}{2f_s}$), gives:

$$2\pi\frac{f}{2f_s} - \pi\left(\frac{2f_s}{2f_s}\right) = 2\pi\left(\frac{f - f_s}{2f_s}\right) = 2\pi\frac{\left(\frac{f-f_s}{2}\right)}{f_s}$$

Resulting in $\frac{f-f_s}{2}$

Hence, the following equation is obtained:

$$S_{nn,\text{downsampled}}(f) = \frac{1}{2}\left[S_{nn,\text{diff}}\left(\frac{f}{2}\right) + S_{nn,\text{diff}}\left(\frac{f - f_s}{2}\right)\right]$$

D.6. Derivation of Equation 6.1

Definition Variance of a vector can be defined as:

$$\text{var}(\hat{x}) = E[(\hat{x} - x)^2]$$

Here $\hat{\vec{x}}$ is defined as:

$$\hat{\vec{x}} = \frac{\tilde{\vec{a}}^T \tilde{\vec{y}}}{\tilde{\vec{a}}^T \tilde{\vec{a}}}$$

Using

$$\tilde{\vec{a}} = \mathbf{R}_n^{-1/2} \vec{a}$$

The formula can be rewritten to:

$$\hat{\vec{x}} = \frac{\tilde{\vec{a}}^T \mathbf{R_n}^{-\frac{1}{2}} (\vec{a}x + \vec{n})}{\tilde{\vec{a}}^T \tilde{\vec{a}}}$$

Which can be rewritten to:

$$\hat{x} = \frac{\tilde{\vec{a}}^T \tilde{\vec{n}}}{\tilde{\vec{a}}^T \tilde{\vec{a}}} + \vec{x}$$

So filling this into the variance formula:

$$\text{var}(\hat{x}) = E\left[\left(\frac{\tilde{\vec{a}}^T \tilde{\vec{n}}}{\tilde{\vec{a}}^T \tilde{\vec{a}}}\right)^2\right]$$

Putting this into the formula:

$$\text{var}(\hat{x}) = E\left[\left(\frac{\vec{a}^T \mathbf{R}_n^{-T/2} \vec{n}}{\tilde{\vec{a}}^T \tilde{\vec{a}}}\right)\left(\frac{\vec{a}^T \mathbf{R}_n^{-T/2} \vec{n}}{\tilde{\vec{a}}^T \tilde{\vec{a}}}\right)^T\right]$$

Which is then rewritten to:

$$\text{var}(\vec{x}) = \left(\frac{\tilde{\vec{a}}^T}{\tilde{\vec{a}}^T \tilde{\vec{a}}}\right) E\left[\tilde{\vec{n}} \tilde{\vec{n}}^T\right] \left(\frac{\tilde{\vec{a}}^T}{\tilde{\vec{a}}^T \tilde{\vec{a}}}\right)^T$$

Here $E[\tilde{\vec{n}} \tilde{\vec{n}}^T] = \sigma^2 \mathbf{I}$, which simplifies to where σ^2 is a scaling factor:

$$\text{var}(\vec{x}) = \sigma^2 \frac{\tilde{\vec{a}}^T \tilde{\vec{a}}}{(\tilde{\vec{a}}^T \tilde{\vec{a}})^2}$$

Which simplifies to:

$$\text{var}(\vec{x}) = \sigma^2 \frac{1}{\tilde{\vec{a}}^T \tilde{\vec{a}}}$$

D.7. Proof that $\sigma = 1$

A whitening filter transforms coloured noise to white noise. By definition, the PSD of the output signal $S_{\text{out}}(f)$ after applying a linear filter is given by:

$$S_{out}(f) = |W(f)|^2 \cdot S_{nn}(f)$$

And the Whiting filter is defined as:

$$W(f) = \frac{1}{\sqrt{S_{nn}(f)}}$$

Filling this in:

$$S_{out}(f) = \left| \frac{1}{\sqrt{S_{nn}(f)}} \right|^2 \cdot S_{nn}(f)$$

And solving it the following is found:

$$S_{out}(f) = \frac{1}{S_{nn}(f)} \cdot S_{nn}(f) = 1$$

According to Wiener-Khinchin theorem and Parseval's identity the total varience can be found with:

$$\sigma^2 = \int S_{out}(f)\, df = 1$$

References

[1] M. Rybak et al. “Deshima 2.0: Rapid Redshift Surveys and Multi-line Spectroscopy of Dusty Galaxies”. In: *Journal of Low Temperature Physics* 209 (May 5, 2022), pp. 1–13. DOI: 10.1007/s10909-022-02730-y.

[2] K. Karatsu et al. *DESHIMA 2.0: A 200-400 GHz Ultra-wideband Integrated Superconducting Spectrometer*. Jan. 29, 2026. DOI: 10.48550/arXiv.2601.21603. arXiv: 2601.21603[astro-ph]. URL: http://arxiv.org/abs/2601.21603.

[3] Kaushal Marthi. “Modelling kinetic inductance detectors and associated noise sources”. In: ().

[4] J.R. Pardo, J. Cernicharo, and E. Serabyn. “Atmospheric transmission at microwaves (ATM): an improved model for millimeter/submillimeter applications”. In: *IEEE Transactions on Antennas and Propagation* 49.12 (Dec. 2001), pp. 1683–1694. ISSN: 0018926X. DOI: 10.1109/8.982447. URL: http://ieeexplore.ieee.org/document/982447/.

[5] Omid Noroozian et al. “Two-level system noise reduction for Microwave Kinetic Inductance Detectors”. In: THE THIRTEENTH INTERNATIONAL WORKSHOP ON LOW TEMPERATURE DETECTORS—LTD13. Stanford (California), 2009, pp. 148–151. DOI: 10.1063/1.3292302. URL: https://pubs.aip.org/aip/acp/article/1185/1/148-151/692603.

[6] Z. Pan et al. “Noise Optimization for MKIDs with Different Design Geometries and Material Selections”. In: *IEEE Transactions on Applied Superconductivity* 33.5 (Aug. 2023), pp. 1–8. ISSN: 1051-8223, 1558-2515, 2378-7074. DOI: 10.1109/TASC.2023.3250167. arXiv: 2304.01133[astro-ph]. URL: http://arxiv.org/abs/2304.01133.

[7] Akio Taniguchi et al. “DESHIMA 2.0: Development of an Integrated Superconducting Spectrometer for Science-Grade Astronomical Observations”. In: *Journal of Low Temperature Physics* 209.3 (Nov. 1, 2022), pp. 278–286. ISSN: 1573-7357. DOI: 10.1007/s10909-022-02888-5. URL: https://doi.org/10.1007/s10909-022-02888-5.

[8] Akira Endo et al. “First light demonstration of the integrated superconducting spectrometer”. In: *Nature Astronomy* 3.11 (Aug. 5, 2019), pp. 989–996. ISSN: 2397-3366. DOI: 10.1038/s41550-019-0850-8. arXiv: 1906.10216[astro-ph.IM]. URL: http://arxiv.org/abs/1906.10216.

[9] maybeetree and Arend. *tifuun/gateau: v0.2.8*. Version v0.2.8. Apr. 24, 2026. DOI: 10.5281/ZENODO.17183878. URL: https://zenodo.org/doi/10.5281/zenodo.17183878.

[10] A. Moerman et al. *gateau: an observation simulator for ground-based submillimeter astronomy with integral field units and kinetic inductance detectors*. Apr. 25, 2026. DOI: 10.48550/arXiv.2604.23305. arXiv: 2604.23305[astro-ph.IM]. URL: http://arxiv.org/abs/2604.23305.

[11] Arend Moerman et al. “Alignment and optical verification of DESHIMA 2.0 at ASTE”. In: (2025). URL: https://repository.tudelft.nl/record/uuid:29693def-e51d-4360-8a25-7133d675764d.

[12] Stefanie Brackenhoff. “SPLITTER”. In: (2021). URL: https://repository.tudelft.nl/record/uuid:097bbb7a-ca0c-4847-9773-edb1c9e1d4ab.

[13] George B. Rybicki and Alan P. Lightman. *Radiative processes in astrophysics*. Weinheim, [Germany]: Wiley-VCH Verlag GmbH & Co. KGaA, 2004. 1 p. ISBN: 978-0-471-82759-7 978-3-527-61818-7.

[14] Y Sueno et al. “Characterization of two-level system noise for a microwave kinetic inductance detector comprising niobium film on a silicon substrate”. In: *Progress of Theoretical and Experimental Physics* 2022.3 (Mar. 1, 2022), 033H01. ISSN: 2050-3911. DOI: 10.1093/ptep/ptac023. URL: https://doi.org/10.1093/ptep/ptac023.

[15] Clemens Müller et al. “Interacting two-level defects as sources of fluctuating high-frequency noise in superconducting circuits”. In: *Physical Review B* 92.3 (July 31, 2015), p. 035442. ISSN: 1098-0121, 1550-235X. DOI: 10.1103/PhysRevB.92.035442. URL: https://link.aps.org/doi/10.1103/PhysRevB.92.035442.

[16] Steven M. Kay. *Fundamentals of statistical signal processing*. 1 online resource (3 volumes) : illustrations vols. Prentice Hall signal processing series. Englewood Cliffs, N.J.: Prentice-Hall PTR, 2013. URL: http://catalog.hathitrust.org/api/volumes/oclc/26504848.html.

[17] *6.3. Weighted least-squares estimation — MUDE textbook*. URL: https://mude.citg.tudelft.nl/book/2025/observation_theory/03_WeightedLSQ.html.

[18] Roy D. Yates and David J. Goodman. *Probability and stochastic processes: a friendly introduction for electrical and computer engineers*. Third edition. 1 online resource (xvi, 496 pages) vols. Hoboken, NJ: John Wiley & Sons, Inc., 2014. ISBN: 978-1-118-80438-4. URL: https://www.safaribooksonline.com/library/view/title/9781118324561/?ar?orpq&email=%5Eu.

[19] Proakis. *Digital Signal Processing, 4e*. Google-Books-ID: xvC0DwAAQBAJ. Pearson Education India. 1157 pp. ISBN: 978-93-325-2653-2.

[20] Steven Smith. *The Scientist and Engineer’s Guide to Digital Signal Processing*. ISBN: 0-9660176-3-3. URL: https://www.dspguide.com/.

[21] Frederik Michel Dekking et al. *A Modern Introduction to Probability and Statistics: Understanding Why and How*. Springer Texts in Statistics. London: Springer, 2005. ISBN: 978-1-85233-896-1 978-1-84628-168-6. DOI: 10.1007/1-84628-168-7. URL: https://link.springer.com/10.1007/1-84628-168-7 (visited on 06/16/2026).