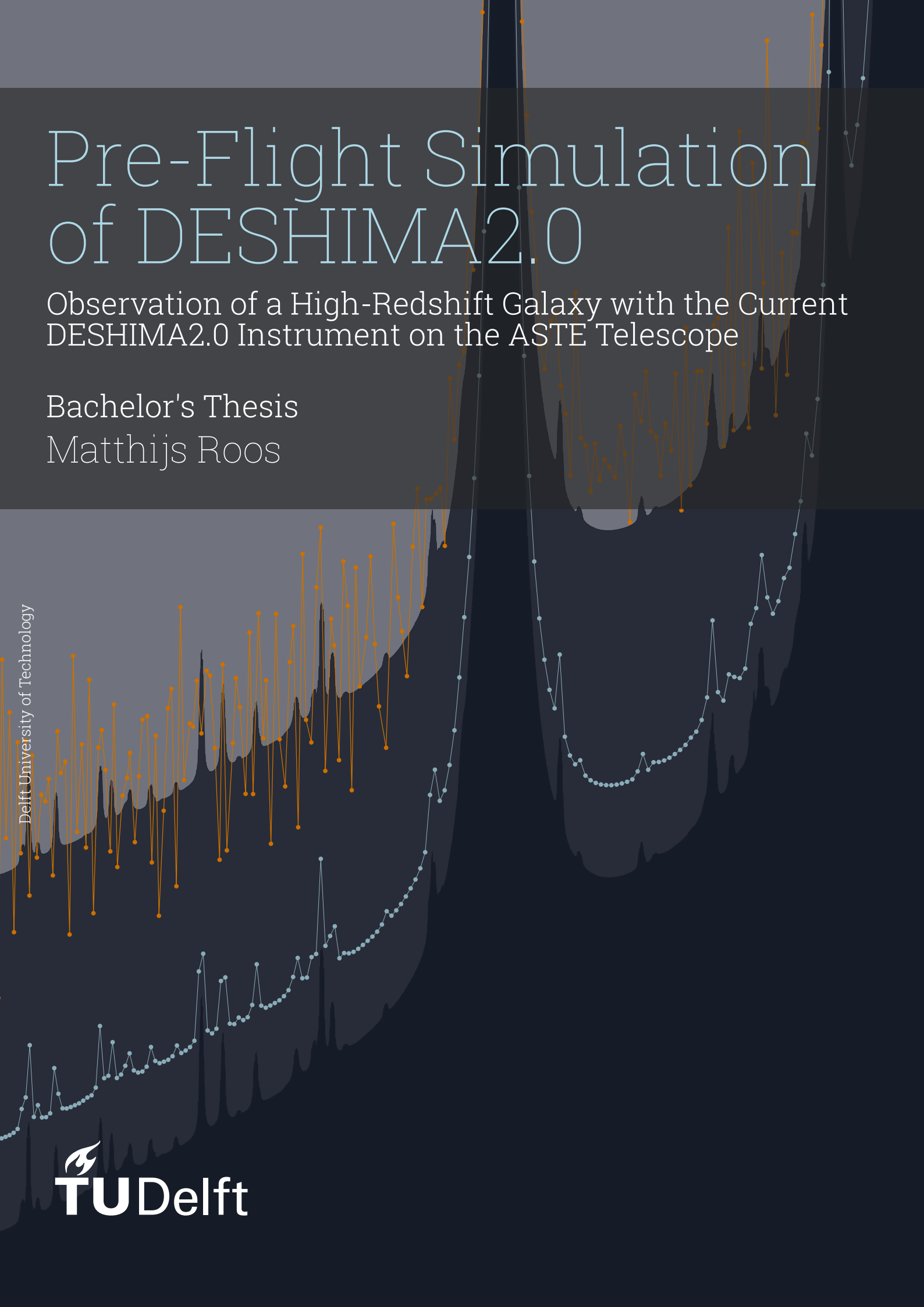
Pre-Flight Simulation of DESHIMA2.0

Observation of a High-Redshift Galaxy with the Current DESHIMA2.0 Instrument on the ASTE Telescope

Bachelor's Thesis

Matthijs Roos



Delft University of Technology

TU Delft

Pre-Flight Simulation of DESHIMA2.0

Observation of a High-Redshift Galaxy with the Current DESHIMA2.0 Instrument on the ASTE Telescope

by

Matthijs Roos

to obtain the degree of Bachelor of Science in Applied Physics
at the Delft University of Technology

Responsible Instructors:	Dr. A. (Akira) Endo	Supervisor
	Prof. Dr. A.F. (Sander) Otte	Examiner
	Dr. M. (Matus) Rybak	Examiner
Study Programme:	Bachelor of Applied Physics	(AP)
Faculty:	Faculty of Applied Sciences	(AS)

Cover: A selected view of the Minimum Detectable Line Flux of the current DESHIMA2.0 instrument (orange curve). Characterizing the sensitivity of the instrument, it gives an indication of when spectral lines in its frequency range can be observed.

Abstract

To better understand the formation of stars in dusty star-forming galaxies (DSFGs), and the evolution of this type of submillimeter galaxies (SMGs), it is of high significance to perform submillimeter/far-infrared surveys. The Deep Spectroscopic High-redshift Mapper 2.0 (DESHIMA2.0) on the ASTE telescope in Chile has been designed to observe the redshifted emission lines of these DSFGs to measure both their (spectroscopic) redshifts and molecular compositions. DESHIMA2.0 is designed to observe across the 220-440 GHz frequency band using its 347 channel integrated superconducting spectrometer (ISS) chip.

The current DESHIMA2.0 instrument has been tested in the lab. Its 332 filter channels with center frequencies in the range of 204-391 GHz have a spectral resolution of $f/\delta f \approx 340$. Their coupling efficiencies can be approximated by Lorentzian functions.

The gravitationally lensed ultraluminous high-redshift DSFG J1329+2243 with redshift $z = 2.04$ shows emission lines of molecular gases like CO and H_2O in the frequency band of DESHIMA2.0. Using the Time-dependent End-to-end Model for Post-process Optimization (TiEMPO), an 8-hour observation of the J1329+2243 galaxy has been simulated for both the lab-measured chip and the designed chip. Two measures of sensitivity, the noise equivalent flux density (NEFD) and the minimum detectable line flux (MDLF), are theoretically derived and subsequently used to be compared with the results of the simulations. The results yielded from these simulations are ultimately used to report on the overall performance of the lab-measured chip.

To run a TiEMPO simulation using the lab-measured chip, adaptations had to be made to the model for it to accept customizable chip data. Within TiEMPO, variable spectral resolutions and coupling efficiencies had to be introduced as well as a crucial addition to enable the creation of a new filterbank. Given the results of the simulations, it can be stated that this implementation of the lab-measured chip has been successful.

After applying an ON-OFF (dual) sky chopping technique to the simulated observation to cancel fluctuations of the atmospheric transmission, the atmosphere-corrected antenna temperature of the J1329+2243 galaxy could be found. The standard deviation of the noise showed to be scaling inversely proportional to the square root of the integration time. For five separate emission lines of the galaxy within the range of 200-310 GHz, a signal-to-noise ratio (SNR) analysis was performed and compared to the estimated theoretical proportionality to the square root of the integration time. For lines that show overlap in the spectrum, the detection was proven to be more difficult. Optimizations in the strategy used to define these signal-to-noise ratios would improve this finding.

The performance of the current DESHIMA2.0 instrument is sufficient to detect (SNR$\geq$5) the bright CO$(7-6)$ line of an ultraluminous high-redshift galaxy like J1329+2243 after 5.5 minutes of observation time. After 50 minutes of observation time, it is also capable of identifying the CO$(6-5)$, the $H_2O(211-202)$, the [CI]$(2-1)$, and the CO$(8-7)$ emission line.

Contents

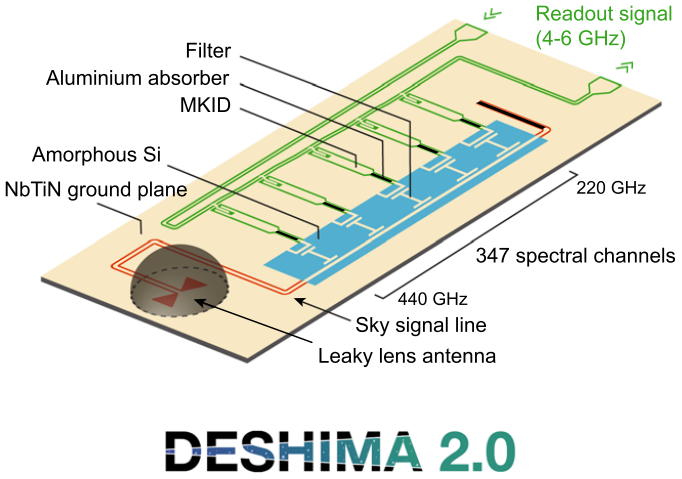
1

Introduction

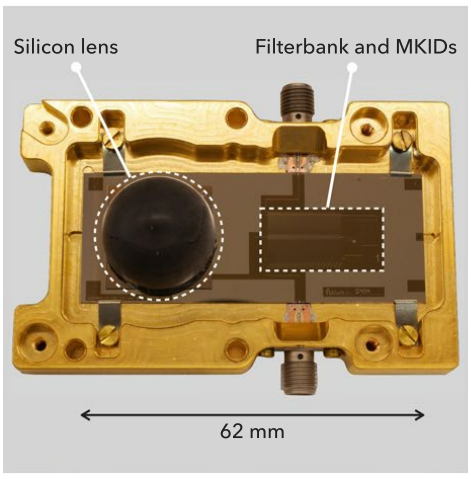
Looking at the night sky from one's backyard is like using a time machine. Light from even the closest stars, other than our Sun, take several years to reach the Earth. This allows us to look back in time and observe key processes that took place in the early Universe.

To unravel the cosmic history of star- and galaxy-formation, it is vital to perform observations of early stages of galaxies. The ones with the highest rates of star formation, the dusty star-forming galaxies (DSFGs), are primarily luminous in the submillimeter (far-infrared) band [1]. This poses a challenge, as the large redshift range of interest ($1+z \sim$ 1-10) corresponds to a spectral bandwidth spanning several hundreds of GHz [2].

To be able to map dusty galaxies in terms of their physical conditions and spectroscopic redshifts across this wide bandwidth, the DEep Spectroscopic HIgh-redshift MApper (DESHIMA) was developed [2], [3]. DESHIMA, an integrated superconducting spectrometer using superconducting filterbank technology as well as microwave kinetic inductance detectors (MKIDs) [4], would allow rapid (overnight) observations of DSFGs using bright emission lines [5]. This measurement would be faster than what is currently possible on this scale using conventional spectrometers, while only using a few cm^2 in chip area [2]. The first realization of DESHIMA, covering a frequency band of 332-377 GHz, proved to be successful in its first light [2]. For the development of DESHIMA2.0, the design of the first was improved upon, and the scalability of it was used to cover a larger range of 220-440 GHz (see figure 1.1) with better coupling efficiencies [5]. Following its predecessor, DESHIMA2.0 will be mounted in the ASTE [6], a 10-m submillimeter telescope located in the Atacama desert (Chile), later this year.



(a) Chip design of DESHIMA2.0



(b) First-fabricated DESHIMA2.0 chip

Figure 1.1: The design and the first realization of DESHIMA2.0. The incoming sky signal is coupled by a leaky-lens antenna to an array of filter channels, the filterbank. Each individual (band-pass) filter is connected to a microwave kinetic inductance detector (MKID), which allows coupling to incoming signals of specific frequencies. The readout signal will then be passed on to be used for analysis of the measurement. Both images obtained from [5].

The goal of this project is to estimate the total system performance of DESHIMA2.0 on the ASTE telescope in observing DSFGs using the most recent filterbank chip that has been tested in the lab. To do so, the sensitivity of the chip will be characterized first. Subsequently, an observation of the spectrum of a high-redshift galaxy will be simulated to ultimately predict the time needed to detect the emission lines of the galaxy. This galaxy will be chosen based on its ultraluminous character, making it a strong candidate for first observations during the flight[1] of DESHIMA2.0.

Throughout this report, several figures contain plots that were made using Plotly Graph Objects. Plotly allows users to create interactive figures. This feature has been made use of. By offering a copy of the majority of the plots, one can interact through zooming, panning, and hide/show curves to get a better sense of the information hidden in these. The plots for which an interactive copy is available, have a hyperlink added in their captions of the figure as [Interact].

[1] DESHIMA2.0 will not literally 'fly'. DESHIMA is a collaboration between the Delft University of Technology, SRON, Leiden University, the University of Tokyo, Nagoya University, and NAOJ. As DESHIMA closely collaborates with the space institute SRON, it tends to use its jargon. Therefore, it is their habit to use the term 'flight' to refer to the commissioning of DESHIMA2.0.

2

Observing Galaxies

While most of the Universe is empty space, a distribution of matter is seen throughout it as matter tends to group together due to gravitational attraction. Most of the time, matter is seen in large gas clouds, which contain mostly light elements that were formed right after the nucleosynthesis. When such a gas cloud reaches a density high enough to collapse, the formation of stars and even a galaxy might be initiated [7]. Large clusters of matter can be found structured in those galaxies, where gas, dust, and stars collected in a disk rotate around a center region, where the density is at its highest. Metals are seen less often in the Universe, as they can only form within stars and be spread when those stars die through supernovae explosions [8].

To better understand the formation and stages of stars and galaxies, as well as processes in the early Universe, there is need to observe gas clouds that are yet to collapse and galaxies that show high levels of star formation. A type of the latter are the dusty star-forming galaxies (DSFGs). These galaxies are characterized by having abundant molecular gas and form stars at a rapid rate. They are a type of submillimeter galaxies (SMGs), that are luminous (bright) at wavelengths (λ) shorter than a millimeter or, equivalently, frequencies (f or ν) that are higher than roughly 300 GHz, specified as far-infrared [8]. Spectral lines, caused by (discrete) emissions or absorptions of photons, appear in these bright spectra. Namely, in this range of frequencies, emission lines of molecular gases like CO and H_2O can be observed [1]. Since these frequencies are known and thus recognizable, it is possible to figure out which gases and other matter an observed galaxy consists of and even indicate how much of that specific molecular gas is present just by looking at their spectra.

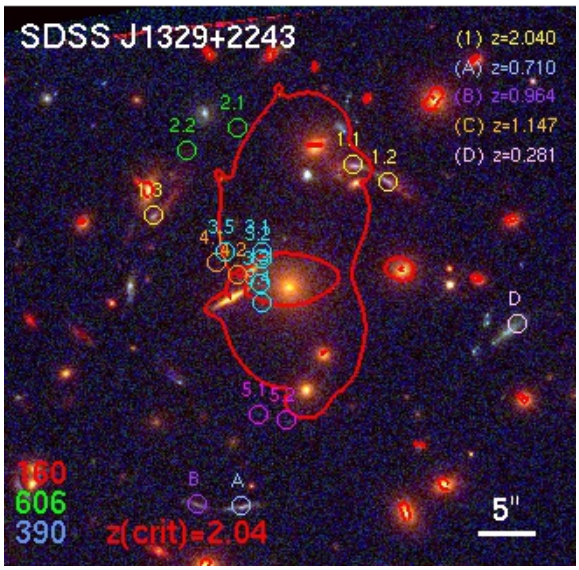


Figure 2.1: A map of the SDSS J1329+2243 cluster with z =0.443 [9]. The yellow encircled points 1.1, 1.2, and 1.3 all show a bright gravitationally lensed galaxy at redshift z =2.04. Image obtained from [10].

2.1. Galaxy Spectrum

Of the total power radiated by a source, also called the luminosity L [W], only a fraction can reach an observer. To better describe this effect, flux F [W m^{-2}] is introduced. Flux describes the luminosity of the source that passes through a unit area at distance d [8]. As radiation can go in all three spatial directions, the total flux of an isotropic source is thus found by dividing the luminosity by the surface area of a sphere with radius d.

$$F = \frac{L}{4\pi d^2} \tag{2.1}$$

In the field of astronomy, it is quite common to express the luminosity of a source in terms of the total solar luminosity $L_\odot = 3.828 \cdot 10^{26}$ W.

The flux from the source is in reality a distribution of fluxes with infinitesimal bandwidths. The summation of those fluxes would yield the total flux. This way, the flux density[1] can be introduced: a flux per unit frequency F_ν [Jy][2]. Here, the unit jansky is defined as Jy $= 10^{-26}$ W m^{-2} Hz^{-1}.

$$F = \int_0^\infty F_\nu d\nu \tag{2.2}$$

One can also use the brightness or specific intensity I_ν [W m^{-2} sr^{-1} Hz^{-1}], which relates to the flux density as follows:

$$I_\nu = \frac{\mathrm{d}F_\nu}{\mathrm{d}\Omega} \iff F_\nu = \int I_\nu \mathrm{d}\Omega \tag{2.3}$$

Here, $\mathrm{d}\Omega = \sin\theta \mathrm{d}\theta \mathrm{d}\phi$ is the element of solid angle[3]. For a blackbody, a theoretical object that is both a perfect absorber and emitter [8], the following distribution of specific intensity as a function of its temperature holds:

$$I_\nu = \frac{2h\nu^3}{c^2} \frac{1}{e^{\frac{h\nu}{k_B T}} - 1} \tag{2.4}$$

Here, $h = 6.626 \cdot 10^{-34}$ J s is Planck's constant and $k_B = 1.38 \cdot 10^{-23}$ J K^{-1} is the Boltzmann constant. In this equation, the temperature T [K] is also called the blackbody temperature. Combining equations 2.3 and 2.4 yields the following representation of flux density which can be used to model and look at a galaxy spectrum, using that a blackbody has the characteristic that it has brightness independent of solid angle.

$$F_\nu = \Omega \left(\frac{\nu}{c}\right)^2 \frac{2h\nu}{e^{\frac{h\nu}{k_B T}} - 1} \tag{2.5}$$

2.1.1. Redshift

As most galaxies formed earlier on in the history of the Universe, it is necessary to look at those that are relatively far away from us. Light travels at the finite speed of approximately $c = 2.998 \cdot 10^8$ m s^{-1}, so the light received here on Earth is from the past. Although this is what makes it possible to look at the history of the Universe in general, it also means that the effect of time is seen on the incoming light. In fact, due to expansion of the Universe, wavelengths (and frequencies) of the incoming photons scale accordingly. This scaling is captured in redshift z and is defined through the difference in the emitted wavelength λ_e and the observed wavelength λ_o [8].

$$z = \frac{\lambda_o - \lambda_e}{\lambda_e} = \frac{\Delta\lambda}{\lambda_e} \Rightarrow \nu_o = \frac{\nu_e}{1+z} \tag{2.6}$$

[1] Flux density can also be defined as flux per unit wavelength F_λ which relates to F_ν as $\nu F_\nu = \lambda F_\lambda$. This property is further explored in appendix B.2.

[2] For the sake of clarity, ν is used to denote frequency while flux is used in formulae, to differentiate it properly from F. Once the switch to temperature T is made, frequency will be denoted by f.

[3] Solid angle Ω, being the three-dimensional analogy to the two dimensional angle, has unit steradian sr $= \left(\frac{180^\circ}{\pi}\right)^2$. A sphere is 4π steradians.

As a result, spectral lines will shift to lower frequencies [7]. In general: the larger the redshift, the further away and the 'older' the source[4]. So, galaxies that are more distant have higher redshifts than nearby galaxies.

Peculiar Motion

On top of the radial velocity induced by expansion of the Universe at a rate of H_0, a velocity v_{pec} caused by (random) peculiar motion needs to be added:

$$v_r = H_0 d + v_{pec} \tag{2.7}$$

It should be noted that redshift from a particular galaxy at distance d generally is not one discrete value due to this added peculiar motion. Take a galaxy that is being observed from the plane of its disk. While one side of the disk will have a positive average peculiar velocity (moving away from the observer), the other side will subsequently have a negative average peculiar velocity (moving toward the observer) [8]. There is indeed a certain distribution in the velocity and therefore in the redshift of a galaxy.

Line width

An astronomical spectral line in the flux density spectrum of a galaxy is observed as a distribution with finite width: a function of frequency, with the center frequency defined as having the highest likelihood [7]. The flux density typically takes the form of a Gaussian distribution, making the full width at half maximum (FWHM) of said distribution the width of the line. The width of such a line is commonly expressed in km s^{-1}, which is the unit of velocity width - another measure of line width. These two show the following proportionality:

$$\frac{\Delta\nu}{\nu_0} = \frac{\Delta V}{c} \Rightarrow \Delta V = \frac{c\Delta\nu}{\nu_0} \tag{2.8}$$

Here, ν_0 represents the center frequency of a spectral line, $\Delta\nu$ defines the width of the flux density as a function of frequency, and c is the speed of light. An important feature here is that the FWHM of a line will consequently scale with its center frequency if the velocity width is kept constant.

2.1.2. The Spectrum: Flux Density and Blackbody Temperature

The galaxy that will be used as a reference in this report and as a model for simulations done later in chapter 5, is the ULIRG (ultraluminous infrared galaxy) J1329+2243 shown in figure 2.1. This galaxy is observed to have a redshift of 2.04 and is gravitationally lensed by a foreground cluster at $z = 0.443$ [9], [10]. It is defined as being an ULIRG, because its far-infrared luminosity is $L_{\text{FIR}} = 3.9 \cdot 10^{14}\text{L}_\odot \gtrsim 10^{13}\text{L}_\odot$ [11]. Its luminosity distance is taken to be $D_L = 16060$ kpc (1 pc $\approx 3.086{\cdot}10^{16}$m).

Continuum Radiation

The spectrum of a galaxy does not solely consist of a series of spectral lines, but in fact builds on a continuum created by thermal radiation of dust present in the interstellar medium (ISM) of the galaxy [8]. To model this continuum radiation, results from the ALESS survey [12] (figure 2.2a) were used and rescaled to fit the line fluxes of the J1329+2243 galaxy. The rescaling is done using equation 2.1 and works via a method formulated by Matus Rybak (private communication) that works as follows.

$$\frac{F^*_{\nu,\,\text{J1329}}}{F_{\nu,\,\text{DUST}}} = \frac{L_{\text{FIR,\,J1329}}}{4\pi D^2_{L,\,\text{J1329}}} \frac{4\pi D^2_{L,\,\text{DUST}}}{L_{\text{FIR,\,DUST}}} = \frac{L_{\text{FIR,\,J1329}}}{L_{\text{FIR,\,DUST}}} \left(\frac{D_{L,\,\text{DUST}}}{D_{L,\,\text{J1329}}}\right)^2 \tag{2.9}$$

Here, the median FIR luminosity of the dust in the ALESS survey is $L_{\text{FIR}} = 3.5 \cdot 10^{12}\text{L}_\odot$. The respective luminosity distance is taken to be $D_L = 22400$ kpc.

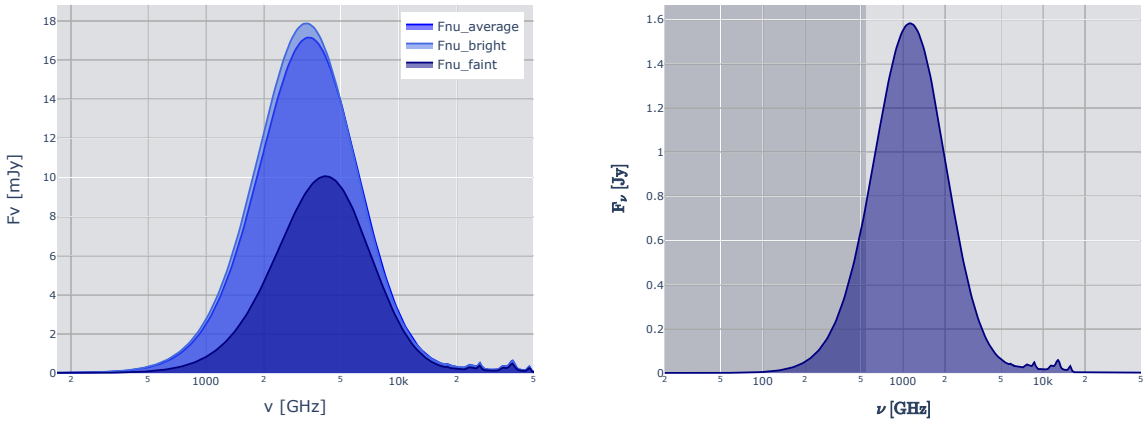
$$F^*_{\nu,\,J1329} \approx \frac{3.9 \cdot 10^{14}\ \text{L}_\odot}{3.5 \cdot 10^{12}\ \text{L}_\odot} \left(\frac{22400\ \text{kpc}}{16060\ \text{kpc}}\right)^2 F_{\nu,\,DUST} \approx 216.77 F_{\nu,\,DUST} \tag{2.10}$$

The found ratio in equation 2.10 of the flux densities approximates 216.77. After rescaling the dust flux density by this factor, the spectrum is scaled once more to have $F_\nu = 0.273$ Jy as for its flux density at a rest frame frequency of 350 GHz, or approximately 850 μm, which was taken from [11].

[4] A derivation of the redshift is given in appendix B.1.

$$F_{\nu,\,J1329} = F^*_{\nu,\,J1329}\frac{0.273\ \text{Jy}}{F^*_{\nu,\,J1329}(\nu_{z=0} = 350\ \text{GHz})} \tag{2.11}$$

The result of the applied rescaling can be seen in figure 2.2b. The distribution in flux density shows resemblance to that of blackbody radiation, as characterized by equation 2.4. However, a second source of continuum radiation may let it deviate from the curve: namely, synchrotron radiation caused by (relativistic) electrons and other ionized matter moving through magnetic fields present in the interstellar medium [8].



(a) Flux density continuum of high-redshift galaxies [Interact]

(b) Model of thermal flux density [Interact]

Figure 2.2: (a) Flux density as a function of rest frame ($z = 0$) frequency based on the results of high-redshift galaxy observations done in [12]. (b) The rescaled average flux density F_ν of (a) to match the line fluxes of spectral lines in the J1329+2243 galaxy. The shaded area shows frequencies up to 550 GHz. Based on method formulated by Matus Rybak, private communication.

Spectrum of a Galaxy

The spectrum shown figure 2.2b can now be used as the base for the spectrum of the J1329+2243 galaxy. By adding the emission line fluxes at their appropriate redshift-corrected center frequencies from [13] with a constant velocity width of 600 km s^{-1}, the model is created (see figure 2.3).

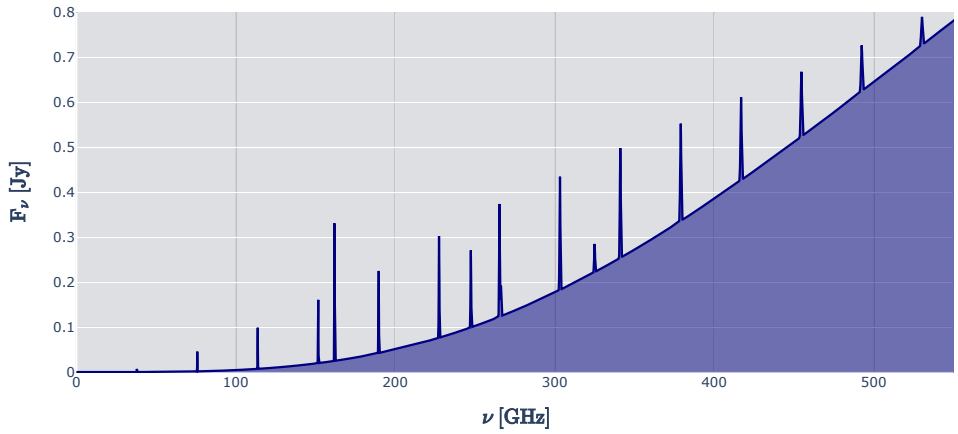


Figure 2.3: Model of the continuous flux density spectrum of the J1329+2243 galaxy with spectral lines shown up to 550 GHz. [Interact]

It can be useful to express the spectrum as brightness temperature instead of flux density. Brightness temperature is the equivalent temperature of a blackbody with the same specific intensity or brightness as the source. For this conversion, the following equation is found when rearranging equation 2.5.

$$T = \frac{h\nu}{k_B \ln\left(\frac{2\Omega h\nu^3}{F_\nu c^2} + 1\right)} \tag{2.12}$$

By introducing the power spectral density as PSD, some terms can be replaced and equation 2.12 can be simplified.

$$\text{PSD} = \frac{\lambda^2}{2\Omega} F_\nu = \frac{c^2}{2\Omega\nu^2} F_\nu \tag{2.13}$$

Substituting equation 2.13 in equation 2.12, the equation for temperature as a function of frequency f is found:

$$T = \frac{hf}{k_B \ln\left(\frac{hf}{\text{PSD}} + 1\right)} \tag{2.14}$$

Applying equation 2.14 to the flux density spectrum shown in figure 2.3, the model of the temperature spectrum of the galaxy is yielded. The PSD also contains the beam solid angle Ω of the used instrument (in this case the ASTE telescope), which in itself is a function of frequency. This results in the deceleration in the slope of the continuum of the spectrum for higher frequencies, to show a maximum toward 600 GHz. The spectrum is shown in figure 2.4.

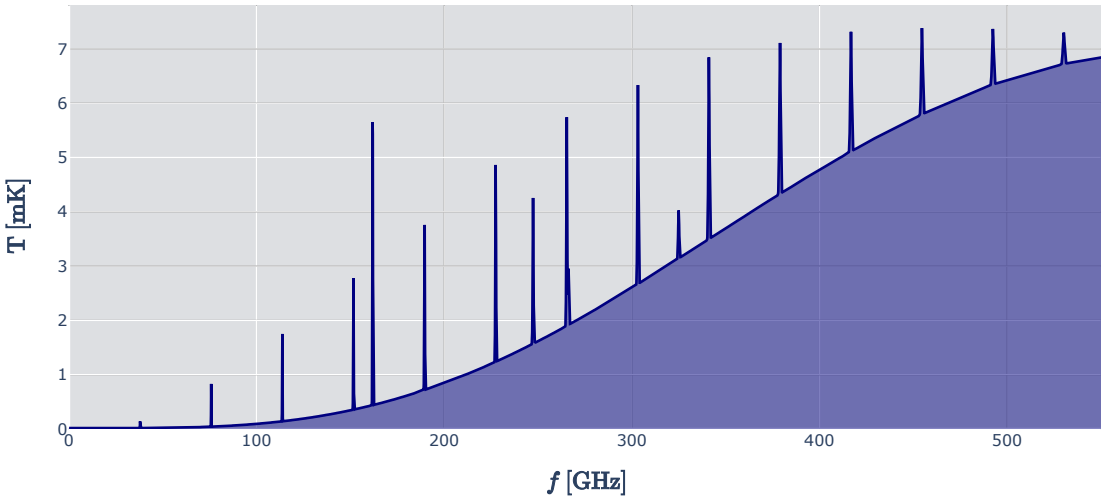


Figure 2.4: Model of the continuous (blackbody) temperature spectrum of the J1329+2243 galaxy with spectral lines shown up to 550 GHz. The spectrum is affected by the frequency dependent beam solid angle of the ASTE telescope. [Interact]

2.1.3. Atmospheric Transmission

Before the radiation of a galaxy reaches the Earth's surface, it will be influenced by the Earth's atmosphere. As the atmosphere is filled with gases, part of the radiation is absorbed according to the atmospheric transmission coefficient $\eta_{atm}(f)$ [2]. Not only is $1 - \eta_{atm}$ of the signal absorbed to affect the observed signal, but also a spectrum of the atmosphere $(1 - \eta_{atm})T_{p,\,atm}$ is added, since the atmosphere itself emits radiation (see figure 2.6b). To illustrate this, equation 2.15 shows what components the observed temperature spectrum consists of [2].

$$T_{sky} = (1 - \eta_{atm})T_{p,\,atm} + \eta_{atm}T_A^* \tag{2.15}$$

Here, T_A^* is used to denote the galaxy spectrum before it is affected by the atmospheric transmission and represents what a telescope would observe if it were placed right outside of the Earth's atmosphere.

The atmospheric transmission coefficient η_{atm} is in principle a function of frequency f, precipitable water vapor PWV [mm], and telescope elevation angle EL [°]. The PWV is the depth of the volume created by collecting all water vapor within an air column of the atmosphere overhead [14]. The higher the PWV, the more radiation will be absorbed by the atmosphere, resulting in a lower η_{atm}, which is the effect seen in figure 2.5a. The EL is the angle the telescope makes with respect to the surface of the Earth. An EL of 0° would consequently be parallel to the surface, while 90° would be perpendicular to the surface. As a lower EL would mean a longer path through the atmosphere, the η_{atm} is at its lowest for the lowest possible angle. The shortest path is thus most ideal, shown in figure 2.5b.

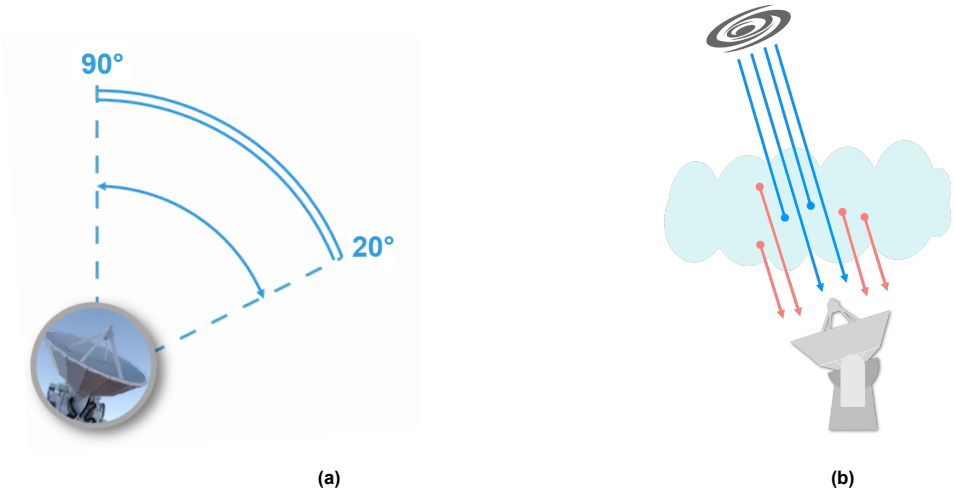
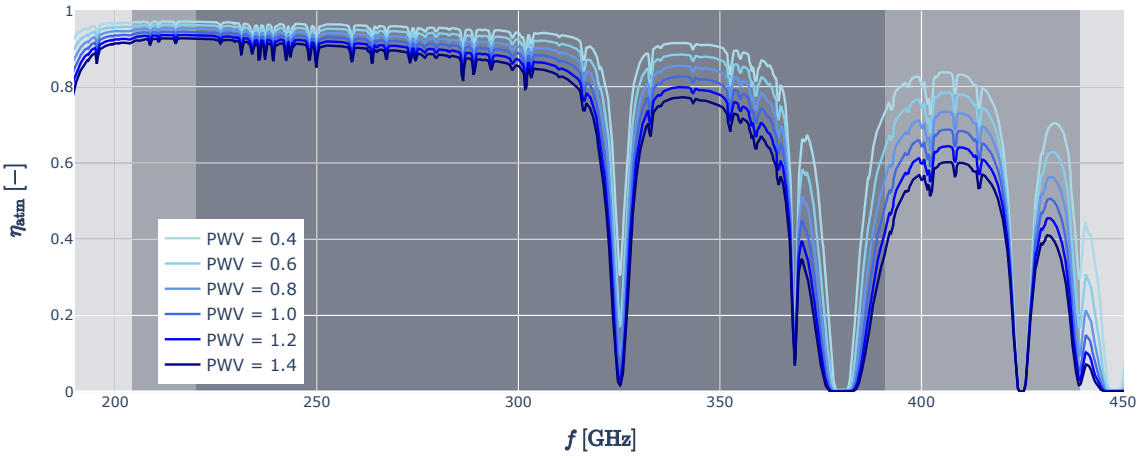
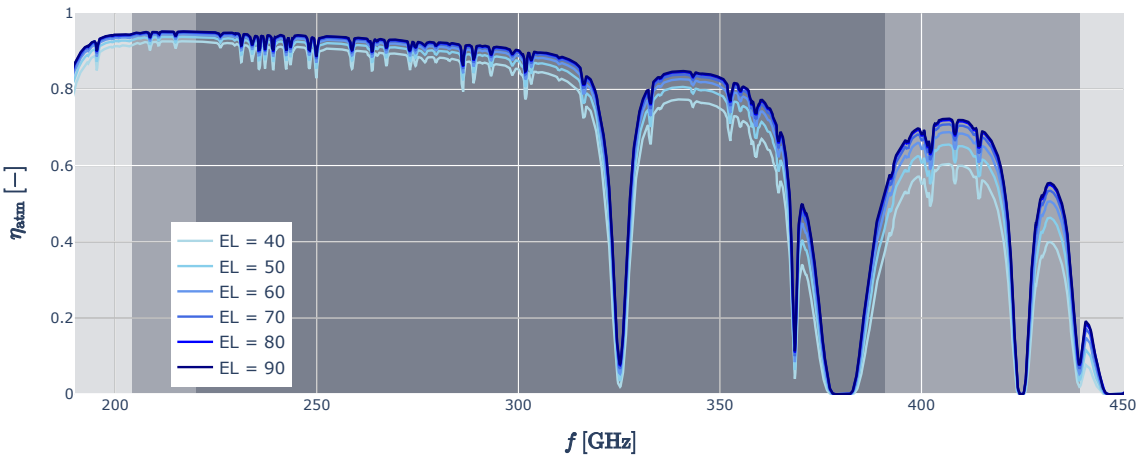


Figure 2.6: (a) A visualization of the elevation angle (EL) of the ASTE telescope. (b) Schematic of radiation from a galaxy reaching a telescope. While going through the atmosphere, certain frequencies of the radiation are absorbed, while other frequencies are emitted. Both images obtained from [15].



(a) η_{atm} plotted for several values of the PWV [mm], while keeping EL at 60°. [Interact]



(b) η_{atm} plotted for several values of the EL [°], while keeping PWV at 1.0 mm. [Interact]

Figure 2.5: η_{atm} plotted for several values of both the precipitable water vapor PWV (a) and the telescope elevation angle EL (b) as a function of frequency. Data obtained using the `deshima-sensitivity` Python package. The background shows the overlap of two frequency bands: roughly 204-391 GHz and 220-440 GHz. Predominant dips in the atmospheric transmission can be seen in the vicinity of 325 GHz, 368 GHz, 380 GHz, 425 GHz, and 439 GHz, making it hard or even impossible to detect emission lines here.

2.2. DESHIMA2.0

Bright dusty star-forming galaxies are seen over a wide range of redshifts, namely $1+z \sim$ 1-10 [1]. To be able to observe galaxies of this type, like the J1329+2243 galaxy, an instrument must be able to detect spectra for the better part of this redshift range. The DEep Spectroscopic High-redshift MApper 2.0 is designed to observe within a frequency band of 220-440 GHz, making it capable of detecting a [CII] emission line (158 μm or 1.9 THz) in the wide redshift range of $z \sim$ 3.3-7.6 [5]. This frequency band would include nine emission lines of the J1329+2243 galaxy spectrum shown in figure 2.4. Being mounted in the ASTE telescope, it benefits from the significantly low PWV values ($\sim$ 1 mm) in the Atacama desert, Chile [16].

2.2.1. The Filterbank

The filterbank of the DESHIMA2.0 chip consists of an array of filter channels that act as band-pass filters; each responds to a different resonance frequency. Each filter is coupled to a microwave kinetic inductance detector (MKID), which are all read out via a single microwave readout line [17], [4]. A single unit cell of the filterbank is shown in figure 2.7. The filter channel characteristics are further explored in chapter 3.

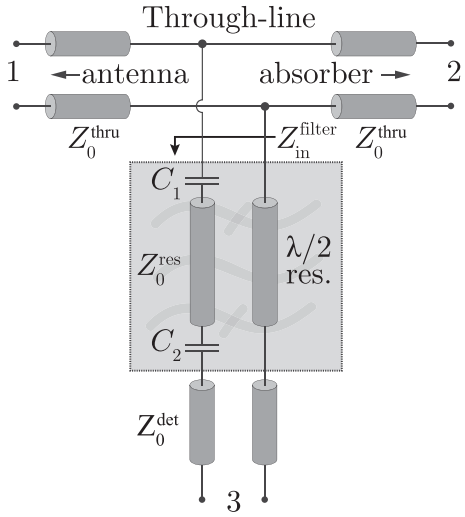
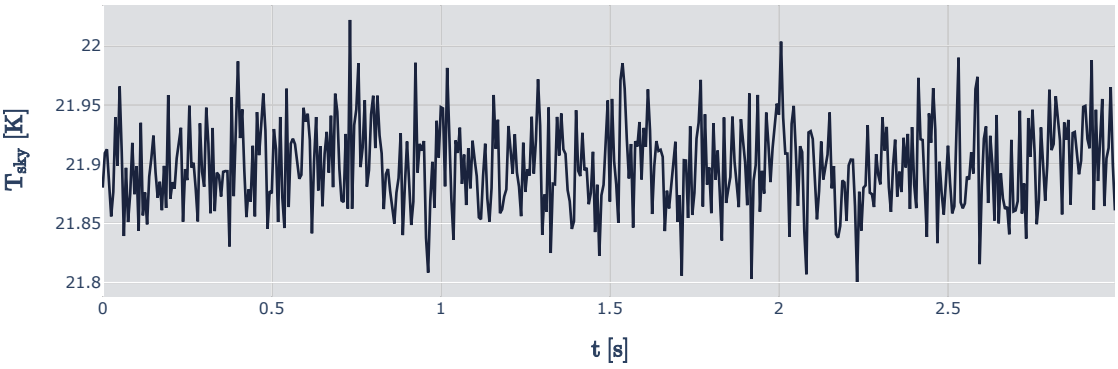


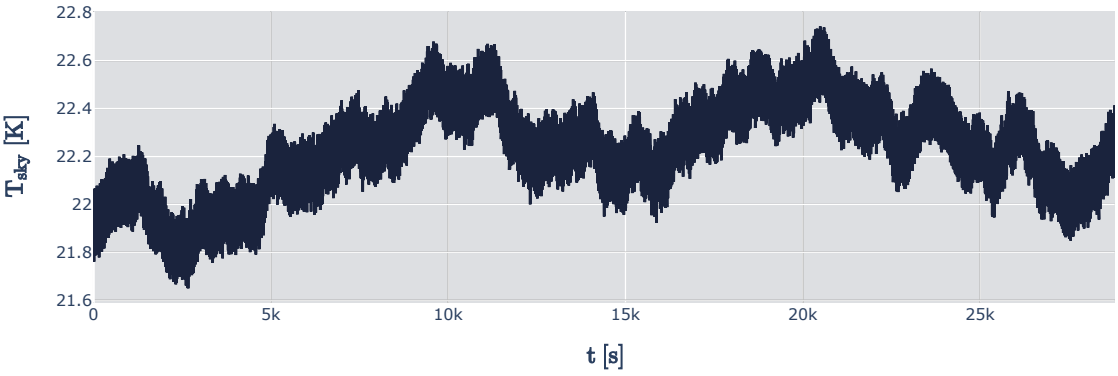
Figure 2.7: Network model of a single three-port resonator band-pass filter. The resonance frequency of this filter is defined by the submillimeter wavelength λ. At port 3, the filter is coupled to the read-out line via an MKID. Image obtained from [17].

2.2.2. Two Types of Noise

When the ASTE telescope positions itself toward a galaxy, the readout signal of the DESHIMA2.0 chip will not directly yield the desired spectrum as sky temperature. Instead, disturbances in the signal can be found in the signal. The first type consists of both noise originating from the incoming light itself (photon noise) and noise inside of the detector (recombination noise), which averages to zero and is therefore removable by taking the average of the signal over a long observation time (it would cancel in the limit) [4]. The major problem that arises when doing a measurement is the existence of fluctuations in the atmosphere: the second type of noise in the signal. As air passes over the observation site, precipitable water vapor values along the line of sight change, causing the atmospheric transmission to change over time (see figure 2.5a). Due to the nature of the two noise sources, they are seen on different time scales: photon and recombination noise are seen on short time intervals, whilst the atmosphere causes fluctuations relatively slowly. The two different time scales are visualized in figure 2.8.



(a) The first type of noise in a DESHIMA2.0 sky measurement. Given as zero-mean noise caused by photon noise of the incoming light and recombination noise in the detector.



(b) The second type of noise in a DESHIMA2.0 sky measurement. Its effect is best seen on a longer time-scale, as it is caused by fluctuations in the transmission of the atmosphere.

Figure 2.8: Both types of noise present in the sky temperature spectrum. [Interact]

2.2.3. Sky Chopping

Fortunately, it is also possible to get rid of the second source of noise (see figure 2.8b). Since the changes in the spectrum of the atmosphere $(1-\eta_{atm})T_{p,\,atm}$ are happening at a relatively slow rate, and measurements of the sky are done at a sampling rate of 160 Hz [18], the PWV can be approximated as being constant over a short time interval (see figure 2.8a). The changes will therefore happen gradually, and taking the time-average of two subtracted sky measurements will tend to converge to zero if they are spaced just close enough. This method of subtracting two sky measurements is called dual chopping.

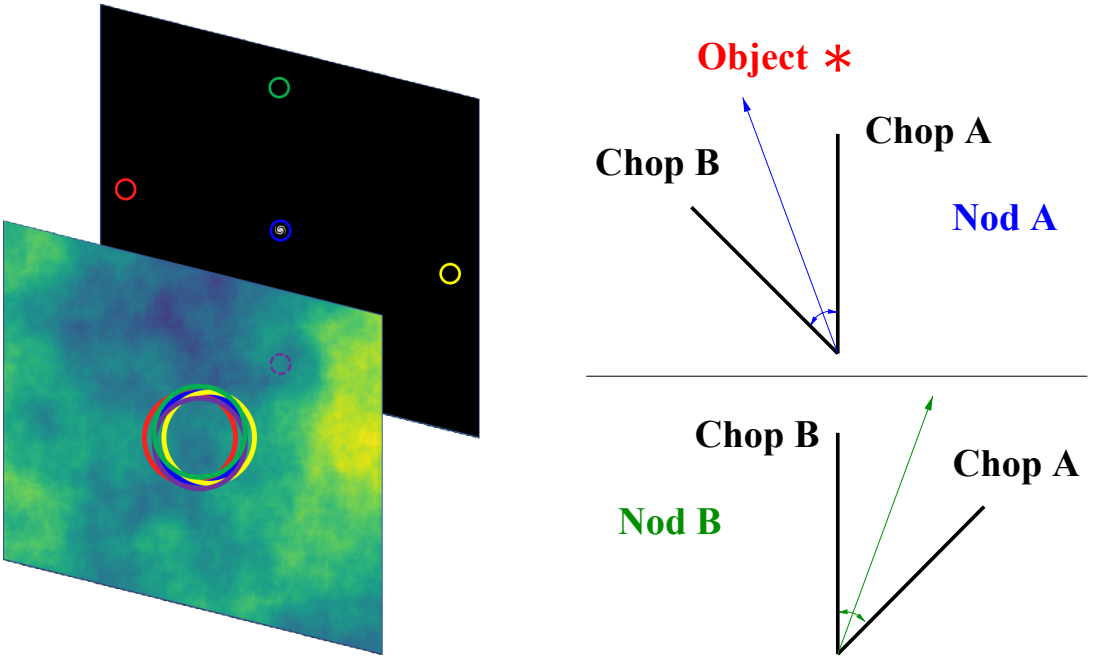
Dual sky chopping

When dual sky chopping is applied to a measurement of the sky with a galaxy (a so-called 'ON' sky measurement) and a measurement of the sky without a galaxy (a so-called 'OFF' sky measurement), this subtraction is called ON-OFF chopping. For the two sky measurements whose difference converges to zero in the limit, it is called OFF-OFF chopping, as neither of these two sky positions contains a galaxy. With the use of an optical instrument, DESHIMA2.0 will operate at a chopping rate of 10 Hz, switching back and forth between two sky positions at ten times per second [18]. Consequently, one dual chopping cycle consists of (sampling rate)/(chopping rate) $= \frac{160\text{ Hz}}{10\text{ Hz}} = 16$ measurements. Using equation 2.15, equation 2.16 follows:

$$T_{sky,\,ON} - T_{sky,\,OFF} \approx \eta_{atm,\,ON} T_A^* \Rightarrow \overline{\Delta T_{sky}} \approx \overline{\eta_{atm}} \cdot \overline{T_A^*} \tag{2.16}$$

Here, the time-average atmospheric transmission can be approximated by $\eta_{atm}(\text{PWV}_0)$, where PWV_0 represents a baseline around which the precipitable water vapor fluctuates over time. Dividing the dual chopped sky temperature by this atmospheric transmission would subsequently yield the atmosphere-corrected antenna temperature T_A^*:

$$T_A^* = \frac{\Delta T_{sky}}{\eta_{atm}(\text{PWV}_0)} = \frac{T_{sky,\,ON} - T_{sky,\,OFF}}{\eta_{atm}(\text{PWV}_0)} \tag{2.17}$$



(a) The five beam positions

(b) Sky chopping

Figure 2.9: (a) The five beam positions on the galaxy (ON) and left, right, up, and down (OFF) colored as blue, red, yellow, green, and purple respectively. It shows their positions on the ARIS-simulated [19] sky (foreground) and their positions at the same distance from the point of observation as the galaxy (background). Image obtained from [15]. (b) A schematic view of sky chopping where the telescope switches between positions ON (chop in the direction of the object) and OFF (chop in a direction next to the object). Dual chopping entails a single nodding position, while ABBA chopping entails alternating between nod A and nod B. Original image from [20].

The approximation in equation 2.16 is needed to account for the spacing between the ON-position and the OFF-position. Theoretically, if they would be at the exact same point in the sky, the chopping would extract the T_A^* of the galaxy almost perfectly. This causes a problem however, as it is not physically possible to get both an ON and an OFF sky measurement from the point in the sky where the galaxy is located (the OFF sky measurement would include the temperature of the galaxy). It is thus vital to choose the right balance between picking an OFF-position that is too close and one that is too distanced. Moreover, the two measurements cannot be performed simultaneously, leading to a difference in the noise added by the sky on the two sky positions.

Ideally, one would align the wind direction and the two sky positions to improve on the approximation. In practice, this is not feasible. Furthermore, extra inefficiencies due to differences in the optics for each sky position have to be taken into account. This would make the dual chopping less accurate.

ABBA sky chopping

To compensate for these inefficiencies, another sky chopping method can be applied: ABBA chopping [20]. For this method, which is a type of chop-nodding, two different OFF-positions are chosen, say to the left (L-position) and to the right (R-position) of the galaxy (ON-position). The instrument would alternate between two nodding positions were it performs n cycles (16 measurements) of dual chopping. Namely, L-ON chopping at nod A, and ON-R chopping at nod B. It would do so in the order A-B-B-A, hence the name. Notice how the ON-position is measured on a different chop (see figure 2.9b), resulting in different optical efficiencies for the ON-position in nod A and nod B respectively.

In practice, the T_A^* can now be better approximated if taken the time-average of the four consecutive dual chopped signals: optical differences, gradual sky fluctuations, and any other systematic effects will be canceled. The atmospheric transmission using the baseline PWV_0 for the galaxy will also match the via chop/nodding approximated atmospheric transmission way better, as the average atmospheric transmission is not in between the ON and OFF-position, but rather in between the two symmetrically chosen OFF-positions, which is equivalent to the ON-position. A better approximation for the atmosphere-corrected antenna temperature would thus be:

$$T_A^* = \frac{2T_{sky,\,ON} - T_{sky,\,L} - T_{sky,\,R}}{4\eta_{atm}(\mathrm{PWV}_0)} \tag{2.18}$$

2.2.4. Simulating the Performance of DESHIMA

deshima-sensitivity

To be able to make estimations on the observation sensitivity of DESHIMA-type spectrometers, a Python package called `deshima-sensitivity` was developed [21]. Using input parameters such as spectrometer characteristics (filter center frequency, spectral resolution, etc.), transmissions and efficiencies (circuit efficiency, on-source fraction, etc.), and atmospheric conditions (telescope elevation angle and precipitable water vapor), it is capable of theoretically modeling DESHIMA-type spectrometers. It then returns measures for their sensitivity like the noise equivalent flux density and the minimum detectable line flux. The last two will be looked into in chapter 3.

TiEMPO

While `deshima-sensitivity` gives an indication of the sensitivity, it still assumes some ideal conditions while performing observations. As a result, the theoretical predictions are not accurate enough to make predictions on the performance of an instrument like DESHIMA2.0. One major source of fluctuations, that is not considered in `deshima-sensitivity`, is the continuous change in the atmospheric conditions on the telescope site. To produce more realistic observation simulations of DESHIMA-type spectrometers, the Time-dependent End-to-end Model for Post-process Optimization (`TiEMPO`) was developed [15], [22]. In `TiEMPO`, the entire instrument, including optics and the telescope itself, is considered. Moreover, a time-dependent phase screen, which is obtained via ARIS [19], is added to simulate the moving atmosphere that affects observations due to its change in precipitable water vapor. A single time frame of the ARIS-simulated sky is shown in figure 2.9a. The model proved to compare to real on-sky measurements [22], making it a perfect tool to use for simulating observations done with DESHIMA2.0.

3

DESHIMA2.0 Chip Characteristics

In this chapter, an analysis of the characteristics of the lab-measured DESHIMA2.0 filterbank chip LT263 will be performed, comparing it to the Preset chip. This Preset chip is one with mostly design parameters of DESHIMA2.0 [18], as well as 'preset' or default parameters used in the `TiEMPO` model. The main features of the Preset chip, in which it deviates from the LT263 chip, are the ideal filter channels used: it has a constant spectral resolution and coupling efficiency of 500 and 0.4 respectively.

3.1. The Chip: LT263

The DESHIMA2.0 instrument makes use of the filterbank chip LT263, which is lab-measured using a photomixer, capable of illuminating the chip with a tunable discrete THz frequency [17]. With a frequency sweep of this instrument performed, the frequency response of the chip was obtained. This raw data was then further analyzed and processed by Kenichi Karatsu, after which the (Lorentzian) filter channel functions found by curve fitting were provided by Akira Endo, private communication. A chip with the same design as the LT263 chip is shown in figure 3.1.

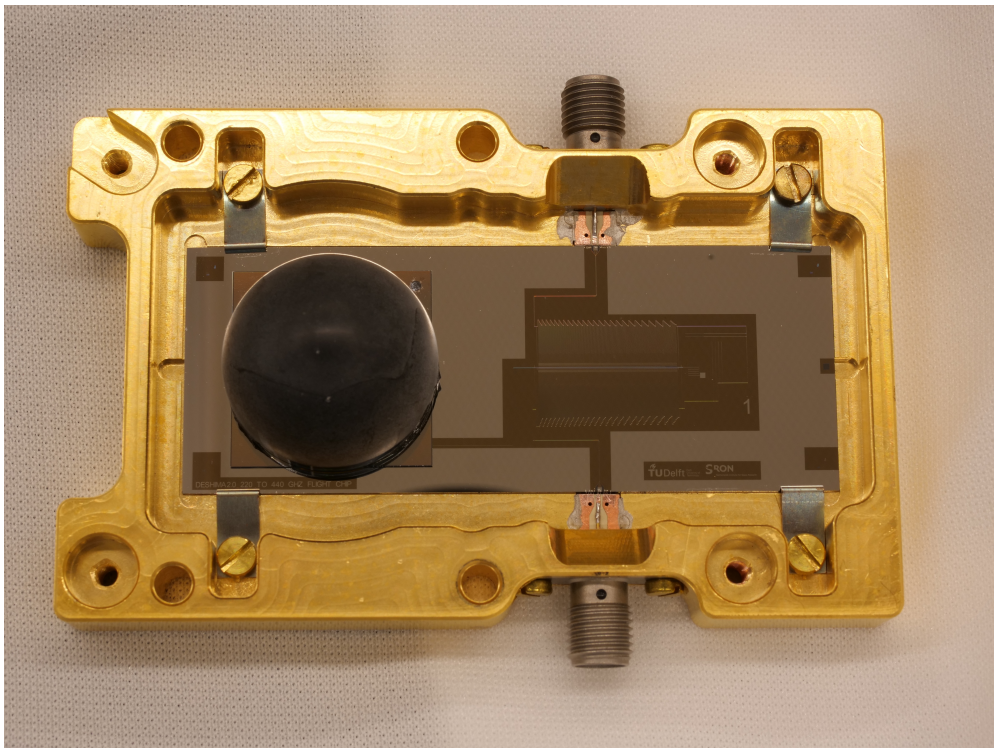


Figure 3.1: An image of the LT223 chip, which has the same design as the LT263 chip. The filterbank is seen to the right of the lens. Image provided by Kenichi Karatsu, private communication.

The design stated that the center frequencies of the 347 filter channels of the DESHIMA2.0 instrument would be spaced logarithmically in the range of 220-440 GHz [5]. The center frequency of each filter channel of the LT263 channel is plotted in figure 3.2 alongside those of the Preset.

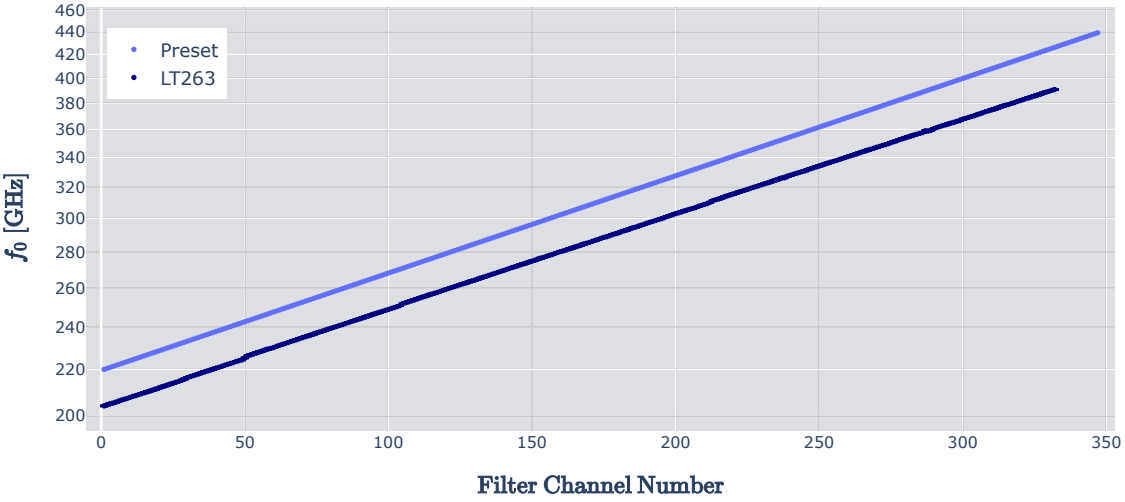


Figure 3.2: Center frequency as a function of filter channel number for both the Preset chip and the LT263 chip. While the Preset has 347 filter channels in the band of 220-440 GHz, the LT263 has 332 filter channels in the band of 204-391 GHz. As designed, and therefore just like the Preset, the center frequencies of the LT263 are close to be spaced evenly on a logarithmic scale. [Interact]

As is visible in figure 3.2, the LT263 chip closely resembles the design feature of the logarithmically spaced filter channels, as a straight line is formed when the f_0 is plotted on a logarithmical axis. However, the number of filter channels, 332, is smaller than desired and in a lower frequency band than designed: roughly 204-391 GHz. Furthermore, the spacing is rather irregular. Some center frequencies have significant offsets, resulting in gaps between consecutive filter channels.

The spectral resolution at the center frequency Q_l is plotted for each filter channel with center frequency f_0 in figure 3.3.

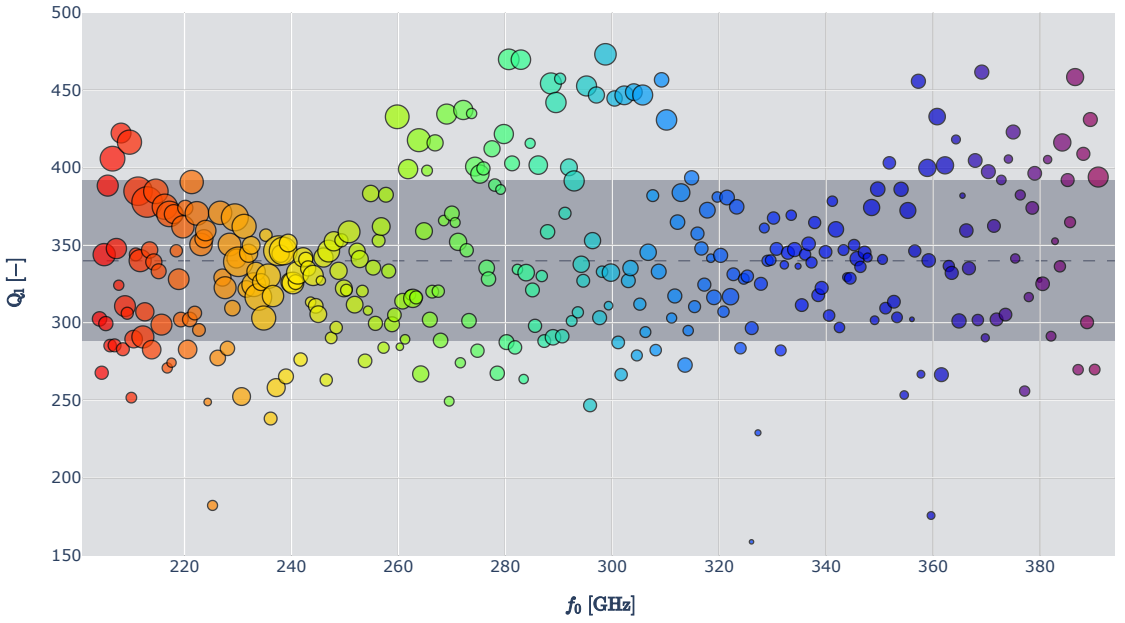


Figure 3.3: Loaded quality factor ($Q_l = 340 \pm 50$) plotted against center frequency for all 332 filter channels of the LT263 chip. The size of the plotted markers represents the circuit efficiency corresponding to the filter channel. [Interact]

The width of the curve fitted filter channel functions is defined by the full width at half maximum FWHM, which is inversely proportional to Q_l (this will be explored in detail in section 3.2). The latter has a mean value of $Q_l = 340 \pm 50$ for the resonant filters. This is still seen as a respectable value, as it was seen as a challenge to obtain a minimum of 300 [23]. This quality factor would correspond to an average velocity width of approximately ΔV = 900 km s^{-1}. The Preset chip has a perfectly constant spectral resolution of 500 (or: ΔV = 600 km s^{-1}), which meets the design goal. Furthermore, the Preset chip is set to a constant optimal circuit efficiency of approximately 0.32 (0.4 times $\frac{\pi}{4}$).

3.2. Channel Response: Coupling Efficiency

For each channel, a resonator is used as a band-pass filter, which was visualized in figure 2.7, such that the power transferred from the antenna (at Port 1 in figure 2.7) to the detector (at Port 3 in figure 2.7, the MKID) is maximized at its resonance frequency. The coupling efficiency or transmission $|S_{31}|^2$, which defines the fraction of the power transmitted, can thus be obtained by analyzing the circuitry of the three-port network. This yields the relation between the coupling efficiency as a function of frequency f, the resonance frequency f_0 of the resonator, the loaded quality factor Q_l, and the internal quality factor Q_i [23], [17]:

$$|S_{31}|^2 = \left| \frac{Q_l}{\sqrt{\frac{2Q_i^2 Q_l^2}{(Q_i - Q_l)^2}} \left(1 + 2jQ_l \left(\frac{f - f_0}{f_0}\right)^2\right)} \right|^2 \tag{3.1}$$

Here, the internal quality factor Q_i reflects dissipation loss in the resonator, which besides power leaks to the ports also influences the loaded quality factor Q_l of the resonator. At maximum or, equivalently, at resonance frequency f_0, the coupling efficiency is equal to $\frac{(Q_i - Q_l)^2}{2Q_i^2}$, which can be substituted in the above given relation. As will be visible in the following formula, the response of the resonator can thus be described by a Lorentzian function:

$$|S_{31}|^2 = \frac{|S_{31}(f = f_0)|^2}{1 + \left(\frac{f - f_0}{\gamma}\right)^2} \tag{3.2}$$

Where γ is half of the full width at half maximum $\left(\frac{\text{FWHM}}{2}\right)$, which results in FWHM $= \frac{f_0}{Q_l}$. Stated as one of the goal parameters, the spectral resolution $R = \frac{f}{\delta f}$ is aimed to be 500. Note that $Q_l = \frac{f_0}{\delta f}$ is thus the spectral resolution at the resonance frequency.

As mentioned in [17], the theoretical maximum of the coupling efficiency is 50%, leaving the remaining 50% of the input power to be split between ports 1 and 2:

$$\lim_{Q_i \to \infty} |S_{31}(f = f_0)|^2 = \lim_{Q_i \to \infty} \frac{(Q_i - Q_l)^2}{2Q_i^2} = \frac{1}{2} \tag{3.3}$$

The coupling of a filter channel to a (spectral) line within the FWHM of the Lorentzian function can be approximated by a box with equal area. The maximum of the coupling efficiency can be taken as the circuit efficiency η_{circuit}, defining the height of the box-approximation of the band-pass filter. However, a factor of $\frac{\pi}{4}$ has to be included to take the deviation from a perfect boxcar function of a Lorentzian as a band-pass filter into account[1]. Consequently, the theoretical maximum circuit efficiency will be $\frac{\pi}{8} \approx$ 39.3%. For the definition of the box-approximation of the band-pass filter, the FWHM is used to define the width, so the origin of the factor can be shown as in equation 3.4.

$$\int_{f_0 - \gamma}^{f_0 + \gamma} |S_{31}|^2 df = \int_{f_0 - \gamma}^{f_0 + \gamma} \frac{|S_{31}(f = f_0)|^2}{1 + \left(\frac{f - f_0}{\gamma}\right)^2} df = \frac{|S_{31}(f = f_0)|^2 \pi \gamma}{2} = \text{FWHM} \cdot \frac{\pi}{4} |S_{31}(f = f_0)|^2 \tag{3.4}$$

The area of the Lorentzian function within the FWHM is thus equal to FWHM $\cdot \eta_{\text{circuit}}$. A visualization of such a box-approximation of a band-pass filter is shown in figure 3.4 as 'Box Line'.

[1] The circuit efficiency used in the `deshima-sensitivity` code is equivalent to the maximum coupling efficiency times a factor of $\frac{\pi}{4}$. This conversion factor is necessary, since η_{circuit} is defined as the mean transmission within the half-power bandwidth, whilst $|S_{31}(f = f_0)|^2$ is the peak height of the Lorentzian. See note in [24].

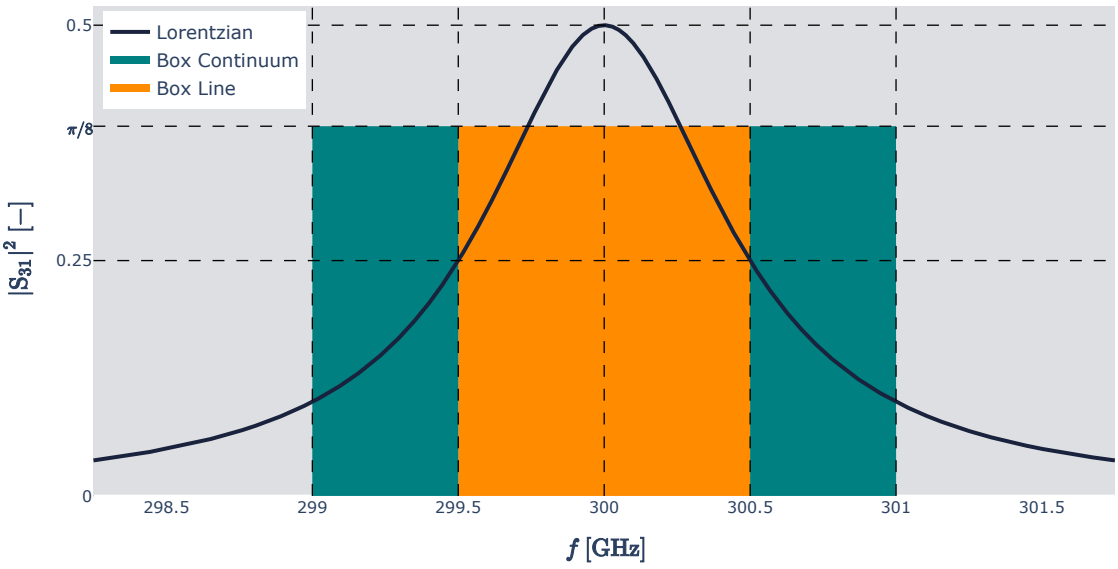


Figure 3.4: A Lorentzian function with its center at f = 300 GHz and maximum value of $|S_{31}|^2 = 0.5$. The full width at half maximum FWHM, in this case at $|S_{31}|^2 = 0.25$, is of size 1 GHz. A box with the same area within the FWHM is plotted as 'Box Line'. Extending this box to 'Box Continuum', whose width is twice the FWHM, the full area of the Lorentzian function is represented. [Interact]

For all 332 channels, the transmission $|S_{31}|^2$ has been plotted as a function of frequency f using Lorentzian functions in figure 3.5. These are defined by the parameters found by curve fitting the responses. The aforementioned theoretical maximum of 50% is not reached by any of the channels due to dissipation of the resonators in the circuit. The coupling efficiency $|S_{31}(f = f_0)|^2$ averages at 14 $\pm$ 4 %.

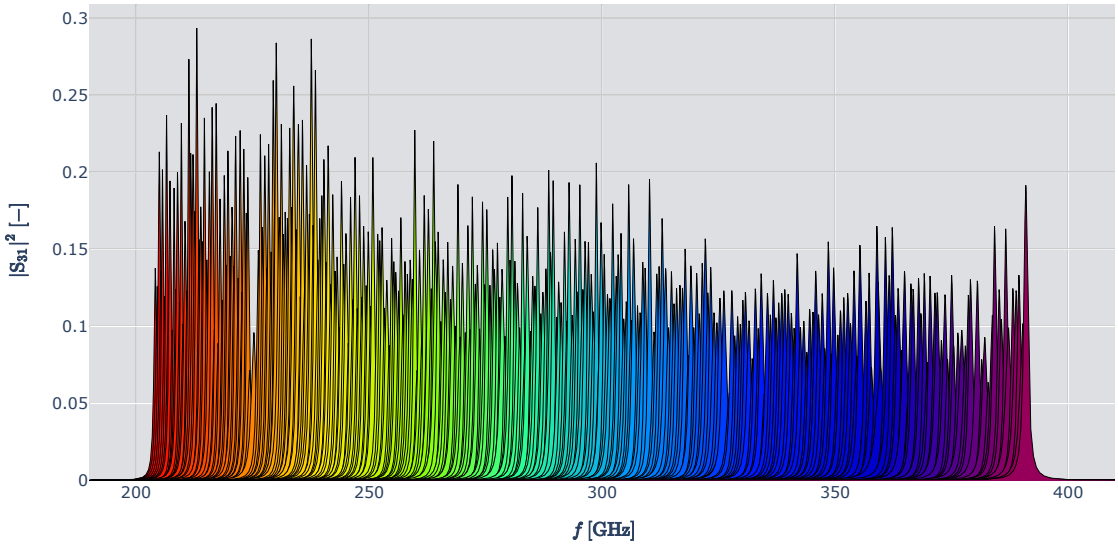


Figure 3.5: Model transmission $|S_{31}|^2$ plotted as a function of frequency f for all 332 filter channels of the LT263 chip, where the transmission curves are approximated by Lorentzians with parameters center frequency f_0 (frequency for which the transmission is maximum), peak height $|S_{31}(f = f_0)|^2$ (maximum transmission), and loaded quality factor Q_l (defined as $\frac{f_0}{\text{FWHM}}$ with FWHM being the full width at half maximum). [Interact]

Overall, peak height seems to decrease as center frequency increases. The center frequency of the channel equals the resonance frequency of the Lorentzian function and can thus be seen as the equivalent resonance frequency of the resonator used for that channel.

The curves centered around lower center frequencies (roughly in the band of 204-240 GHz) perform better at efficiencies ranging from roughly 15% to 20% with some notable outliers of less than 10% and some of more than 25%. For higher center frequencies (roughly in the band of 320-391 GHz), the peak heights drop to roughly 10% to 15%. Some channels, however, still yield peak heights well above 15%.

3.3. Quantifying Sensitivity: NEFD and MDLF

Now that the channel transmission for the spectrum for the chip is known, it is possible to characterize the sensitivity of the chip. This is done through two different measures:

- The noise equivalent flux density (NEFD)
- The minimum detectable line flux (MDLF)

Both rely on and are directly proportional to the detector performance through the noise equivalent source flux (NEF), which defines a threshold for the source flux affecting the detector to overcome the noise equivalent power ($\mathrm{NEP_{KID}}$) [2]. In other words, it equals the flux of the source that is needed to differentiate its signal from noise in the detector, consequently resulting in a signal-to-noise ratio (SNR) of 1. From this, the following equation can be formed, relating the $\mathrm{NEP_{KID}}$ to the NEF [25]:

$$\frac{\mathrm{NEP_{KID}}}{\sqrt{2}_{\mathrm{INT}}}\sqrt{2}_{\mathrm{ON/OFF}} = \eta_{sw}\,\eta_{inst}\,A_g\,\mathrm{NEF} \tag{3.5}$$

Here, $\sqrt{2}_{\mathrm{INT}}$ (with dimension $\mathrm{s}^{-0.5}$) accounts for $\mathrm{NEP_{KID}}$ [$\mathrm{W\,s^{-0.5}}$] being defined for an integration time of 0.5 s while NEF [$\mathrm{W\,m^{-2}}$] is defined for an integration time of 1 s. $\sqrt{2}_{\mathrm{ON/OFF}}$ compensates for ON-OFF chopping of the signal, which lowers the SNR by a factor of $\sqrt{2}$ [24]. A_g represents the geometric area $\pi\left(\frac{D}{2}\right)^2$ of the telescope, which is taken as a disk with diameter D. All relevant efficiencies are captured in the factors η_{sw} and η_{inst}, which respectively project the coupling from a spectral point source to the DESHIMA cryostat window, and the instrument optical efficiency within the bandwidth (the FWHM) of a given channel.

The only factor that is yet to be determined is the $\mathrm{NEP_{KID}}$ itself. There are two profound noise sources in the KID. Namely, photon noise and recombination noise. Since these two sources are uncorrelated, the squares of their noise equivalent powers can be added to give the square of the $\mathrm{NEP_{KID}}$ as a function of the line center frequency f_0 [2], [26].

$$\mathrm{NEP_{KID}} = \sqrt{\underbrace{2\mathrm{P_{KID}}\left(hf_0 + \frac{\mathrm{P_{KID}}}{\mathrm{FWHM}}\right)}_{\text{Photon Noise}} + \underbrace{4\Delta_{\mathrm{Al}}\frac{\mathrm{P_{KID}}}{\eta_{\mathrm{pb}}}}_{\text{Recombination Noise}}} \tag{3.6}$$

In this equation, h represents the Planck constant, Δ_{Al} denotes the superconducting gap energy of aluminium, and η_{pb} is the pair-breaking efficiency [2]. $\mathrm{P_{KID}}$ is the power coupled to the detector.

3.3.1. Effect of the Atmospheric Transmission

Note that the effect of the atmospheric transmission within the bandwidth of a given channel is part of η_{sw} and $\mathrm{P_{KID}}$. More specifically, the former is given as $\eta_{sw} = \eta_{pol}\,\eta_{atm}\,\eta_a\,\eta_{fwd}$, the product of the polarization factor of 0.5 due to the instrument working with a single polarization, the transmission of the atmosphere, the aperture efficiency, and the forward efficiency within the FWHM [27]. The latter is given as $\mathrm{P_{KID}} = \big(\eta_{fwd}(1-\eta_{atm})+(1-\eta_{fwd})\big)\eta_{inst}\cdot\mathrm{B(f,T)}\cdot\mathrm{FWHM}$ where $\mathrm{B(f,T)}$ is the single polarization Planck brightness at frequency f and ambient temperature T, similar to equation 2.4 [2]. Throughout the equation for the NEF, the atmospheric transmission thus really leaves a mark. As a result, the 'fingerprint' of the atmospheric transmission will be quite noticeable in the sensitivity of the detector as well, as the atmosphere is (almost) opaque to some frequencies in the bands of the channels. η_{atm} is plotted as a function of the center frequency f_0 for all the channels of both the LT263 and the Preset chip in figure 3.6 for PWV = 1 mm and EL = 60°.

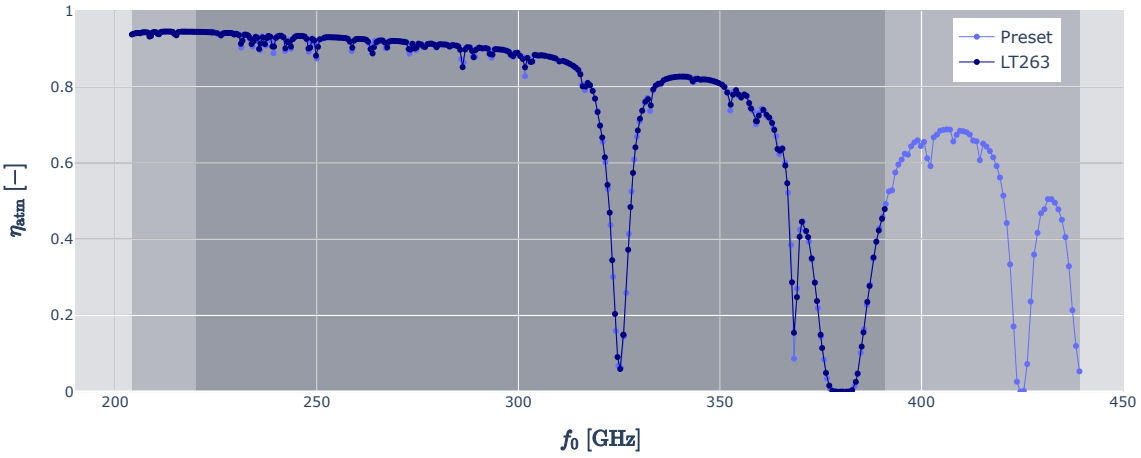


Figure 3.6: The atmospheric transmission η_{atm} for each filter channel with respective center frequency f_0 considered in determining the sensitivity of the LT263 chip and the Preset chip, for which the frequency bands are marked by the shaded regions. For the model of the atmospheric transmission, PWV of 1.0 mm and EL of 60° were used. [Interact]

3.3.2. Noise Equivalent Flux Density for a Line and a Continuum

Now that a definition for the noise equivalent flux is found, the desired measures for sensitivity can be derived. To get to the noise equivalent flux density for a line, the NEF is simply divided by the equivalent bandwidth (FWHM) of the filter channel. Note that this entails a (spectral) line that is not wider than this bandwidth. This yields the following final definition for the NEFD with dimension W m^{-2} Hz^{-1} s$^{0.5}$ as given by [2]:

$$\mathrm{NEFD}_{\mathrm{line}} = \frac{\mathrm{NEP}_{\mathrm{KID}}\,\sqrt{2}_{\mathrm{ON/OFF}}}{\sqrt{2}_{\mathrm{INT}}\,\eta_{sw}\,\eta_{inst}\,A_g\,\mathrm{FWHM}} \tag{3.7}$$

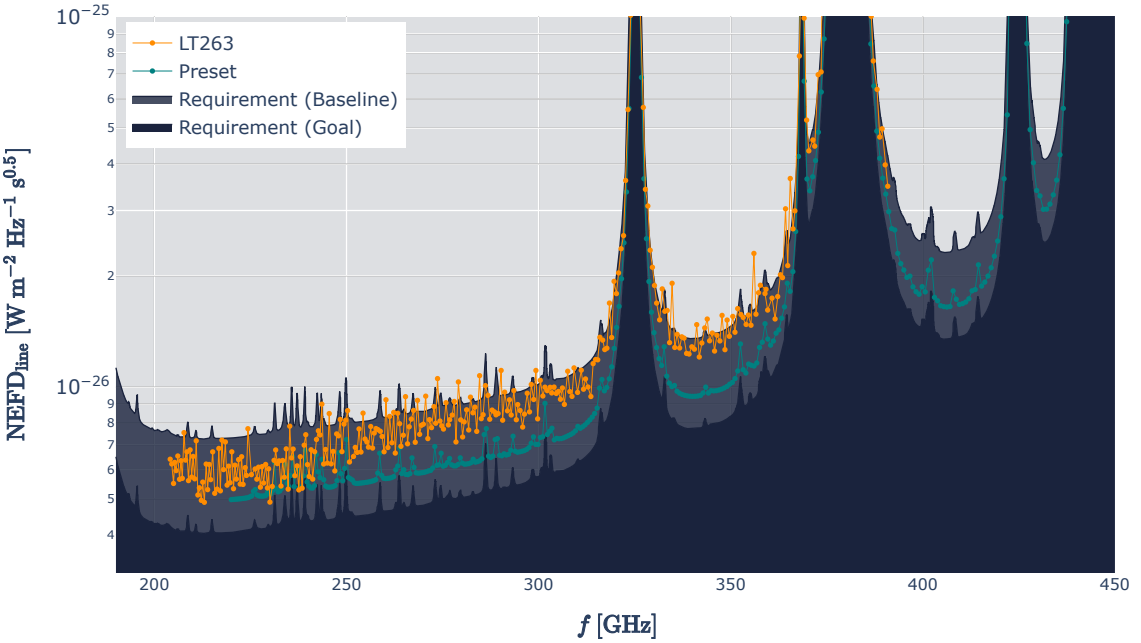


Figure 3.7: The spectral Noise Equivalent Flux Density for a line ($\mathrm{NEFD}_{\mathrm{line}}$) for both the LT263 chip and the Preset chip, plotted as a function of the center frequency of the respective filter channels alongside the requirement curves (with both baseline and goal parameters as inputs). [Interact]

In the limit, a (spectral) emission line is close to an ideal delta function, whereas observed lines have finite widths. Moreover, for the galaxy that was introduced in section 2.1, a velocity width of 600 km s^{-1} was used. For the matter of this research, it is more convenient to look at the noise equivalent flux density for a continuum instead of a line due to the finite nature of the width that defines the Gaussian shaped emission line, as well as the continuum created by thermal radiation of hot dust. A continuum is better described here as a flux density spanning as wide as the entire frequency spectrum.

An adaptation to the description of the filters has to be made. Before, the box-approximation of the band-pass filter had height η_{circuit} and width FWHM, as shown in equation 3.4. Since the full frequency spectrum is now considered, the area equals:

$$\int_{-\infty}^{+\infty} |S_{31}|^2 df = |S_{31}(f = f_0)|^2 \pi\gamma = 2\text{FWHM} \cdot \frac{\pi}{4} |S_{31}(f = f_0)|^2 \tag{3.8}$$

Here, it is used that the area of the Cauchy distribution defining a Lorentzian function is normalized to be 1. As a result, the area equals the scaling factor, which is in this case $|S_{31}(f = f_0)|^2 \pi\gamma$. As one can see, the box-approximation is two times as large as for the case where a line within the FWHM was observed. Keeping the height constant, the width doubles in size. By this redefinition of NEFD and applying it to equation 3.7, the following equation is found for $\text{NEFD}_{\text{continuum}}$:

$$\text{NEFD}_{\text{continuum}} = \text{NEFD}_{\text{line}} \frac{\text{FWHM}}{2 \cdot \text{FWHM}} = \frac{\text{NEP}_{\text{KID}} \sqrt{2}_{\text{ON/OFF}}}{\sqrt{2}_{\text{INT}} \, \eta_{sw} \, \eta_{inst} \, A_g \, 2\text{FWHM}} \tag{3.9}$$

A visualization of such a box-approximation of a band-pass filter is shown in figure 3.4 as 'Box Continuum'.

As can be seen in figure 3.8, the filter channels of the LT263 chip only meet the baseline for low center frequencies. The ratio between $\text{NEFD}_{\text{continuum}}$ and $\text{NEFD}_{\text{line}}$ is exactly equal to 0.5 for both the LT263 chip and the Preset chip, which is in line with equation 3.9. For the baseline and goal, this ratio is different: for the former, the ratio is 0.4, while it is 0.6 for the latter - both in accordance with their requirement parameters. These different ratios result in the Preset chip almost meeting the goal requirement for the $\text{NEFD}_{\text{continuum}}$, while performing slightly worse for $\text{NEFD}_{\text{line}}$. For the LT263, the baseline requirement is met at almost all frequencies for $\text{NEFD}_{\text{line}}$, while it performs slightly worse than the baseline requirement for $\text{NEFD}_{\text{continuum}}$.

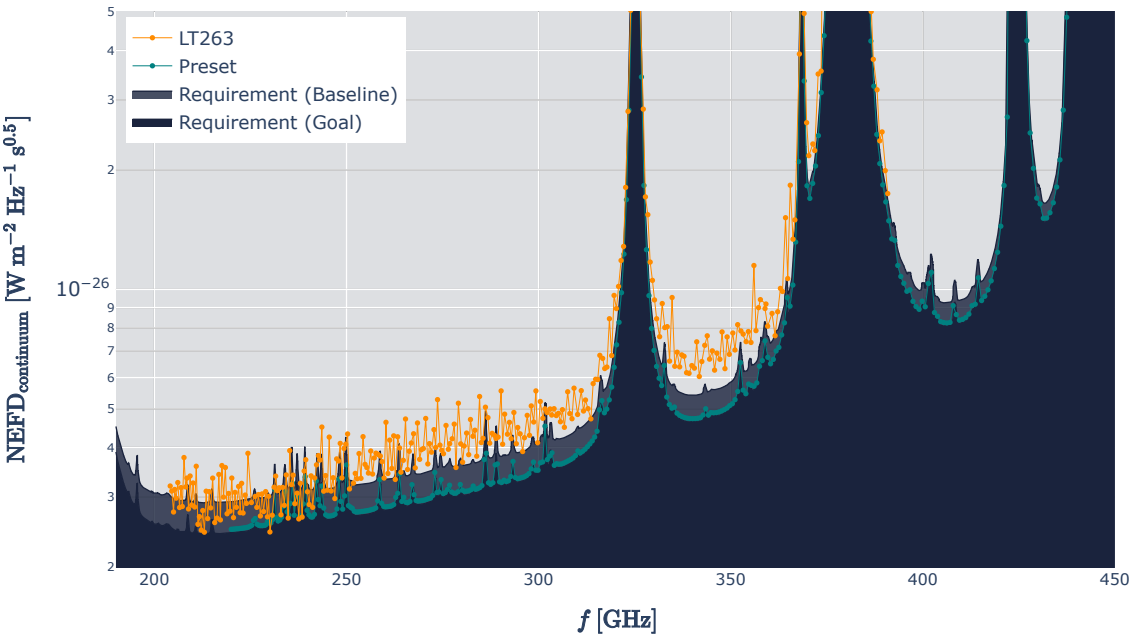


Figure 3.8: The spectral Noise Equivalent Flux Density for a continuum ($\text{NEFD}_{\text{continuum}}$) for both the LT263 chip and the Preset chip, plotted as a function of the center frequency of the respective filter channels alongside the requirement curves (with both baseline and goal parameters as inputs). [Interact]

3.3.3. Minimum Detectable Line Flux

The second measure for sensitivity, the MDLF, involves observation parameters, as it defines a threshold for the flux of a (spectral) line to be measured within an observation of duration t_{obs}, resulting in an integration time of $\tau = \eta_{on}\, t_{obs}$ [s], given a desired signal-to-noise ratio (SNR). η_{on} is the on-source fraction and t_{obs} [s] is the number of seconds of observation time [18]. Consequently, the definition of MDLF is found to be:

$$\mathrm{MDLF} = \frac{\mathrm{NEP}_{\mathrm{KID}}\,\sqrt{2}_{\mathrm{ON/OFF}}}{\sqrt{2}_{\mathrm{INT}}\,\eta_{sw}\,\eta_{inst}\,A_g}\,\frac{\mathrm{SNR}}{\tau} \tag{3.10}$$

The NEFD and the MDLF for the chip can be plotted against (spectral) line center frequency f_0 to be compared with the given goal and baseline parameters for DESHIMA2.0 [18]. The results are shown in figures 3.7, 3.8, and 3.9. As both aforementioned measures for sensitivity relate to minimum required values for the source flux to result in a successful measurement, lower magnitudes are desirable.

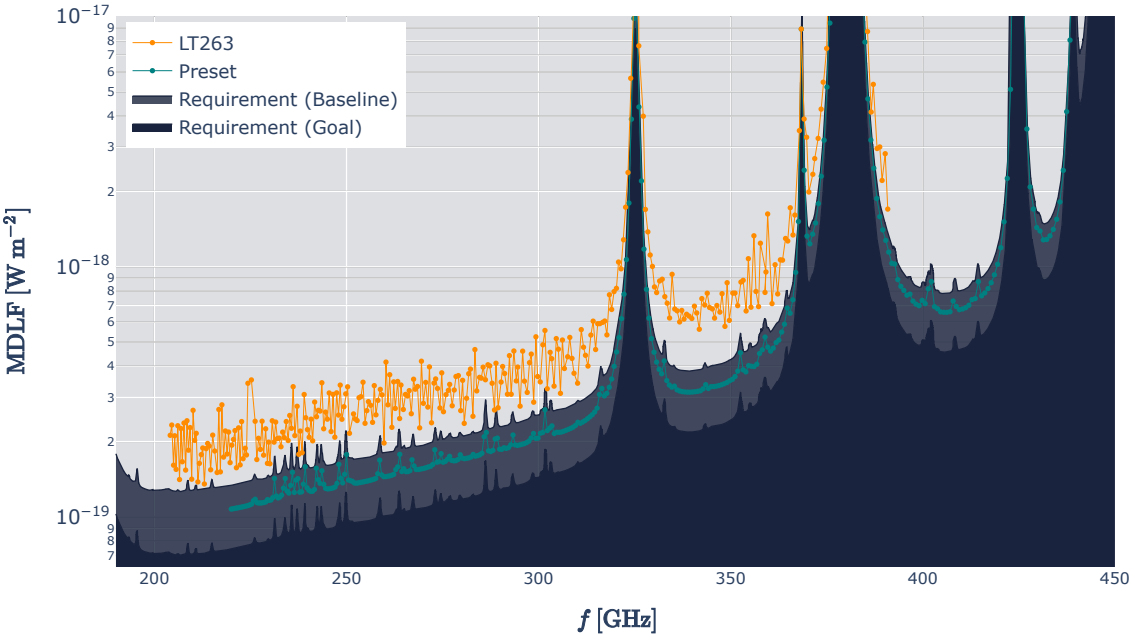


Figure 3.9: The Minimum Detectable Line Flux (MDLF) for $SNR = 5$ and $t_{obs} = 8$ hours for both the LT263 chip and the Preset chip, plotted as a function of the center frequency of the respective filter channels alongside the requirement curves (with both baseline and goal parameters as inputs). [Interact]

From the NEFD plots, it can be concluded that the baseline requirement is reached to some degree - almost all data points are scattered around the baseline curves or perform even better. However, the more ideal goal requirement curves are not yet within reach for all channels, given the data of the LT263 chip. On top of that, the MDLF is seen to be worse than the baseline. As expected, the Preset closely resembles that of a somewhat 'perfect' chip: it mostly meets the design parameters and thus outperforms the baseline requirements.

The effect of the atmospheric transmission is clearly visible with (asymptotic) peaks in the vicinity of the frequencies where the atmosphere is known to be almost opaque. Another remark can be made, namely the spread in the data points. The data points do not form a well-defined curve, but instead scatter around a curve shaped similarly to the baseline and goal curves. This is primarily due to significant irregularities in the basic features like the quality factor Q_l of the chip as described in section 3.1. The scatter may thus originate from imperfections in the manufacturing process of the chip. Notably, more imperfections arose in the visualization of the data. Probably the first approximation made, is the assumption that the channel responses are perfect Lorentzian functions and thus fit perfectly to a curve with parameters discussed in section 3.2. This, however, is not the case due to how the neighboring channels interact, altering their responses from perfect Lorentzian functions [17].

4

TiEMPO Customization

To perform realistic simulations of an observation of the J1329+2234 galaxy, `TiEMPO` is the perfect candidate. The sophisticated model is designed for DESHIMA-type spectrometers and allows users to use a custom galaxy spectrum. It is also designed to be modular, which means that any user could easily adapt the code to their needs [22]. In this chapter, the structure of the model is summarized by briefly looking at the input parameters. Adaptations made to the code to accept custom chip data are also discussed.

4.1. Structure of TiEMPO

The `TiEMPO` model consists of several components, as shown in figure 4.1.

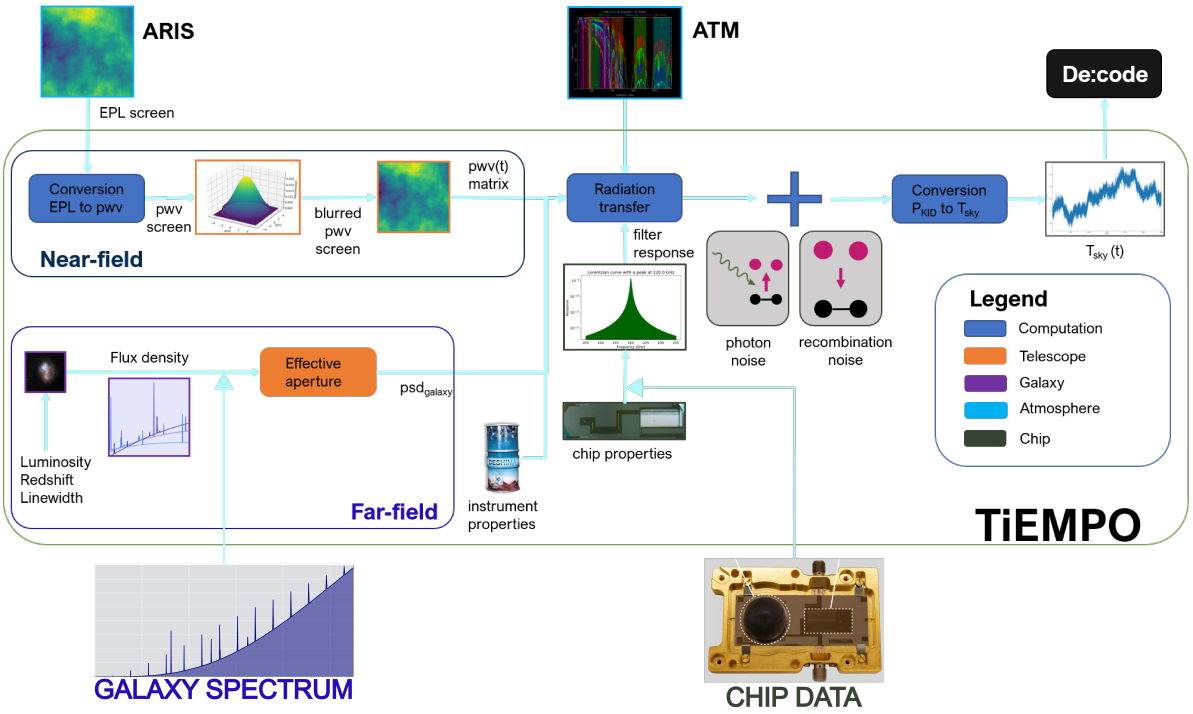


Figure 4.1: The adapted `TiEMPO` structure. The option to include custom chip data was added to the model. Original image from [15].

Each component of TiEMPO either retrieves data or simulates the workings of (a part of) the DESHIMA instrument. The functions of each component are as follows [15]:

- Atmosphere model:
  - Using ARIS [19], fluctuations of PWV of the sky are simulated;
  - This way, the precipitable water vapor will be a function of time as $pwv(t) = dpwv + pwv\_0$ where pwv_0 is the baseline.
- Galaxy model:
  - Simulates a custom galaxy spectrum on the ON-position (or uses the GalSpec Python package to create one);
  - Converts the galaxy spectrum into the power spectral density PSD (see equation 2.13).
- Telescope beam:
  - Covers the transmission of the telescope at multiple sky positions.
- Spectrometer chip:
  - Assumes Lorentzian filters with a constant spectral resolution and circuit efficiency.
  - The center frequencies of the filters are defined as an array containing frequencies that are spaced evenly on a logarithmic scale.
- Radiation transfer:
  - Combines all previous components to find the signal of the galaxy after it is affected by the atmospheric transmission (see equation 2.14).
  - Adds photon and recombination noise to the simulated signal (see equation 3.6).

The output of TiEMPO is a matrix containing the fluctuation in temperature signal for each filter channel over time, for six sky positions. Consequently, the matrix is shaped as 6 x #filters x #samples. The last is defined by the sampling frequency, which was equal to 160 Hz, and the observation time t_{obs} in seconds. The sky positions consist of one ON-position (center ON, with galaxy) and the five OFF-positions OFF (center, without galaxy), L (left), R (right), U (up), and D (down) (see figure 2.9a).

4.2. Customizing

The current version of TiEMPO only considers a perfect DESHIMA2.0 chip, which assumes a constant spectral resolution of 500 and a filter peak height of 0.4 (equal to the $|S_{31}(f = f_0)|^2$ seen in equation 3.2). To have simulations running using the LT263 chip, two adaptations had to be made to the TiEMPO code.

First, a parameter was introduced to make importing of the chip data possible. This would then subsequently result in replacing the default filter channels. The center frequencies of the custom chip are used, as well as the varying spectral resolutions and filter peak heights. Additionally, a parameter was introduced to update the filterbank properties. The function responsible for creating such a new filterbank had to be adapted as well (see Appendix A.3).

Table 4.1: Parameters used for the TiEMPO simulations of the chips. The data of the LT263 chip is imported through chipdata.

Input for TiEMPO	Description	Preset Chip	LT263 Chip
obs_time	t_{obs}[s]	28800	28800
pwv_0	Baseline for PWV [mm]	1.0	1.0
EL	Elevation angle [°]	60	60
F_min	Minimum f_0 [Hz]	220e9	-
spec_res	R or Q_l	500	-
f_spacing	Logarithmic spacing f	500	-
num_filters	Number of channels	347	-
frequency_gal	f_{gal} [Hz]	f from adj.data	f from chip.data
spectrum_gal	F_ν [Jy]	FD from adj.data	FD from chip.data
chipdata	Filepath to read chip data	-	LT263_filters.csv
run_new_filterbank	Update filterbank properties	False	True

5

Simulating the Performance of the Lab-Measured Filterbank Chip

Using the customized version of `TiEMPO`, it is possible to perform a simulation of the LT263 filterbank chip, whose characteristics were measured in the lab and analyzed in chapter 3. For both the LT263 chip and the Preset chip, a simulation of an observation of the high-redshift J1329+2243 galaxy (see figure 5.1) using $t_{obs} = 8$ hours was performed. For these simulations, a baseline precipitable water vapor `pwv_0` of 1.0 mm, and a telescope elevation angle `EL` of 60° were used. See table 4.1 for further details on the parameters.

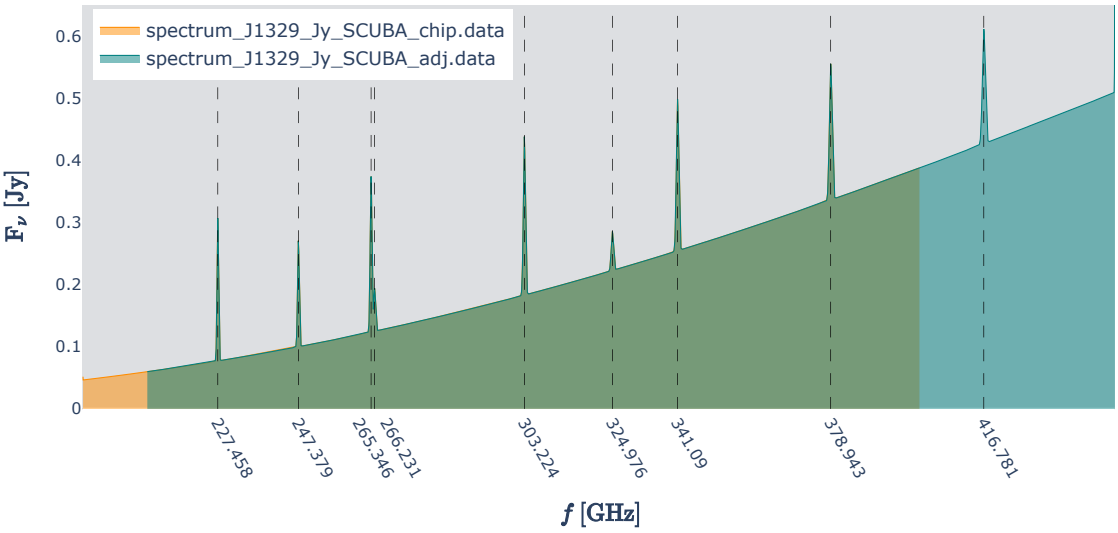


Figure 5.1: Model flux density spectrum of the high-redshift J1329+2243 galaxy using rescaled continuum flux measurements from [12] and adding the redshift-corrected lines (z =2.04 [10]) with line width 600 km s^{-1}. The redshift-corrected center frequencies of the lines are given on the f-axis. The data from the file 'spectrum_J1329_Jy_SCUBA_chip.data' contains the spectrum for the frequency band of the LT263 chip plus an added margin of 10 GHz on each end. Likewise, 'spectrum_J1329_Jy_SCUBA_adj.data' contains the spectrum for the Preset chip. The spectra overlap in the green region. Building of the model was based on method created by Matus Rybak, private communication. [Interact]

Table 5.1: The line fluxes of the J1329+2243 galaxy taken from [13], where the redshift-corrected line frequencies lie within the frequency spectrum of the Preset chip and the LT263 chip. The redshift of the galaxy is taken to be 2.04 and the line width is taken constant at 600.0 km s^{-1}.

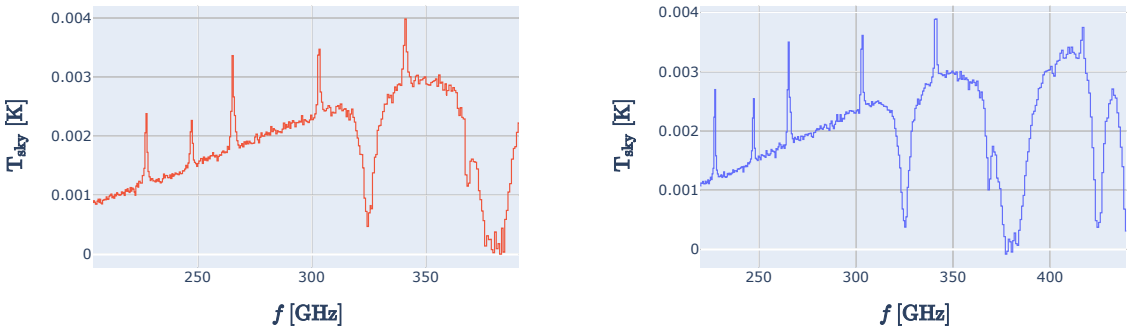
Line Name	Line Frequency f_{line} [GHz]	Redshift-Corrected f_{line} [GHz]	Line Flux [W m^{-2}]
CO(6-5)	691.4730763	227.4582487	$1.11454546 \cdot 10^{-18}$
H_2O (211-202)	752.0332	247.3793	$9.02 \cdot 10^{-19}$
CO(7-6)	806.651806	265.345988	$1.4151786 \cdot 10^{-18}$
[CI](2-1)	809.34197	266.23091	$3.91 \cdot 10^{-19}$
CO(8-7)	921.7997	303.2235	$1.64751481 \cdot 10^{-18}$
H_2O (202-111)	987.9267	324.9758	$4.34 \cdot 10^{-19}$
CO(9-8)	1036.912393	341.089602	$1.77366595 \cdot 10^{-18}$
CO(10-9)	1151.985452	378.942582	$1.76839879 \cdot 10^{-18}$
CO(11-12)	1267.014486	416.781080	$1.63933894 \cdot 10^{-18}$

5.1. Temperature from the Simulations

The primary output of `TiEMPO` is the sky temperature measured on six sky positions for each filter channel as a function of time. This section will focus on what this data from both of the simulations looks like, how it behaves as a function of frequency and time and how the galaxy spectrum as described in 2.1 can be retrieved from it.

5.1.1. Sky Temperature

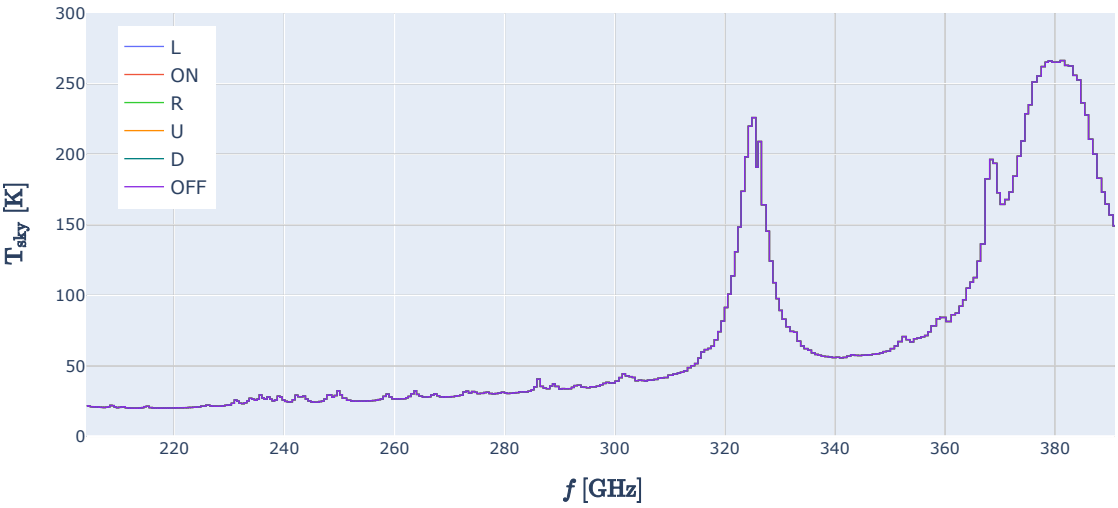
The raw data obtained from the `TiEMPO` simulations using the J1329+2243 galaxy is plotted in figure 5.3 for the LT263 chip and in figure 5.4 for the Preset chip (on the next pages). Figure 5.3a and figure 5.4a show the time-averaged signals for all beam positions. On this scale, no difference in signal can be seen. The effect of the atmospheric transmission is clearly visible, as it defines the almost asymptotic behavior around the regions where it was found to be close to zero in figure 3.6. When looking at the signal of individual frequency channels in figure 5.3b and in figure 5.4b, the atmospheric noise can be clearly seen. For these two plots, channels of the chips were chosen such that they have approximately the same center frequencies. Indeed, the respective frequency channels show quite some resemblance as their signals fluctuate over time. One thing to note is that the variation of the signal on a shorter timescale appears to be larger for the LT263 than for the Preset chip. ON-OFF (dual) sky chopping (see section 2.2.3) is applied to the time-averaged sky temperature as a function of frequency, clearly showing how the spectrum increasingly resembles that of the J1329+2243 galaxy for longer observation times. Four different timestamps in the processing of the observation for t_{obs} = 28800 s (or 8 hours) are shown in figure 5.5, namely t_{obs} = 1 s, 120 s, 600 s, and 3600 s. Notice how the noise in the spectrum reduces as integration time increases.



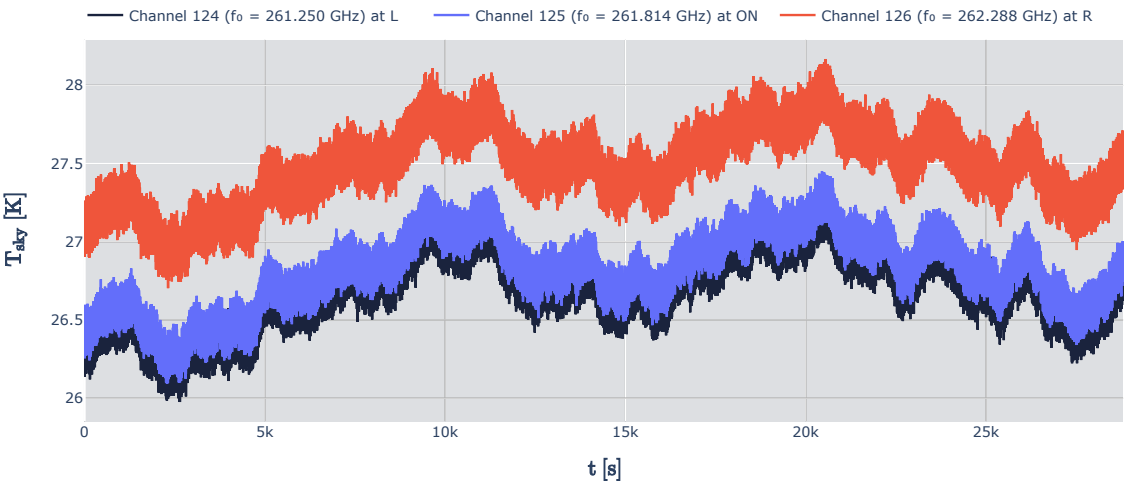
(a) LT263 chip, $T_{sky}(f)$ for $\tau = 14400$s

(b) Preset chip, $T_{sky}(f)$ for $\tau = 14400$s

Figure 5.2: The time-averaged chopped $T_{sky}(f)$ of a `TiEMPO` simulation plotted for t_{obs}= 28800 s, or, equivalently, integration time τ = 14400 s for both the (a) LT263 DESHIMA2.0 filterbank chip and (b) the Preset chip at a sampling rate of 160 Hz. Here, the `TiEMPO` parameters `pwv_0` = 1.0 mm and `EL` = 60° were used. A model of the J1329+2243 galaxy was used to be put on the ON-position. The L-position was used as the OFF-position to perform dual chopping, taking the chopping rate of 10 Hz into account. The spectrum shown is therefore the measured galaxy brightness temperature spectrum affected by the atmospheric transmission.

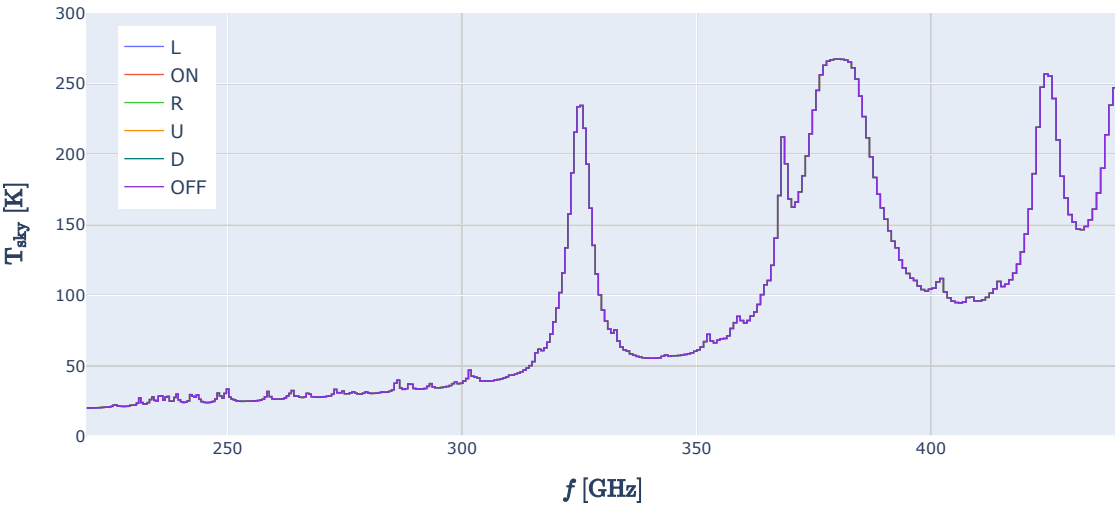


(a) LT263 chip, $\overline{T_{sky}}(f)$ for all six beam positions. [Interact]

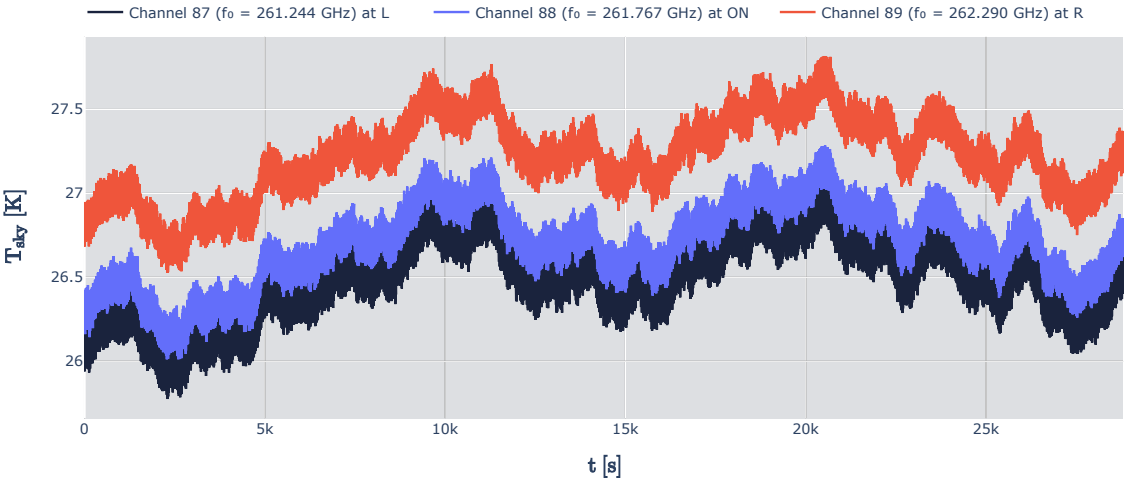


(b) LT263 chip, $T_{sky}(t)$ for three filter channels

Figure 5.3: (a) The time-averaged $T_{sky}(f)$ for all six beam positions of a $t_{obs} = 28800$ s (8 hour) or, equivalently, $\tau = 14400$ s (4 hour) `TiEMPO` simulation of the lab-measured LT263 DESHIMA2.0 filterbank chip. The simulation was done with a model of the J1329+2243 galaxy on the ON-position, and `TiEMPO` parameters `pwv_0` $= 1.0$ mm and `EL` $= 60°$. In the plot, the center frequencies of the 332 filter channels are used for f. (b) The measured $T_{sky}(t)$ for single beam positions of three filter channels with center frequencies close to 261.8 GHz.

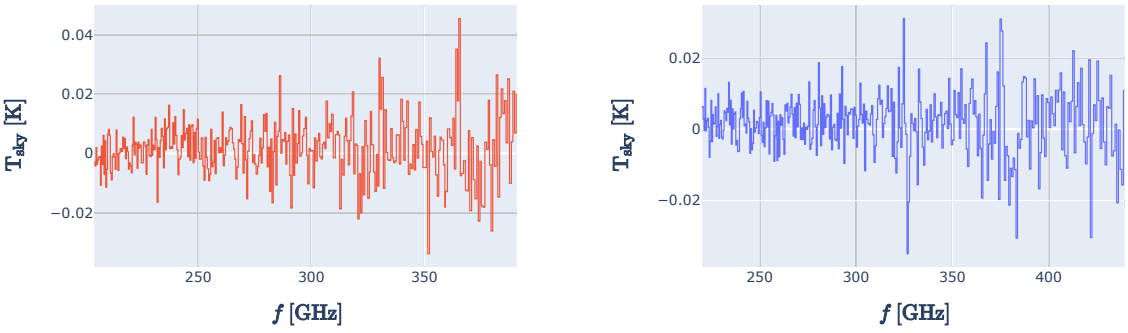


(a) Preset chip, $\overline{T_{sky}}(f)$ for all six beam positions. [Interact]



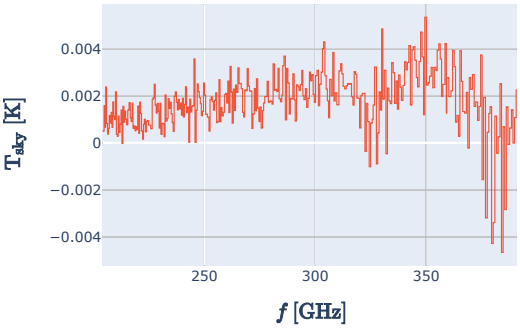
(b) Preset chip, $T_{sky}(t)$ for three filter channels

Figure 5.4: (a) The time-averaged $T_{sky}(f)$ for all six beam positions of a $t_{obs} = 28800$ s (8 hour) or, equivalently, $\tau = 14400$ s (4 hour) `TiEMPO` simulation of the Preset chip. The simulation was done with a model of the J1329+2243 galaxy on the ON-position, and `TiEMPO` parameters `pwv_0` $= 1.0$ mm and `EL` $= 60°$. In the plot, the center frequencies of the 347 filter channels are used for f. (b) The measured $T_{sky}(t)$ for single beam positions of three filter channels with center frequencies close to 261.8 GHz.

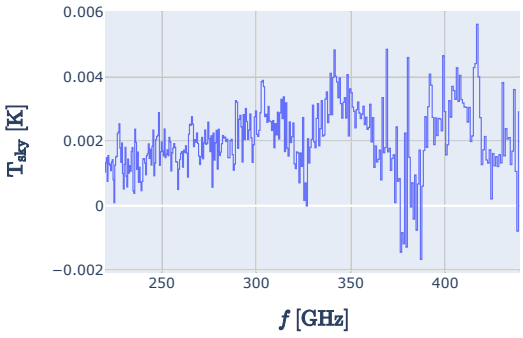


(a) LT263 chip, $T_{sky}(f)$ for $\tau = 0.5$s

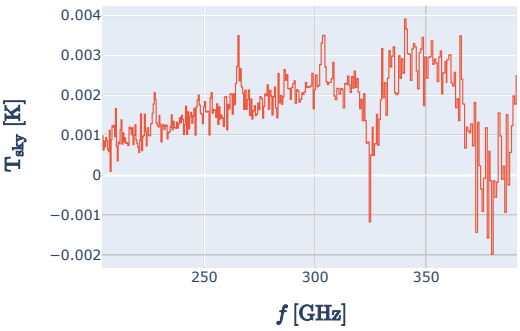
(b) Preset chip, $T_{sky}(f)$ for $\tau = 0.5$s



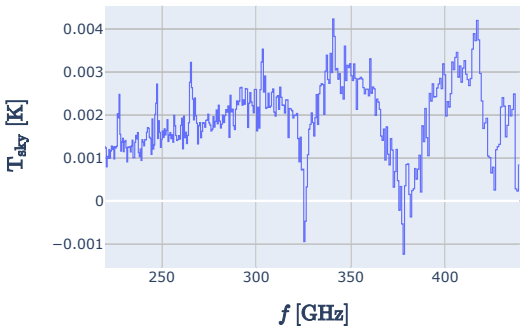
(c) LT263 chip, $T_{sky}(f)$ for $\tau = 60$s



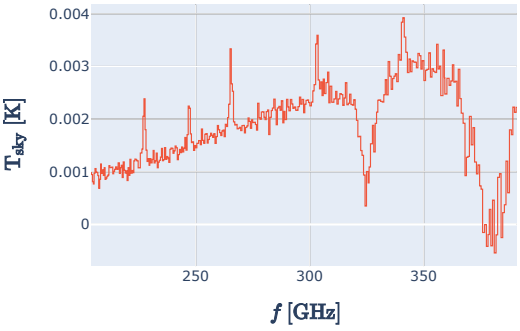
(d) Preset chip, $T_{sky}(f)$ for $\tau = 60$s



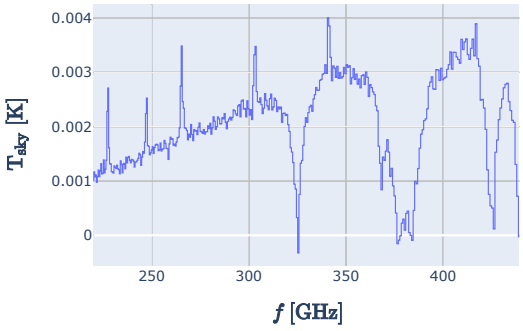
(e) LT263 chip, $T_{sky}(f)$ for $\tau = 300$s



(f) Preset chip, $T_{sky}(f)$ for $\tau = 300$s



(g) LT263 chip, $T_{sky}(f)$ for $\tau = 1800$s



(h) Preset chip, $T_{sky}(f)$ for $\tau = 1800$s

Figure 5.5: The time-averaged chopped $T_{sky}(f)$ plotted for t_{obs} = 1 s, 120 s, 600 s, and 3600 s, or integration time τ = 0.5 s, 60 s (= 1 min), 300 s (= 5 min), and 1800 s (= 30 min) respectively. The red curves on the left are the results from a `TiEMPO` simulation of the LT263 DESHIMA2.0 filterbank chip with `pwv_0` = 1.0 mm and `EL` = 60°. The blue curves on the right are the results of the same simulation, but by using the Preset chip instead. A model of the J1329+2243 galaxy was used to be put on the ON-position. The L-position was used as the OFF-position to perform dual chopping, taking a sampling rate of 160 Hz and a chopping rate of 10 Hz into account.

5.1.2. Atmosphere-Corrected Antenna Temperature

By applying equation 2.17 and dividing the ON-OFF chopped signal by the atmospheric transmission, the spectra shown in figure 5.6 are found. Notice how the only peaks that are left on the continuum (close to a first order polynomial in this frequency band) are either lines of the J1329+2243 galaxy, or distortions caused by the atmospheric transmission (basically extreme amplification of the noise at those frequency bands).

Applying Different Methods of Chopping

In figure 5.7, three different curves of the atmosphere-corrected antenna temperature found through methods of sky chopping (see section 2.2.3) of different sky positions are plotted alongside the model of the J1329+2243 galaxy model. For the method of dual chopping, a sampling rate of 160 Hz and a chopping rate of 10 Hz was applied to switch between the ON-position, where the galaxy is situated, and the L-position, which is one of the OFF-positions (there is no galaxy here). This method of chopping, which is also called ON-OFF chopping, results in an on-source fraction of 0.5. Dual chopping can also be applied on two OFF-positions, such that only the atmosphere-corrected noise left in the signal remains. This variant of dual chopping is also called OFF-OFF chopping. It should be noted that the on-source fraction of 0.5 used here and throughout the rest of the report is just a theoretically possible value. In reality, it takes time for the optical system to adjust when switching between positions. This would introduce other efficiencies to take into account, lowering the on-source fraction. For the method of ABBA chop/nodding, dual chopping is applied on two different nodding positions. The beam switches between the L-position and the ON-position for nod A, while it switches between the ON-position and the R-position for nod B. An n-factor of 3 was used to have three dual chop cycles per nod.

As is visible in figure 5.7, the effect of ABBA chopping here is very similar to that of the dual chopping for both chips. Since `TiEMPO` does not consider different optical efficiencies and foreground noises for different nodding positions, both methods of chopping will yield the same result in the limit. In practice, ABBA chopping should perform better, as shown in [20] and already examined in [28] for `TiEMPO` simulations: it is successful in compensating for the differences in the nodding positions. So, for the sake of conciseness of and relevance to this report, this method of chopping won't be looked into any further.

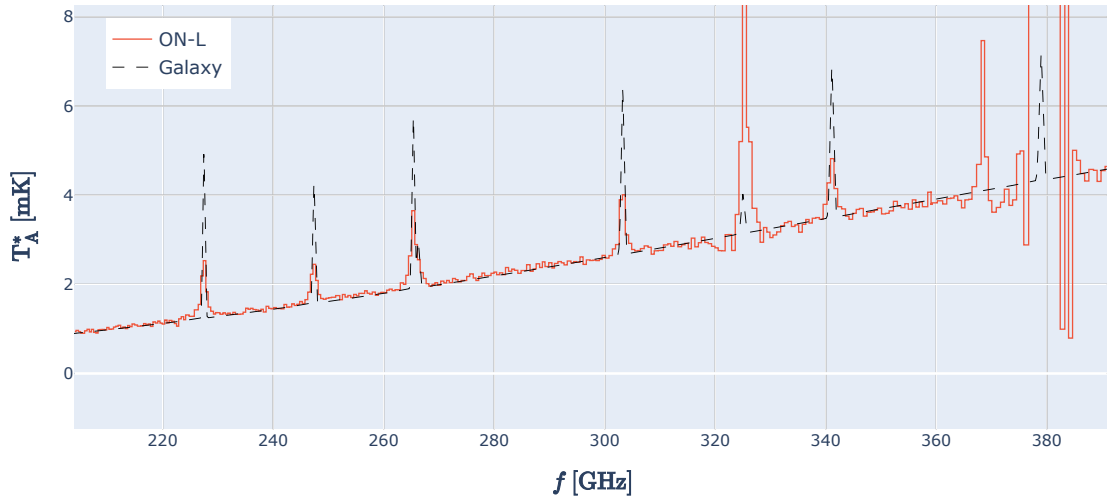
The signal of the OFF-OFF chopping as shown in both 5.7a and 5.7b proves the zero-mean nature of the noise. Though heavily distorted within the frequency ranges of approximately 320-330 GHz, 360-400 GHz, and 420-430 GHz due to the close-to-zero atmospheric transmission (see 3.6), the other frequency regions show to be centered around zero with little standard deviation left. The standard deviation as a function of time will be looked into in section 5.2.1.

5.2. Sensitivity: Signal-to-Noise Ratio

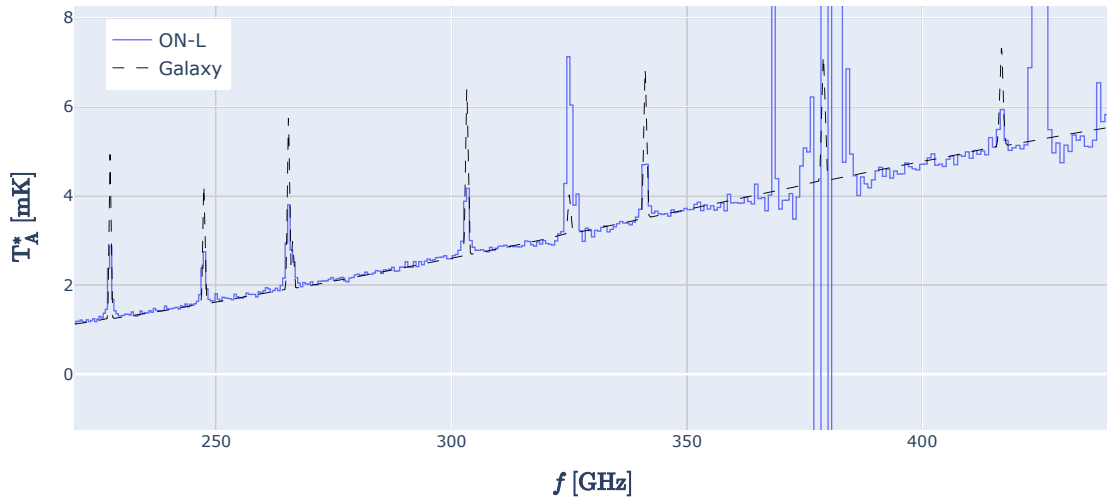
In observing the spectrum of a galaxy, it is important to know whether and after how much integration time τ a line with redshift-corrected center frequency f_{line} can be observed. To obtain this description of sensitivity, there are several methods of describing the ratio between observed signal and magnitude of noise. Given the atmosphere-corrected antenna temperature spectrum of the galaxy, the ratio will be defined in this context as follows.

$$\text{SNR}(\tau) = \frac{T_A^*(f_0 \approx f_{line}, t) - T_{A,\,\text{continuum}}^*(f = f_0, t)}{\sigma_{T_A^*}(t)} \tag{5.1}$$

Here, the signal-to-noise ratio is defined as a function of integration time τ, which is half of the observation time due to an on-source fraction of 0.5 caused by dual chopping. The line flux of a line with center frequency f_{line} is approximated by taking the temperature measured by the filter channel with its center frequency f_0 closest to that of the line, with respect to the continuum temperature at the filter channel center frequency. This continuum is approximated at each time step by a curve-fitted first-order polynomial, where the curve-fit only considers points outside of the region defined by the line. This region is considered to be four standard deviations of the Gaussian distribution that is the line flux (see section 2.1.1) away from the center frequency f_{line}. Lastly, this found line flux is divided by the standard deviation of the noise present in the temperature. The standard deviation is found by taking the same region as defined for the curve-fit, subtract the found first-order polynomial, and take the standard deviation of it as it is now assumed to be zero-mean (as if it were an OFF-OFF chopped signal).

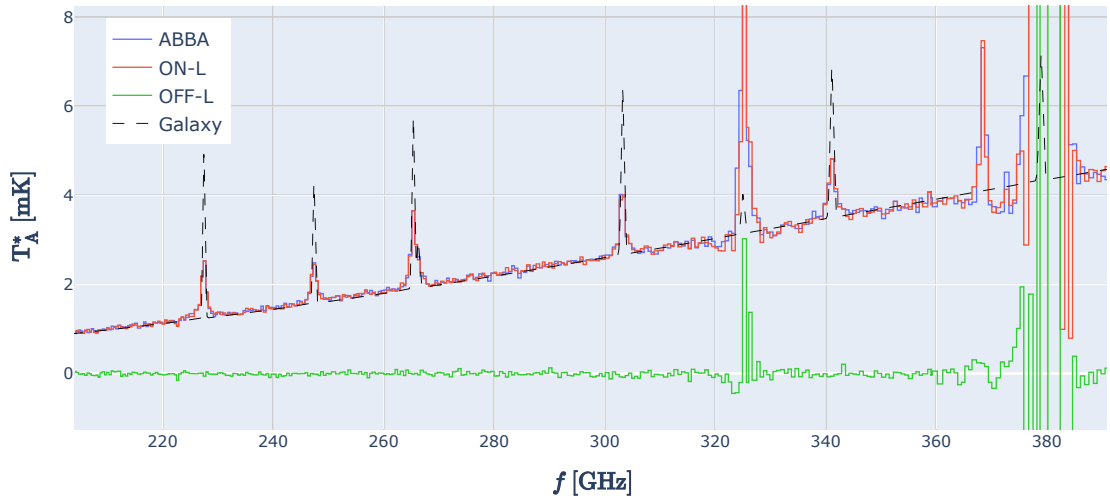


(a) LT263 chip, $T_A^*(f)$ [Interact]

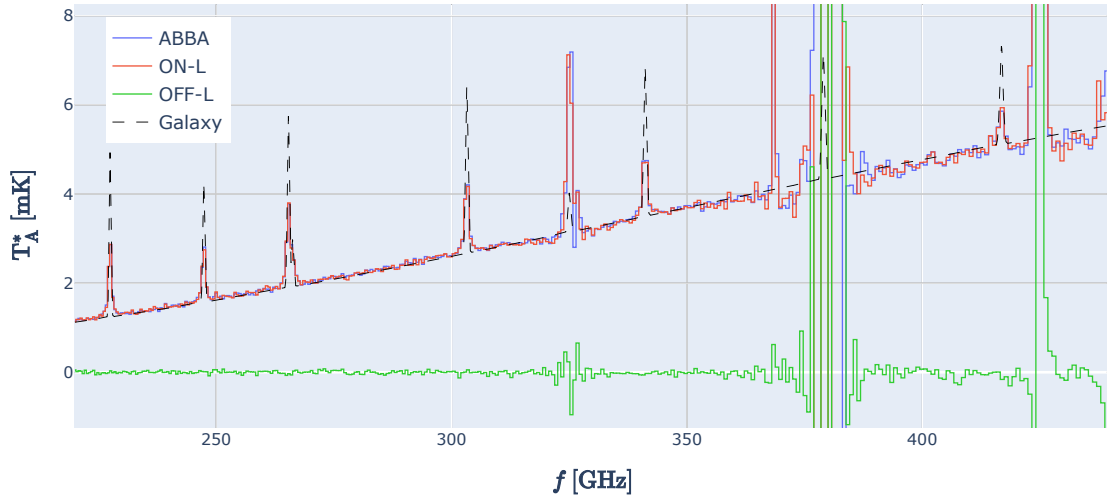


(b) Preset chip, $T_A^*(f)$ [Interact]

Figure 5.6: The simulated atmosphere-corrected antenna temperature $T_A^*(f)$ spectrum plotted alongside the model of the J1329+2243 galaxy for both the (a) lab-measured LT263 DESHIMA2.0 filterbank chip and the (b) Preset chip. The brightness temperature spectrum was found by applying dual chopping to `TiEMPO` simulated data of a $t_{obs} = 28800$ s (8 hour) or, equivalently, $\tau = 14400$ s (4 hour) observation. For the method of dual chopping, a sampling rate of 160 Hz and a chopping rate of 10 Hz was applied to switch between the ON-position, where the galaxy is situated, and the L-position. This results in an on-source fraction of 0.5. The used `TiEMPO` simulated data had parameters `pwv_0` $= 1.0$ mm and `EL` $= 60°$, which were used as inputs for `deshima-sensitivity` to get the atmospheric transmission η_{atm}, which the through dual chopping time-averaged T_{sky} was corrected by.



(a) LT263 chip, $T_A^*(f)$ [Interact]



(b) Preset chip, $T_A^*(f)$ [Interact]

Figure 5.7: The simulated atmosphere-corrected antenna temperature $T_A^*(f)$ spectra plotted alongside the model of the J1329+2243 galaxy for both the (a) lab-measured LT263 DESHIMA2.0 filterbank chip and the (b) Preset chip for three different sky chopping applications. The brightness temperature spectrum was found by applying dual chopping to `TiEMPO` simulated data of a $t_{obs} = 28800$ s (8 hour) or, equivalently, $\tau = 14400$ s (4 hour) observation. Here, the ON-L line is that of a dual chopping applied on the ON-position, where the galaxy is situated, and the L-position. The OFF-L line is that of OFF-OFF dual chopping applied on two OFF-positions: namely, the OFF-position (same position as the ON-position, but without the galaxy) and the L-position. The ABBA (n=3) line is the result of simulated chop/nodding between a nod A where the signal of the L-position and the ON-position are chopped, and a nod B where the signal of the ON-position and the R-position are chopped. The used `TiEMPO` simulated data had parameters `pwv_0` $= 1.0$ mm and `EL` $= 60°$, which were used as inputs for `deshima-sensitivity` to get the atmospheric transmission η_{atm}, which the through chopping time-averaged T_{sky} spectra were corrected by.

An estimation of the standard deviation in the noise can be made by looking at the noise equivalent flux density presented in section 3.3.2 and defined as equation 3.9 for a continuum and converting it to noise equivalent temperature (NET) through the use of equation 2.14. The behavior of the standard deviation of the noise will be discussed in section 5.2.1.

In section 3.3.3, a way of measuring the minimum detectable line flux MDLF was discussed and formulated through equation 3.10. Since the MDLF is scalable, a rescaled version of the MDLF($t_{obs} = 28800$ s, $\eta_{on} = 0.5$, SNR $= 5$) in figure 3.9 is used, substituting an SNR of 1 as well as an observation time t_{obs}= 1s. This property of scalability is used to define a theoretically found SNR for the lines in equation 5.2 where line flux LF [W m^{-2}] as given in table 5.1 is divided by the MDLF.

$$\text{SNR}(\tau) = \frac{\text{LF}(f_{line})}{\text{MDLF}(t_{obs} = 1\text{s}, \eta_{on} = 0.5, \text{SNR} = 1)} \sqrt{\frac{2\tau}{1\text{s}}} \tag{5.2}$$

Here, in equation 5.2, the scaling of time is done as follows. Using the property MDLF$(1\text{s})\sqrt{1\text{s}} =$ MDLF$(t_{obs})\sqrt{t_{obs}}$ and rearranging it as well as substituting $t_{obs} = 2\tau$ yields MDLF$(\tau) =$ MDLF$(1\text{s})\sqrt{\frac{1\text{s}}{2\tau}}$. Characteristically, the theoretical curve will show a proportionality to the square root of integration time, given that the ratio of line fluxes in equation 5.2 are constant over time.

5.2.1. Noise Standard Deviation

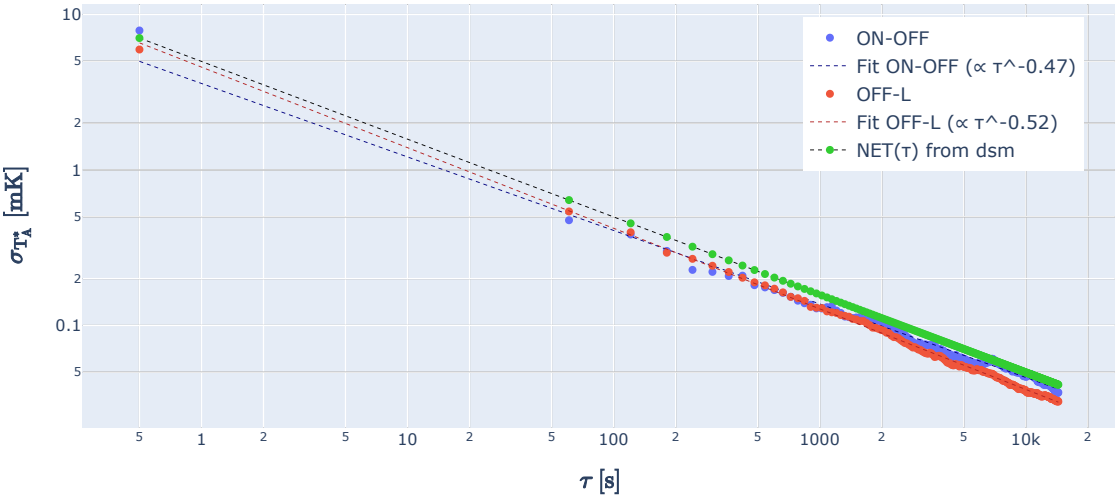
The standard deviations of the noise measured in the atmosphere-corrected antenna temperature $\sigma_{T_A^*}$ [K] is proven to be inversely proportional to the integration time, as it was curve-fitted for both the Preset and the LT263 chip and plotted as such in figures 5.8a and 5.8b[1]. Using the NEFD of `deshima-sensitivity`, predicted curves could be plotted as well. For these predicted curves, the scaling over time is as in equation 5.3. Here, the NEFD(τ) is converted to NET(τ) by using equation 2.14.

$$\text{NEFD}(\tau) = \frac{\text{NEFD}(t_{obs} = 1\text{s}, \eta_{on} = 0.5)}{\sqrt{\tau}} \tag{5.3}$$

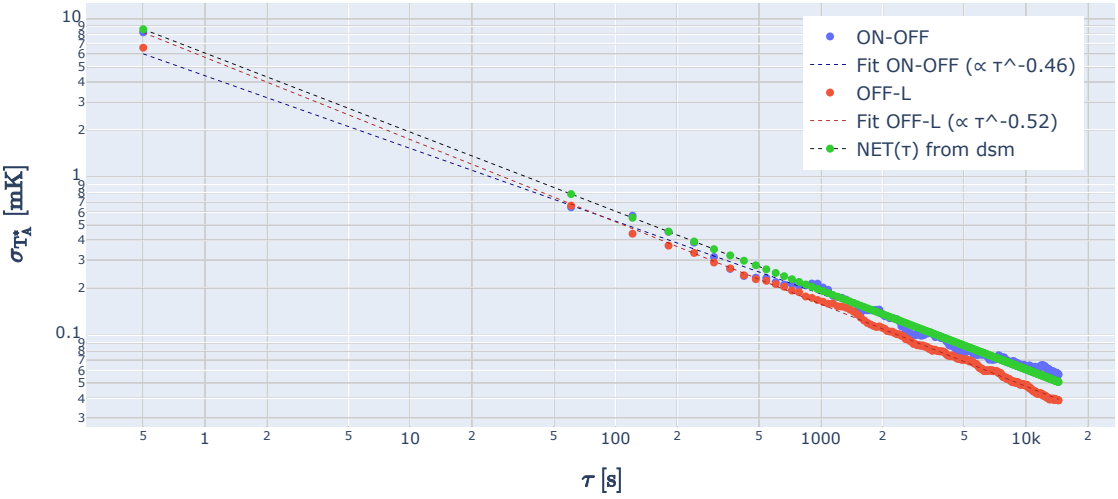
The two chips show close to the same curves in approximating the standard deviation of the noise as a function of time, with curve-fit parameters that are nearly equal. The Preset chip performs just slightly better, as its standard deviation is lower at all times. This was also seen in figures 5.3b and 5.4b.

For both the LT263 chip and the Preset chip, the NET predicted by `deshima-sensitivity` as a function of integration time is shown to resemble the curves. Regarding the curve-fits, the OFF-OFF chopped signal even has its standard deviation at $\tau = 0.5$ s at nearly the exact same point as the predicted NET(τ). Alternatively, the ON-OFF signal shows to have its curve-fit to be a factor of $\approx$1.4 times smaller than the NET(τ) for $\tau = 0.5$ s. This result can be further looked into[2].

[1] Versions of these plots using linear axes are included in appendix B.4.1.
[2] A short analysis is included in appendix B.3.



(a) Preset chip, $\sigma_{T_A^*}(\tau)$ [Interact]



(b) LT263 chip, $\sigma_{T_A^*}(\tau)$ [Interact]

Figure 5.8: The standard deviation $\sigma_{T_A^*}$ of the atmosphere-corrected antenna temperature for the (a) Preset chip and (b) LT263 chip plotted as a function of integration time τ. The minimum τ considered is 0.5 s, and the maximum is 14400 s (as the total observation time was 8 hours). The integration times shown are spaced linearly at integration time intervals of 60 s. The ON-OFF line represents the found standard deviation in the ON-OFF (dual) chopped signal, whereas the OFF-L line represents it of an OFF-OFF (dual) chopped signal. Both were derived in the frequency band of 270-300 GHz. Their proportionalities to the integration time were curve fitted and plotted as well. The noise equivalent temperature NET was derived from the theoretical noise equivalent flux density NEFD obtained from `deshima-sensitivity` using parameters `pwv_0` $= 1.0$ mm and `EL`$= 60^\circ$.

5.2.2. Measured Signal-to-Noise Ratios

First, the signal-to-noise ratio for the Preset chip and the LT263 chip will be examined for the lines of the J1329+2243 galaxy in the frequency band of 200-310 GHz (see table 5.1 for reference), as this region shows a close-to-linear scaling, and furthermore without too much disturbance caused by correcting for the atmospheric transmission - visible in figure 5.6a for the LT263 and in figure 5.6b for the Preset chip. This region also makes it possible to compare the two chips, as both cover these frequencies. Next, the behavior of the SNR as a function of time will be analyzed by looking at the filter channels with center frequencies close to the redshift-corrected individual line frequencies.

In figure 5.9, the SNR functions for the Preset chip are plotted alongside the theoretical predictions made by `deshima-sensitivity`. The latter scales with the square root of integration time, as given by equation 5.2. Recall that the integration time is half of the observation time, as dual chopping was performed, reducing the on-source fraction to 0.5. For all lines, the SNR(τ) found from the `TiEMPO` simulations closely resembles the relation of square root of integration time proportionality for an integration time up to approximately an hour ($\tau \approx 3600$ s), after which it decelerates in its growth rate and starts to slowly incline with a factor less than the square root relation found prior for the hours thereafter. The ratios at which they decelerate are approximately 22.0, 15.5, 25.5, 11.5, and 21.0 for the CO $(6-5)$, the H_2O $(211-202)$, the CO $(7-6)$, the [CI] $(2-1)$, and the CO $(8-7)$ line respectively. Overall, the curves are close to being at half of the prediction curves, which seems reasonable when looking at the peaks visible in figure 5.6b and comparing to the peaks of the galaxy spectrum. The SNR of the [CI] $(2-1)$ line is remarkable as it is higher with respect to the prediction curve than the other curves shown. The reason for this will be looked into further on in section 5.2.3.

In figure 5.10, the SNR functions are plotted again alongside the theoretical predictions made by `deshima-sensitivity`, but now for the LT263 chip. For all lines, again, the SNR(τ) found from the `TiEMPO` simulations closely resembles the expected relation, but now for integration time up to approximately three hours ($\tau \approx 10800$ s). It is expected to behave similarly to the Preset chip after this time, as it decelerates in its growth and shows signs of a starting incline with a factor less than the square root relation. The ratios at which the lines decelerate are approximately 19.0, 12.5, 26.0, 8.5, and 21.0 for the CO $(6-5)$, the H_2O $(211-202)$, the CO $(7-6)$, the [CI] $(2-1)$, and the CO $(8-7)$ line respectively. So, even though the two chips decelerate after different observing times, the decelerating happens at ratios that are fairly similar. Besides the slightly higher than predicted curves for the CO $(7-6)$ and the CO $(8-7)$ line, the SNR of the [CI] $(2-1)$ is also remarkable as it is way higher than its prediction. The reason for this will be looked into further on in section 5.2.3.

When comparing the integration times for the SNR to reach certain values, the Preset chip performs about twice as good as the LT263 chip. This is in line with what was expected by looking at the MDLF plot in figure 3.9, where a factor of approximately 2 separates the two different chips. Knowing this, the difference in 'deceleration times' of approximately one hour and three hours can be better understood: the LT263 chip takes roughly twice as long to reach this observed decelerating SNR. As the standard deviation was shown to have the expected inverse square root proportionality with the integration time in 5.2.1, it can only be the estimation of the peak height that is disturbed or stagnates the incline. The MDLF also explains another trend: a lower line frequency tends to result in a higher SNR.

Another note to be made is that the line flux is not simply observed by a single filter channel. Instead, more than one filter channel may cover a frequency: as designed, two channels would have any particular frequency within their equivalent bandwidths of size 2FWHM, since the line is approximated to be a continuum (see section 3.3.2) and the spacing between channels is designed to be equal to FWHM. However, for the LT263, more than two equivalent bandwidths may overlap as the spectral resolutions are less than the spacing of the filter channels, as seen in section 3.1. While the coupling efficiency of the channels might be lower, the line flux is still measured by several filter channels. While `TiEMPO` takes this effect into account, `deshima-sensitivity` does not consider this and only observes a single channel at a time. This feature is also seen in figures 5.11, 5.12, 5.13, and 5.14.

Since the method used to calculate the SNR only looks at the filter channel with its center frequency closest to the line frequency, it is really just an approximation. An improvement on this method would be to curve-fit a Gaussian distribution through the temperatures measured by the filter channels surrounding the line flux and find the line flux that way. In practice, a method like this would actually be the only way to observe lines, as the redshift is not necessarily known in advance. Moreover, the existence of the line might not even be known in the first place. This could cause observations to be misleading or faulty. An example of this is actually seen for the lines CO $(7-6)$ and [CI] $(2-1)$.

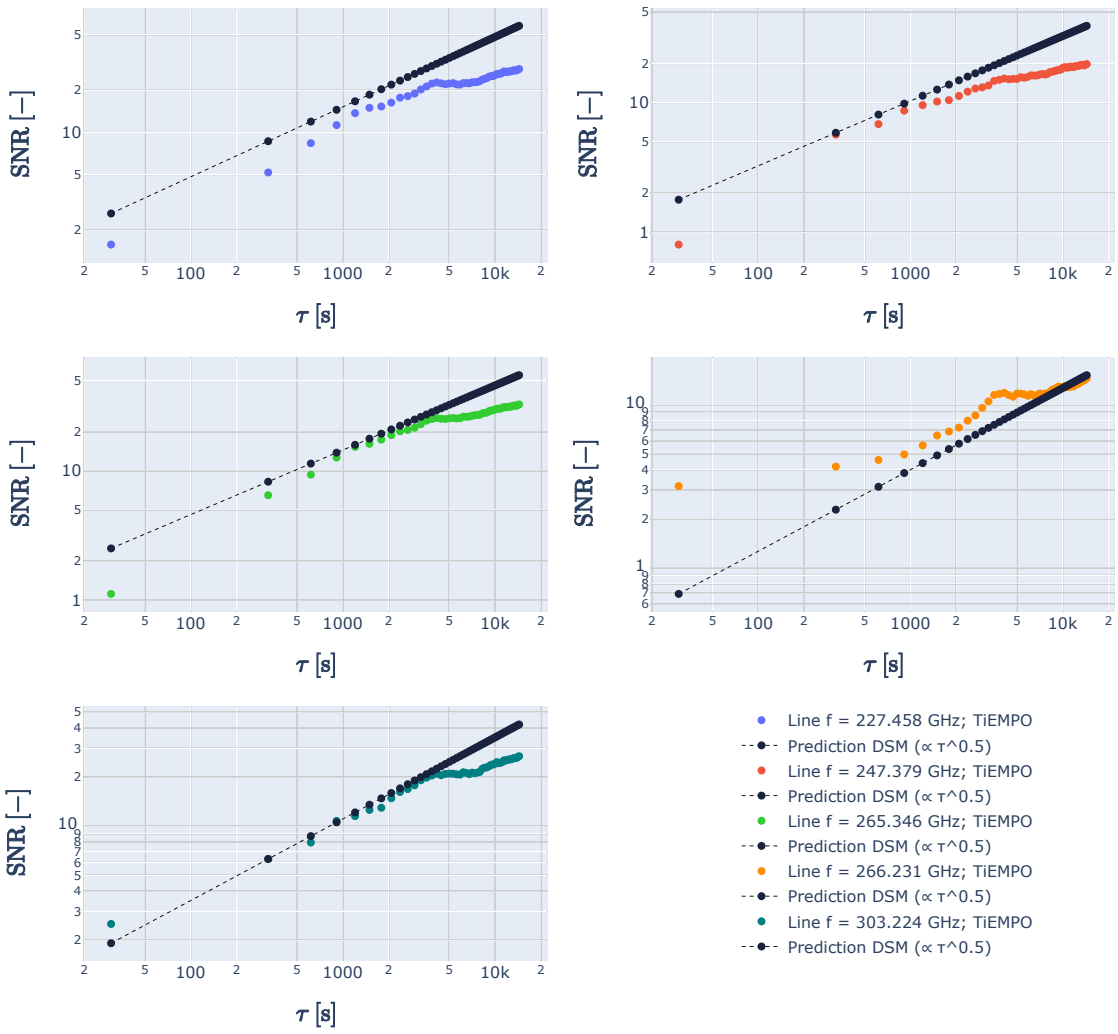


Figure 5.9: The signal-to-noise ratio as a function of integration time $\tau = 0.5 \cdot t_{obs}$ (on-source fraction is 0.5 due to dual chopping of the ON-position and the L-position) found in a `TiEMPO` simulation of the Preset chip (t_{obs}=28800s, `pwv_0`=1.0mm, and `EL`=60°) for five different lines (CO $(6-5)$, H_2O $(211-202)$, CO $(7-6)$, [CI] $(2-1)$, and CO $(8-7)$ in order of increasing line frequency) of a model of the J1329+2243 galaxy in the region of 200-310 GHz. The integration time was plotted from 30 s (one minute of observation time) up to 14400 s (eight hours of observation time). To compare the simulated results to the theory, the signal-to-noise ratios found by comparing the MDLF from `deshima-sensitivity` (DSM) to the respective line flux of the line. For the MDLF, the same input parameters were used as for the simulations. [Interact]

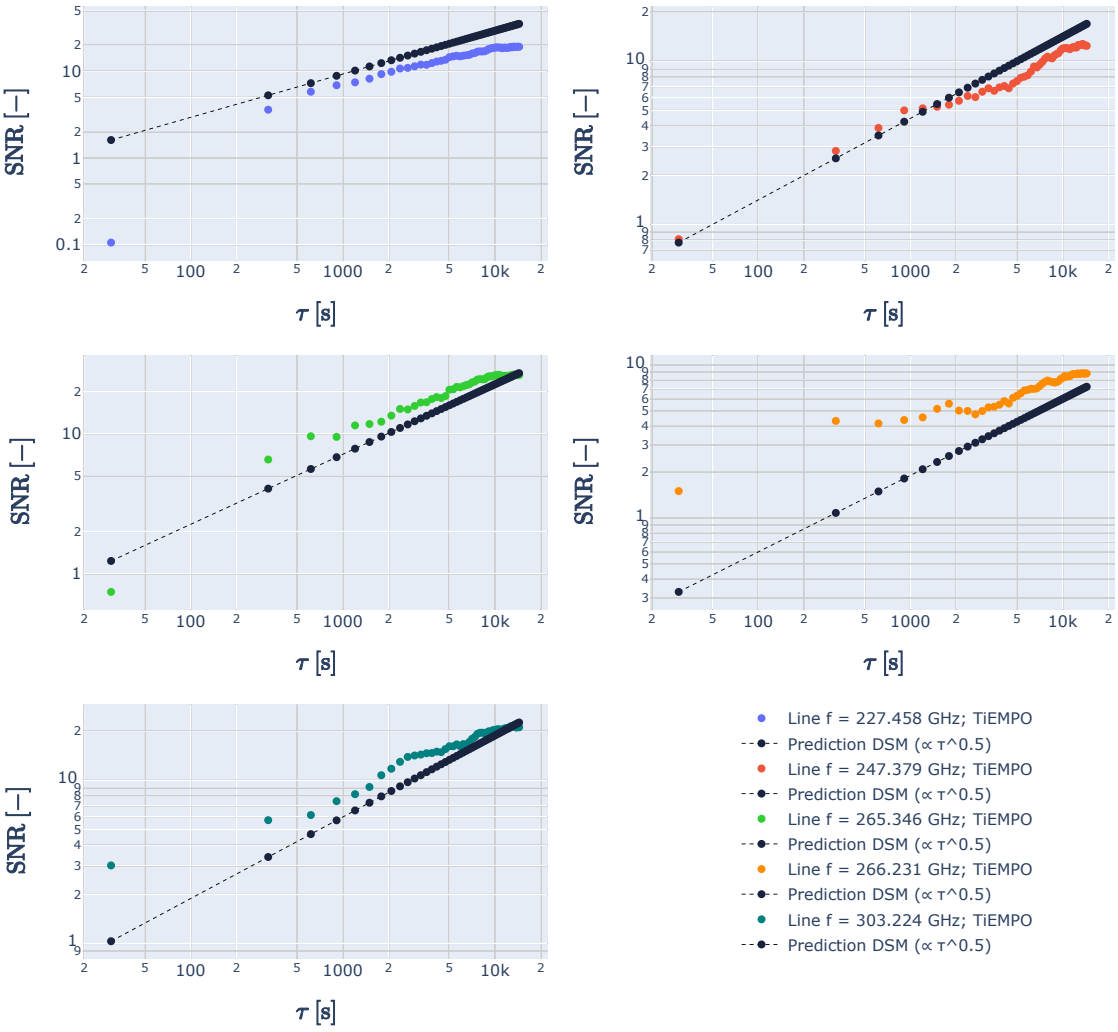


Figure 5.10: The signal-to-noise ratio as a function of integration time $\tau = 0.5 \cdot t_{obs}$ (on-source fraction is 0.5 due to dual chopping of the ON-position and the L-position) found in a `TiEMPO` simulation of the LT263 chip (t_{obs}=28800s, `pwv_0`=1.0mm, and `EL`=60°) for five different lines (CO $(6-5)$, H_2O $(211-202)$, CO $(7-6)$, [CI] $(2-1)$, and CO $(8-7)$ in order of increasing line frequency) of a model of the J1329+2243 galaxy in the region of 200-310 GHz. The integration time was plotted from 30 s (one minute of observation time) up to 14400 s (eight hours of observation time). To compare the simulated results to the theory, the signal-to-noise ratios found by comparing the MDLF from `deshima-sensitivity` (DSM) to the respective line flux of the line. For the MDLF, the same input parameters were used as for the simulations. [Interact]

5.2.3. Line Observations

Knowing the relations of the signal-to-noise ratios for each of the five emission lines as in figures 5.9 and 5.10, it is useful to relate them to the observed spectra[3].

CO $(6-5)$ line

By zooming in on the CO $(6-5)$ line (f_{line} = 227.458 GHz) of the J1329+2243 galaxy, the plots in figure 5.11 are found. For the LT263 chip, the channel closest to the line frequency has f_0 = 227.596 GHz and FWHM = 0.705 GHz, which means that it fully covers the entire line within the bandwidth of the box-approximation of the channel for a continuum (recall that it is shaped as shown in figure 3.4). A line is considered to be observed once an SNR $\geq$ 5 is reached. This happens after approximately 7.5 minutes of integration time, which would result in about 15 minutes of observation time. For the Preset chip, the channel closest to the line frequency is at f_0 = 227.601 GHz and has FWHM = 0.455 GHz, resulting in less of the line flux being covered by the effective bandwidth of the channel. Due to their positionings and measured temperatures, the filter channels both yield SNR curves that are to be expected: about half of the predicted curve. Using the Preset chip, the line is said to be observed after roughly 6 minutes of observation time.

H_2O $(211-202)$ line

By zooming in on the H_2O $(211-202)$ line (f_{line} = 247.379 GHz) of the J1329+2243 galaxy, the plots in figure 5.12 are found. For the LT263 chip, the FWHM of the channel closest to the line frequency, having f_0 = 247.437 GHz, has a value of 0.853 GHz, which is a little bit larger than for the CO $(6-5)$ line. The effect of this larger bandwidth can be seen in the SNR, as it more closely resembles the predicted curve instead of staying approximately halfway, like the CO $(6-5)$ (see figure 5.10). This results in an approximate observation time of 30 minutes required to get an SNR $\geq$ 5. For the Preset chip, the channel used for the calculation of the SNR has f_0 = 247.525 and FWHM = 0.495 GHz, which is really to that of the CO $(6-5)$ line. This similarity is also seen for the two SNR plots in figure 5.9. The line will reach an SNR $\geq$ 5 after roughly 9.5 minutes of observation time using the Preset chip.

CO $(7-6)$ and [CI] $(2-1)$ lines

The CO $(7-6)$ and [CI] $(2-1)$ lines of the J1329+2243 galaxy lie very close to each other with respective line frequencies of f_{line} = 265.346 GHz and f_{line} = 266.231 GHz. Due to the line width of 600 km s^{-1}, the Gaussian distributions result in the lines to overlap. When looking at how both chips respond to this in figure 5.13, one can see that connecting (interpolating) the measured temperatures approximates a skewed Gaussian distribution. When a real observation would be done, it is harder to find these lines as independent lines as they thus merge together. Through curve-fitting with a double Gaussian distribution, the two can be identified, as the smaller Gaussian distribution of [CI] $(2-1)$ is not fully merged with the one of CO $(7-6)$. Although they would thus be independently observable, it is problematic for the method used here. This effect of merging would even worsen when the lines would have larger line widths than the one used for the model here (600 km s^{-1}).

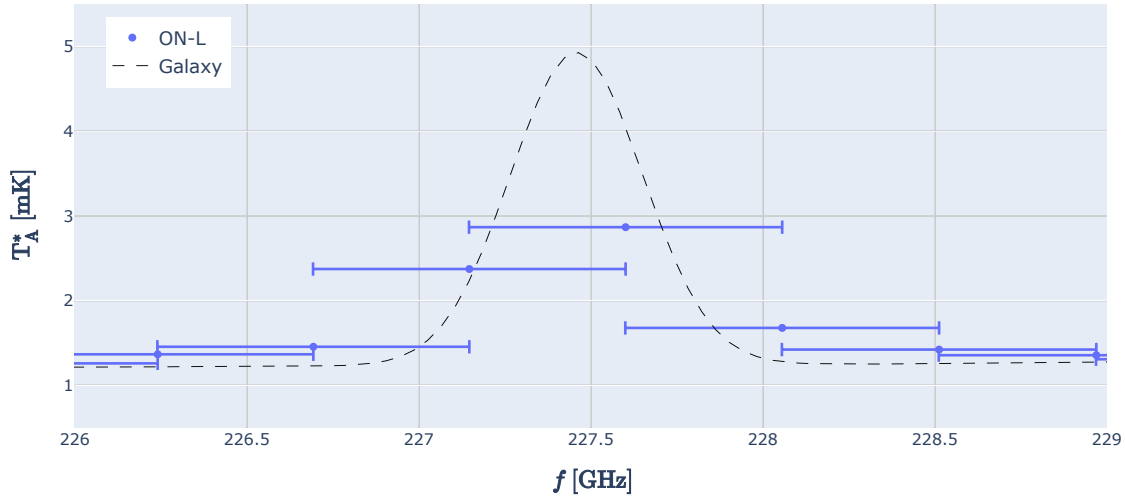
The Preset chip has its closest center frequencies at f_0 = 265.454 GHz (with FWHM = 0.531 GHz) and at f_0 = 265.984 GHz (with FWHM = 0.532 GHz). Notice that the latter is thus right between the two lines and is influenced by both line fluxes. This is also seen for the SNR plot of this line in figure 5.9, as the curve is way higher than the predicted line before it decelerates. It matches the predicted line afterwards, as its measured temperature is at almost the same height as the temperature of the actual line of the galaxy. So, while it looks promising that it is the only case of a matching SNR curve, this would actually be a misconception due to the effect the neighboring line has on this channel. The SNR curve for the CO $(7-6)$ line is actually well in line with what is expected: about halfway the predicted line. This is due to its channel center frequency being relatively close to the line frequency and its effective bandwidth spanning across the entire Gaussian distribution of the line. This also includes some of the [CI] $(2-1)$ line however, which results in the SNR being slightly too high. Still, while consequently being somewhat inaccurate, it can be assumed that an observation time of a little bit over 5.5 minutes would be enough to detect the CO $(7-6)$ line with an SNR $\geq$ 5. The required observation time for the [CI] $(2-1)$ line of approximately 30 minutes is way more questionable, as the filter channel used to find the SNR did not match the line frequency.

[3] Versions of these plots using linear axes are included in appendix B.4.2.

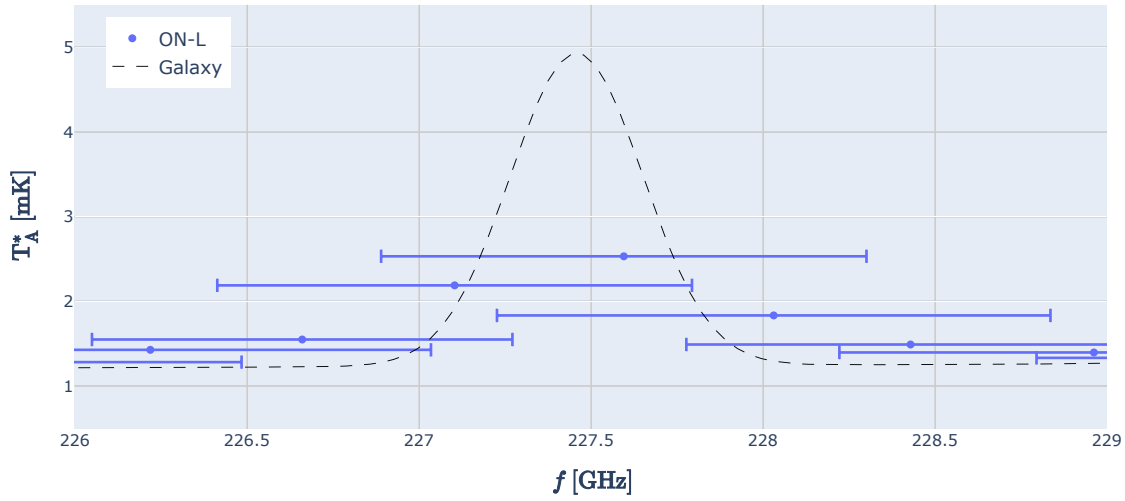
While the center frequencies of the LT263 used for the SNR fit the line frequencies better with f_0 = 265.363 GHz and f_0 = 266.331 GHz respectively, the SNR curve for the latter is still out of the ordinary (see figure 5.10). This is due to the large FWHM = 0.832 GHz of the second filter channel used. The one for the CO $(7-6)$ line on the other hand is almost perfectly aligned. The FWHM = 0.666 GHz of this channel causes it to have overlap with the neighboring line though, resulting in the SNR being too high for this chip as well. Again, the observation time required to officially observe these lines are questionable. For the CO $(7-6)$ line, an observation time of about 5.5 minutes is required to do a proper observation of the line. The SNR curve for [CI] $(2-1)$ line reads an integration time of about 25 min or an observation time of about 50 minutes to get to a value greater than 5, which should be a better approximation than the one found for the Preset chip, as the channel used to find the SNR matched the line frequency better.

CO $(8-7)$ line

For the last line that was analyzed, having a redshift corrected line frequency of f_{line} = 303.224 GHz and shown in figure 5.14, the filter channels with the closest center frequencies were the ones with f_0 = 303.477 GHz (and FWHM = 0.607 GHz) for the Preset chip and with f_0 = 303.407 (and FWHM = 0.905 GHz) for the LT263 chip. Both chips show the same effect: two filter channels sit at almost equal distance from the line frequency. Therefore, the one closer to it has a measured temperature which is of only a slightly larger magnitude. Looking at all other filter channels neighboring these two in question, it seems like almost all of the line is being measured by just these two channels. This is seen in the SNR plots in figures 5.9 and 5.10, as the predicted curve is met for both chips up until the aforementioned 'deceleration' time. Given the integration time, only for the Preset chip can be said that the curve tends to halfway the using `deshima-sensitivity` predicted curve for integration times longer than this aforementioned 'deceleration' time. The observation times required to reach an SNR ≥ 5 is about 4.5 minutes for the Preset chip, and approximately 8.5 minutes for the LT263 chip.

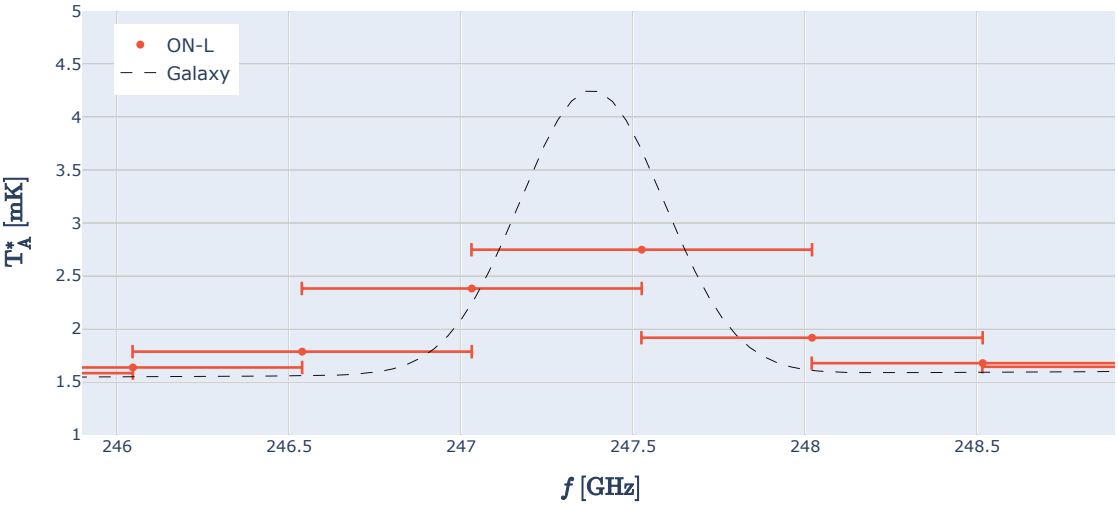


(a) Preset chip, $T_A^*(f)$ [Interact]

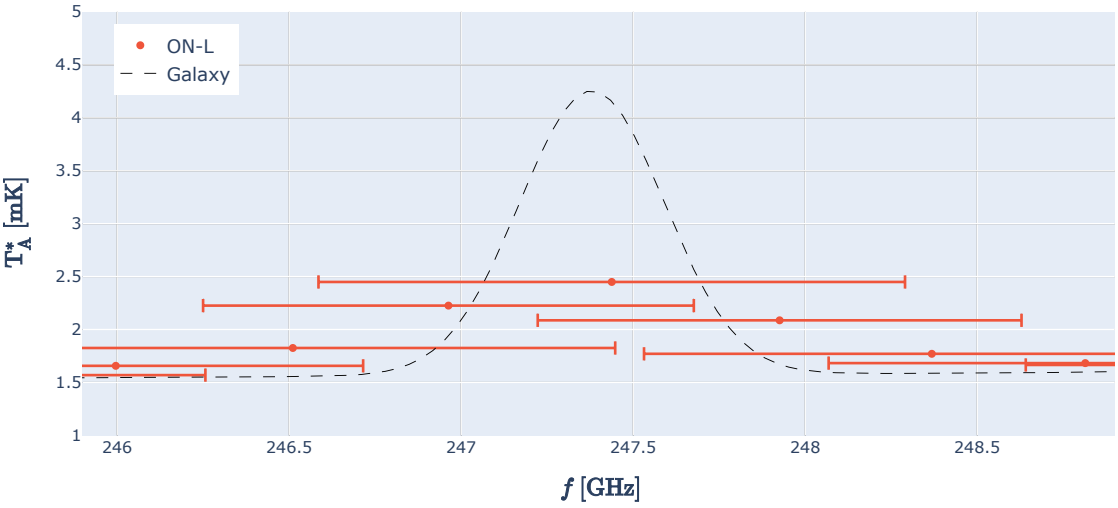


(b) LT263 chip, $T_A^*(f)$ [Interact]

Figure 5.11: Atmosphere-corrected antenna temperature for both the (a) Preset chip and the (b) LT263 chip, used in calculating the signal-to-noise ratio for the CO $(6-5)$ line of the high-redshift J1329+2243 galaxy visible at a redshift-corrected line frequency of approximately f_{line} = 227.458 GHz. The error bars shown for center frequencies f_0 of the filter channels define the full width at half maximum (FWHM) that shapes the Lorentzian of the filter.

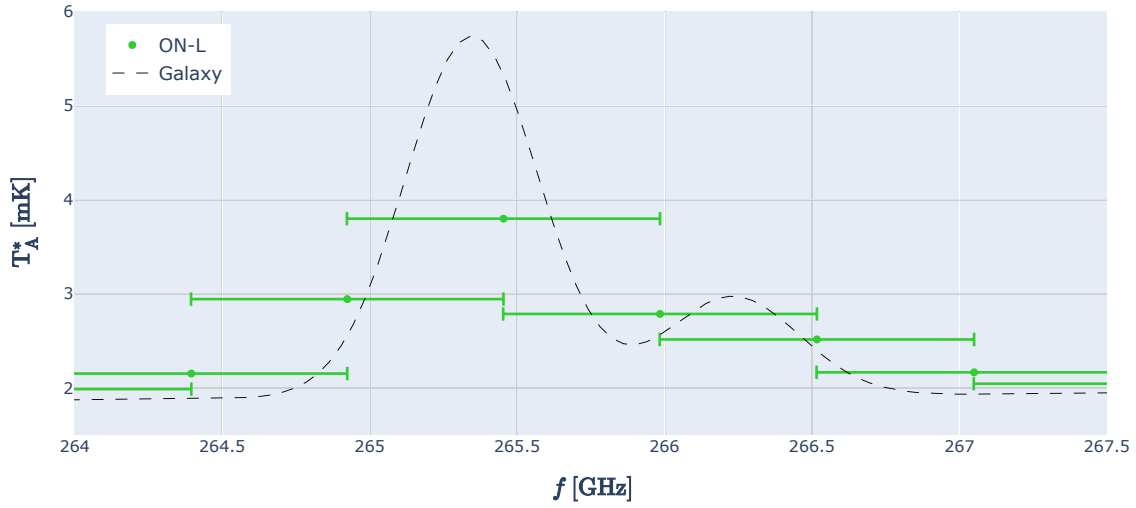


(a) Preset chip, $T_A^*(f)$ [Interact]

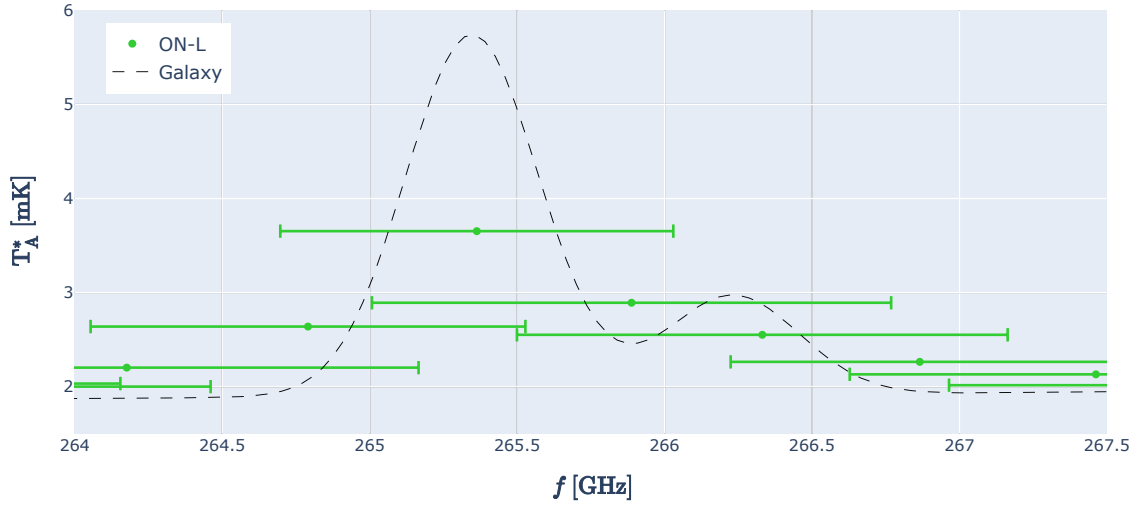


(b) LT263 chip, $T_A^*(f)$ [Interact]

Figure 5.12: Atmosphere-corrected antenna temperature for both the (a) Preset chip and the (b) LT263 chip, used in calculating the signal-to-noise ratio for the H_2O $(211 - 202)$ line of the high-redshift J1329+2243 galaxy visible at a redshift-corrected line frequency of approximately f_{line} = 247.379 GHz. The error bars shown for center frequencies f_0 of the filter channels define the full width at half maximum (FWHM) that shapes the Lorentzian of the filter.

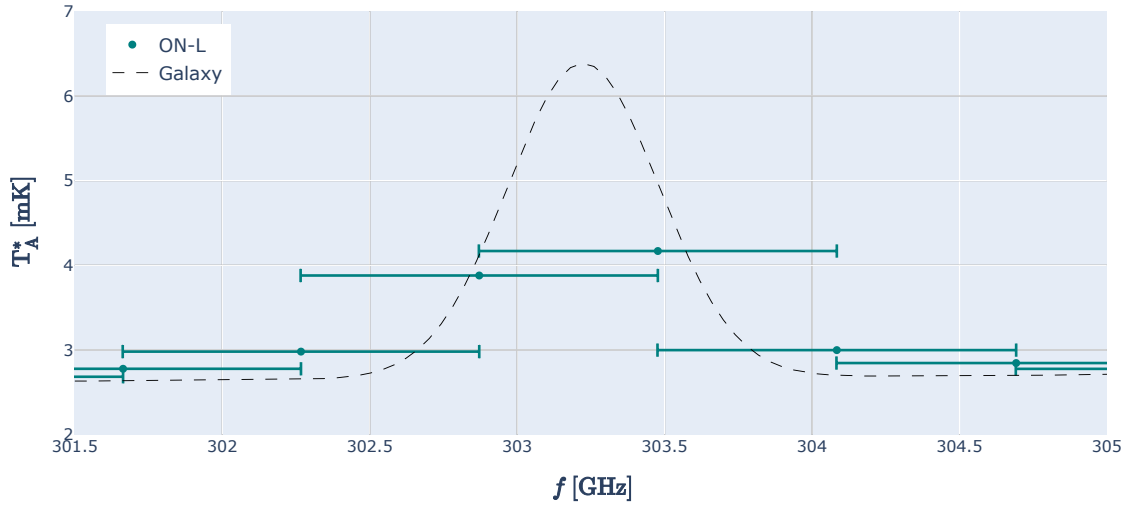


(a) Preset chip, $T_A^*(f)$ [Interact]

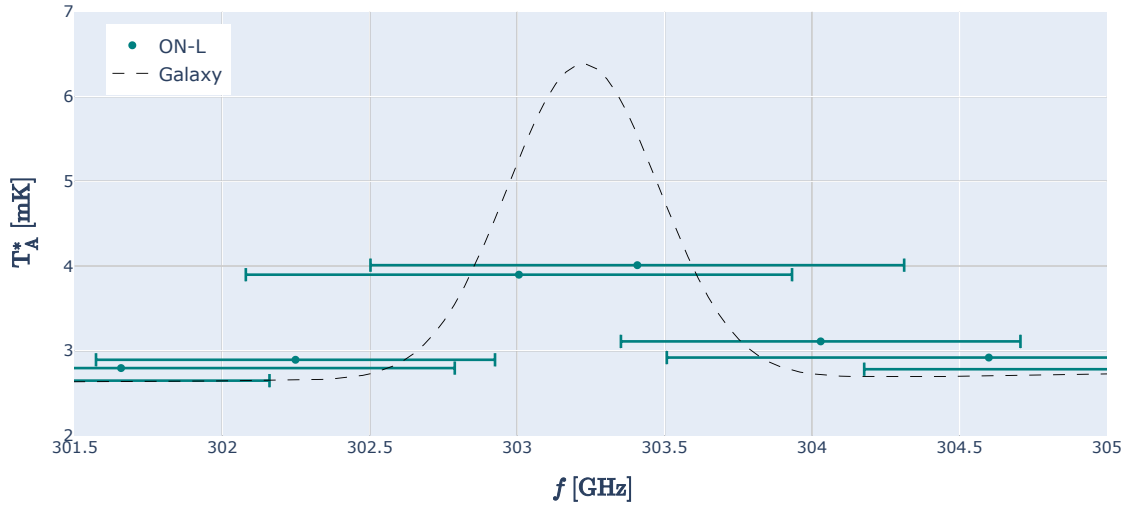


(b) LT263 chip, $T_A^*(f)$ [Interact]

Figure 5.13: Atmosphere-corrected antenna temperature for both the (a) Preset chip and the (b) LT263 chip, used in calculating the signal-to-noise ratio for the CO $(7-6)$ and [CI] $(2-1)$ lines of the high-redshift J1329+2243 galaxy visible at a redshift-corrected line frequencies of approximately f_{line} = 265.346 GHz and f_{line} = 266.231 GHz respectively. The error bars shown for center frequencies f_0 of the filter channels define the full width at half maximum (FWHM) that shapes the Lorentzian of the filter.



(a) Preset chip, $T_A^*(f)$ [Interact]



(b) LT263 chip, $T_A^*(f)$ [Interact]

Figure 5.14: Atmosphere-corrected antenna temperature for both the (a) Preset chip and the (b) LT263 chip, used in calculating the signal-to-noise ratio for the CO $(8-7)$ line of the high-redshift J1329+2243 galaxy visible at a redshift-corrected line frequency of approximately f_{line} = 303.224 GHz. The error bars shown for center frequencies f_0 of the filter channels define the full width at half maximum (FWHM) that shapes the Lorentzian of the filter.

6

Conclusion

In this report, a pre-flight simulation of the current lab-measured DESHIMA2.0 instrument containing the LT263 chip, which will be mounted in the ASTE telescope later this year, is performed. The results of this simulation are then used to make an estimate of its total system performance in observing dusty star-forming galaxies.

First, the lab-measurements of the LT263 chip are used to analyze the characteristics of the chip. The 332 filter channels that it consists of, are approximated by Lorentzian functions whose center frequencies lie in the band of 204-391 GHz. The spectral resolution of the filters is found to be $f/\delta f = 340 \pm 50$. The maximum coupling efficiencies of the filters average at 14 ± 4 % and tend to decrease for increasing center frequencies. These compare with the idealized chip, referred to as the Preset chip, which has a constant spectral resolution and coupling efficiency of 500 and 0.4 respectively – therefore meeting goal parameters set for DESHIMA2.0.

Two measures of the sensitivity are examined using `deshima-sensitivity`. While the LT263 chip meets the baseline requirement of the noise equivalent flux density (NEFD) for lower frequencies, it is seen to be worse than the baseline requirement of the minimum detectable line flux (MDLF).

By successfully adapting `TiEMPO` to be able to implement the lab-measured data of the chip, the model could be used to simulate the observations of 8 hours for both the LT263 and the Preset chip. Here, a model galaxy based on reported flux densities of the ultraluminous dusty star-forming galaxy J1329+2243 with redshift $z = 2.04$ is used as the input galaxy spectrum. The output of those simulations is the measured sky temperature at six different sky positions.

After applying an ON-OFF (dual) sky chopping technique to the simulated observation to cancel atmospheric noise, the atmosphere-corrected antenna temperature of the J1329+2243 galaxy is found with an on-source fraction of 0.5. The zero-mean noise left in the spectrum, which also includes photon and recombination noise, shows to have its standard deviation scaling inversely proportional to the square root of time.

The J1329+2243 galaxy has emission lines of molecular gases in the frequency band of DESHIMA2.0. The signal-to-noise ratio is analyzed for five of these emission lines, all in the band of 200-310 GHz: CO$(6-5)$, $H_2O(211-202)$, CO$(7-6)$, [CI]$(2-1)$, and CO$(8-7)$. A line is said to be detected once a signal-to-noise ratio of $\geq$5 is reached. The less luminous line flux of [CI]$(2-1)$ shows to be affected by its neighboring line (CO$(7-6)$) with whom it partially merges.

Based on the results, it can be verified that the current DESHIMA2.0 instrument is capable of detecting all analyzed lines of the ultraluminous high-redshift galaxy J1329+2243 within an observation time of 50 minutes (SNR $\geq$5). The performance is even sufficient to detect the bright CO$(7-6)$ line after 5.5 minutes of observation time. It takes 15 minutes for CO$(6-5)$, 30 minutes for $H_2O(211-202)$, 50 minutes for [CI]$(2-1)$, and 8.5 minutes for CO$(8-7)$. For the Preset chip, a signal-to-noise ratio $\geq$5 is reached roughly twice as fast, which is in line with the ratio of the minimum detectable line fluxes of $\approx$2.

Large line widths might cause issues in detecting emission lines that are close together in the spectrum, like the case of the merging CO$(7-6)$ emission line and relatively faint [CI]$(2-1)$ line. In detecting these, the effective bandwidths of the filter channels used to detect them is too large, causing the detected signal-to-noise ratio to be too optimistic.

6.1. Future Prospects

The results found in this project can be compared to and used as reference for real observations to be made later this year, when DESHIMA2.0 is expected to be mounted in the ASTE telescope for its campaign. It would be interesting to see how closely the results from this project resemble the real total system performance in observing a DSFG similar to the J1329+2243 galaxy.

In the estimation of the performance, several assumptions were made that can be improved on.

First, it is assumed that the band-pass filter transmissions consist of Lorentzian functions that each can be approximated by a boxcar function in `deshima-sensitivity`. However, the actual coupling efficiencies are not perfect and might even deviate from the Lorentzian functions. While `TiEMPO` already uses the actual Lorentzian functions, the input for the LT263 used in this project is still a curve-fit of the actual responses measured in the lab. As `TiEMPO` is proven to be adaptable, it is possible to perform simulations using the actual raw lab-measured data. It would be interesting to see if and by how much the estimation of the performance improves.

For this model galaxy, it is assumed that the $\mathrm{NEFD}_{\mathrm{continuum}}$ from `deshima-sensitivity` would show the best approximation of the noise found in the sky chopped signal. While this is shown to be true for the continuum of the model galaxy spectrum in the validation of the standard deviation in the noise, the NEFD of the emission lines of the galaxy is expected to behave differently, as these have a certain line width. Looking into what the actual NEFD would be, is the next step here.

Another addition to `TiEMPO` would be to include different optical efficiencies and foreground noises for each position. This would allow better approximations of the real read sky temperature signals, as chop/nodding will be applied for DESHIMA2.0. An analysis of the revisioned model could then be looked into to confirm the effectiveness of certain sky chopping techniques.

In this project, the calculation of the signal-to-noise ratio of an emission line was done by taking the measured temperature of the filter channel with center frequency closest to the line frequency. This showed to affect the detection for lines that show overlap with neighboring lines in the spectrum. As real measurements use methods like curve-fitting of the galaxy spectrum to compare the signal with its noise, the signal-to-noise ratios can be better estimated using such a technique. In particular, a technique that involves combining the responses of neighboring filter channels would be more ideal, since it has to be taken into account that the filter channels are neither spaced perfectly logarithmically, nor show constant spectral and coupling efficiencies. The effect of incorporating this into the analysis on sensitivity measures such as the signal-to-noise ratio would improve the similarity to actual results.

In general, the signal-to-noise ratios are proportional to the square root of time, but show a certain time after which it suddenly decelerates in its growth. While executing a different method to find the signal-to-noise ratio, this certain 'deceleration' time one finds in the time-dependent signal-to-noise curves can be revisioned, as they might be caused by the method through which the signal-to-noise ratio was determined. For now, this observation stays inconclusive.

7

Acknowledgements

First of all, I want to thank Akira Endo for his great enthusiasm and for offering me the opportunity to do my bachelor end project under his supervision. While you were in the midst of the final stages of the DESHIMA2.0 project and in preparing to go to Chile, you were always there for me to answer questions and to give advice. Especially, I want to thank you for your patience and faith in me. While I was struggling with health problems, you always supported me and still motivated me to get the most out of my project. You were always optimistic and are the reason this project is in the finished state that it is in right now.

Additionally, I want to thank Matus Rybak for his help in adapting TiEMPO, as well as kindly providing the data necessary to get a simulation up and running. Moreover, I want to thank both Matus Rybak and Sander Otte for their effort in forming the Thesis committee. I also want to thank Kenichi Karatsu for his help when I was practicing for my presentation.

I would like to thank my brother in particular, as he was always there to help me if I had to figure out more specific problems that I had or just wanted to discuss the theory behind the research. The exemplary dedication you have in your own studies has always been inspiring to me. I also want to thank my parents for always being there to support me and for taking care of me.

Lastly, I would like to thank all my friends that I have made over the past three years through the Bachelor of Applied Physics. Even when studying was hard, we supported each other, remotely or from the 'Plein'. You made me enjoy these years studying in Delft. I would like to thank those that were always in for a game of pool in particular.

References

[1] C.M. Casey, Desika Narayanan, and Asantha Cooray. “Dusty star-forming galaxies at high redshift”. In: *Physics Reports* 541.2 (2014). Dusty star-forming galaxies at high-redshift, pp. 45–161. ISSN: 0370-1573. DOI: https://doi.org/10.1016/j.physrep.2014.02.009. URL: https://www.sciencedirect.com/science/article/pii/S0370157314000477.

[2] A. Endo et al. “First light demonstration of the integrated superconducting spectrometer”. In: *Nature Astronomy* 3 (2019), pp. 989–996. DOI: 10.1038/s41550-019-0850-8.

[3] M. Rybak et al. “Deshima 2.0: Rapid Redshift Surveys and Multi-line Spectroscopy of Dusty Galaxies”. In: *J Low Temp Phys* 209 (2022), pp. 766–778. DOI: 10.1007/s10909-022-02730-y. URL: https://doi.org/10.1007/s10909-022-02730-y.

[4] J.J.A. Baselmans. “Kinetic Inductance Detectors”. In: *J Low Temp Phys* 167 (2012), pp. 292–304. DOI: 10.1007/s10909-011-0448-8. URL: https://doi.org/10.1007/s10909-011-0448-8.

[5] A. Taniguchi et al. “DESHIMA 2.0: Development of an Integrated Superconducting Spectrometer for Science-Grade Astronomical Observations”. English. In: *Low Temperature Physics* 209 (Nov. 2022), pp. 278–286. ISSN: 1063-777X. DOI: 10.1007/s10909-022-02888-5.

[6] H. Ezawa et al. “The Atacama Submillimeter Telescope Experiment (ASTE)”. In: *Ground-based Telescopes*. Ed. by Jacobus M. Oschmann Jr. Vol. 5489. International Society for Optics and Photonics. SPIE, 2004, pp. 763–772. DOI: 10.1117/12.551391. URL: https://doi.org/10.1117/12.551391.

[7] J. Binney and S. Tremaine. *Galactic Dynamics*. Princeton, USA: Princeton University Press, 1987.

[8] L.S. Sparke and J.S. Gallagher. *Galaxies in the Universe - An Introduction*. Cambridge, UK: Cambridge University Press, 2007.

[9] M.B. Bayliss et al. “Line-of-Sight Structure Toward Strong Lensing Galaxy Clusters”. In: *The Astrophysical Journal* 783.1 (Feb. 2014), p. 41. DOI: 10.1088/0004-637X/783/1/41. URL: https://dx.doi.org/10.1088/0004-637X/783/1/41.

[10] K. Sharon et al. “Strong Lens Models for 37 Clusters of Galaxies from the SDSS Giant Arcs Survey*”. In: *The Astrophysical Journal Supplement Series* 247.1 (Feb. 2020), p. 12. DOI: 10.3847/1538-4365/ab5f13. URL: https://dx.doi.org/10.3847/1538-4365/ab5f13.

[11] K.C. Harrington et al. “Turbulent Gas in Lensed Planck-selected Starbursts at z 1–3.5”. In: *The Astrophysical Journal* 908.1 (Feb. 2021), p. 95. DOI: 10.3847/1538-4357/abcc01. URL: https://dx.doi.org/10.3847/1538-4357/abcc01.

[12] E. da Cunha et al. “An ALMA Survey of Sub-millimeter Galaxies in the Extended Chandra Deep Field South: Physical Properties Derived from Ultraviolet-to-radio Modeling”. In: *The Astrophysical Journal* 806.1, 110 (June 2015), p. 110. DOI: 10.1088/0004-637X/806/1/110. arXiv: 1504.04376 [astro-ph.GA].

[13] M. Rybak. *LinesTable.data*. Observed / Predicted line fluxes for few select galaxies. Dec. 2021.

[14] R. Stull. *Practical Meteorology: An Algebra-based Survey of Atmospheric Science*. Vancouver, B.C., Canada: University of British Columbia, 2017. URL: https://www.eoas.ubc.ca/books/Practical_Meteorology/.

[15] E. Huijten. “TiEMPO:Time-Dependent End-to-End Model for Post-process Optimization of the DESHIMA Spectrometer”. Bachelor’s Thesis [Delft University of Technology]. 2020. URL: http://resolver.tudelft.nl/uuid:5302e10e-3b56-4d4d-a1fd-6a1d58f57abd.

[16] T. Takekoshi et al. “DESHIMA on ASTE: On-Sky Responsivity Calibration of the Integrated Superconducting Spectrometer”. In: *J Low Temp Phys* 199 (2020), pp. 231–239. DOI: 10.1007/s10909-020-02338-0. URL: https://doi.org/10.1007/s10909-020-02338-0.

[17] A. Pascual Laguna et al. “Terahertz Band-Pass Filters for Wideband Superconducting On-Chip Filter-Bank Spectrometers”. In: *IEEE Transactions on Terahertz Science and Technology* 11.6 (2021), pp. 635–646. DOI: 10.1109/TTHZ.2021.3095429.

[18] A. Endo. *WP1.2 DESHIMA 2.0 Hardware Requirements*. Sept. 2018. URL: https://deshima.kibe.la/shared/entries/ff96eb1f-ef80-41e2-adf8-178769a72e3f.

[19] Y. Asaki et al. “Verification of the Effectiveness of VSOP-2 Phase Referencing with a Newly Developed Simulation Tool, ARIS”. In: *Publications of the Astronomical Society of Japan* 59.2 (Apr. 2007), pp. 397–418. ISSN: 0004-6264. DOI: 10.1093/pasj/59.2.397. eprint: https://academic.oup.com/pasj/article-pdf/59/2/397/18531864/pasj59-0397.pdf. URL: https://doi.org/10.1093/pasj/59.2.397.

[20] K. Volk. *Chopping and Nodding for Mid-Infrared Astronomy*. http://www.gemini.edu/sciops/instruments/mir/chopnod.pdf. 2007.

[21] A. Endo et al. *deshima-dev/deshima-sensitivity: Update release*. Version v0.4.1. June 2022. DOI: 10.5281/zenodo.6642924. URL: https://doi.org/10.5281/zenodo.6642924.

[22] E. Huijten et al. “TiEMPO: open-source time-dependent end-to-end model for simulating ground-based submillimeter astronomical observations”. In: *Journal of Astronomical Telescopes, Instruments, and Systems* 8.2 (2022), p. 028005. DOI: 10.1117/1.JATIS.8.2.028005. URL: https://doi.org/10.1117/1.JATIS.8.2.028005.

[23] A. Endo et al. “On-chip filter bank spectroscopy at 600–700 GHz using NbTiN superconducting resonators”. In: *Applied Physics Letters* 103.3 (July 2013). 032601. ISSN: 0003-6951. DOI: 10.1063/1.4813816. eprint: https://pubs.aip.org/aip/apl/article-pdf/doi/10.1063/1.4813816/14288900/032601_1_online.pdf. URL: https://doi.org/10.1063/1.4813816.

[24] A. Endo. *Performance predictions for LT223 ”flight” chip 1 for emission-line detection (interactive plots)*. Sept. 2021. URL: https://deshima.kibe.la/shared/entries/60d95a38-744e-44ce-b1d4-a1e00f66c760.

[25] A. Endo et al. “Wideband on-chip terahertz spectrometer based on a superconducting filterbank”. In: *Journal of Astronomical Telescopes, Instruments, and Systems* 5.3 (2019), p. 035004. DOI: 10.1117/1.JATIS.5.3.035004. URL: https://doi.org/10.1117/1.JATIS.5.3.035004.

[26] L. Ferrari et al. “Antenna Coupled MKID Performance Verification at 850 GHz for Large Format Astrophysics Arrays”. In: *IEEE Transactions on Terahertz Science and Technology* 8.1 (2018), pp. 127–139. DOI: 10.1109/TTHZ.2017.2764378.

[27] A. Endo. *Measured parameters of DESHIMA 1.0 on ASTE*. Sept. 2018. URL: https://deshima.kibe.la/shared/entries/87fec07f-d8c7-46e6-8557-7a83c46caaa2.

[28] Y. Roelvink. “Simulation of a High-Redshift Line-Emitting Galaxy Detection with DESHIMA using TiEMPO”. Bachelor’s Thesis [Delft University of Technology]. 2020. URL: http://resolver.tudelft.nl/uuid:c878c9d5-f9af-44fd-83c4-a4bd8a7496b5.

A

Python Code

This appendix contains the Python code behind the main steps taken and processes covered in the project. It covers the input parameters to run the TiEMPO simulations for this project, as well as a useful class plot_TiEMPO and set of functions built to ease plotting of the data from the simulations. A detailed report on one of the adaptations in the TiEMPO code is also added as the last section.

A.1. execute_TiEMPO_custom.py

Using the structure given by Matus Rybak, private communication, the input parameters for TiEMPO are set here, such that a simulation can be run. This is also where the user can choose between using a custom chip, like the LT263 chip, or use the Preset chip.

```
import numpy as np
import time
import tiempo_deshima_custom_chipMR as tiempo_deshima_custom_input

# Number of parallel processes (often number of threads minus a few, most cores have 2
    threads)
threads = 30 # BLADE server handles 30 threads fine, as it has 32 cores

# Total observing time:
Tobs = 8*3600 # s (1800 sec is half an hour)
n_batches = 8 # Number of batches the total observing time will be split into

# Use LT263 chip data:
usechip = True

# Path of the 'All TiEMPO files' folder
basefolder = "./TiEMPO/"

# Paths of the input data folders (ARIS and source)
arisfolder = basefolder + "Output_ARIS"
sourcefolder = "Spectrum_MatthijsRoos/"

# Path of the folder where the datafiles will be saved
savefolder = basefolder + "simResults"

# Saving time_vector, center_freq in savefolder:
save_tf = True

# For when LT263 chip data is used:
LT263data = "https://raw.githubusercontent.com/MatthijsRoos/thesis/main/DESHIMA2.0
    _FlightChip_Filters.run991.wb-31dB.csv"
dF_Toptica = -0.96 # Toptica frequency offset
run_new_filterbank = False # True when new filterbank has to created (run just once),
    otherwise False

# source: assign right spectrum data to frequency_gal and spectrum_gal
# savename: format of the name of the simulation files
if usechip:
```

```
    source = sourcefolder + "spectrum_J1329_Jy_SCUBA_chip.data"
    savename = f"TiEMPO_LT263_{Tobs}s_"+time.strftime("%d-%m-%Y_%H%M")
else:
    source = sourcefolder + "spectrum_J1329_Jy_SCUBA_adj.data"
    savename = f"TiEMPO_preset_{Tobs}s_"+time.strftime("%d-%m-%Y_%H%M")

# Load spectrum data
with open(source,'r') as s:
    lines = s.readlines()
    f, FD = zip(*[line.split() for line in lines])

frequency_gal = np.array(f,dtype='float32')
spectrum_gal = np.array(FD,dtype='float32')

# Create the input for tiempo_deshima_custom_input.run_tiempo()
# F_min, spec_res, f_spacing, and num_filters represent the preset for a (perfect) chip
input_run_tiempo = dict(input_dictionary = 'manual',\
                        prefix_atm_data = 'aris200602.dat-',\
                        sourcefolder = arisfolder,\
                        save_name_data = savename,\
                        savefolder = savefolder,\
                        save_P = False,\
                        save_T = True,\
                        n_jobs = threads,\
                        n_batches = n_batches,\
                        obs_time = Tobs,\
                        grid = 1,\
                        separation = 1.,\
                        EL = 60,\
                        pwv_0 = 1,\
                        F_min = 220e9,\
                        num_bins = len(frequency_gal),\
                        spec_res = 500,\
                        f_spacing = 500,\
                        num_filters = 347,\
                        windspeed = 10,\
                        use_galspec = False,\
                        frequency_gal = frequency_gal,\
                        spectrum_gal = spectrum_gal,\
                       )

if usechip:
    input_run_tiempo["chipdata"] = LT263data
    input_run_tiempo["dF"] = dF_Toptica
    input_run_tiempo["run_new_filterbank"] = run_new_filterbank

# Start to run TiEMPO
TimeNow = time.time()

print("Running TiEMPO, this will take a while, please do not close...")
time_vector, center_freq = tiempo_deshima_custom_input.run_tiempo(**input_run_tiempo)

print(f"Time for iteration = {(time.time()-TimeNow)/60:.1f} min")
TimeNow = time.time()

if save_tf:
    savefolder = tiempo_deshima_custom_input.interface.convert_folder(savefolder)
    np.save(savefolder.joinpath(savename+'_t_'), time_vector)
    np.save(savefolder.joinpath(savename+'_f_'), center_freq)

print(f"Done. Files saved in {savefolder} as {savename}")
```

A.2. plot_TiEMPO.py

This class together with the provided functions can be used to create all plots shown in chapter 5. The basics on how it is used will be covered in the following subsections.

```
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots
import pandas as pd
import sys
from tiempo_deshima_custom_input.DESHIMA.desim import minidesim as dsm

# ABBA function created by Yannick Roelvink, adapted to fit format
from simPlots.ABBA_chop import simulate_ABBA_chop

import math
h = 6.62607004e-34
k = 1.38064852e-23

colors = ["#636EFA", # Royal Blue
          "#EF553B", # Tomato Red
          "#32CD32", # Lime Green
          "#FF8C00", # Dark Orange
          "#008080", # Teal
          "#8A2BE2", # Blue Violet
          "#C71585", # Medium Violet Red
          "#1A233D", # Midnight Blue
         ]

class plot_TiEMPO:
    def __init__(self, simresult, n_batches = 1, Q = 500, beam_radius = 5.,\
                 sampling_rate = 160, chopping_rate = 10, n = 3):
        """
        simresult: string (path to .npy files containing the results from TiEMPO)
            Only the base of the path should be given (i.e., without '_T_0.npy').
            Example format: "simResults/TiEMPO_LT263_[Tobs]s_[dd]-[mm]-[yyyy]_[hhmm]"
        n_batches: int
            Number of batches the entire observation is divided up into. Default is 1.
        Q: int/float OR array, unit: no unit
            The loaded quality factor to derive the FWHM of each filter channel, used
            for plot(). Applies FWHM = F0/FWHM, where F0 is the center frequency.
            Default is 500.
        beam_radius: float, unit: m
            Radius of the Gaussian beam. Default is 5.
        sampling_rate: int, unit: Hz
            Rate at which measurements are made; or, equivalently, number of
            measurements per second. Default is 160.
        chopping_rate: int, unit: Hz
            Rate at which position switching on the sky is performed by the
            rotating mirror in the cabin optics. Default is 10.
        """
        self.t = np.load(simresult+'_t_'+'.npy') # time_vector
        f = np.load(simresult+'_f_'+'.npy') # center_freq
        self.f = f*1e-9 # GHz
        self.FWHM = self.f / Q # full width half maximum, GHz
        #self.P = np.load(simresult+'_P_0'+'.npy') # matrix of the power
        T_batches = [np.load(simresult+'_T_'+str(i)+'.npy') for i in range(n_batches)]
        T = np.concatenate(T_batches, axis=2) # matrix of the sky temperature
        self.T = dict(L = T[0,:,:], # sky measurement left OFF
                      ON = T[1,:,:], # sky measurement center (with galaxy) ON
                      R = T[2,:,:], # sky measurement right OFF
                      U = T[3,:,:], # sky measurement up OFF
                      D = T[4,:,:], # sky measurement down OFF
                      OFF = T[5,:,:], # sky measurement center OFF
                     )
        self.simresult = simresult
        self.beam_radius = beam_radius
        self.sampling_rate = sampling_rate
        self.chopping_rate = chopping_rate

    def plot(self, mode = ["ON-OFF"], spectrum = None, eta_atm = None, n = 3,\
```

```
             showFWHM = False, t_max = None, zoom = None,\
             plotfolder = None, html = True):
        """
        mode: list of strings
            Each string may contain one of the following three mode options.
            - One of the sky measurements for Tsky ("L", "ON", "R", "U", "D", "OFF");
            - Two of the sky measurements for Tsky, separated by a minus sign ("-"),
              that will be used for sky chopping using chopDUAL();
            - "ABBA", such that the output of chopABBA() will be used.
            Default is ["ON-OFF"].
        spectrum: string (path to .data file built by SpectrumGenerator.py) OR None
            When given the path to the file containing the galaxy spectrum [Jy],
            an overlay of the galaxy spectrum is plotted alongside the Tsky(f).
            When given None, no overlay of the galaxy spectrum is plotted.
            Default is None.
        eta_atm: array OR None, unit: no unit
            When given an array containing the eta_atm for each frequency of
            the filterbank chip, the Tsky is corrected by this data accordingly.
            When given None, no correction is done and the raw Tsky is plotted.
            Default is None.
        n: int
            n-factor, the number of chopping pairs to be observed per nodding
            position (used for chopABBA()). Default is 3.
        showFWHM: bool
            If True, the FWHM for each filter channel is plotted. Default is False.
        t_max: int, unit: s
            To plot the temperature up until this given timestamp (slices the T array)
            if it is not equal to None. Default is None.
        zoom: dict OR None
            If given a dictionary of the form dict(x=[x_min,x_max], y=[y_min,y_max]),
            The saved plot is zoomed to fit the described rectangle. There will be no
            zooming in (or out) when zoom is set to None. Default is None.
        plotfolder: string
            When not equal to None, an html file of the plot will be saved in this folder.
            Default is None.
        html: bool
            When True, the figure is saved as an interactive .html file.
            When False, the figure is saved as a static .svg file.
            Default is True.

        outputs: Plotly figure
        """
        # For saving:
        chip, sec = self.simresult.split("_")[-4:3]
        savename = f"T_{chip}_{sec}_{mode[0]}"
        w, h = 900, 500

        fig = go.Figure()

        fig.update_yaxes(title='$\mathrm{T_{sky}\:[K]}$',range=[0,300])

        if not (t_max is None):
            fig.update_yaxes(autorange=True)
            fig.update_layout(margin=dict(b=20,t=20))
            w, h = 500, 300
            t_max = int(t_max)
            savename += f"_{t_max}s"
            t_max *= self.sampling_rate

        for i,m in enumerate(mode):
            if m == "ABBA":
                y = chopABBA(self.T["L"][:,:t_max],\
                             self.T["ON"][:,:t_max],\
                             self.T["R"][:,:t_max],\
                             sampling_rate = self.sampling_rate,\
                             chopping_rate = self.chopping_rate,\
                             n = n)
            elif m in self.T:
                y = np.mean(self.T[m][:,:t_max], axis=1)
            elif len(m.split("-")) > 1:
                T_on, T_off = m.split("-")
```

```
138                 y = chopDUAL(self.T[T_on][:,:t_max],\
139                              self.T[T_off][:,:t_max],\
140                              sampling_rate = self.sampling_rate,\
141                              chopping_rate = self.chopping_rate)
142             else:
143                 sys.exit(f"Format of the mode {m} is incorrect, try for example 'L', 'ON-OFF
                        ', or 'ABBA'.")
144             if not (eta_atm is None):
145                 y /= eta_atm
146                 y *= 1e3
147                 fig.update_yaxes(title='$\mathrm{T_A^*\:[mK]}$',range=[-1.25,8.25])
148             fig.add_trace(go.Scatter(x=self.f, y=y, name=m,
149                                      line=dict(width=1, color=colors[i%len(colors)],\
150                                                shape='hvh')))
151             if showFWHM:
152                 fig.update_traces(mode = 'markers',
153                                   error_x = dict(width=5, thickness=2, type='data', array =
                                        self.FWHM))
154 
155         if not (spectrum is None):
156             with open(spectrum,'r') as file:
157                 lines = file.readlines()
158                 f_gal, FD_gal = zip(*[line.split() for line in lines])
159 
160             f_gal = np.array(f_gal,dtype='float32')
161             FD_gal = np.array(FD_gal,dtype='float32')
162             T_gal = T_from_FD(f_gal, FD_gal, self.beam_radius)
163 
164             fig.add_trace(go.Scatter(x=f_gal, y=T_gal*1e3, name='Galaxy',
165                                      line=dict(width=.7, color='black', dash='dash')))
166 
167         fig.update_xaxes(title='$f\:\mathrm{[GHz]}$',type="linear",range=[self.f[0],self.f
                [-1]])
168         if showFWHM:
169                 savename += "_FWHM"
170         if not (zoom is None):
171             fig.update_yaxes(autorange=False)
172             if zoom["x"] != [None,None]:
173                 savename += "_zoom"+str(round(sum(zoom["x"])/2))
174             fig.update_xaxes(range=zoom["x"])
175             fig.update_yaxes(range=zoom["y"])
176         fig.update_layout(legend=dict(yanchor="top",y=0.97,xanchor="left",x=0.03,font_size
                =14),
177                           template='plotly', margin=dict(l=60,r=60), width = w, height = h,
178                           xaxis_title_font_size=18, yaxis_title_font_size=18)
179 
180         if not (plotfolder is None):
181             if html:
182                 fig.update_xaxes(title='f [GHz]',type="linear")
183                 if not (eta_atm is None):
184                     fig.update_yaxes(title='T* [mK]')
185                 else:
186                     fig.update_yaxes(title='Tsky [K]')
187                 fig.write_html(plotfolder + savename + ".html", include_plotlyjs="cdn")
188             else:
189                 fig.write_image(plotfolder + savename + ".pdf", width=w, height=h)
190         fig.show()
191 
192     def noisePlot(self, mode = ["ON"], channel = [50], t_max = None,\
193                   plotfolder = None, html = True):
194         """
195         mode: list of strings
196             Each string should contain one of the sky measurements for Tsky
197             ("L", "ON", "R", "U", "D", "OFF"). Default is ["ON"].
198         channel: list of int
199             Each integer in the list represents a channel number, starting
200             from channel 1. Default is [50].
201         t_max: int OR None, unit: s
202             Maximum timestamp to plot. If given an integer, it slices up
203             until that timestamp. Otherwise, if given None, the maximum
204             possible timestamp will be shown. Default is None.
```

```
205         plotfolder: string
206             When not equal to None, an html file of the plot will be saved in this folder.
207             Default is None.
208         html: bool
209             When True, the figure is saved as an interactive .html file.
210             When False, the figure is saved as a static .svg file.
211             Default is True.
212 
213         outputs: Plotly figure
214         """
215         if len(mode) != len(channel):
216             sys.exit(f"Input mode (len={len(mode)}) and channel (len={len(channel)}) are of
                    different sizes.")
217 
218         chip, sec = self.simresult.split("_")[-4:3]
219         savename = f"noise_{chip}_{sec}"
220 
221         if t_max is None:
222             t_max = len(self.T["ON"][0])
223         else:
224             t_max = int(t_max)
225             savename += f"_{t_max}s"
226             t_max *= self.sampling_rate
227 
228         fig = go.Figure()
229 
230         for c,(ch,m) in enumerate(zip(channel,mode)):
231             fig.add_trace(go.Scatter(x=self.t[:t_max],y=self.T[m][ch-1][:t_max],\
232                                      line_color=colors[c%len(colors)-1],\
233                                      name=f"Channel {ch} (f\u2080 = {self.f[ch-1]:.3f} GHz)
                                             at {m}"))
234 
235         fig.update_xaxes(title='$\mathrm{t\:[s]}$',title_font_size=18)
236         fig.update_yaxes(title='$\mathrm{T_{sky}\:[K]}$',title_font_size=18)
237         fig.update_layout(template='plotly', margin=dict(l=60,r=60),
238                           plot_bgcolor='rgba(26,35,61,0.15)')
239         if (c < 3):
240             fig.update_layout(legend=dict(orientation="h",entrywidth=1/(c+1),
241                                           entrywidthmode="fraction",
242                                           yanchor="bottom",y=1.02,
243                                           xanchor="center",x=0.5))
244         elif (c > 9):
245             fig.update_traces(showlegend=False)
246 
247         if not (plotfolder is None):
248             if html:
249                 fig.update_xaxes(title='t [s]')
250                 fig.update_yaxes(title='Tsky [K]')
251                 fig.write_html(plotfolder + savename + ".html", include_plotlyjs="cdn",
252                                default_width = 1000, default_height = 500)
253             else:
254                 fig.write_image(plotfolder + savename + ".pdf", width=1000, height=500)
255         fig.show()
256 
257     def sigmaPlot(self, mode = ["ON-L"], f_min = 200, f_max = 300,\
258                   NEFD = None, eta_atm = None, plotfolder = None, html = True):
259         """
260         mode: list
261             Each string in the list should contain two of the sky measurements
262             for Tsky ("L", "ON", "R", "U", "D", "OFF") separated by a minus
263             sign ("-") to indicate the inputs for chopDUAL().
264             Default is ["ON-L"].
265         f_min: int/float, unit: GHz
266             Minimum frequency (in GHz) considered for calculation of standard
267             deviation. Default is 200.
268         f_max: int/float, unit: GHz
269             Maximum frequency (in GHz) considered for calculation of standard
270             deviation. Default is 300.
271         NEFD: array OR None, unit: W m^-2 Hz^-1 s^0.5
272             When given an array containing the NEFD for each frequency of the
273             filterbank chip, the NET(tau) is plotted alongside the standard
```

```
274             deviation curves.
275             When given None, no extra curve is plotted. Default is None.
276         eta_atm: array OR None, unit: no unit
277             When given an array containing the eta_atm for each frequency of
278             the filterbank chip, the Tsky and thus the standard deviation is
279             corrected by this data accordingly.
280             When given None, no correction is done and the raw standard deviation
281             is plotted. Default is None.
282         plotfolder: string
283             When not equal to None, an html file of the plot will be saved in this folder.
284             Default is None.
285         html: bool
286             When True, the figure is saved as an interactive .html file.
287             When False, the figure is saved as a static .svg file.
288             Default is True.
289 
290         outputs: Plotly figure
291         """
292         fig = go.Figure()
293 
294         for i,m in enumerate(mode):
295             if not len(m.split("-")) > 1:
296                 sys.exit(f"Format of the mode {m} is incorrect, try for example 'ON-OFF'.")
297             T1, T2 = m.split("-")
298             T_on, T_off = self.T[T1], self.T[T2]
299 
300             if (T1 or T2) == "ON":
301                 offoff = False
302             else:
303                 offoff = True
304 
305             t_max = self.t[-1-(T_on.shape[1])%(int(self.sampling_rate/self.chopping_rate))]
306 
307             tArray = np.logspace(0,np.log10(t_max),20)
308 #             tArray = np.linspace(1,t_max,50)
309             sArray = [sigma(self.f, T_on, T_off, t_obs,\
310                             f_min = f_min, f_max = f_max,\
311                             eta_atm = eta_atm, offoff = offoff,\
312                             sampling_rate = self.sampling_rate,\
313                             chopping_rate = self.chopping_rate)[0]
314                       for t_obs in tArray]
315 
316             a, b = np.polyfit(np.log10(tArray), np.log10(sArray), 1)
317             tau = tArray/2 # on-source fraction = 0.5
318 
319             fig.add_trace(go.Scatter(x = tau, y = np.array(sArray)*1e3,
320                                      name = m, mode = 'markers', marker_color = colors[i%len(
                                             colors)],
321                                      hovertemplate="\u03C4 = %{x:.2g} s<br>"+
322                                      "\u03C3 = %{y:.3g} mK<br><extra></extra>"))
323             fig.add_trace(go.Scatter(x = tau, y = 10**(a*np.log10(tArray) + b + 3),
324                                      name = f'Fit {m} (\u221D \u03C4^{a:.2f})',
325                                      line = dict(width=.7,dash='dot',color='black')))
326 
327         if not (NEFD is None):
328             ch_min = 0 # (np.abs(self.f-f_min)).argmin()
329             ch_max = (np.abs(self.f-f_max)).argmin()
330 
331             NET = [T_from_FD(self.f[ch_min:ch_max+1],\
332                              NEFD[ch_min:ch_max+1]/np.sqrt(t),\
333                              self.beam_radius)
334                    for t in tau] # NEFD/sqrt(tau)
335 
336             if not (eta_atm is None):
337                 NET = [net/eta_atm[ch_min:ch_max+1] for net in NET]
338 
339             NET_tau = np.mean(NET,axis=1)
340 
341             fig.add_trace(go.Scatter(x = tau, y = NET_tau*1e3, name = "NET(\u03C4) from dsm",
342                                      mode = 'markers+lines', marker_color = colors[(i+1)%len(
                                             colors)],
```

```
                                    line = dict(width=.7,dash='dot',color='black'),
                                    hovertemplate="\u03C4 = %{x:.2g} s<br>"+
                                    "\u03C3 = %{y:.3g} mK<br><extra></extra>"))

        fig.update_xaxes(title="$\mathrm{\\tau\:[s]}$",type="log")
        if not (eta_atm is None):
            fig.update_yaxes(title="$\mathrm{\sigma_{T_A^*}\:[mK]}$",type="log")
        else:
            fig.update_yaxes(title="$\mathrm{\sigma_{T_{sky}}\:[mK]}$",type="log")
        fig.update_layout(legend=dict(yanchor="top",y=0.97,xanchor="right",x=0.97,font_size
            =14),
                          width=900, height=500, template='plotly', margin=dict(l=60,r=60),
                          xaxis_title_font_size=18, yaxis_title_font_size=18)

        if not (plotfolder is None):
            chip, sec = self.simresult.split("_")[-4:3]
            if html:
                fig.update_xaxes(title="\u03C4 [s]")
                fig.update_yaxes(title="\u03C3 [mK]")
                fig.write_html(plotfolder + f"sigma_{chip}_{sec}.html", include_plotlyjs="cdn
                    ")
            else:
                fig.write_image(plotfolder + f"sigma_{chip}_{sec}.pdf",width=900,height=500)
        fig.show()

    def snrPlot(self, lines, OFFstr = "L", f_min = 200, f_max = 310, t_min = 60,\
                eta_atm = None, MDLF = None, t_dsm = 1, redshift = 2.04, LineWidth_kms =
                    600.0,\
                n_cols = 2, plotfolder = None, html = True):
        """
        lines: string (path to .data file used for SpectrumGenerator.py)
            String of the path to the file containing the columns:
            Line Name, Line res-frame frequency [GHz], Line flux for Galaxy X [W m^-2]
            This function uses both the line frequencies (Lfreq) and the line flux (LFlux)
            and applies the redshift to scale the frequencies appropriately.
        OFFstr: string
            One of the 'off' sky measurements for Tsky, denoted by "L", "R", "U", "D", "OFF".
            Default is "L".
        f_min: int/float, unit: GHz
            Minimum frequency considered for finding spectral lines from the input lines.
            Default is 200.
        f_max: int/float, unit: GHz
            Maximum frequency considered for finding spectral lines from the input lines.
            Default is 310.
        t_min: int/float, unit: s
            Minimum observation time (Tobs) considered for calculating the signal-to-noise
            ratio. The maximum time is always set to the maximum possible using chopDUAL().
            Remember that the integration time tau will be half of the observation time.
            Default is 60.
        eta_atm: array OR None, unit: no unit
            When given an array containing the eta_atm for each frequency of
            the filterbank chip, the Tsky and thus the standard deviation is
            corrected by this data accordingly.
            When given None, no correction is done and the raw standard deviation
            is plotted. Default is None.
        MDLF: array OR None, unit: W m^-2
            Array containing the MDLF for Tobs = t_dsm and snr = 1, such that it can
            be scaled accordingly.
            When given None, no SNR for method 'dsm' can be plotted.
            Default is None.
        t_dsm: int, unit: s
            Input observing time for deshima_sensitivity to define MDLF, equal to t_obs.
            Default is 1.
        redshift: float, unit: no unit
            The redshift of the galaxy used as the input for lines.
            Default is 2.04 (redshift of J1329 galaxy).
        LineWidth_kms: float, unit: km s^-1
            The line width used in SpectrumGenerator.py for the galaxy.
            Default is 600.0.
        n_cols: int
            Number of columns for subplot. Default is 2.
```

```
411         plotfolder: string
412             When not equal to None, an html file of the plot will be saved in this folder.
413             Default is None.
414         html: bool
415             When True, the figure is saved as an interactive .html file.
416             When False, the figure is saved as a static .svg file.
417             Default is True.
418 
419         outputs: Plotly figure
420         """
421         # Collect lines in given region
422         Lf, LF = np.genfromtxt(lines, unpack=True, skip_header=1, usecols=(1,2))
423         Lf /= (1. + redshift)
424         Lfreq, LFlux = zip(*[line for line in zip(Lf,LF) if f_min<line[0]<f_max])
425         Lfreq, LFlux = zip(*sorted([i for i in zip(Lfreq,LFlux)]))
426 
427         if (not OFFstr in self.T) or OFFstr == "ON":
428             sys.exit(f"Format of OFFstr {OFFstr} is incorrect, try one of 'L', 'R', 'U', 'D',
                    'OFF'.")
429         T_on = self.T["ON"]
430         T_off = self.T[OFFstr]
431 
432         ch_min = (np.abs(self.f-f_min)).argmin()
433         ch_max = (np.abs(self.f-f_max)).argmin()
434 
435         f_adj = self.f[ch_min:ch_max+1]
436         T_on_adj = T_on[ch_min:ch_max+1]
437         T_off_adj = T_off[ch_min:ch_max+1]
438 
439         # Adjust for sigma() (without lines); region 4*std from Lf is approx. 99.9% of total
                integral
440         for Lf in Lfreq:
441             LineWidth_GHz = LineWidth_kms/3.e5*Lf
442             std_Gaussian = LineWidth_GHz/2.35482
443             outside_Gaussian = np.logical_xor((f_adj < Lf-4*std_Gaussian),(f_adj > Lf+4*
                    std_Gaussian))
444             f_adj = f_adj[outside_Gaussian]
445             T_on_adj = T_on_adj[outside_Gaussian]
446             T_off_adj = T_off_adj[outside_Gaussian]
447 
448         t_max = self.t[-1-(T_on.shape[1])%(int(self.sampling_rate/self.chopping_rate))]
449         tArray = np.logspace(np.log10(t_min),np.log10(t_max),20)
450 #         tArray = np.linspace(t_min,t_max,50)
451 
452         # Getting the standard deviations and baseline parameters from sigma()
453         sArray, baseArray, t_indexArray = zip(*[sigma(f_adj, T_on_adj, T_off_adj, t_obs,\
454                                                       f_min = f_min, f_max = f_max,\
455                                                       eta_atm = eta_atm, offoff = False,\
456                                                       sampling_rate = self.sampling_rate,\
457                                                       chopping_rate = self.chopping_rate)
458                                                 for t_obs in tArray])
459 
460         # Setting plot parameters and calculating the signal-to-noise ratios
461         tau = tArray/2 # on-source fraction = 0.5
462         f_adj = self.f[ch_min:ch_max+1]
463         TArray = [chopDUAL(T_on[ch_min:ch_max+1,t_index],\
464                            T_off[ch_min:ch_max+1,t_index],\
465                            sampling_rate = self.sampling_rate,\
466                            chopping_rate = self.chopping_rate)
467                   for t_index in t_indexArray]
468 
469         if not (eta_atm is None):
470             TArray /= eta_atm[ch_min:ch_max+1]
471 
472         if not (MDLF is None):
473             MDLF = MDLF[ch_min:ch_max+1]
474 
475         n_rows = len(Lfreq)//n_cols + (len(Lfreq)%n_cols!=0)
476         fig = make_subplots(rows=n_rows, cols=n_cols)
477 
478         for i,(Lf,LF) in enumerate(zip(Lfreq,LFlux)):
```

```
479             ch1_Lf = (np.abs(f_adj-Lf)).argmin()
480             # To also check the second closest:
481 #             ch2_Lf = ch1_Lf + int(np.sign(Lf-f_adj[ch1_Lf]))
482 #             if TArray[-1][ch1_Lf] > TArray[-1][ch2_Lf]:
483 #                 ch_Lf = ch1_Lf
484 #             else:
485 #                 ch_Lf = ch2_Lf
486             ch_Lf = ch1_Lf # Comment this out if check second closest
487 
488             SNR_tiempo = [(T[ch_Lf]-(a*f_adj[ch_Lf]+b))/sigma for T,sigma,[a,b] in zip(TArray
                    ,sArray,baseArray)]
489 
490             fig.add_trace(go.Scatter(x = tau, y = SNR_tiempo,
491                                      name = f"Line f = {Lf:.3f} GHz; TiEMPO",
492                                      mode = 'markers', marker_color = colors[i%len(colors)],
493                                      hovertemplate="\u03C4 = %{x:.2g} s<br>"+
494                                      "SNR = %{y:.3g}<br><extra></extra>"),
495                           row = i//n_cols + 1, col = i%n_cols + 1)
496 
497             if not (MDLF is None):
498                 SNR_dsm = np.sqrt(tArray/t_dsm)*LF/(MDLF[ch_Lf])
499                 fig.add_trace(go.Scatter(x = tau, y = SNR_dsm,
500                                          name = "Prediction DSM (\u221D \u03C4^0.5)",
501                                          mode = 'markers+lines', marker_color = colors[-1],
502                                          line = dict(width=.7,dash='dot',color='black'),
503                                          hovertemplate="\u03C4 = %{x:.2g} s<br>"+
504                                          "SNR = %{y:.3g}<br><extra></extra>"),
505                               row = i//n_cols + 1, col = i%n_cols + 1)
506 
507         fig.update_xaxes(title="$\mathrm{\\tau\:[s]}$",title_font_size=18,type="log")
508         fig.update_yaxes(title="$\mathrm{SNR\:[-]}$",title_font_size=18)
509         fig.update_layout(legend_font_size=12, margin=dict(l=60,r=60), template='plotly',
510                           width=900, height=300*n_rows)
511         if (len(Lfreq)%n_cols != 0) and (n_cols < 4):
512             fig.update_layout(legend=dict(yanchor="bottom",y=0.045-0.025*n_cols,
513                                           xanchor="center",x=1.03-1/(2*n_cols)))
514         if not (plotfolder is None):
515             chip, sec = self.simresult.split("_")[-4:3]
516             if html:
517                 fig.update_xaxes(title="\u03C4 [s]")
518                 fig.update_yaxes(title="SNR [-]")
519                 fig.write_html(plotfolder + f"snr_{chip}_{sec}.html", include_plotlyjs="cdn")
520             else:
521                 fig.write_image(plotfolder + f"snr_{chip}_{sec}.pdf",width=900,height=300*
                        n_rows)
522         fig.show()
523 
524 def chopDUAL(T_on, T_off, sampling_rate = 160, chopping_rate = 10):
525     """
526     sampling_rate: int, unit: Hz
527         Rate at which measurements are made; or, equivalently, number of
528         measurements per second. Default is 160.
529     chopping_rate: int, unit: Hz
530         Rate at which position switching on the sky is performed by the
531         rotating mirror in the cabin optics. Default is 10.
532 
533     returns: DUAL chopped signal, unit K
534         Finds the DUAL chopped signal (on/off-chopping).
535     """
536     n_samples = T_on.shape[1]
537     unit_length = sampling_rate/chopping_rate
538 
539     if unit_length%2 != 0:
540         sys.exit("sampling_rate should be an even multiple of chopping_rate")
541     if unit_length > n_samples:
542         sys.exit("Dual-chopping unit too large for given data set")
543     unit_length = int(unit_length)
544 
545     onoff_unit = [1]*int(unit_length/2) + [0]*int(unit_length/2)
546     onoff_sequence = onoff_unit * (n_samples//unit_length)
547     ON_index = np.append(onoff_sequence, [0] * (n_samples%unit_length))
```

```
548     OFF_index = np.append(onoff_sequence, [1] * (n_samples%unit_length))
549 
550     return np.mean(T_on[:,ON_index==True],axis=1)-np.mean(T_off[:,OFF_index==False],axis=1)
551 
552 def chopABBA(T_l, T_c, T_r, sampling_rate = 160, chopping_rate = 10, n = 3): # see below
553     """
554     sampling_rate: int, unit: Hz
555         Rate at which measurements are made; or, equivalently, number of
556         measurements per second. Default is 160.
557     chopping_rate: int, unit: Hz
558         Rate at which position switching on the sky is performed by the
559         rotating mirror in the cabin optics. Default is 10.
560     n: int
561         n-factor, the number of chopping pairs to be observed per nodding
562         position. Default is 3.
563 
564     returns: ABBA chopped signal, unit K
565         Making use of simulate_ABBA_chop() from ABBA_chop.py, it finds
566         the chopped signal.
567     """
568     chopped = simulate_ABBA_chop(T_l, T_c, T_r,\
569                                  sampling_rate = sampling_rate,\
570                                  chop_interval = 1/chopping_rate,\
571                                  n = n)
572     return np.mean(chopped[1],axis=1)-(np.mean(chopped[0],axis=1)+np.mean(chopped[2],axis=1))
        /2
573 
574 def sigma(f, T_on, T_off, t_obs, f_min = 200, f_max = 300, eta_atm = None,\
575           offoff = False, sampling_rate = 160, chopping_rate = 10):
576     """
577     f: array, unit: GHz
578         Input frequency array.
579     T_on: array, unit: K
580         Input Tsky representing the "ON" array in chopDUAL().
581     T_off: array, unit: K
582         Input Tsky representing the "OFF" array in chopDUAL().
583     t_obs: float, unit: s
584         Input observing time Tobs, equivalent to twice the integration
585         time tau (tau = t_obs/2)
586     f_min: int/float, unit: GHz
587         Minimum frequency (in GHz) considered for calculation of standard
588         deviation. Default is 200.
589     f_max: int/float, unit: GHz
590         Maximum frequency (in GHz) considered for calculation of standard
591         deviation. Default is 300.
592     eta_atm: array OR None, unit: no unit
593         When given an array containing the eta_atm for each frequency of
594         the filterbank chip, the Tsky and thus the standard deviation is
595         corrected by this data accordingly.
596         When given None, no correction is done and the raw standard deviation
597         is plotted. Default is None.
598     offoff: bool
599         When True, the standard deviation is calculated using the output of
600         chopDUAL() directly.
601         When False, the standard deviation is calculated using the difference
602         of the output of chopDUAL() and a first order polynomial fitted
603         through this output (to approximate substracting the base curve,
604         which is assumed to be close to linear).
605         Default is False.
606     sampling_rate: int, unit: Hz
607         Rate at which measurements are made; or, equivalently, number of
608         measurements per second. Default is 160.
609     chopping_rate: int, unit: Hz
610         Rate at which position switching on the sky is performed by the
611         rotating mirror in the cabin optics. Default is 10.
612 
613     returns: sigma_T, T-baseline, and t_index, units: K, K, and no unit
614         sigma_T is the standard deviation of the difference between T_on and
615         T_off minus the curve fitted baseline (if offoff is False) and is
616         corrected by eta_atm if given.
617         [a,b] are yielded from the baseline curve-fitting of the (f_adj,T)-curve
```

```
618         of the difference between T_on and T_off. When offoff is True, it is
619         just equal to [0,0], since it is then expected to be zero mean.
620         t_index is the boolean array to slice the T array up until t_obs.
621     """
622     t_sample = 1/sampling_rate # s
623     n_samples = T_on.shape[1] # number of time steps
624     t_index = t_sample * np.arange(n_samples) < t_obs
625 
626     if offoff:
627         ch_min = 0
628     else:
629         ch_min = (np.abs(f-f_min)).argmin()
630     ch_max = (np.abs(f-f_max)).argmin()
631 
632     f_adj = f[ch_min : ch_max+1]
633     T = chopDUAL(T_on[ch_min:ch_max+1,t_index], T_off[ch_min:ch_max+1,t_index], sampling_rate
        , chopping_rate)
634 
635     if not (eta_atm is None):
636         T /= eta_atm[ch_min : ch_max+1]
637 
638     if offoff:
639         return np.std(T), [0,0], t_index
640     else:
641         a, b = np.polyfit(f_adj, T, 1)
642         baseline = a*f_adj + b
643         return np.std(T - baseline), [a,b], t_index
644 
645 def T_from_FD(f, FD, beam_radius = 5.):
646     """
647     f: array, unit: GHz
648         Contains the frequencies corresponding to the galaxy spectrum.
649     FD: array, unit: Jy
650         Contains the spectrum (flux density in Jy) of the galaxy.
651     beam_radius: float, unit: m
652         Radius of the Gaussian beam. Default is 5.
653 
654     returns: T_gal, unit K
655         Temperature derived using Planck from given flux density of the galaxy.
656     """
657     Ae = dsm.calc_eff_aper(f*1e9, beam_radius) #1e9 added to convert the f to Hz
658     psd_gal = FD * Ae * 1e-26 * 0.5
659 
660     # Choice of math.log() over numpy.log() due to behaviour at small values for psd
661     T_gal = np.array([h*f / (k*math.log(h*f/psd + 1.)) for f,psd in zip(f,psd_gal)])
662     return T_gal
```

A.2.1. Loading Simulation Data

Here, the loading of simulation data from `TiEMPO` is shown. Additionally, the used data generated using `deshima-sensitivity`, stored as .npy-files, is shown being loaded here as well.

```
# Used folders:
savefolder = "simResults/"
plotfolder = "simPlots/"
spectrumfolder = "Spectrum_MatthijsRoos/"

# Used galaxy line frequency [GHz] and line flux [W/m^2]
lines, redshift = spectrumfolder + "LinesTable.data", 2.04 # J1329 galaxy


### Example of loading the simulation data of the Preset chip
# Tobs = 28800s, Preset, [pwv = 1, EL = 60]

filename = "TiEMPO_preset_28800s_08-06-2023_2057"

spectrum = spectrumfolder + "spectrum_J1329_Jy_SCUBA_adj.data"
eta_atm = np.load(savefolder + "preset_eta_atm_pwv1.0_EL60.npy")
NEFDcont = np.load(savefolder + "preset_NEFDcont_pwv1.0_EL60.npy") *1e26
MDLF = np.load(savefolder + "preset_MDLF_pwv1.0_EL60_Tobs1s.npy")

simresult = savefolder + filename

run = plot_TiEMPO(simresult, Q = 500)


### Example of loading the simulation data of the LT263 chip
# Tobs = 28800s, Chip LT263, [pwv = 1, EL = 60]

filename = "TiEMPO_LT263_28800s_16-06-2023_1402"

chippath = "https://raw.githubusercontent.com/MatthijsRoos/thesis/main/DESHIMA2.0
    _FlightChip_Filters.run991.wb-31dB.csv"
Q = pd.read_csv(chippath)['Ql'].to_numpy()

spectrum = spectrumfolder + "spectrum_J1329_Jy_SCUBA_chip.data"
eta_atm = np.load(savefolder + "LT263_eta_atm_pwv1.0_EL60.npy")
NEFDcont = np.load(savefolder + "LT263_NEFDcont_pwv1.0_EL60.npy") *1e26
MDLF = np.load(savefolder + "LT263_MDLF_pwv1.0_EL60_Tobs1s.npy")

simresult = savefolder + filename

run = plot_TiEMPO(simresult, n_batches = 8, Q = Q)
```

A.2.2. Use of plot_TiEMPO()

Examples on how to create the interactive plots and, if needed, save them as either .pdf-files or .html-files.

```
run = plot_TiEMPO(simresult, n_batches, Q)

# Create a plot containing the time-averaged
# sky temperature Tsky for all six sky positions
run.plot(mode = ["L", "ON", "R", "U", "D", "OFF"],\
         plotfolder = plotfolder)#, html = False)

# Create a plot containing Tsky data of a list of
# channels for sky positions up until a maximum timestamp
run.noisePlot(mode = ["L", "ON", "R"], channel = [87, 88, 89],\
              t_max = 3, plotfolder = plotfolder)#, html = False)

# Create a plot containing the atmosphere corrected
# antenna temperature of ON-OFF and OFF-OFF dual chopped
# signals, as well as an ABBA chop/nodded signal
run.plot(mode = ["ON-L","OFF-L","ABBA"],\
         spectrum = spectrum, eta_atm = eta_atm,\
         plotfolder = plotfolder)#, html = False)

# Create a plot of the standard deviation as a function
# of time for the ON-OFF and an OFF-OFF dual chopped
# atmosphere corrected antenna temperature
run.sigmaPlot(mode = ["ON-OFF","OFF-L"], f_min = 270,\
              eta_atm = eta_atm, NEFD = NEFDcont,\
              plotfolder = plotfolder)#, html = False)

# Create plots of the signal-to-noise ratio as a function
# of time in the frequency range of 200-310 GHz along with
# theoretical curves for each line
run.snrPlot(lines, OFFstr = "L", f_min = 200, f_max = 310,\
            eta_atm = eta_atm, MDLF = MDLF, redshift = redshift,\
            plotfolder = plotfolder)#, html = False)

# Create plots of the atmosphere corrected antenna temperature along
# with the galaxy spectrum, zoomed in on the lines of the galaxy
for z in [[dict(x=[226,229],y=[0.5,5.5]),\
           dict(x=[245.9,248.9],y=[1,5]),\
           dict(x=[264,267.5],y=[1.5,6]),\
           dict(x=[301.5,305],y=[2,7])]
         ]:
    run.plot(mode = ["ON-L"], spectrum = spectrum, eta_atm = eta_atm, \
             showFWHM = True, zoom = z, plotfolder = plotfolder)#, html = False)

# Create plots of dual chopped Tsky using ON-L for
# tau = 0.5s, 30s, 1m, 5m, 10m, 30m, 1h, 2h, and 4h
for t in [1,60,120,600,1200,3600,7200,14400,28800]:
    run.plot(mode = ["ON-L"], t_max = t,\
             plotfolder = plotfolder)#, html = False)
```

A.3. Revisioning new_filterbank()

Here, it is shown how the analysis and adaptation of the new_filterbank() function inside `TiEMPO` was performed, giving details on how a custom chip can be implemented in the model.

```
"""
In the process of adapting `tiempo_deshima_custom_input` to be able to handle custom chip data inputs alongside the current use of a 'perfect chip', the hardest part was to create a new filterbank corresponding to the new chip data. In general, adaptations included:
 - replacing the perfectly spaced center frequencies (`F` or `filters`) of the filters by the measured center frequencies;
 - replacing the constant Q-factor (`Q`, `R`, or `spec_res`) for all filter channels by the measured Q-factors of each filter channel and rewriting corresponding code such that it could handle the Q-factor as a vector instead of a constant;
 - replacing the constant `eta_filter_peak` for all filter channels by the measured |S31|$^2$ of each filter channel;
 - recreating the input galaxy spectrum such that its frequency corresponds to that used for the bins in the simulation (`frequency_gal` and `spectrum_gal`).

 The following code shows an example of an adaptation made to handle the chip data. Here, the initialization in the `filterbank` class in `filterbank.py` is shown.
"""

### Adjusted by Matthijs Roos:
# def __init__(self, F_min, R, num_filters = 1, f_spacing = 380, num_bins = 1500, D1 = 0):
def __init__(self,input):
    self.input = input # Input is now the dictionary that is returned from running get_dictionary()
    self.F_min = input["F_min"]
    self.F0 = input["F_min"]
    self.R = input["spec_res"]
    self.num_filters = input["num_filters"]
    self.num_bins = input["num_bins"]
    self.DataChip = input["run_new_filterbank"]
    self.chipdata = input["chipdata"]
    if self.DataChip:
        self.datapath = 'DataChip/' # Folder to store new filterbank data (from chip data)
        F = self.chipdata.F
        self.F_max = F[-1]
    else:
        self.datapath = 'Data/' # Folder to store default filterbank data (perfect chip)
        self.f_spacing = input["f_spacing"]
        self.F_max = self.F_min * (1 + 1/self.f_spacing)**(self.num_filters - 1)
        F = np.logspace(np.log10(self.F_min), np.log10(self.F_max), self.num_filters)
    self.filters = F
    self.FWHM = self.filters/self.R
    self.D1 = input["D1"]
    self.path_model = Path(__file__).parent.parent.parent

"""
As I seem to have adapted all the code to be able to handle the chip data (as seen above), the last thing to do is to run `new_filterbank()` to create a new filterbank belonging to the LT263 chip. However, the function `new_filterbank()` in `interface.py` was causing a problem for the implementation of the chip in the code, as there was an error caused within the function `calcT_psd_P()` in `use_desim.py` which is called by `getPoints_TP_curve()` in `filterbank.py`. To my surprise, the same Exception is caused when run using the unadapted version of `tiempo_deshima_custom_input`, as seen at the end of this document. Here is a closer look at the error:

# File "...\tiempo_deshima_custom_input\interface.py", line 72, in new_filterbank
#     ft1.save_TP_data(EL_vector, pwv_vector)
#
# File "...\tiempo_deshima_custom_input\DESHIMA\MKID\filterbank.py", line 146, in save_TP_data
#     Pkid, Tb_sky = self.getPoints_TP_curve(EL_vector, pwv_vector[i])
#
# File "...\tiempo_deshima_custom_input\DESHIMA\MKID\filterbank.py", line 126, in getPoints_TP_curve
#     Tb_sky, psd_KID_desim, F_bins = use_desim_instance.calcT_psd_P(self.eta_atm_df, self.F_highres,
```

```
#     self.eta_atm_func_zenith, self.filters, EL_vector, self.num_filters, pwv, self.R, self.num_bins, self.D1)
#
# File "...\tiempo_deshima_custom_input\DESHIMA\use_desim.py", line 202, in <b>calcT_psd_P</b>
#     [psd_co[j, :], psd_jn_chip[j, :]] = self.obt_data(input, D1)
#
# File "...\tiempo_deshima_custom_input\DESHIMA\use_desim.py", line 155, in obt_data
#     D2goal = dsm.spectrometer_sensitivity(**sensitivity_input) # takes a lot of time
#
# File "...\tiempo_deshima_custom_input\DESHIMA\desim\minidesim.py", line 230, in spectrometer_sensitivity
#     eta_atm = eta_atm_func(F=F, pwv=pwv, EL=EL, eta_atm_df=eta_atm_df, F_highres=F_highres, eta_atm_func_zenith=eta_atm_func_zenith)
#
# File "...\tiempo_deshima_custom_input\DESHIMA\desim\minidesim.py", line 360, in eta_atm_func
#     eta_atm = np.zeros((len(F), len(pwv)))
#
# TypeError: object of type <b>'numpy.float64'</b> has no len()

The error refers to either `F` or `pwv` having no length, as this object is of type np.float64.<br>
Going back a couple of steps for `F`, it can be seen that this is due to `F` being set to a float in the following for-loop in the function `calcT_psd_P()`:
"""
for j in range(0, num_bins):
    input = {'F': F_bins[j], etc...}
    [psd_co[j, :], psd_jn_chip[j, :]] = self.obt_data(input, D1)
"""
However, `F` having no length is simply canceled in `eta_atm_func()` itself (a few lines back):
"""
if not hasattr(F, "__len__"):  # give F a length if it is an integer.
      F = np.asarray([F])
"""
It is thus `pwv` that causes the error! Again, we would like to know where `pwv` is defined. All the way back in `save_TP_data()` in `filterbank.py`, it is visible why it is a np.float64:
"""
for i in range(0, len(pwv_vector)):
      Pkid, Tb_sky = self.getPoints_TP_curve(EL_vector, pwv_vector[i])
"""
Strangely, `eta_atm_func()` in `minidesim.py`, where it all 'goes wrong', mentions `pwv` as a float within the list of parameters, though uses its length for a for-loop and creation of an array within said function. Note: not only in this function, but throughout all of `minidesim.py` it is expected to be a float.

So, what has to be done to make the code work? One may notice that `calcT_psd_P()` has some similarities with `transmit_through_DESHIMA()`, which is also found in `use_desim.py`. That function defines arrays as for `pwv`-inputs for `deshima_sensitivity()`, in which `eta_atm_func()` is used:
"""
pwv_values_no_gal = np.array([pwv_value[0], pwv_value[2], pwv_value[3], pwv_value[4], pwv_value[5]])
pwv_value_gal = np.array([pwv_value[0], pwv_value[1]])
"""
This is why a for-loop is needed in `eta_atm_func()` is necessary: to loop through the different `pwv` values. Now that we have figured that out, it is time to solve the problem at hand. `eta_atm_func()` will be used twice in `calcT_psd_P()`. An easy solution would be to give `pwv` a length via the same method `F` can be given a length in the `eta_atm_func()` function:
"""
if not hasattr(pwv, "__len__"):  # give pwv a length if it is a float.
  pwv = np.asarray([pwv])
"""
Let us look at whether this influences the results in those two instances. Firstly, `eta_atm` is being used in the line where the error was first found:
"""
[psd_co[j, :], psd_jn_chip[j, :]] = self.obt_data(input, D1)
"""
```

```
But, as is seen immediately, `eta_atm` is not gathered from the dictionary `result` that `deshima_sensitivity()` returns (only `psd_co` and `psd_jn_chip`). So, it is not a problem if `eta_atm` suddenly has a length.
"""
numerators[:, k] =  delta_F * np.sum(transmission \
 * dsm.eta_atm_func(F=F_bins, pwv=pwv, EL=EL_vector, eta_atm_df=eta_atm_df, F_highres=F_highres,
 eta_atm_func_zenith=eta_atm_func_zenith), axis = 0)
"""
Whether the output of `eta_atm_func()` is a scalar or vector of length 1, does not matter. The `np.sum(..., axis=0)` will make it of the appropriate length anyways. However, in the calculation of the columns of `numerators`, both `F` and `EL` have a length (`num_bins` and `length_EL_vector` respectively). `F` won't be of length 1 anymore, thus resulting in the output of `eta_atm_func()` having to be of the shape (`num_bins`,`length_EL_vector`), to be in accordance with the format of `numerators`, which shape is (`length_EL_vector`,`num_filters`). To clarify this statement, take a look at the lines in the code in question:
"""
delta_F = F_bins[1] - F_bins[0]
numerators = np.zeros([EL_vector.shape[0], num_filters])
for k in range(0, num_filters):
    transmission = use_desim.calcLorentzian(F_bins, F_filter[k], R[k])
    transmission = transmission.reshape([transmission.shape[0], 1])
    numerators[:, k] =  delta_F * np.sum(transmission \
    * dsm.eta_atm_func(F=F_bins, pwv=pwv, EL=EL_vector, eta_atm_df=eta_atm_df, F_highres=F_highres, eta_atm_func_zenith=eta_atm_func_zenith), axis = 0)
"""
- `delta_F` is thus a float and won't mess with the shapes used;
- `numerators` has shape (`length_EL_vector`,`num_filters`) and its k$^{\text{th}}$ column (`numerators[:, k]`) consequently has length `length_EL_vector`;
- `transmission` has shape (`num_bins`,1);
- `F_bins` has length `num_bins`;
- `EL_vector` has length `length_EL_vector`

The summation through `np.sum()` is along `axis = 0`, so this will yield a vector of length `length_EL_vector` when the output of `eta_atm_func()` is of shape (`num_bins`,`length_EL_vector`). The solution here would be to loop through the elements of `EL_vector` to build the matrix of the appropriate shape. We are ready to implement this in `eta_atm_func()` in `minidesim.py` as follows:
"""
def eta_atm_func(F, pwv, EL=60., eta_atm_df = pd.Series([]), F_highres = pd.Series([]), eta_atm_func_zenith = pd.Series([])):
    if np.average(F) > 10.**9:
        F = F / 10.**9
    if not hasattr(F, "__len__"):  # give F a length if it is an integer.
        F = np.asarray([F])
    if eta_atm_df.empty:
        eta_atm_df, F_highres = load_eta_atm()
    if type(eta_atm_func_zenith) != interp2d:
        eta_atm_func_zenith = eta_atm_interp(eta_atm_df)
    pwv = np.squeeze(pwv)
    ### Adjusted by Matthijs Roos:
    if not hasattr(pwv, "__len__"):  # give pwv a length if it is a float.
        pwv = np.asarray([pwv])
    if not hasattr(EL, "__len__"): # give EL a length if it is a float.
        EL = np.asarray([EL])
    eta_atm = np.zeros((len(F), len(pwv), len(EL)))
    for i in range(len(pwv)):
        for j in range(len(EL)):
            eta_atm[:,i,j] = np.abs(np.squeeze(eta_atm_func_zenith(pwv[i], F))) ** (1./np.sin(EL[j]*np.pi/180.))
    ###
    return np.squeeze(eta_atm)
"""
The function `calcT_psd_P()` is placed in a section of the code, denoted to not be _"(...) used in the model, only for making the interpolation curves and plotting"_. It is indeed only used when having to create a new filterbank, so that explains why it was left unnoticed before: tiempo simulations do not require this function to be working. Based on this, I think that it worked before, just to be ran once to create the requested files (containing the splines to convert P to T).

```

```
One final note, I came across the following 'DeprecationWarning':
interp2d is deprecated in SciPy 1.10 and will be removed in SciPy 1.12.0.

Might be interesting to know when adapting the code in the future.
"""
###
###
###
"""
To recreate the original error:
"""
import tiempo_deshima_custom_input
import numpy as np
import time
from tiempo_deshima_custom_input.interface import get_dictionary
from tiempo_deshima_custom_input.interface import new_filterbank

# We use the core of run_tiempo to recreate the proper input dictionary for new_filterbank()
def run_tiempo(input_dictionary, prefix_atm_data, sourcefolder, save_name_data, \
               savefolder = None, save_P=True, save_T=True, n_jobs = 30, n_batches = 8,\
               obs_time = 3600., grid = .2, x_length_strip = 65536., separation = 1.1326,\
               galaxy_on = True, luminosity = 13.7, redshift = 4.43, linewidth = 600, \
               EL = 60, EL_vec=None, max_num_strips = 32, pwv_0 = 1., F_min = 220e9, \
               num_bins = 6000, spec_res = 500, f_spacing = 500, \
               num_filters = 347, beam_radius = 5., useDESIM = 1, \
               inclAtmosphere = 1, windspeed = 10, D1 = 0, dictionary_name = '', \
               use_galspec = True, frequency_gal = np.NaN, spectrum_gal = np.NaN):

    dictionary = get_dictionary(input_dictionary, prefix_atm_data, sourcefolder,\
                                save_name_data, savefolder, save_P, save_T, n_jobs, n_batches
                                    , \
                                obs_time, grid, x_length_strip, separation, galaxy_on, \
                                luminosity, redshift, \
                                linewidth, EL, EL_vec, max_num_strips, pwv_0, F_min, \
                                num_bins, spec_res, f_spacing, num_filters, \
                                beam_radius, useDESIM, inclAtmosphere, \
                                windspeed, D1, dictionary_name,\
                                use_galspec, frequency_gal, spectrum_gal)
    return dictionary

# Just any input parameters:
threads = 6
Tobs = 1
basefolder = "./TiEMPO_MatthijsRoos_2023/"
arisfolder = basefolder + "Output_ARIS"
sourcefolder = "Spectrum_MatthijsRoos/"
savefolder = basefolder + "simResults"
source = sourcefolder + "spectrum_J1329_Jy_SCUBA_adj.data"
savename = f"TiEMPO_preset_{Tobs}s_"+time.strftime("%d-%m-%Y_%H%M")
with open(source,'r') as s:
    lines = s.readlines()
    f, FD = zip(*[line.split() for line in lines])
frequency_gal = np.array(f,dtype='float32')
spectrum_gal = np.array(FD,dtype='float32')

input_run_tiempo = dict(input_dictionary = 'manual',\
                        prefix_atm_data = 'aris200602.dat-',\
                        sourcefolder = arisfolder,\
                        save_name_data = savename,\
                        savefolder = savefolder,\
                        save_P = False,\
                        save_T = True,\
                        n_jobs = threads,\
                        n_batches = 1,\
                        obs_time = Tobs,\
                        grid = 1,\
                        separation = 1.,\
                        EL = 60,\
                        pwv_0 = 1,\
                        F_min = 220e9,\
                        num_bins = len(frequency_gal),\
```

```
                        spec_res = 500,\
                        f_spacing = 500,\
                        num_filters = 347,\
                        windspeed = 10,\
                        use_galspec = False,\
                        frequency_gal = frequency_gal,\
                        spectrum_gal = spectrum_gal,\
                       )

dictionary = run_tiempo(**input_run_tiempo)


# Initiating the error:
new_filterbank(dictionary)
```

B

More on the Physics and Plots

B.1. More on Redshift

Two reference points at a distance $d(t)$ from one another will separate according to:

$$v_r(t) = d(t)\frac{1}{R}\frac{\mathrm{d}R}{\mathrm{d}t} \equiv H(t) \tag{B.1}$$

Where $v_r(t)$ is the radial velocity, R is a scaling factor representing the separation, and $H(t)$ is the Hubble parameter [7]. The last parameter has a current value of $H(0) = H_0 \simeq 68$ km s^{-1} Mpc^{-1}, which results in two reference points distanced at 1 Mpc $\approx 3.08{\cdot}10^{22}$ m moving at roughly 68 km s^{-1} - meaning that the Universe would currently be expanding. The side-effect of this expansion is that the Doppler effect applies:

$$\frac{\Delta\lambda}{\lambda} = \frac{\Delta v_r}{c} = \frac{H(t)\Delta d}{c} = H(t)\Delta t = \frac{1}{R}\frac{\mathrm{d}R}{\mathrm{d}t}\Delta t \tag{B.2}$$

From equation B.2, it follows that wavelength and the scaling factor are proportional to each other as $\frac{1}{\lambda}\frac{\mathrm{d}\lambda}{\mathrm{d}t} = \frac{1}{R(t)}\frac{\mathrm{d}R(t)}{\mathrm{d}t}$ [8]. When $R(t_0) = 1$ with t_0 being the present cosmic time (a clock that is locally at rest), it is found that:

$$\frac{\nu_e}{\nu_0} = \frac{\lambda_0}{\lambda_e} = \frac{1}{R(t_e)} \equiv 1 + z \tag{B.3}$$

Here, the subscript e denotes a photon emitted by the source. z is the (spectroscopic) redshift and is therefore defined as the ratio between the change in wavelength and the original wavelength $z = \frac{\lambda_0 - \lambda_e}{\lambda_e}$.

B.2. More on Flux Density

Using the property that the equation B.4 for total flux F holds (expanded definition of equation 2.2), the equality in equation B.6 can be found.

$$F = \int F_\nu d\nu = \int F_\lambda d\lambda \tag{B.4}$$

From which follows that:

$$F_\nu|d\nu| = F_\lambda|d\lambda| \Rightarrow F_\nu = F_\lambda\left|\frac{d\lambda}{d\nu}\right| \Rightarrow F_\nu = \frac{\lambda^2}{c}F_\lambda \tag{B.5}$$

Which yields:

$$\nu F_\nu = \lambda F_\lambda \tag{B.6}$$

The result of equation B.6 can be used as another measure of flux as a function of frequency ν, with a unit independent of frequency. A visualization of the spectrum shown in figure 2.2b is given in figure B.1. The same applied to the model of the J1329+2243 galaxy used in this report yields figure B.2.

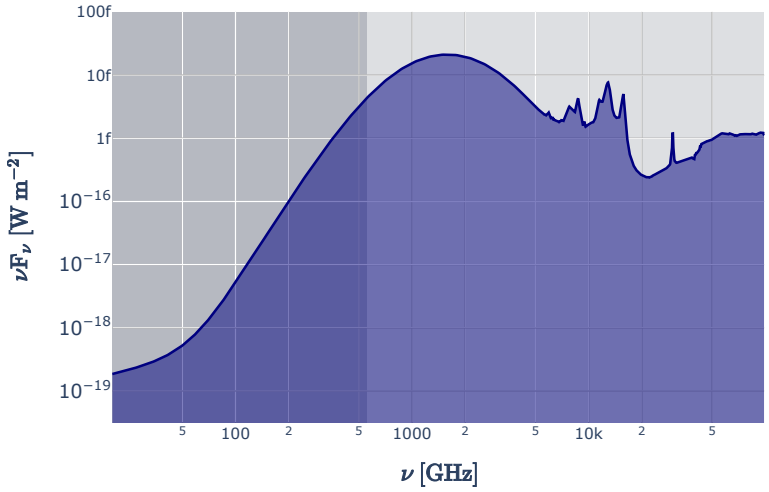


Figure B.1: Average flux density F_ν multiplied by the frequency ν to yield νF_ν. Average flux density data from [12] and rescaled to match the line fluxes of spectral lines in the J1329+2243 galaxy. The shaded area shows frequencies up to 550 GHz. [Interact]

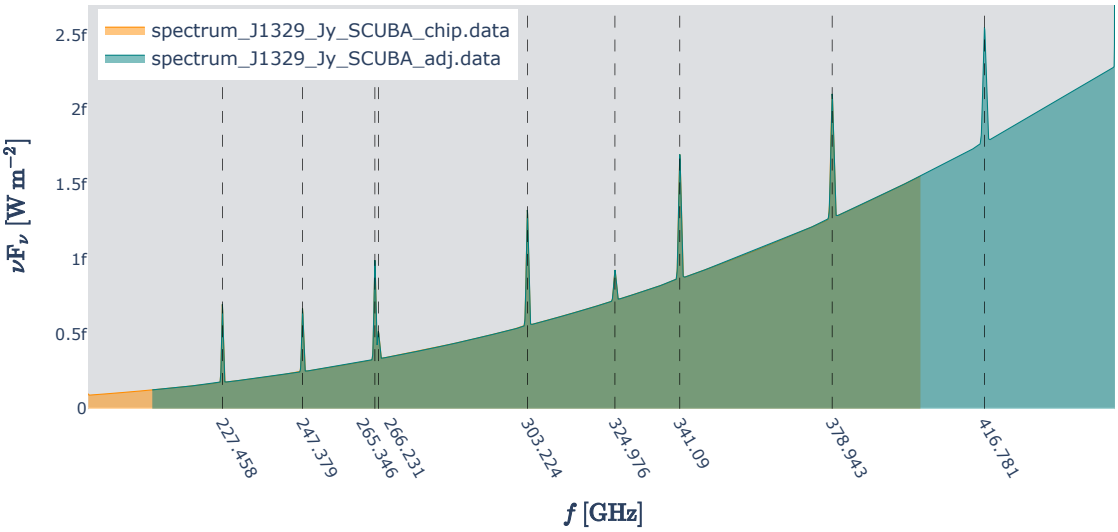


Figure B.2: Model flux density spectrum of the high-redshift J1329+2243 galaxy using rescaled continuum flux measurements from [12] and adding the redshift-corrected lines (z =2.04 [10]) with line width 600 km s^{-1}. Based on method created by Matus Rybak (2012), private communication. The redshift-corrected center frequencies of the lines are given on the f-axis. The flux density is multiplied by the frequency ν to yield νF_ν. The data from the file 'spectrum_J1329_Jy_SCUBA_chip.data' contains the spectrum for the frequency range of the LT263 chip plus an added margin of 10 GHz on each end. Likewise, 'spectrum_J1329_Jy_SCUBA_adj.data' contains the spectrum for the Preset chip. The spectra overlap in the green region. [Interact]

B.3. More on Noise Standard Deviation

In section 3.3.2, it was assumed that $\text{NEFD}_{\text{continuum}}$ would be a better approximation than $\text{NEFD}_{\text{line}}$. It is possible to use a better approximation, with a factor difference from this used NEFD.

Given that η_{atm}, f, and Ω are equal for both the `TiEMPO` simulated standard deviation and the `deshima-sensitivity` predicted standard deviation in the noise, the ratio between the two can be examined. The definition of the temperature as shown in equation 2.14 is used to work back to the ratio of the NEFDs as follows:

$$\frac{\text{NET}_{\text{DSM}}}{\sigma_{T_A^*}} = \frac{hf}{\eta_{atm} \cdot k \cdot \ln\left(\frac{hf}{\text{PSD}_{\text{DSM}}} + 1\right)} \frac{\eta_{atm} \cdot k \cdot \ln\left(\frac{hf}{\text{PSD}_{\text{TiEMPO}}} + 1\right)}{hf} = \frac{\ln\left(\frac{hf}{\text{PSD}_{\text{TiEMPO}}} + 1\right)}{\ln\left(\frac{hf}{\text{PSD}_{\text{DSM}}} + 1\right)} = \alpha \tag{B.7}$$

Here, α is thus the ratio of the predicted and found standard deviations. Next, the terms are rearranged and the property $a \cdot \ln b = \ln b^a$ is used:

$$\alpha \ln\left(\frac{hf}{\text{PSD}_{\text{DSM}}} + 1\right) = \ln\left(\frac{hf}{\text{PSD}_{\text{TiEMPO}}} + 1\right) \Rightarrow \left(\frac{hf}{\text{PSD}_{\text{DSM}}} + 1\right)^{\alpha} = \frac{hf}{\text{PSD}_{\text{TiEMPO}}} + 1 \tag{B.8}$$

Now, an approximation is necessary to continue. Since the fraction $\frac{hf}{\text{PSD}}$ is typically of the order of magnitude of 10^{-6} to 10^{-5} here, a Taylor expansion can be applied.

$$\left(\frac{hf}{\text{PSD}_{\text{DSM}}} + 1\right)^{\alpha} = 1 + \alpha \frac{hf}{\text{PSD}_{\text{DSM}}} + \mathcal{O}\left(\left(\frac{hf}{\text{PSD}_{\text{DSM}}}\right)^2\right) \tag{B.9}$$

$$1 + \alpha \frac{hf}{\text{PSD}_{\text{DSM}}} \approx 1 + \frac{hf}{\text{PSD}_{\text{TiEMPO}}} \Rightarrow \alpha \approx \frac{\text{PSD}_{\text{DSM}}}{\text{PSD}_{\text{TiEMPO}}} \tag{B.10}$$

Next, the definition of PSD given in equation 2.13 can be used to find the ratio between the used $\text{NEFD}_{\text{continuum}}$, which was used to define PSD_{DSM}, and the effective $\text{NEFD}_{\text{TiEMPO}}$, which is represented by what was just found for $\text{PSD}_{\text{TiEMPO}}$.

$$\alpha \approx \frac{\frac{c^2}{2\Omega f^2}\text{NEFD}_{\text{continuum}}}{\frac{c^2}{2\Omega f^2}\text{NEFD}_{\text{TiEMPO}}} = \frac{\text{NEFD}_{\text{continuum}}}{\text{NEFD}_{\text{TiEMPO}}} \Rightarrow \text{NEFD}_{\text{TiEMPO}} \approx \frac{\text{NEFD}_{\text{continuum}}}{\alpha} \tag{B.11}$$

The ratio as in equation B.7 is visualized in figure B.3. Whilst the NET is perfectly inversely proportional to the integration time, the found standard deviations are not (see legend in figures 5.8a and 5.8b for the curve-fitted parameters). This explains the leftover power law of the form $\alpha \cdot \tau^{\beta}$ which is also curve-fitted and plotted in the figure.

An interesting conclusion can be drawn from the curve-fitted power laws. To compensate for the time-dependency left in the ratio, an integration time of $\tau = 1$ s can be taken; this will yield the ratio if the real signal would have the perfect inverse proportionality to the square root of the integration time that was expected. This yields ratios that are somewhat close to ≈ 1 for the OFF-L signals. The ON-OFF signals yield values that are closer to ≈ 1.4. This is an observation that could be further looked into in future research.

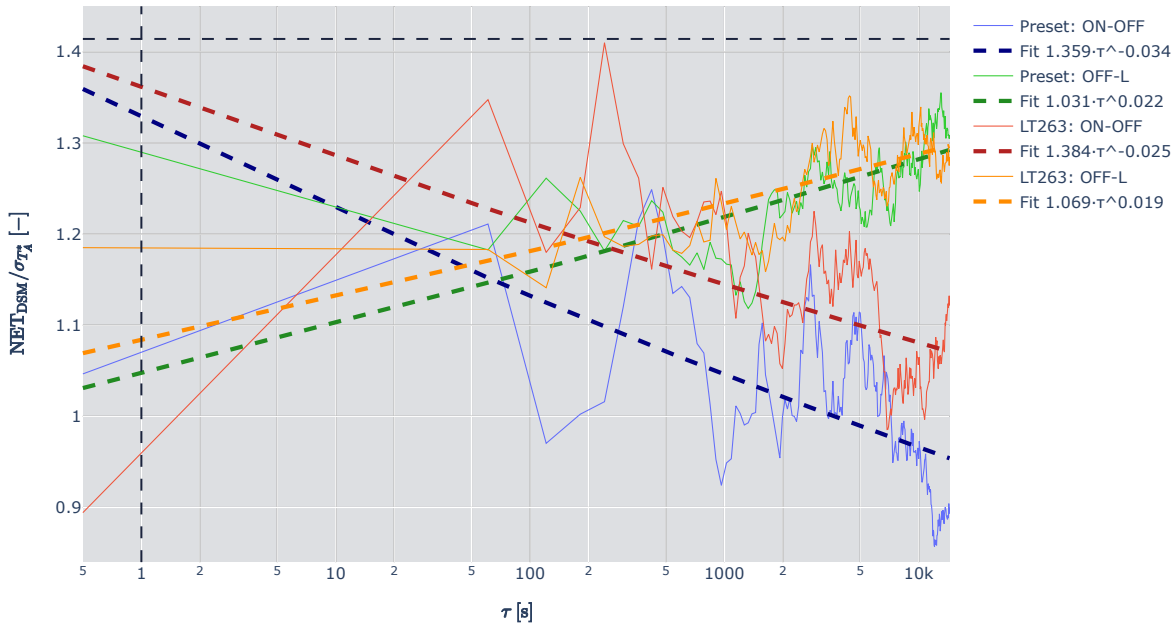


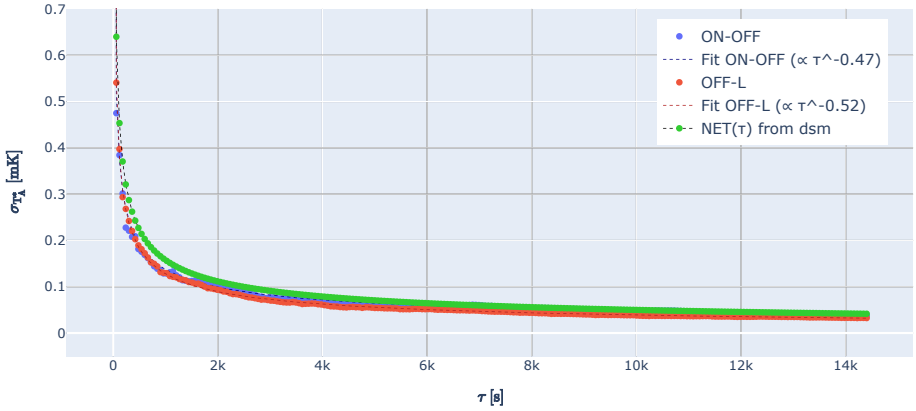
Figure B.3: The ratio between the noise equivalent temperature NET derived from the by `deshima-sensitivity` estimated $\mathrm{NEFD}_{\mathrm{continuum}}$ and the standard deviation of both the ON-OFF and the OFF-L chopped atmosphere-corrected antenna temperature. Both are a function of integration time τ. A horizontal dashed line at $\sqrt{2}$ and a vertical dashed line at 1 s were drawn for comparison. [Interact]

B.4. Linear Axes Plots

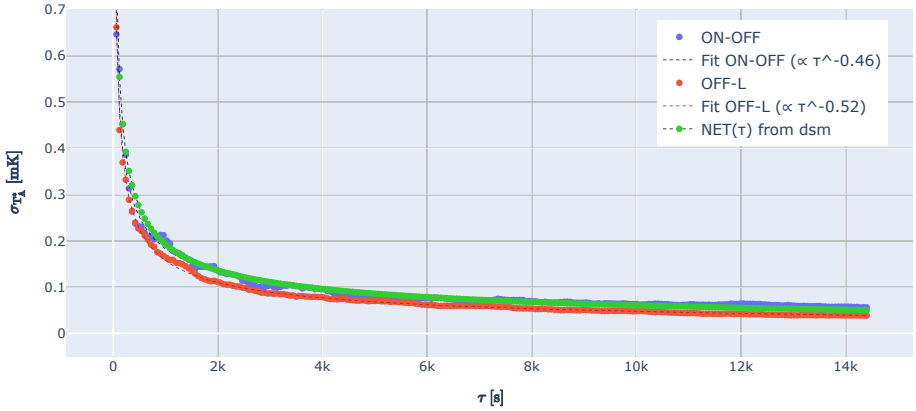
While logarithmic plotting of the $\sigma_{T_A^*}$ covered in section 5.2.1, and the SNR covered in section 5.2.2 show the proportionalities to the integration time more clearly, it is also convenient to show the same plots using natural axes to get a better sense of what is actually happening to the magnitude of what was plotted on the vertical axis.

B.4.1. Linear Axes for Noise Standard Deviation

The main difference for figures B.4a and B.4b compared to figures 5.8a and 5.8b respectively, is that the considered integration times are now also spaced linearly, which is seen influencing the quality of the curve-fits quite significantly. While the shape of the inverse square root proportionality might be more visible on a linear scale, it is still better quantifiable on a logarithmic scale.



(a) Preset chip [Interact]



(b) LT263 chip [Interact]

Figure B.4: The standard deviation $\sigma_{T_A^*}$ of the atmosphere-corrected antenna temperature for the (a) Preset chip and the (b) LT263 chip plotted as a function of integration time τ. The minimum τ considered is 0.5 s, and the maximum is 14400 s (as the total observation time was 8 hours). The integration times shown are spaced linearly. The ON-OFF line represents the found standard deviation in the ON-OFF (dual) chopped signal, whereas the OFF-L line represents it of an OFF-OFF (dual) chopped signal. Both were derived in the frequency range of 270-300 GHz. Their proportionalities to the integration time were curve fitted and plotted as well. The noise equivalent temperature NET was derived from the theoretical noise equivalent flux density NEFD obtained from `deshima-sensitivity` using parameters `pwv_0`=1.0mm and `EL`=60°.

B.4.2. Linear Axes for Signal-to-Noise Ratio

To show the square root proportionality to the integration time, the considered integration time are now spaced linearly.

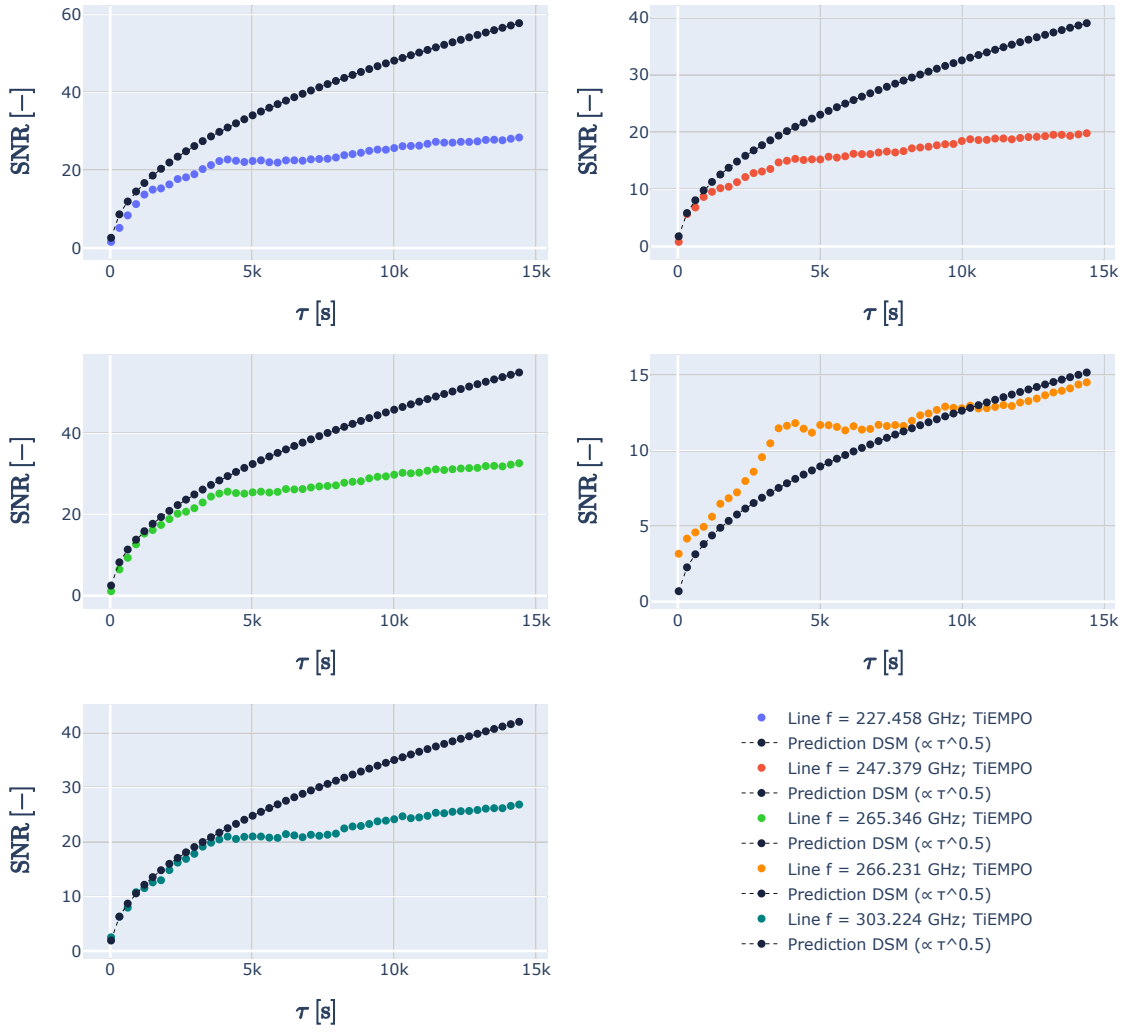


Figure B.5: The signal-to-noise ratio as a function of integration time $\tau = 0.5 \cdot t_{obs}$ (on-source fraction is 0.5 due to dual chopping of the ON-position and the L-position), plotted on linear axes, found in a `TiEMPO` simulation of the Preset chip (t_{obs}=28800s, `pwv_0`=1.0mm, and `EL`=60°) for five different lines (CO $(6-5)$, H_2O $(211-202)$, CO $(7-6)$, [CI] $(2-1)$, and CO $(8-7)$ in order of increasing line frequency) of a model of the J1329+2243 galaxy in the region of 200-310 GHz. The integration time was plotted from 30 s (one minute of observation time) up to 14400 s (eight hours of observation time). To compare the simulated results to the theory, the signal-to-noise ratios found by comparing the MDLF from `deshima-sensitivity` (DSM) to the respective line flux of the line. For the MDLF, the same input parameters were used as for the simulations. [Interact]

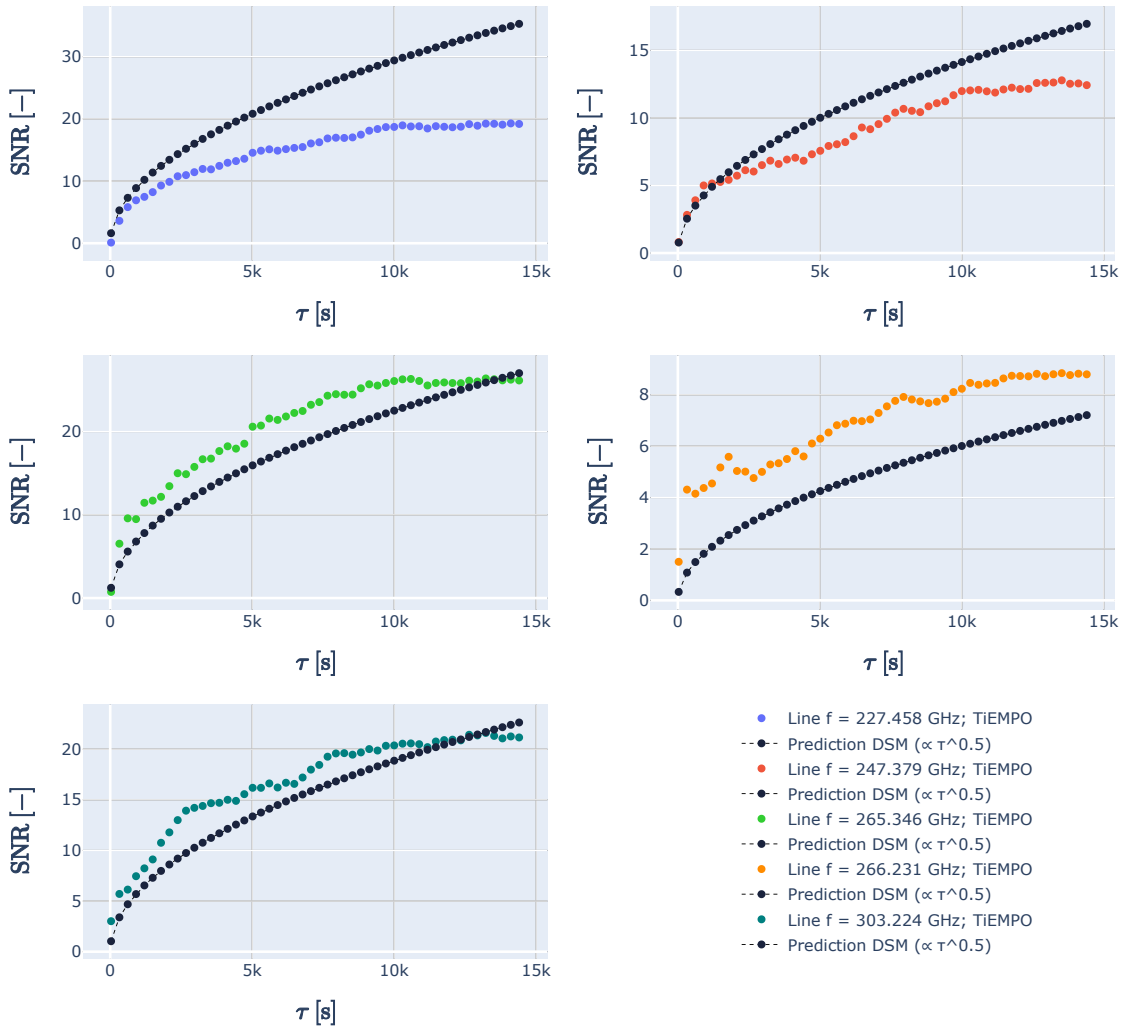
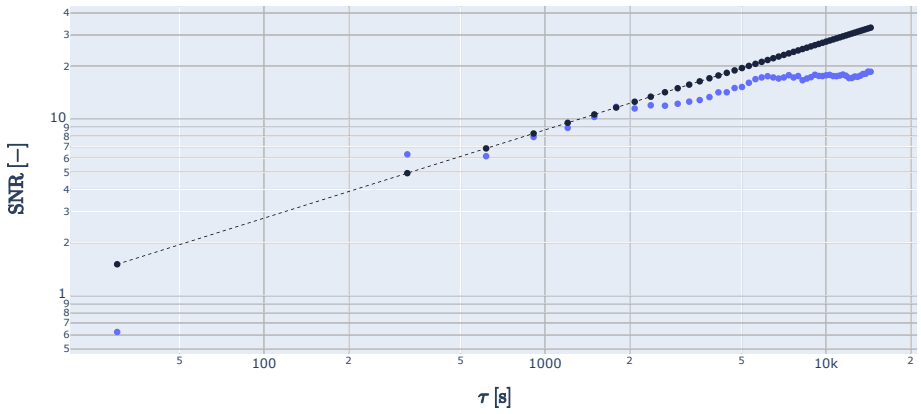


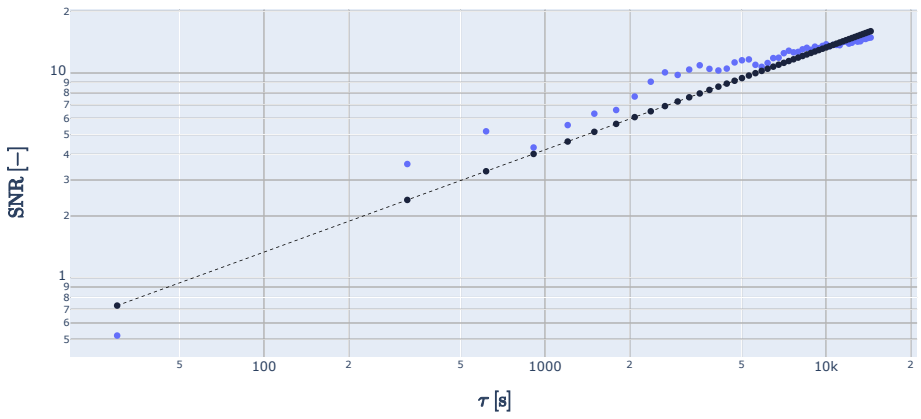
Figure B.6: The signal-to-noise ratio as a function of integration time $\tau = 0.5 \cdot t_{obs}$ (on-source fraction is 0.5 due to dual chopping of the ON-position and the L-position), plotted on linear axes, found in a `TiEMPO` simulation of the LT263 chip (t_{obs}=28800s, `pwv_0`=1.0mm, and `EL`=60°) for five different lines (CO $(6-5)$, H_2O $(211-202)$, CO $(7-6)$, [CI] $(2-1)$, and CO $(8-7)$ in order of increasing line frequency) of a model of the J1329+2243 galaxy in the region of 200-310 GHz. The integration time was plotted from 30 s (one minute of observation time) up to 14400 s (eight hours of observation time). To compare the simulated results to the theory, the signal-to-noise ratios found by comparing the MDLF from `deshima-sensitivity` (DSM) to the respective line flux of the line. For the MDLF, the same input parameters were used as for the simulations. [Interact]

B.5. Signal-to-Noise Ratio of the CO(9-8) Line

With a redshift-corrected frequency of f_{line} = 341.090 GHz, the CO$(9-8)$ line is still within the frequency range of both of the chips, whilst not being in a region heavily distorted by the atmospheric transmission. The range that has a somewhat linear continuum is however quite narrow. Therefore, this line was not included in the report. An SNR plot as a function of integration time can still be created and is given here. It is interesting to see that an SNR of 5 is still reached within the 30 min that was concluded to be enough for the current DESHIMA2.0 to detect the other non-merging lines.



(a) Preset chip



(b) LT263 chip

Figure B.7: The signal-to-noise ratio as a function of integration time $\tau = 0.5 \cdot t_{obs}$ (on-source fraction is 0.5 due to dual chopping of the ON-position and the L-position) found in a `TiEMPO` simulation of the (a) Preset chip and the (b) LT263 chip (t_{obs}=28800s, pwv_0=1.0mm, and EL=60°) for the CO$(9-8)$ line of a model of the J1329+2243 galaxy in the region 335-362.5 GHz. The integration time was plotted from 30 s (one minute of observation time) up to 14400 s (eight hours of observation time). To compare the simulated results to the theory, the signal-to-noise ratios found by comparing the MDLF from `deshima-sensitivity` (DSM) to the respective line flux of the line. For the MDLF, the same input parameters were used as for the simulations.