Optimizing End-to-End System Performance for Millimeter and Submillimeter Spectroscopy of Protostars: Wideband Heterodyne Receivers and Sideband-Deconvolution Techniques for Rapid Molecular-Line Surveys

Thesis by

Matthew C. Sumner

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2011

(Defended September 23, 2010)

Dedicated to Heather, Claire, and David.

Acknowledgements

I would like to offer my deep gratitude to the following people for making this work possible:

Jonas Zmuidzinas for his faith in me, for standing by me during the challenging times, for his generous support of his students, and for seeing every setback as an exciting opportunity to understand the problem better.

My wife, Heather, for her constant love and support, for all the fun times we have had during this process, and for the many, many sacrifices she has made to allow me to complete my degree.

My children, Claire and David, for bringing joy and sunshine into every day, for their patience on all the nights and weekends that I spent in the office finishing this work, and for helping me to keep perspective on the things that really matter in life.

My parents, Jim and Molly, for always believing in me, for starting me on the path that has led to this point, and for lending a sympathetic ear whenever I needed support.

Sue, Roger, Laura, and Greg for never doubting that I would finish, for welcoming me into their family, and for supporting me in the final stretch. I would particularly like to thank Sue for flying down on a moment's notice and for staying weeks at a time to help us out while I finished my thesis.

Susanna Widicus-Weaver for being a colleague and a friend, for teaching me to use the CSO, and for making our many observing runs both productive and fun.

Tom Phillips for his commitment to mentoring students, particularly his generosity with telescope time in support of students' work, and for always being willing to pull up a chair at *Hale Pohaku* for a friendly conversation over dinner.

Geoff Blake for his enthusiasm, for his focus on helping me to graduate quickly, and for his help at the summit when we were taking the data used in this work.

Ernie from the "Ernie's al Fresco" lunch truck, for making sure I had a warm lunch and a kind word every day.

Joseph V. for jumping in and running the final lap at my side, offering moral support and practical advice every step of the way.

Alice S. for her professional guidance and for a friendship that endured throughout my time at Caltech.

The Caltech Graduate Student Council (GSC) for providing many opportunities that I never expected to find in graduate school, for encouraging me to identify and pursue previously unknown interests, for the countless (and often thankless) volunteer hours they provide for Caltech, and for the many great friends I met while working on the Board of Directors.

Abstract

This thesis describes the construction, integration, and use of a new 230-GHz ultra-wideband heterodyne receiver, as well as the development and testing of a new sideband-deconvolution algorithm, both designed to enable rapid, sensitive molecular-line surveys.

The 230-GHz receiver, known as Z-Rex, is the first of a new generation of wideband receivers to be installed at the Caltech Submillimeter Observatory (CSO). Intended as a proof-of-concept device, it boasts an ultra-wide IF output range of $\sim$ 6 - 18 GHz, offering as much as a twelvefold increase in the spectral coverage that can be achieved with a single LO setting. A similarly wideband IF system has been designed to couple this receiver to an array of WASP2 spectrometers, allowing the full bandwidth of the receiver to be observed at low resolution, ideal for extra-galactic redshift surveys. A separate IF system feeds a high-resolution 4-GHz AOS array frequently used for performing unbiased line surveys of galactic objects, particularly star-forming regions. The design and construction of the wideband IF system are presented, as is the work done to integrate the receiver and the high-resolution spectrometers into a working system. The receiver is currently installed at the CSO where it is available for astronomers' use.

In addition to demonstrating wideband design principles, the receiver also serves as a testbed for a synthesizer-driven, active LO chain that is under consideration for future receiver designs. Several lessons have been learned, including the importance of driving the final amplifier of the LO chain into saturation and the absolute necessity of including a high-Q filter to remove spurious signals from the synthesizer output. The on-telescope performance of the synthesizer-driven LO chain is compared to that of the Gunn-oscillator units currently in use at the CSO. Although the frequency agility of the synthesized LO chain gives it a significant advantage for unbiased line surveys, the cleaner signal and broader tuning range of the Gunn continue to make it the preferred choice.

The receiver and high-resolution spectrometer system were brought into a fully operational state late in 2007, when they were used to perform unbiased molecular-line surveys of several galactic sources, including the Orion KL hot core and a position in the L1157 outflow. In order to analyze these data, a new data pipeline was needed to deconvolve the double-sideband signals from the receiver and to model the molecular spectra. A highly automated sideband-deconvolution system has been created, and spectral-analysis tools are currently being developed.

The sideband deconvolution relies on chi-square minimization to determine the optimal single-sideband spectrum in the presence of unknown sideband-gain imbalances and spectral baselines. Analytic results are presented for several different methods of approaching the problem, including direct optimization, nonlinear root finding, and a hybrid approach that utilizes a two-stage process to separate out the relatively weak nonlinearities so that the majority of the parameters can be found with a fast linear solver. Analytic derivations of the Jacobian matrices for all three cases are presented, along with a new *Mathematica* utility that enables the calculation of arbitrary gradients.

The direct-optimization method has been incorporated into software, along with a spectral simulation engine that allows different deconvolution scenarios to be tested. The software has been validated through the deconvolution of simulated data sets, and initial results from L1157 and Orion are presented.

Both surveys demonstrate the power of the wideband receivers and improved data pipeline to enable exciting scientific studies. The L1157 survey was completed in only 20 hours of telescope time and offers moderate sensitivity over a $>$ 50-GHz range, from 220 GHz to approximately 270 or 280 GHz. The speed with which this survey was completed implies that the new systems will permit unbiased line surveys to become a standard observational tool. The Orion survey is expected to offer $\sim$ 30 mK sensitivity over a similar frequency range, improving previous results by an order of magnitude. The new receiver's ability to cover such broad bandwidths permits very deep surveys to be completed in a reasonable time, and the sideband-deconvolution algorithm is capable of preserving these low noise levels. Combined, these tools can provide line spectra with the sensitivity required for constraining astrochemical models and investigating prebiotic molecules.

Contents

Chapter 1

Introduction

1.1 Star Formation

Throughout most of the interstellar medium, atomic species dominate. The density is so low that interactions between atoms are rare, and any molecules that do form are destroyed almost immediately by stellar radiation. Within this harsh environment, however, there are oases in the form of molecular clouds. These clouds consist of dense collections of dust and gas in which the outer layers of dust grains protect the interiors of the clouds from stellar radiation. The higher density provides more opportunities for atoms to interact and form molecules while the gentler environment allows those molecules to survive.

Current theories of stellar formation hold that stars are born in the depths of these clouds. In particular, low-mass stars ($\sim 1\ M_{Sun}$) are believed to follow a life cycle similar to the one shown in Figure 1.1 [e.g., Andre et al., 2000, Lada, 1987, Shu et al., 1993, Boogert, 1999]. Deep in the cloud, dust and gas form pockets of higher-density material. Over time, these dense cores become gravitationally unstable, initiating an inside-out collapse to create a protostar at the center. Following the nomenclature of Andre et al. [1993], an object in this stage of low-mass stellar evolution is usually referred to as a Class 0 protostar.

As can be seen in the artist's conception in Figure 1.2, Class 0 protostars are still deeply enshrouded in the dust and gas that gave birth to them. Material continues to fall onto the central source, creating an ever-expanding bubble in the molecular cloud around the protostar. The infalling matter carries angular momentum with it; in order for it to form a compact protostar, some mechanism is needed to shed the excess angular momentum. As discussed in Shu et al. [1993, 2000], disk-mediated mass transfer provides one method of

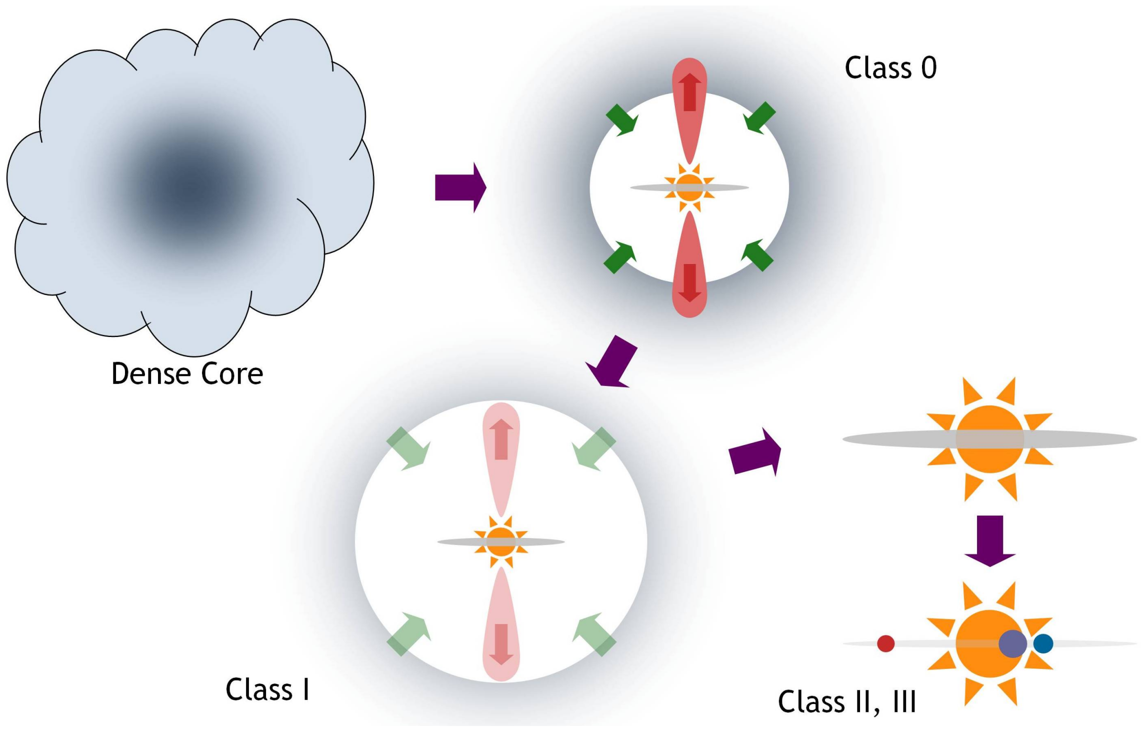


Figure 1.1: Overview of low-mass star formation.

allowing matter to accrete onto the central source. In this model, excess angular momentum is carried off by magnetically driven jets emanating from both poles of the protostar.

Most of the gas inside a molecular cloud is relatively cold, with temperatures on the order of $T \sim 10$ - 20 K [Gueth et al., 1997, Shu et al., 1993]. At this stage of stellar development, the protostar is not massive enough to support internal burning. However, the gravitational potential energy given up by material falling into the central protostar releases a significant amount of energy, which is eventually converted into heat in the surrounding gas.[1] This increased thermal energy allows the relatively simple molecules of the molecular cloud to interact with one another, driving a surprisingly rich chemistry.

As the protostar evolves into a so-called Class I object, the amount of matter falling onto the central source decreases. The angular extent of the bipolar jets expands, gradually clearing the remains of the dense core. The protostar continues to evolve into a Class II

[1]In high-mass protostars, some of the richest chemistry occurs in the hot core, the warm region near the protostar. It is unclear whether low-mass protostars support miniature hot cores, although there is evidence favoring their existence (e.g., Schoier et al., 2002).

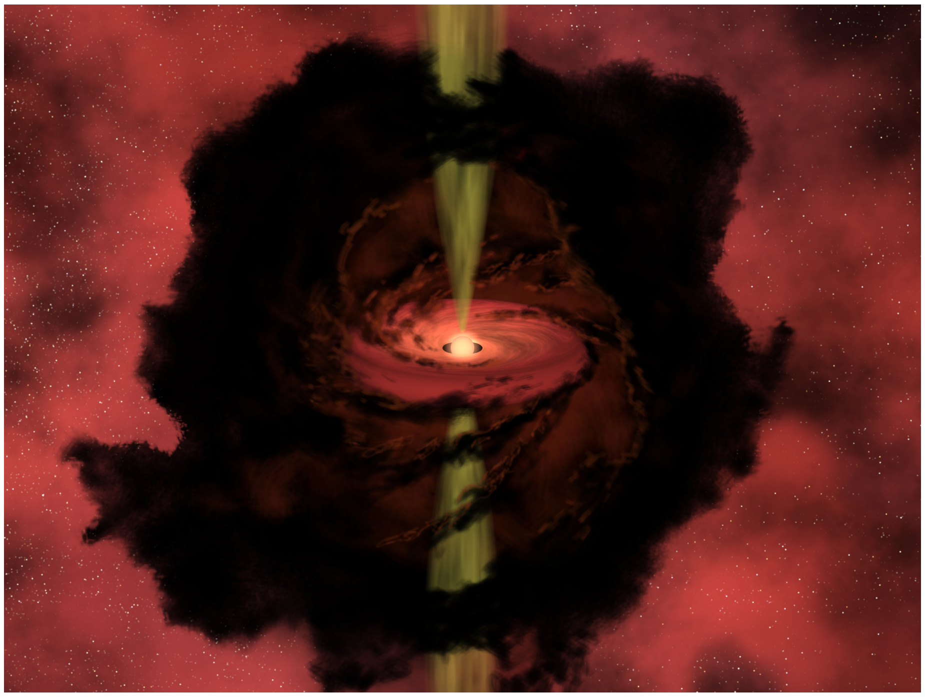


Figure 1.2: Artist's conception of a heavily enshrouded protostar. (Image courtesy of NASA/JPL-Caltech/R. Hurt, SSC.)

object, characterized by a fully exposed central source surrounded by a dusty, protoplanetary disk, and finally a Class III object, in which disk-clearing has condensed much of the diffuse material of the disk into planets.

1.2 Studying Molecular Clouds

Molecular clouds exist because dust grains shield the interiors from intense optical and ultraviolet stellar radiation; however, this also makes them difficult to study at many of the traditional wavelengths. In particular, the same dust grains that keep light from entering also prevent it from escaping, making it impossible to study the interiors of molecular clouds with optical observations.

The physical process behind the dust grain's extinction of light is Rayleigh scattering, which intensifies as a strong function of frequency; radiation with short wavelengths is scattered much more strongly than that of longer wavelengths. Using lower frequencies allows observers to peer deeper into molecular clouds.

As an example, consider the images of the Carina Nebula in Figure 1.3. The top image shows a dust pillar in the visible spectrum while the bottom shows the same image in infrared. Because of their longer wavelengths, the infrared photons are not scattered as heavily by the dust. They escape from the depths of the molecular cloud, making the dust virtually invisible and clearly revealing an embedded protostar (and its associated outflows) at the tip of the dust column.

The situation improves further when even longer wavelengths are used, as shown in Figure 1.4, comparing visible and submillimeter images of the Antennae Galaxies.[2] The interactions between these colliding galaxies has stirred the dust and gas, triggering star formation. This fact is not obvious in the visible image, but appears clearly in the submillimeter, where the bright red regions show dense dust that has been warmed by obscured star formation.

Submillimeter observations benefit not only from an improved ability to penetrate through dust, but also because they are directly sensitive to emissions from the gas and dust in the star-forming regions. At the temperatures typical of these objects, thermal emission from the dust peaks at frequencies of a few to tens of THz, generating a significant amount of flux at the upper end of the submillimeter range [Kraus, 1986, esp. Fig 3.16]. Because the peak of the emission occurs to the short side of the submillimeter range, submillimeter imaging arrays (such as SHARC II, which generated the data shown in Figure 1.4) are ideal for searching for distant star-forming regions in which the spectrum has been redshifted.

Not surprisingly, molecular clouds also generate a significant amount of flux in molecular emission lines. In particular, the temperatures are just right to excite the low-lying rotational (and sometimes vibrational) quantum levels of small molecules, and the resulting rovibrational emission spectral lines lie squarely in the submillimeter range. For instance, CO, one of the most common interstellar molecules, has strong emission lines spaced at 115 GHz, and the lines at 230 GHz and 345 GHz are particularly useful tracers of molecular gas.

[2]Submillimeter and millimeter wavelengths fall between microwave and far-infrared. The submillimeter image shown here was taken by the SHARC II camera, operating at 350 μm ($\approx$ 850 GHz). The term "submillimeter astronomy" refers to observations in the range $\sim$ 300 GHz to a few THz (corresponding to wavelengths of $\sim$ 1 mm to $\sim$ 0.1 mm, respectively). The Caltech Submillimeter Observatory, the telescope used for the observations in this thesis, has instruments covering frequencies from approximately 180 GHz to 1 THz. The term "submillimeter" will be used throughout this work to refer to observations in those frequencies, although the low-frequency end of the range might be more properly termed "millimeter astronomy."

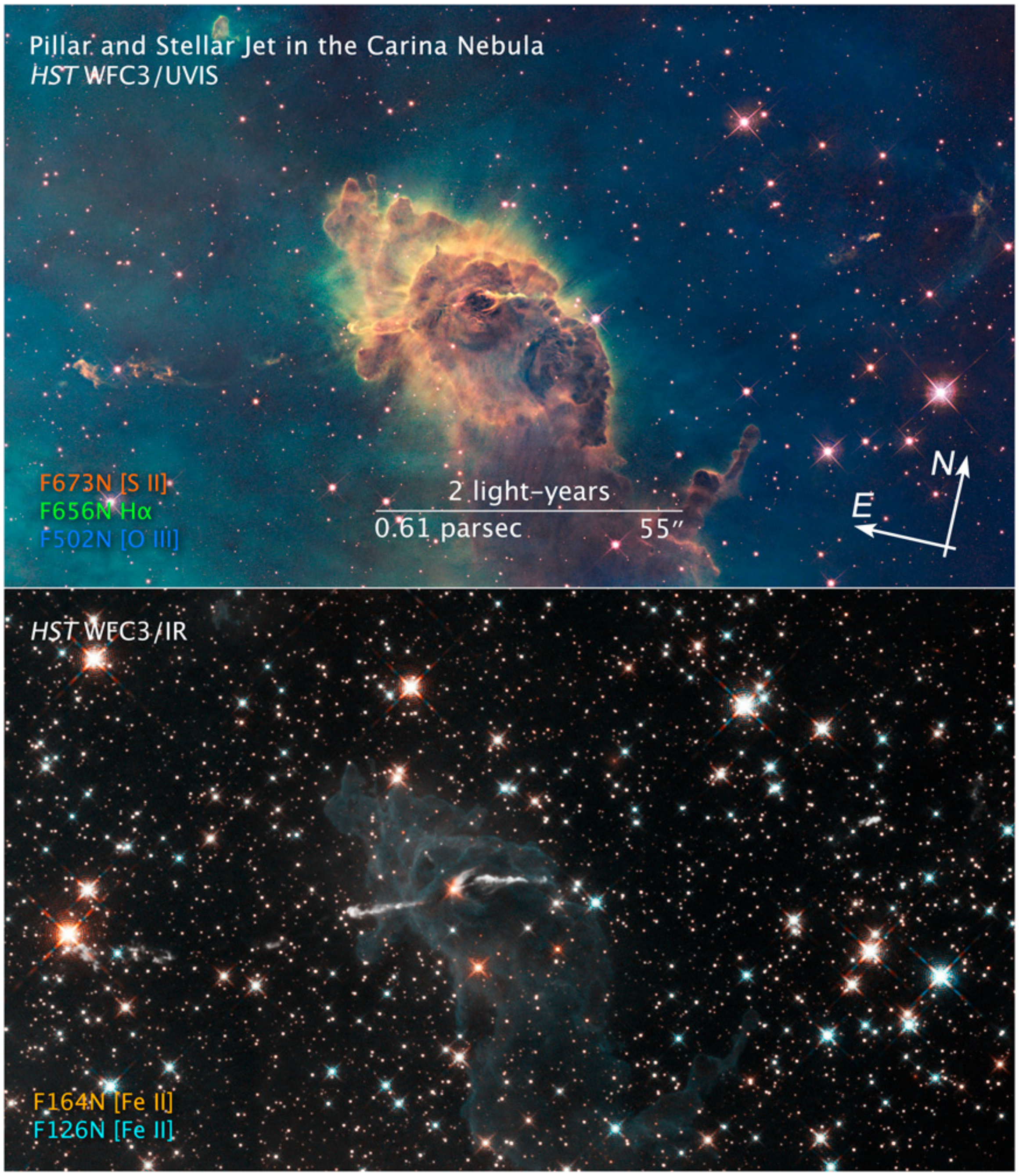
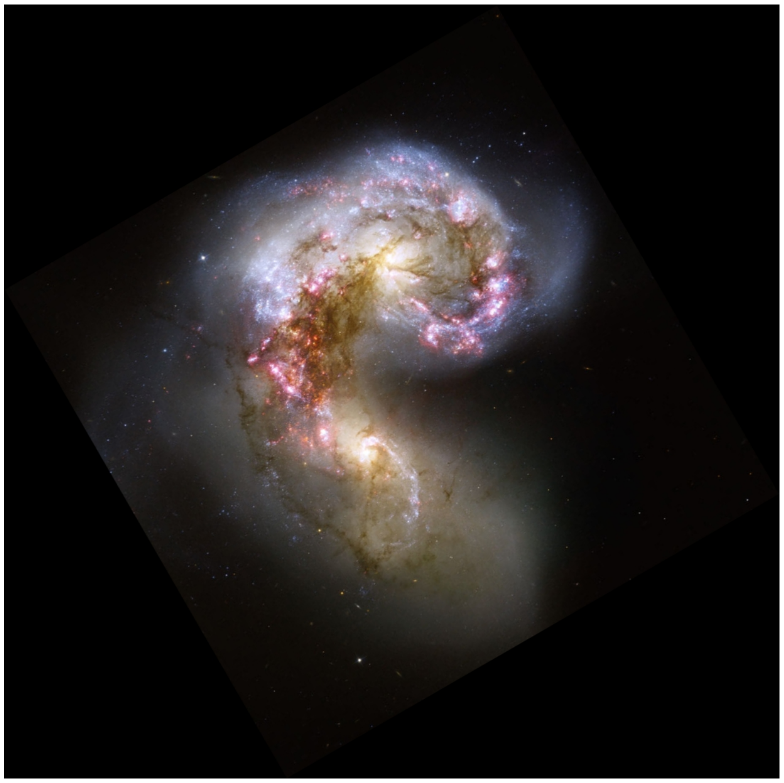


Figure 1.3: Hubble images of dust pillar in the Carina Nebula in visible (top) and infrared (bottom). The longer wavelength of the infrared photons allows them to escape from the molecular cloud, revealing a protostar that is hidden by dust in the visible spectrum. (Image credit: NASA, ESA, and the Hubble SM4 ERO Team.)



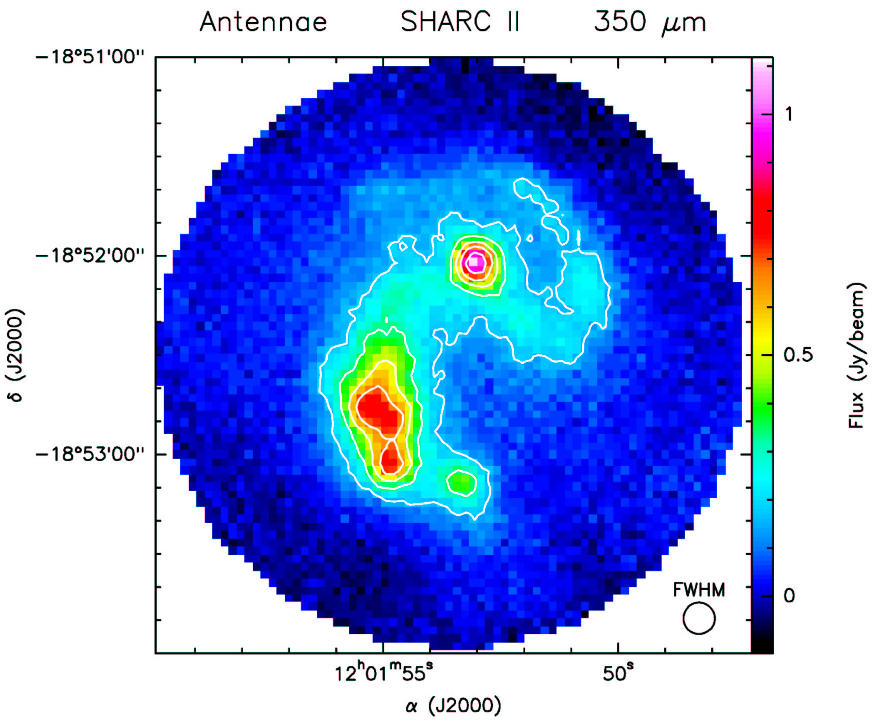


Figure 1.4: Visible and submillimeter images of the Antennae Galaxies. (Visible image credits: NASA, ESA, and the Hubble Heritage Team (STScI/AURA)-ESA/Hubble Collaboration, with acknowledgment to B. Whitmore (STScI). SHARC II image credit: Darren Dowell.)

1.3 Interplay between Physics and Chemistry

As Groesbeck et al. [1994] demonstrated, the integrated flux in the molecular lines may represent a significant fraction of, or even the majority of, the energy emitted from star-forming regions in the submillimeter range. This means that instruments sensitive to that emission make great tools for studying such regions, but it also implies something important about the physics of star formation. As stated earlier, the gas and dust falling onto the central core of a protostar give up a considerable amount of gravitational potential energy that eventually is converted into heat. In order to continue to collapse, the gas in and around the protostar must have some way of dissipating this heat. While thermal emission from dust plays a key role, cooling via molecular-line emission is also critical to the process. Thus, in order to fully describe star formation, one must understand the line emission from the region, which in turn, requires a knowledge of the molecular constituents.

Understanding the astrochemistry of star formation is also important for establishing the physical state of material surrounding a protostar. Throughout much of a molecular cloud, a significant portion of the molecular gas is believed to be frozen into ices that coat the surfaces of dust grains [Boogert, 1999]. The warmth generated by stellar formation causes these ices to sublimate, increasing the density of the gas-phase material near the protostar. In order to calculate the details of this density profile, it is necessary to understand the constituents of the molecular ices.

Similarly, the physics of star formation has a strong effect on the associated astrochemistry. Without the thermal energy generated by the infalling matter, most of the molecules would remain frozen on dust grains, constraining the types of reactions they could undergo. The gas-phase density has direct consequences for the likelihood of collisions between molecules while the physical size of a star-forming region sets a distance scale over which one molecule must encounter another if it is to react before drifting out into the colder reaches of the molecular cloud. In addition, the powerful outflows from a protostar create their own chemistry, both within the ejected material and at the shock front that results from the outflow plowing into the quiescent envelope.

From these examples, it can be seen that detailed modeling of star formation requires a genuinely interdisciplinary approach; in order to understand how a dense core condenses into a protostar, it is necessary to simultaneously address the chemistry and the physics.

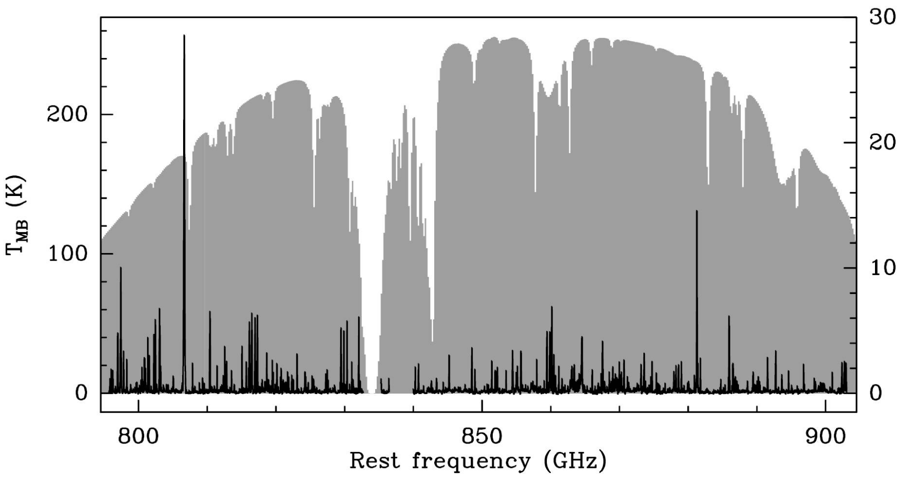


Figure 1.5: 795 - 903 GHz survey of Orion by Comito et al. [2005]. The atmospheric transmission typical of the observing conditions for the survey (shown in gray in the background of the plot) testifies to the difficulty of this work.

1.4 Submillimeter Line Surveys

One particularly powerful way to study star-forming regions is through the use of unbiased submillimeter line surveys, in which molecular lines are observed over a broad range of frequencies. Figure 1.5 shows a survey of the Orion molecular cloud from 795 - 903 GHz [Comito et al., 2005], showing a wealth of lines, even at these relatively high frequencies.

In contrast to targeted line studies, which seek to confirm or refute the existence of a particular molecular line, an unbiased survey seeks to inventory all of the lines within the observing window. This information can be used to build up a “chemical catalog” for the star-forming region, placing important constraints on models and helping to distinguish between competing descriptions of astrochemical networks.

Closer to home, these molecules represent progenitors of the circumstellar disk, which in turn forms the planets, comets, and asteroids. Whether these molecules survive in their initial form to become part of the solar system, or whether they are destroyed in some of the intervening steps, remains an open question. However, there is evidence that biologically relevant molecules can be found in the hot cores surrounding high-mass protostars [Widicus Weaver, 2005]. This raises the tantalizing possibility that astrochemistry could play an important role in understanding the origin of life, providing a viable mechanism for the introduction of biological molecules to Earth.

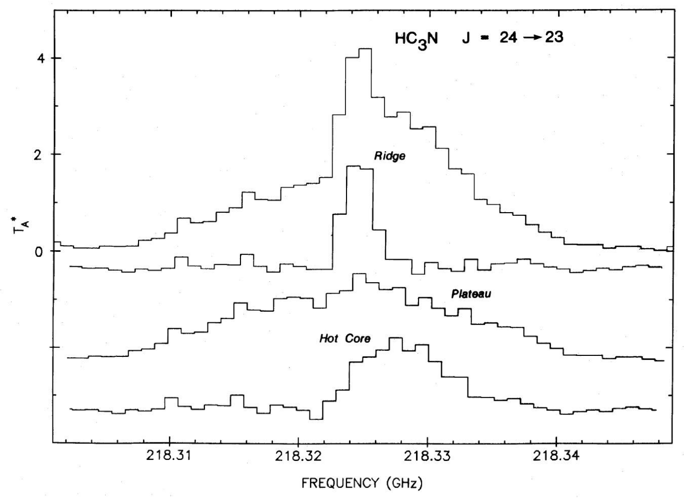


Figure 1.6: Decomposition of spectral line in OMC-1 survey. From Blake et al. [1987, Fig. 2].

By directly probing the molecular gas, a line survey provides a wealth of information about chemical and physical conditions within the star-forming region. Line frequencies can be used to identify molecular species while the line profiles give important information about source dynamics. Amplitudes of lines can be used to estimate physical conditions in the source, such as temperature, pressure, and density, particularly when many molecules, each with multiple lines, are observed.

As an example, consider the peak shown in Figure 1.6, taken from a survey of the Orion molecular cloud (OMC-1). As described in Blake et al. [1987], the observed spectral line, shown at the top, can be decomposed into contributions from several physically distinct regions based on the line profile, as shown in the remaining three traces in the figure. The bottom line (labeled "Hot Core") corresponds to gas warmed by the central source while the narrow "Ridge" line represents quiescent gas unaffected by the protostar's creation in its midst. The very broad tails of the "Plateau" line tie it to the powerful outflows driven by the protostar. Such analysis not only helps to identify the dynamics of the source, but also allows the chemistry of each region to be analyzed separately, even when the lines are blended together.

Despite the wealth of scientific information that can be derived from an unbiased line survey, only a handful of these surveys have been performed. The primary reason is that such surveys are difficult to perform, requiring a significant amount of telescope time that often spans multiple observing runs, sometimes extending across several years. Not only does this represent a significant allocation of resources, but changes in the telescope or receiver configuration can make it difficult to stitch the different data sets together into a single spectrum.

One of the first major surveys of a star-forming region was completed by Blake and Sutton, who studied the Orion molecular cloud from 215 - 263 GHz [Blake et al., 1986, Sutton et al., 1985]. That survey consisted of more than 100 independent observations over 28 nights spread across two adjacent winters. The 795 - 903 GHz survey of the same source, shown in Figure 1.5, required a total of 300 independent observations spread over an approximately five-year period [Comito et al., 2005]. Just the upper half of the spectrum (from 845 - 903 GHz) required 30 nights of observing time, largely due to the challenge of observing at near-THz frequencies from a ground-based telescope.

In Schilke et al. [1997], the authors point out that completing the observations and assembling them into a coherent spectrum is just the beginning and that the truly time-consuming part is the spectral analysis and interpretation. Thus, while line surveys provide a wealth of data, the information is hard-won.

1.5 Instrumentation for Line Surveys

Submillimeter line surveys of star-forming regions are nearly always performed using heterodyne spectroscopy, with an instrumental configuration similar to that shown in Figure 1.7. Photons collected by the telescope are focused onto a mixer, where they are combined with a reference signal generated by a local oscillator (LO). The mixer consists of a device with a nonlinear relationship between current and voltage (a nonlinear I-V curve), which results in the multiplication of the two input signals. The output consists of two signals, one at a frequency equal to the sum of the two input frequencies, and one equal to the difference between them. Only the difference signal is needed for this application, so the sum frequency is removed using a filter. From there, the signal is amplified and passed to a spectrometer, which generates the desired spectrum.

From a practical point of view, heterodyne downconversion provides several important advantages. The mixer brings the signal down from hundreds of GHz to a few GHz, where

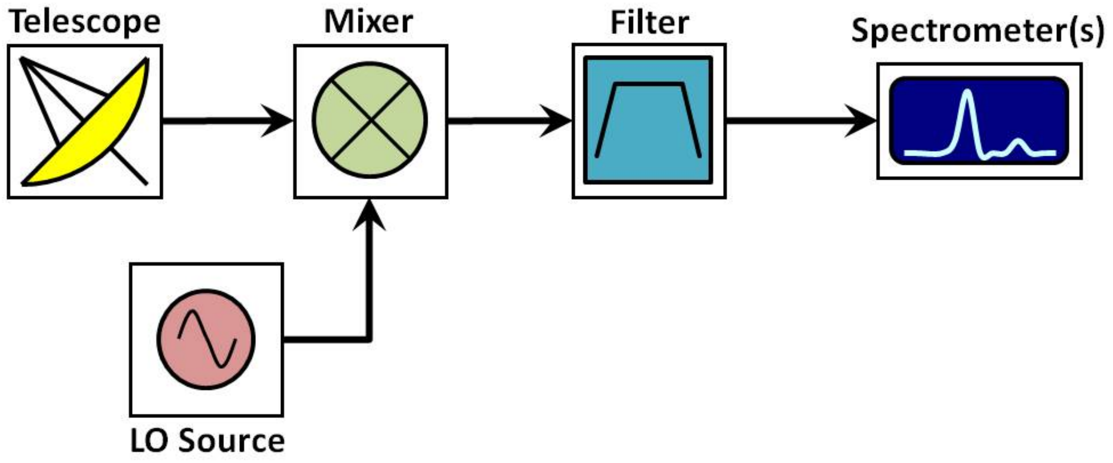


Figure 1.7: Basic instrumentation needed for heterodyne spectroscopy.

commercial, off-the-shelf amplifiers, filters, and connectors are readily available, making it vastly easier and cheaper to work with the signal. This configuration also provides modularity that allows the front-end mixers to be designed separately from the back-end spectrometers. For instance, at the CSO, there are multiple receivers, covering frequencies from 180 GHz up to 900 GHz. They are configured to convert the sky radiation to a common frequency on the output, allowing a single set of spectrometers to be used for all of them.

Scientifically, heterodyne spectroscopy is desirable because it offers extremely high frequency resolution, which is critical for studying molecular-line spectra of star-forming regions. A sample spectrum from the Orion survey is shown in Figure 1.8, demonstrating the fine details that must be preserved. The narrower peaks are only a few MHz wide and the CH_3OH and HNCO lines near the center of the spectrum have frequencies of 241.767 GHz and 241.774 GHz, corresponding to a 7-MHz separation.

Preserving these kinds of details requires an instrument capable of separating ~ 1 MHz differences at frequencies of ~ 300 GHz, corresponding to resolving power of $R \sim 3 \times 10^5$. Fortunately, a heterodyne receiver can easily meet this challenge. In fact, the actual resolution is typically set by the channel spacing in the spectrometer rather than the underlying resolving power of the receiver.

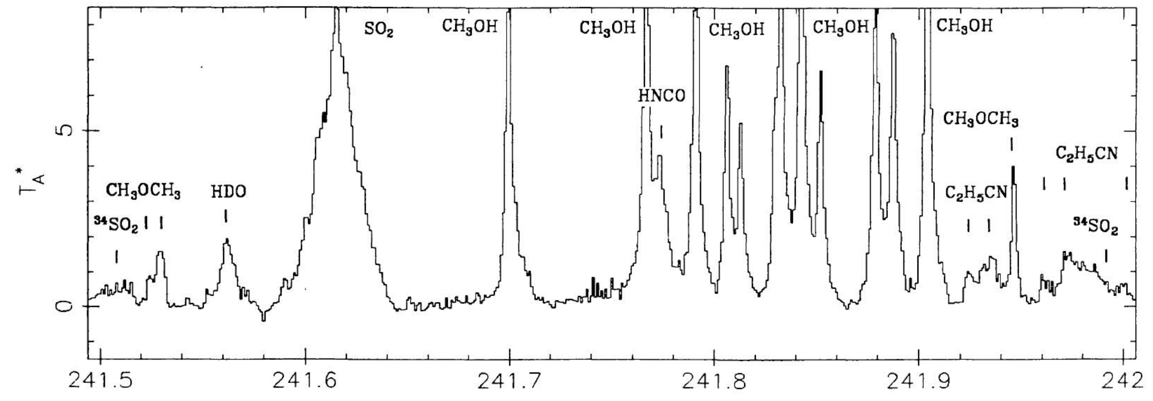


Figure 1.8: A sample molecular-line spectrum from 241.5 - 242 GHz [Sutton et al., 1985].

1.6 Sideband Ambiguity

Heterodyne spectroscopy is well suited to the study of submillimeter molecular lines, but there is one difficulty that it introduces. The goal of a line survey is to produce frequency-calibrated spectra like the one shown in Figure 1.8, in which each channel of the spectrum can be concretely identified with a single frequency. However, at most telescopes (including the CSO), each channel in a heterodyne spectrum actually corresponds to *two* frequencies; it can only be reduced to the form shown above with a significant amount of data processing.

More detail is given in Chapter 4, but the basic problem can be demonstrated by reconsidering the simple system shown in Figure 1.7. In Figure 1.9, we show the same system with the addition of frequency labels. Start by considering the general variables shown in the boxes attached to each of the black arrows. The telescope collects photons with frequency f_{RF} while the LO generates a signal at frequency f_{LO}. In the mixer, these are multiplied to produce the sum and difference frequencies, $f_{SUM} = f_{LO} + f_{RF}$ and $f_{DIFF} = |f_{LO} - f_{RF}|$, respectively. The filter removes the sum frequency, leaving only the difference frequency, f_{DIFF}, at the input to the spectrometer.

As a concrete example, consider what happens when the telescope observes 235-GHz photons while the LO is set to a frequency of 240 GHz. The mixer produces sum and difference frequencies at 5 GHz and 475 GHz, respectively. After the filter removes the sum frequency, the spectrometer is presented with a 5-GHz signal. However, the same

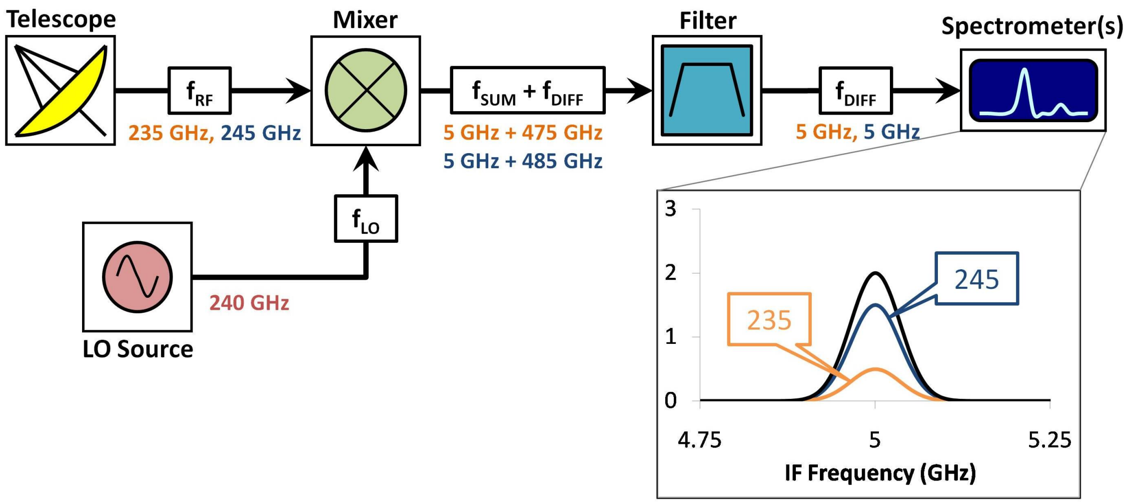


Figure 1.9: Frequency conversions in a simple double-sideband receiver, showing the origin of the sideband ambiguity.

result can be achieved by considering a 245-GHz input to the telescope. The difference frequency will still be 5 GHz, so the input to the spectrometer will be identical in both cases.

Given the knowledge that the spectrometer observed a 5-GHz peak while using a 240-GHz LO setting, we can only say that the input must have contained some combination of 235-GHz and 245-GHz signals. This uncertainty is known as the sideband ambiguity, and it is inherent to the single-mixer receiver design outlined above. It is possible to remove this ambiguity during the data-analysis stage via a process known as sideband deconvolution, which will be covered extensively in Chapter 4.

Frequencies greater than f_{LO} are said to come from the upper sideband while those less than f_{LO} come from the lower sideband. Because this receiver design is sensitive to both sidebands, it is known as a double-sideband receiver. By adding an additional mixer to the receiver, making it a two-mixer design, it is possible to develop a receiver that eliminates this ambiguity. Traditionally, this design has not been used, partly due to the added complexity and partly because double-sideband receivers are actually more efficient for certain types of observations.

1.7 Goals of This Work

This thesis describes an effort to enable rapid, unbiased molecular-line surveys at submillimeter frequencies with the goal of allowing such surveys to be performed more frequently and to achieve higher sensitivities.

Because of the difficulties of performing such work, only a few sources have been surveyed to date, and significant pieces of our understanding of star formation have been extrapolated from these objects. Making it easier to complete a line survey would allow astronomers to study many more objects, eventually building up a statistical sample of sources that could help to elucidate the different mechanisms at play.

One particularly intriguing line of inquiry is the search for "chemical clocks," molecular tracers that would help to identify the ages of different sources. Currently, estimating the age of a protostar requires subtle inference from a variety of observations; finding a molecular tracer that could serve as a proxy for the protostar's age would greatly simplify the process. While efforts have been made to find such tracers through targeted line observations, a broad sample of unbiased line surveys would make the task significantly easier.

Improved survey methods would also offer the ability to take more sensitive line surveys. As models of astrochemistry improve, more sensitive observations are needed to distinguish between them, usually to look for peaks from complex molecules that are predicted by the models but are lost in the noise of prior surveys. Likewise, unambiguously identifying prebiotic molecules requires sufficient sensitivity to dig multiple molecular lines out of the noise.

Accomplishing these goals has required the development of new hardware and software. Improvements in computer models, chip fabrication, and micromachining capabilities have allowed our group to design a new generation of ultra-wideband heterodyne receivers. In a single observation, these receivers can produce several times as much data as previous receivers, allowing a line survey to be completed with many fewer independent observations.

Improving the ability to generate data is not sufficient unless we also improve our ability to process that data. Traditionally, line surveys have been analyzed by cleaning individual observations by hand, using a computer to perform the sideband deconvolution and assemble the observations into a single spectrum, and then manually fitting each peak in the resulting spectrum.

We have streamlined the analysis process by developing deconvolution algorithms that dramatically reduce the amount of manual processing required for each observation while still preserving the high sensitivity needed to search for complex and/or prebiotic modules. Susanna Widicus Weaver's group at Emory University is addressing the final part of the problem by developing spectral-fitting software that not only fits all of the lines of a given molecule simultaneously, but also fits several molecules at once.

Subsequent chapters will discuss the first two stages of this work, briefly describing the hardware improvements that have enabled the underlying observations to be completed more quickly and taking a detailed look at the new deconvolution algorithms that have been developed.

Chapter 2

Submillimeter Observations

Before describing the construction of heterodyne receivers, we include a quick discussion of how submillimeter spectra are actually measured. Most of the signal for a submillimeter heterodyne receiver consists of thermal noise. The primary source of background is thermal emission by molecules in the atmosphere (especially water). The astronomical source adds a small bit of power on top of this, and the challenge is to separate out that small signal against the much larger background. This is the primary reason submillimeter telescopes are typically located at high elevations. The amount of water in the atmosphere falls off rapidly as a function of elevation; by observing at higher locations, astronomers can reduce the total amount of water between the telescope and the astronomical source.

As a concrete example, consider observations made under typical conditions at the Caltech Submillimeter Observatory (CSO), located at an elevation of 13,350 feet, near the summit of Mauna Kea (Figure 2.1). The 225-GHz opacity at zenith is usually $\tau_{225} \sim 0.1$, and observations are performed at moderate zenith angles, corresponding to an airmass of $A \sim 1.5$. The signal from an astronomical source at 225 GHz would therefore be attenuated by a factor of $e^{-A\,\tau_{225}} \approx 0.86$. In addition to attenuating the signal, the atmosphere introduces noise equal to $T_{sky} = (1 - e^{-A\,\tau_{225}})\,T_{atm}$, where T_{atm} is the temperature of the atmosphere, usually assumed to be roughly the same as the ambient temperature at ground level ($T_{atm} \sim 300$ K.[1] Thus, the atmosphere contributes a signal of ~ 28 K of noise per

[1]This assumption is justified by the fact that atmospheric pressure follows an exponential drop-off as a function of elevation. The majority of molecules are close to the telescope, and therefore close to the telescope's ambient temperature. This is particularly true for water molecules, which represent the most significant atmospheric absorber. The partial pressure of water is such that it follows a steeper exponential decline, causing it to be concentrated even closer to the ground. When looking at atmospheric-absorption plots (such as Figure 3.1), water lines can be identified by their large width (due to pressure broadening). Other atmospheric contaminants extend to lower-pressure regions of the atmosphere, resulting in sharper absorption lines.

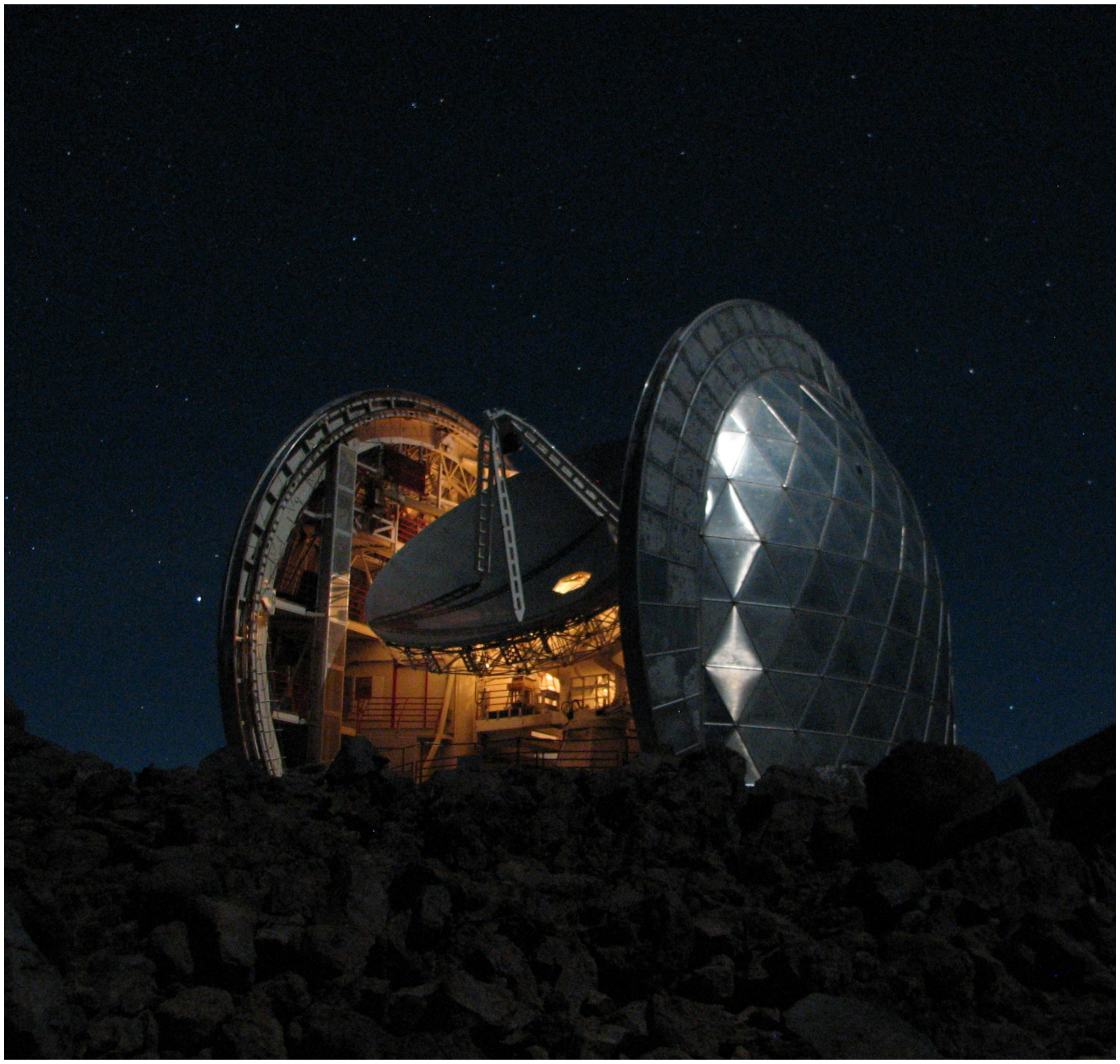


Figure 2.1: Caltech Submillimeter Observatory at night.

sideband, or $\sim$ 56 K total. This is quite large in comparison to the desired spectral lines, which typically have peaks of amplitude $\sim$ 0.1 - 10 K, or the continuum radiation, which is usually a few Kelvins.

To detect these small signals against the larger background, the CSO uses chopping subtraction, in which the source is observed for several seconds, and then an off-source position is observed for the same amount of time. As long as the length of each observation is much smaller than the timescale on which the atmospheric opacity changes, the average sky noise can be subtracted off, and the difference between these two signals represents the desired spectrum.

While the emission from the sky is viewed as "noise," and the source's emission is viewed as "signal," it is important to emphasize that both arise from thermal emission. Therefore, the difference between them is also a random quantity, and it is this randomness that determines the sensitivity of the observation. The RMS noise in the signal after the subtraction is described by the Dicke radiometer equation,

$$T_{RMS}^{SSB} = \frac{2T_{sys}}{\sqrt{\Delta f \, t_{on}}}, \tag{2.1}$$

where Δf represents the bandwidth of the observation (e.g., one spectral channel), T_{sys} represents the system noise temperature and t_{on} represents the on-source integration time.[2]

[2]Actually, a few intermediate steps are needed to arrive at the results shown in Equation 2.1. The Dicke radiometer equation describes the best possible performance that could be achieved by a device that takes an input signal, limits its frequency range to Δf (e.g., with a bandpass filter), measures the power with a square-law detector, and then averages that result over a time t_{on}. If we convert the measured power into an equivalent temperature, the uncertainty in that value must be at least

$$\sigma_T = \frac{T_{sys}}{\sqrt{\Delta f \, t_{on}}} \tag{2.2}$$

[Rohlfs and Wilson, 2003]. A spectrometer channel receiving input from a DSB receiver sums the input from two sidebands, increasing the noise by a factor of $\sqrt{2}$. The desired signal represents the difference of two such measurements taken in on-source and off-source positions: $T_{source} = T_{on} - T_{off}$. If we observe the "on" and "off" positions for equal time, each has an uncertainty equal to

$$\sigma_{T_{on}} = \sigma_{T_{off}} = \frac{\sqrt{2}T_{sys}}{\sqrt{\Delta f \, t_{on}}}. \tag{2.3}$$

Standard propagation-of-error techniques can be used to determine the uncertainty in T_{source}:

$$\sigma_{T_{source}} = \sqrt{2}\,\sigma_{T_{on}} = \frac{2T_{sys}}{\sqrt{\Delta f \, t_{on}}}. \tag{2.4}$$

This is the result shown in Equation 2.1.

2.1 Calibration at the CSO

Submillimeter spectra are calibrated in a fashion that allows the underlying source spectrum to be determined, independent of atmospheric conditions. Conceptually, the calibrated data can be viewed as the spectrum that would be seen by an ideal telescope placed above Earth's atmosphere, except with an additional amount of noise.

The CSO uses the chopper-wheel calibration method, as discussed in Kutner and Ulich [1981], Ulich and Haas [1976], and Peng [2002, Appendix A]. Before taking a spectrum, a calibration scan is calculated by comparing the spectrum of a hot load to that of a cold load. The loads are presumed to be perfect blackbodies over the frequencies of interest, so they can be characterized by a single, frequency-independent temperature. The hot load consists of an ambient-temperature absorber inserted into the beam of the receiver between the receiver and secondary mirror while the cold load is simply an empty patch of sky close to the astronomical source. As described in the following discussion, these observations can be used to determine two unknowns: the system's gain and the end-to-end noise level.

A perfect receiver would be equally sensitive to signals in the upper and lower sidebands. In reality, the receiver often has slightly different sensitivity at the two frequencies. We can model this difference by letting $G\left(f\right)$ represent the receiver's gain as a function of frequency. The gain represents the conversion factor between the spectrometer output and an astrophysically meaningful brightness temperature, T_A^*.[3]

Consider the output of a single channel of the spectrometer, represented as V.[4] When presented with the hot load at temperature T_{hot}, the spectrometer's response is

$$V_{hot} = G^{LSB}\left(T_{hot} + T_{Rx}^{LSB}\right) + G^{USB}\left(T_{hot} + T_{Rx}^{USB}\right), \tag{2.5}$$

[3]There are a variety of temperature scales used for submillimeter spectra. T_A^* is a common scale, as it can be derived directly from the chopper-wheel calibration. Values of T_A^* have been corrected for atmospheric losses, ohmic losses in the telescope, and rear (warm) spillover. They have *not* been corrected for forward (cold) spillover or the coupling between the telescope's beam pattern and the source. See Kutner and Ulich [1981] for additional details.

[4]The quantity V is traditionally considered as a voltage, but it could represent other forms of spectrometer output, such as digital signals from an ADC, counts on a CCD, etc., as long as they are proportional to the spectral power at that frequency.

where the gains

$$G^{LSB} = G\left(f_{LSB}\right) \text{ and}$$
$$G^{USB} = G\left(f_{USB}\right)$$

represent the LSB and USB gains at frequencies

$$f_{LSB} = f_{LO} - f_{IF} \text{ and}$$
$$f_{USB} = f_{LO} + f_{IF}.$$

The two terms $G^{LSB}\ T_{hot}$ and $G^{USB}\ T_{hot}$ simply represent the receiver's downconversion of power in the lower and upper sidebands. The receiver noise temperature T_{Rx} represents the noise added by the receiver during the downconversion. Conceptually, the receiver can be viewed as a perfect receiver, adding no noise to the signal, plus a noise source at the input with temperature T_{Rx}. The receiver noise temperature is not necessarily frequency-independent, so the equation includes different values for the upper and lower sidebands.

The response when looking at a blank patch of sky is a bit more complicated. While the majority of the signal comes from the primary beam, there are also contributions from spillover, scattering, and diffraction. Some of these effects produce rays that terminate on the sky, but not on the intended observation point while others result in rays that terminate on warm objects, such as the ground or the telescope structure. To model these effects, we introduce the warm and cold efficiencies, η_{warm} and η_{cold}. These quantities are defined such that the fraction of the beam that terminates on a warm load is $(1 - \eta_{warm})$ while the fraction reaching the sky is η_{warm}. Of that, η_{cold} forms the primary beam while the remaining $(1 - \eta_{cold})$ ends up on other sections of the sky.

The sky signal includes the following contributions for each side band:

$$\underbrace{(1 - \eta_{warm})\, T_{hot}}_{\text{Scattered into warm load}} + \underbrace{\eta_{warm}\,(1 - \eta_{cold})\, T_{sky}}_{\text{On sky but outside main beam}} + \underbrace{\eta_{warm}\eta_{cold} T_{sky}}_{\text{Main beam}} + \underbrace{T_{Rx}}_{\text{Receiver noise}} .$$

The spectrometer signal from the sky is therefore

$$V_{sky} = G^{LSB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}\left(1-\eta_{cold}\right)T_{sky}^{LSB} + \eta_{warm}\eta_{cold}T_{sky}^{LSB} + T_{Rx}^{LSB}\right]$$
$$+ G^{USB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}\left(1-\eta_{cold}\right)T_{sky}^{USB} + \eta_{warm}\eta_{cold}T_{sky}^{USB} + T_{Rx}^{USB}\right],$$

which can be rearranged to

$$V_{sky} = G^{LSB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}T_{sky}^{LSB} + T_{Rx}^{LSB}\right]$$
$$+ G^{USB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}T_{sky}^{USB} + T_{Rx}^{USB}\right]. \quad (2.6)$$

The sky temperature, T_{sky}, represents the effective temperature of the sky at the input to the receiver. The sky introduces noise into the spectrum with a brightness temperature of

$$T_{sky} = \left(1 - e^{-A\,\tau}\right)T_{atm} + \left(e^{-A\,\tau}\right)T_{space},$$

where τ represents the optical depth at the given frequency. A small amount of radiation can be traced to even an "empty" patch of sky, primarily due to the cosmic microwave background. However, the contribution is sufficiently small that it can be neglected: $T_{space} \approx 0$. As discussed in Footnote 1, the atmospheric temperature is usually approximated as the ambient temperature on the ground, $T_{atm} \approx T_{hot}$ [Peng, 2002]. The sky opacity can be significantly different for the USB and LSB, particularly if one sideband is near a strong absorption line from atmospheric molecules. Therefore, while T_{hot} may be treated as a constant in the equation for V_{hot}, T_{sky} must be defined separately for each sideband. The sky attenuation in the lower sideband is given by $e^{-A\,\tau_{LSB}}$ for the lower sideband and $e^{-A\,\tau_{USB}}$ for the upper sideband. We can define $\bar{\tau}$ and $\delta\tau$ such that $\tau^{LSB} = \bar{\tau} - \delta\tau$ and $\tau^{USB} = \bar{\tau} + \delta\tau$. Then the attenuation factors become $e^{-A(\bar{\tau}-\delta\tau)}$ and $e^{-A(\bar{\tau}+\delta\tau)}$ for the LSB and USB, respectively. To further simplify the notation, we let $\bar{\eta}_{sky} = e^{-A\,\bar{\tau}}$, $\eta_{sky}^{LSB} =^{+A\,\delta\tau}$, and $\eta_{sky}^{USB} =^{-A\,\delta\tau}$, giving

$$\begin{aligned} T_{sky}^{LSB} &= \left(1 - \bar{\eta}_{sky}\eta_{sky}^{LSB}\right)T_{hot} \text{ and} \\ T_{sky}^{USB} &= \left(1 - \bar{\eta}_{sky}\eta_{sky}^{USB}\right)T_{hot}. \end{aligned} \quad (2.7)$$

These values can be inserted into Equation 2.6 to give

$$V_{sky} = G^{LSB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}\left(1-\bar{\eta}_{sky}\eta_{sky}^{LSB}\right)T_{hot} + T_{Rx}^{LSB}\right] \\ + G^{USB}\left[(1-\eta_{warm})T_{hot} + \eta_{warm}\left(1-\bar{\eta}_{sky}\eta_{sky}^{USB}\right)T_{hot} + T_{Rx}^{USB}\right],$$

which can be simplified to

$$V_{sky} = G^{LSB}\left[\left(1-\eta_{warm}\bar{\eta}_{sky}\eta_{sky}^{LSB}\right)T_{hot} + T_{Rx}^{LSB}\right] \\ + G^{USB}\left[\left(1-\eta_{warm}\bar{\eta}_{sky}\eta_{sky}^{USB}\right)T_{hot} + T_{Rx}^{USB}\right]. \quad (2.8)$$

When the telescope is pointed at an astronomical source, the signal contains all the components present in V_{sky}, with an additional signal that can be attributed to the source:

$$V_{src} = V_{sky} + \eta_{src}\eta_{warm}\eta_{cold}\bar{\eta}_{sky}\left(G^{LSB}\eta_{sky}^{LSB}T_{src}^{LSB} + G^{USB}\eta_{sky}^{USB}T_{src}^{USB}\right). \quad (2.9)$$

The source-coupling efficiency, η_{src}, represents the convolution of the source structure with the telescope's main-beam pattern and accounts for effects such as beam dilution while T_{src} is the effective temperature of the source.[5]

The spectra created by the CSO are calibrated to the T_A^* scale using the following equation [Peng, 2002, Equation A.3]:

$$T_A^* = 2T_{hot}\frac{V_{src}-V_{sky}}{V_{hot}-V_{sky}}. \quad (2.10)$$

[5]The efficiencies defined here are comparable to those defined by other authors:

This work	Peng [2002]	Kutner and Ulich [1981]
η_{warm}	α	η_{fss}
η_{cold}	β	η_{rss}
η_{src}	γ	η_c

For a more detailed discussion of the physical interpretation of these efficiencies, see Kutner and Ulich [1981, Equations 5 and 9]. Also note that the efficiencies defined here are not precisely equal to those in Kutner and Ulich [1981], which include an additional efficiency, η_r, that represents losses due to resistive heating of the telescope. However, based on the results in their Table I, this efficiency appears to have a minimal impact with typical values $\eta_r \approx 1$. A thorough definition of T_{src} can be found in Ulich and Haas [1976, Equations 8 and 9], which implicitly defines the conceptually similar quantity T_E.

From Equation 2.9, it is easy to see that the numerator is equal to

$$V_{src} - V_{sky} = \eta_{src}\eta_{warm}\eta_{cold}\bar{\eta}_{sky}\left(G^{LSB}\eta_{sky}^{LSB}T_{src}^{LSB} + G^{USB}\eta_{sky}^{USB}T_{src}^{USB}\right).$$

The denominator can be found using Equations 2.5 and 2.8:

$$\begin{aligned} V_{hot} - V_{sky} = G^{LSB}&\left[T_{hot} + T_{Rx}^{LSB} - \left(1 - \eta_{warm}\bar{\eta}_{sky}\eta_{sky}^{LSB}\right)T_{hot} - T_{Rx}^{LSB}\right] \\ &+ G^{USB}\left[T_{hot} + T_{Rx}^{USB} - \left(1 - \eta_{warm}\bar{\eta}_{sky}\eta_{sky}^{USB}\right)T_{hot} - T_{Rx}^{USB}\right], \end{aligned}$$

which simplifies to

$$V_{hot} - V_{sky} = \eta_{warm}\bar{\eta}_{sky}\left(\eta_{sky}^{LSB}G^{LSB} + \eta_{sky}^{USB}G^{USB}\right)T_{hot}.$$

Using these intermediate results, we find that

$$T_A^* = 2T_{hot}\frac{\eta_{src}\eta_{warm}\eta_{cold}\bar{\eta}_{sky}\left(G^{LSB}\eta_{sky}^{LSB}T_{src}^{LSB} + G^{USB}\eta_{sky}^{USB}T_{src}^{USB}\right)}{\eta_{warm}\bar{\eta}_{sky}\left(\eta_{sky}^{LSB}G^{LSB} + \eta_{sky}^{USB}G^{USB}\right)T_{hot}}.$$

Cancelling like factors gives the rather simple result

$$T_A^* = \frac{2\eta_{src}\eta_{cold}\left(G^{LSB}\eta_{sky}^{LSB}T_{src}^{LSB} + G^{USB}\eta_{sky}^{USB}T_{src}^{USB}\right)}{\left(\eta_{sky}^{LSB}G^{LSB} + \eta_{sky}^{USB}G^{USB}\right)}.$$

This can be further simplified by renormalizing the gains G,

$$\begin{aligned} \bar{G}^{LSB} &= \frac{2G^{LSB}\eta_{sky}^{LSB}}{\eta_{sky}^{LSB}G^{LSB} + \eta_{sky}^{USB}G^{USB}} \text{ and} \\ \bar{G}^{USB} &= \frac{2G^{USB}\eta_{sky}^{USB}}{\eta_{sky}^{LSB}G^{LSB} + \eta_{sky}^{USB}G^{USB}}, \end{aligned} \tag{2.11}$$

allowing us to write

$$T_A^* = \eta_{src}\eta_{cold}\left(\bar{G}^{LSB}T_{src}^{LSB} + \bar{G}^{USB}T_{src}^{USB}\right).$$

The calibration process imposes an important constraint on these gains, namely

$$\bar{G}^{LSB} + \bar{G}^{USB} = 2, \tag{2.12}$$

as can be seen by inspection from Equation 2.11. While this might seem an insignificant result, it allows an important simplification of the problem. If we define γ such that

$$\bar{G}^{LSB} = 1 - \gamma, \tag{2.13a}$$

we immediately find that

$$\bar{G}^{USB} = 1 + \gamma. \tag{2.13b}$$

Thus, the constraint from Equation 2.12 allows us to model the effect of sideband imbalances using only a single parameter, γ:

$$T_A^* = \eta_{src}\eta_{cold}\left[(1-\gamma)\, T_{src}^{LSB} + (1+\gamma)\, T_{src}^{USB}\right]. \tag{2.14}$$

In this equation, all of the sideband-dependent effects $\left(G^{LSB}, G^{USB}, \eta_{sky}^{LSB}, \text{and } \eta_{sky}^{USB}\right)$ have been rolled into γ.

2.2 Measuring System Temperature

The calibration represented by Equation 2.10 produces a value of T_A^* that is independent of the flat portion of the sky attenuation $(\bar{\eta}_{sky})$, the warm scattering and spillover (η_{warm}), and the receiver noise $\left(T_{Rx}^{LSB} \text{ and } T_{Rx}^{USB}\right)$, leaving only the effects of cold scattering and spillover (η_{cold}) and the coupling efficiency between the telescope and the source (η_{src}). In principle, η_{cold} can be determined to yield T_R^* [Kutner and Ulich, 1981], but η_{src} cannot be determined without knowing detailed information about the source's structure.

While the calibration removes the *average* flux introduced by each of the background sources, it cannot remove the randomness introduced by their presence. Both the signal and background sources are created by thermal noise; therefore, T_A^* is also a random quantity with RMS noise levels given by Equation 2.1. In order to calculate T_{RMS}, we need to know T_{sys}, the temperature of a noise source that would add an equivalent amount of noise to an ideal receiving system. In this case, the "receiving system" consists not only of the receiver and the telescope, but also the atmosphere. A perfect telescope (no spillover or scattering) used with a noiseless receiver and placed above the Earth's atmosphere, but exposed to a noise source of temperature T_{sys}, would be equivalent to our real-world system.

Considering such an idealized telescope provides an easy way to estimate the value of T_{sys}. Imagine making a Y-factor measurement, in which V_H is measured by filling the telescope's beam with a perfect absorber at temperature T_H and comparing it to the receiver's response when looking at a blank patch of sky, V_{space}. Taking the ratio of those two values gives the Y factor:

$$\begin{aligned}
Y_{sky} &= \frac{V_H}{V_{space}} \\
&= \frac{\left(G_{LSB}T_H^{LSB} + G_{USB}T_H^{USB}\right) + \left(G_{LSB}T_{sys}^{LSB} + G_{USB}T_{sys}^{USB}\right)}{\left(G_{LSB}T_{space}^{LSB} + G_{USB}T_{space}^{USB}\right) + \left(G_{LSB}T_{sys}^{LSB} + G_{USB}T_{sys}^{USB}\right)} \\
&= \frac{\left(G_{LSB}T_H^{LSB} + G_{USB}T_H^{USB}\right) + \left(G_{LSB}T_{sys}^{LSB} + G_{USB}T_{sys}^{USB}\right)}{\left(G_{LSB}T_{sys}^{LSB} + G_{USB}T_{sys}^{USB}\right)},
\end{aligned}$$

where we have again approximated $T_{space} \approx 0$.

At this point, we introduce several other approximations. First, we assume that the sideband gains of the entire system are equal so that $G_{LSB} \approx G_{USB}$.[6] Likewise, we assume that the system temperature is roughly equal for the two sidebands so that $T_{sys}^{LSB} \approx T_{sys}^{USB}$, which allows us to replace either of the sideband-specific values with the quantity T_{sys}^{DSB}, representing the system noise injected from a single sideband. Finally, we assume that the IF bandwidth is small compared to the RF observing frequency so that $T_{hot}^{LSB} \approx T_{hot}^{USB}$. We can then write the previous equation as

$$\begin{aligned}
Y_{sky} &\approx \frac{2GT_H + 2GT_{sys}^{DSB}}{2GT_{sys}^{DSB}} \\
&= \frac{T_H + T_{sys}^{DSB}}{T_{sys}^{DSB}}.
\end{aligned}$$

[6]Ignoring the distinction between the sidebands does not work near atmospheric absorption lines, but should be a reasonable approximation elsewhere.

This equation can easily be solved for T_{sys}^{DSB} to give

$$T_{sys}^{DSB} = \frac{T_H}{Y-1}. \tag{2.15}$$

Since moving the entire telescope above the atmosphere is not a convenient option, we could achieve the same effect by finding a calibration source that has temperature T_H, completely fills the telescope's beam, and is above the atmosphere. By selecting T_H appropriately, we can make this measurement even easier. If the absorber were above the atmosphere, the atmosphere would attenuate its signal by a factor of $e^{-A\tau}$, but it would also inject noise at the temperature T_{atm} into the signal:

$$V_H = G\left[e^{-A\tau}T_H + \left(1 - e^{-A\tau}\right)T_{atm}\right]. \tag{2.16}$$

If we choose $T_H = T_{atm}$, then it doesn't matter whether the absorber is above the atmosphere or below it, allowing us to perform the calibration by inserting an absorber into the beam at the observatory. Therefore, calibration at the CSO is performed rather simply by inserting an absorber into the beam between the receiver and the secondary to measure V_H, which can be compared to V_{sky}, obtained by moving the telescope slightly off-source and performing a short integration.

For later use, it is also worth mentioning that each spectrum taken at the CSO also has an associated calibration scan consisting of

$$C = \frac{V_{hot} - V_{sky}}{V_{sky}} = Y_{sky} - 1 \tag{2.17}$$

[Peng, 2002]. The calibration scan can be used to form the denominator of Equation 2.10. The scan also provides a channel-by-channel method of determining the system temperature, which can be quite useful in evaluating end-to-end performance.

Chapter 3

Ultra-Wideband Submillimeter Receivers

3.1 CSO Facility Receivers

Ground-based submillimeter observing is complicated by the fact that Earth's atmospheric is only partially transparent at submillimeter frequencies. Figure 3.1 shows the atmosphere transmission as a function of frequency for the CSO; the individual lines correspond to different weather conditions, as parameterized by the amount of precipitable water vapor (PWV) in the atmosphere.

The effects of the atmosphere are twofold. Since the atmosphere is not 100% transmissive, the desired astrophysical signal is attenuated on its way to the telescope. In addition, the sky emits its own thermal noise at submillimeter frequencies, in effect creating a "glowing" haze between the telescope and the source. As the opacity worsens, the amount of thermal noise attributable to the sky increases.

Atmospheric interference is particularly strong at frequencies that correspond to absorption lines of atmospheric molecules, particularly water. These lines eliminate our ability to see through the atmosphere at certain frequencies, leaving several "windows" of moderate transparency in between that can be used for ground-based astronomy. Not surprisingly, heterodyne receivers for ground-based telescopes are typically designed to operate within these windows of semi-transparency.

The bottom of Figure 3.1 shows the coverage bands of five new receivers that have been designed to replace the current facility receivers at the CSO. Although the suite of

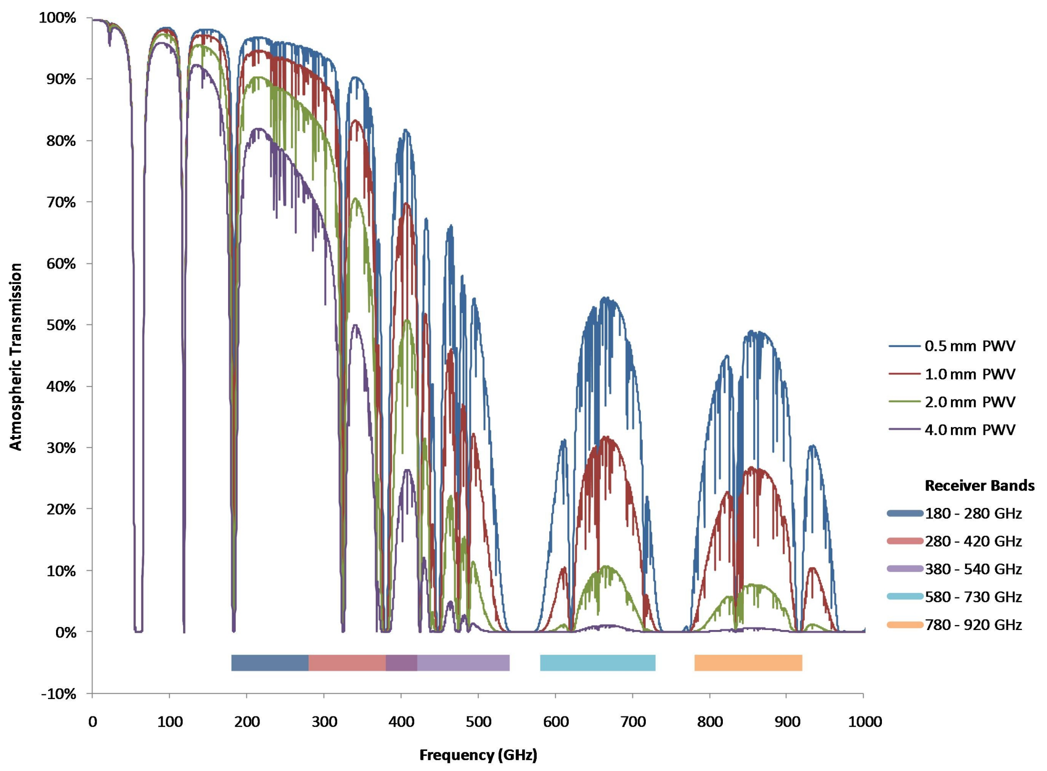


Figure 3.1: Atmospheric transmission at the summit of Mauna Kea. The lines correspond to different amounts of precipitable water vapor (PWV) in the air, and the colored boxes along the bottom show the frequency bands covered by the planned suite of ultra-wideband receivers.

new receivers is still under construction, a 345-GHz prototype using the new mixer design has been installed at the CSO [Kooi et al., 2007]. The receiver used for the observations in this thesis is another prototype instrument, known as Z-Rex, that operates in the 230-GHz atmospheric window [Rice et al., 2003]. (See Figure 3.2.) All of these receivers benefit from new design methods and technologies that allow them to complete line surveys in significantly less time than previous receivers.

3.2 Optimizing Heterodyne Receivers for Line Surveys

As discussed in Chapter 1, submillimeter heterodyne receivers are ideal for performing molecular-line surveys of star-forming regions; however, because such surveys require so much observing time, there is strong incentive to improve the receiver hardware to make surveys more efficient.

3.2.1 IF Bandwidth

The area in which we have made the most significant gains is the instantaneous bandwidth of the receivers. The instantaneous bandwidth can be viewed as the width of the bandpass filter shown between the mixer and the spectrometer in Figure 1.7. It represents the size of the spectrum that can be captured in a single observation, so that doubling this bandwidth halves the number of observations needed to cover a given frequency range. Because this figure represents the bandwidth of the signal at the intermediate-frequency (IF) port of the mixer, it is typically referred to as the receiver's IF bandwidth.

The heterodyne systems used for the original survey of Orion [Sutton et al., 1985, Blake et al., 1986] had an IF bandwidth of approximately 500 MHz, and the facility receivers currently installed at the CSO offer an IF bandwidth of 1 GHz. The new generation of receivers that our group has designed offer IF bandwidths of at least 4GHz, and the prototype receiver used for much of the work in this thesis has an IF bandwidth of 12 GHz.

Consider a hypothetical survey covering the frequency range from 220 - 268 GHz. Figure 3.3 demonstrates the dramatic improvement in surveying speed that would be possible with a broader IF bandwidth. The left axis gives the value of the LO frequency while

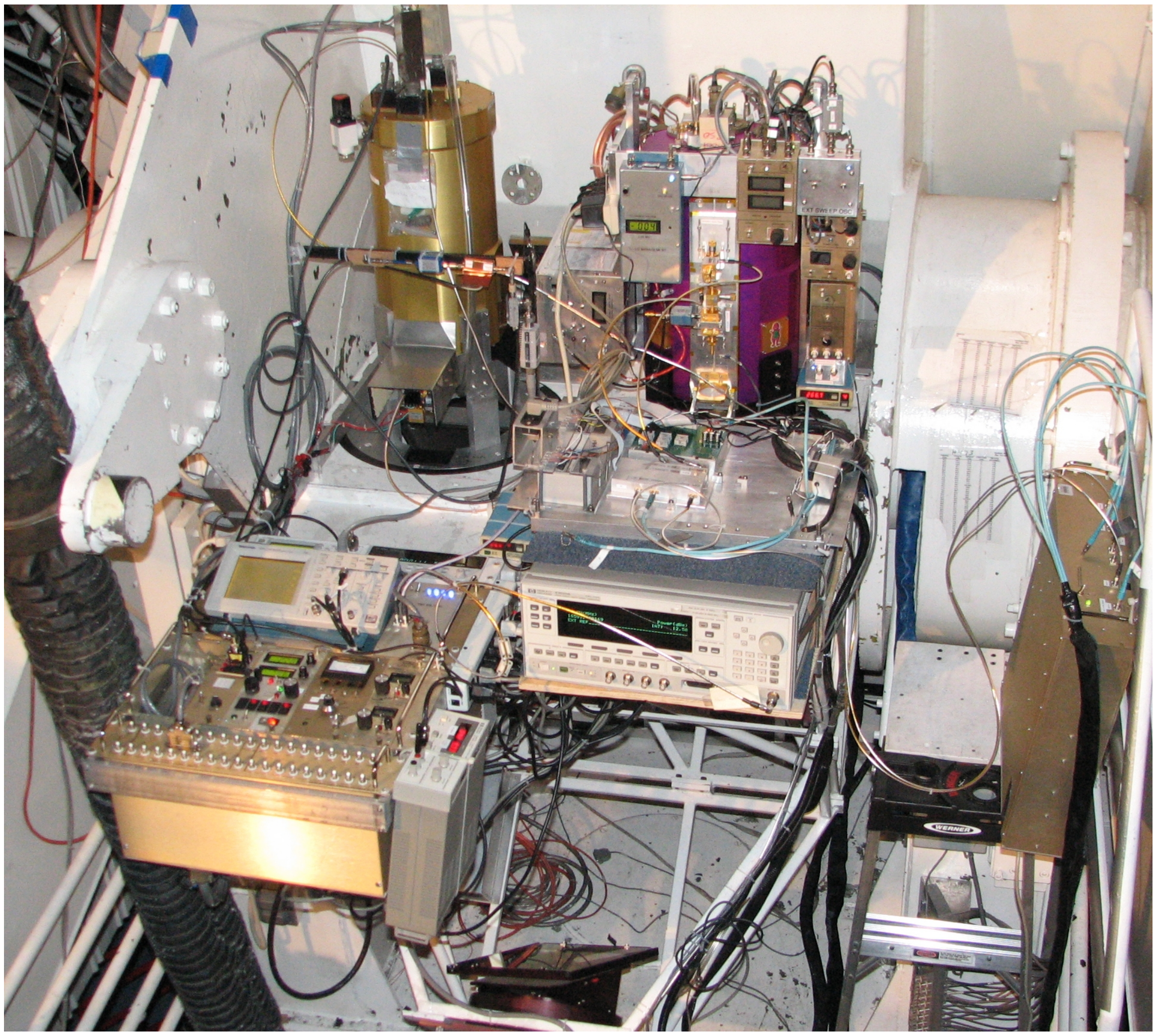


Figure 3.2: Prototype wideband receivers mounted on the telescope at the CSO. The receiver mounted on the left port is Z-Rex, the 230-GHz prototype while the receiver on the right is the 345-GHz prototype. The synthesized LO, described in Section 3.6, can be seen at the bottom of Z-Rex (underneath an aluminum cryogen shield), and a more traditional Gunn-based LO can be seen on the front side of the 345-GHz prototype.

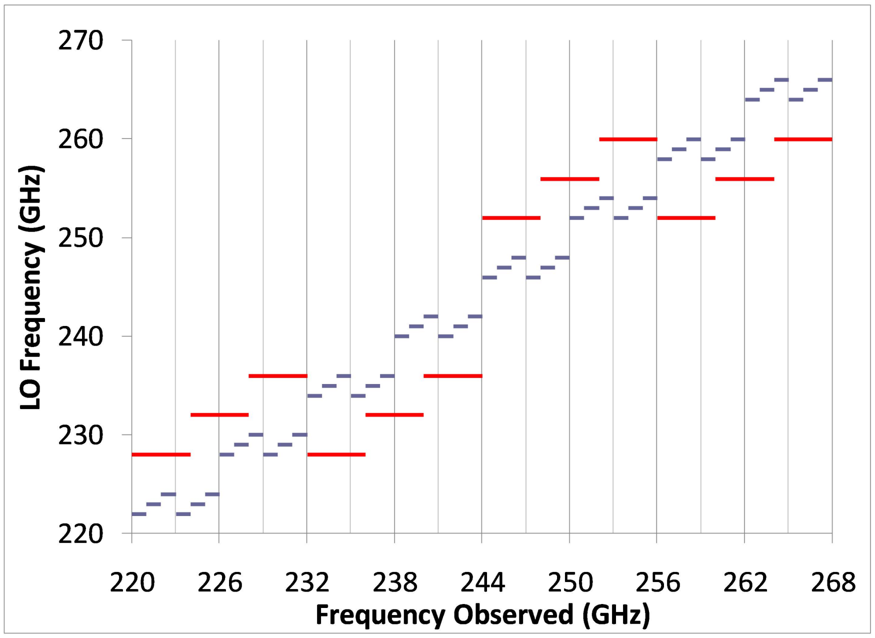


Figure 3.3: Increasing the IF bandwidth of the receiver dramatically reduces the number of observations required to survey a given frequency range. Red lines correspond to a double-sideband receiver with a 4-GHz IF bandwidth while the blue lines correspond to a 1-GHz IF bandwidth.

the bottom axis indicates the RF frequencies that would be observed at each LO setting. The red lines show the frequencies that would be covered by a double-sideband receiver with a 4-GHz IF bandwidth while the blue lines correspond to a 1-GHz IF bandwidth. There are two lines for each LO frequency to represent the upper and lower sidebands. With the larger bandwidth, the entire range could be covered with 6 LO settings while the narrower-band receiver would require 24 individual observations. Assuming the receivers had the same noise temperatures, the integration time per observation would be the same in both cases; therefore, increasing the IF bandwidth from 1 GHz to 4 GHz would lead to an immediate 75% reduction in observing time.

The receiver design can be pushed to even broader bandwidths. We have successfully used a prototype receiver with a 12-GHz IF bandwidth, which could cover the entire frequency range of Figure 3.3 in only two LO settings. At the moment, however, the extra bandwidth cannot be used for the type of surveys discussed here, as the CSO’s high-resolution spectrometer is limited to 4 GHz.

3.2.2 Receiver Sensitivity

Another way to increase survey efficiency is to improve the sensitivity of the receiver by lowering its noise temperature. However, the receivers are approaching a quantum limit that defines the minimum noise levels,[1] and the majority of the noise for ground-based receivers comes from the atmosphere; therefore, this is a path of diminishing returns. Instead, the goal for the new generation of receivers has been to maintain sensitivities similar to previous, narrower-band receivers. As long as we can achieve the broader bandwidths without sacrificing sensitivity, we can realize the impressive gains previously described.

3.2.3 Frequency Agility

Another way to increase survey efficiency is to improve the frequency agility of the receivers. Every time the LO frequency is adjusted, the receivers must be manually tuned to optimize their performance. Minimizing the amount of tuning can significantly decrease the overhead associated with performing a line survey. Successful sideband deconvolution requires multiple observations of each frequency; in our surveys, we typically observe each frequency at $\sim$ 6 - 10 different LO settings. If 6 LO settings are needed to cover the entire band, $\sim$ 50 settings are needed to generate enough data for successful sideband deconvolution. Thus, minimizing the tuning time at each setting becomes even more important.

We have improved the tuning efficiency with two different methods. Advanced modeling techniques have allowed us to make a tunerless mixer block. This significantly decreases the number of adjustments that need to be made at each tuning, leaving the SIS bias voltage and the magnetic-field current as the only mixer-related settings to be optimized. Moreover, our experience during observing runs has indicated that these settings can be tuned easily. It is often possible to proceed through several adjacent LO settings with only minimal adjustment, and when retuning is is necessary, it can usually be achieved quite quickly.

We have also experimented with using a more agile LO source. Previous receivers have relied on Gunn oscillators to generate signals in the range $\sim$ 70 - 110 GHz, which are then

[1]Quantum mechanics imposes a minimum noise temperature of $T_{min} \sim h\nu/k$ on any device that preserves phase information, including SIS mixers [Phillips and Woody, 1982]. For further discussion, see Caves [1982] and Clerk et al. [2010].

fed to a passive multiplier to generate the desired frequency. The Gunn oscillator requires its own manual tuning, which can be quite time consuming. In an attempt to resolve this problem, we have investigated active multiplier chains, which rely on a commercial microwave synthesizer to generate input frequencies in the range $\sim$ 13 - 18 GHz. When using an active multiplier chain, changing the LO frequency is as easy as setting the microwave synthesizer to a new frequency, which can even be done remotely over a network. As discussed later in this chapter, however, the active multiplier chain introduces its own complications, and during observing runs, we often resorted to using the Gunn, despite its slower tuning speed.

3.3 Z-Rex

The receiver used for the line surveys in this thesis is the ultra-wideband, 230-GHz prototype known as Z-Rex [Rice et al., 2003]. An overview of the end-to-end system is shown in Figure 3.4. Photons arriving at the telescope are focused through a beam splitter and into a cryostat. The beam splitter is almost entirely transmissive, with just enough reflectivity to couple in a small fraction of the power from an LO source. Using a beam splitter to combine the astronomical beam and the local oscillator allows both signals to be sent through the same window into the cryostat. The beam splitter does couple a small amount of excess noise into the system, but the resulting simplification of the overall design was considered worthwhile, particularly for a prototype receiver.

Inside the cryostat, the beam is focused through cooled transmissive optics into a circular waveguide horn attached to the mixer block, as shown in Figure 3.5. After a short length of waveguide, the signal is picked up by a specially designed wideband probe and fed into a superconducting junction diode, which mixes the astronomical signal with the LO and outputs the downconverted signal at its IF port. From there, the signal undergoes initial amplification within the cryostat before being passed to additional warm IF amplifiers.

Outside the cryostat, the signal is routed to an IF processor and an array of spectrometers. Because of the very large IF bandwidth of 12 GHz offered by this receiver, it is necessary to combine several spectrometers in parallel to use as much of the spectral bandwidth

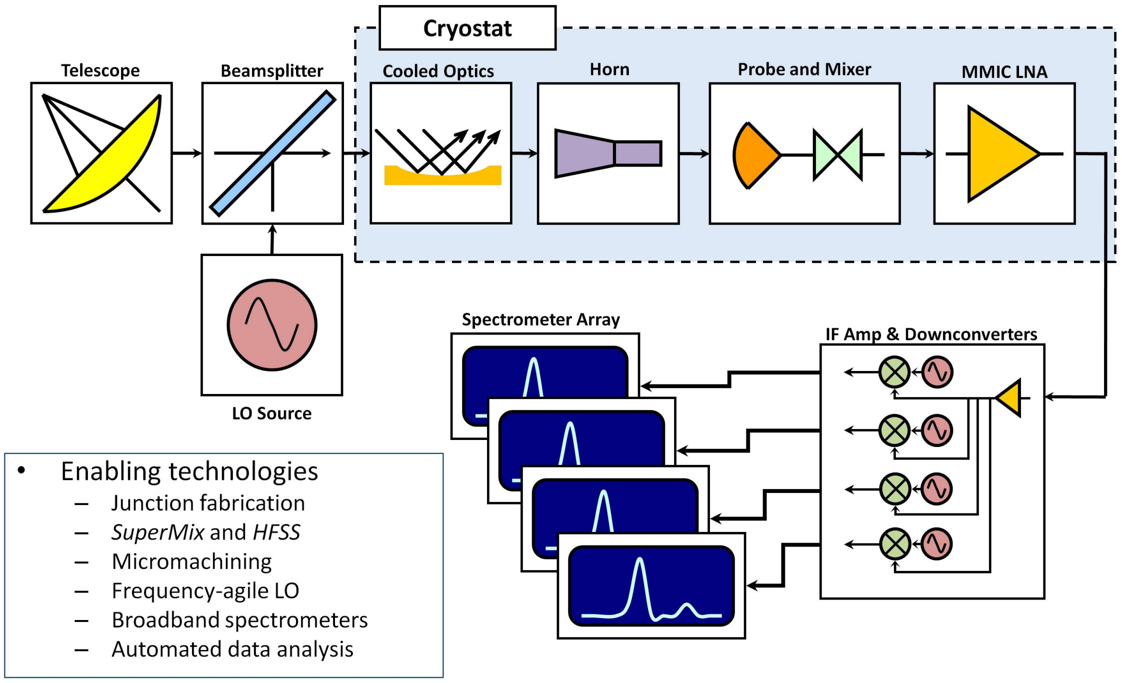


Figure 3.4: Overview of Z-Rex.

as possible. The IF processor is responsible for breaking up the output of the receiver into smaller bands, each at the correct frequency and power level for the individual spectrometers.

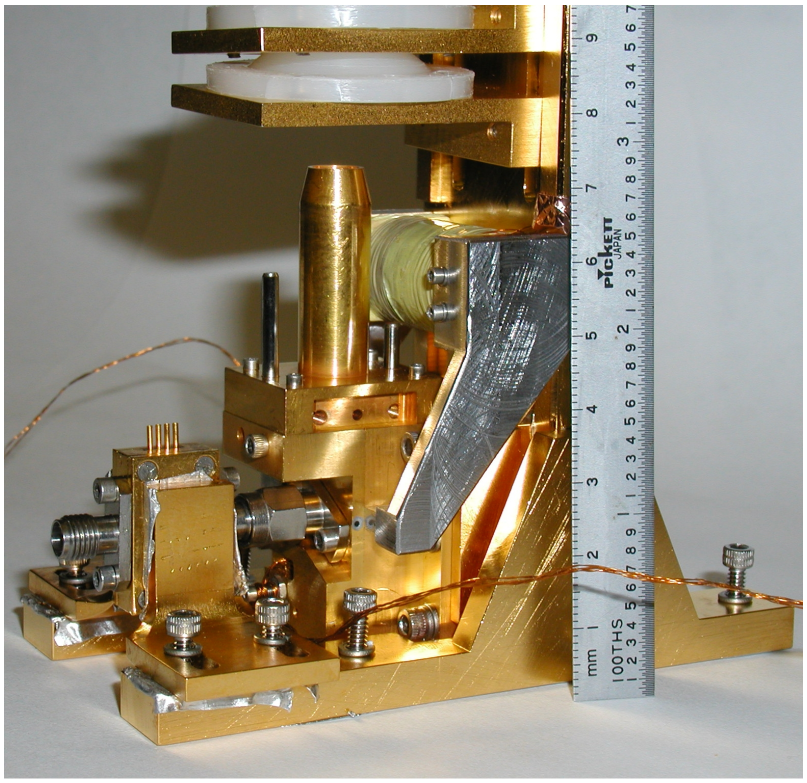
The individual components of Z-Rex are discussed in detail in the following sections.

3.3.1 Mixer Chip and Waveguide Block

The heart of the receiver is the mixer chip, shown in Figure 3.6. The actual mixing is performed by a superconducting tunnel diode, constructed from an superconductor-insulator-superconductor (SIS) junction.

The first element of the mixer chip is the broadband radial-stub probe, which couples the RF signal from the waveguide into the mixer chip. (See Figure 3.8.) Extensive simulation and scale-model testing were used to ensure that the probe would work effectively across the relatively broad RF bandwidth of the receiver (180 - 300 GHz).[2]

[2]Although designed to operate from 180 - 300 GHz, the prototype receiver currently covers a somewhat smaller range. Instead of constructing a new waveguide horn, we used an existing horn. At the low end, the design of that horn limits useful observations to $\gtrsim$225 GHz while the range of the LO often sets the upper frequency limit.



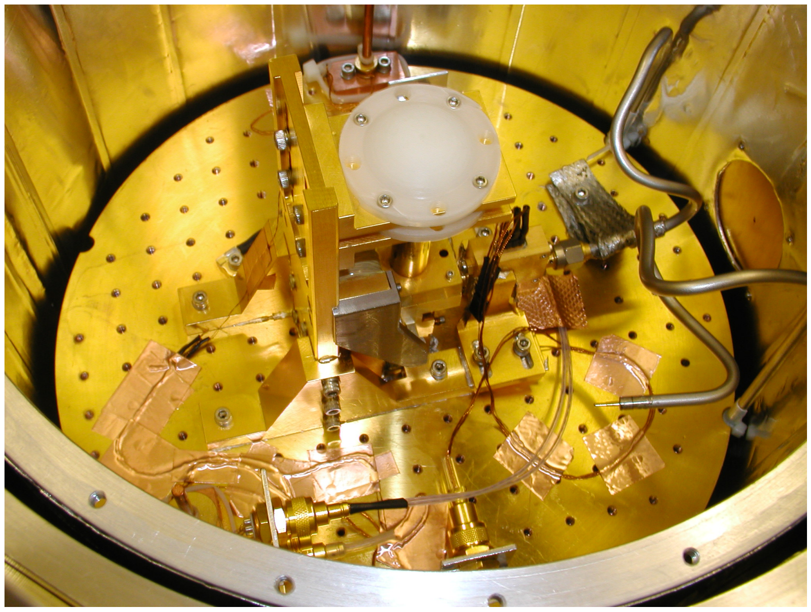


Figure 3.5: Mixer-block assembly (top) and 4-K cold plate inside the cryostat (bottom). (Images courtesy of Frank Rice.)

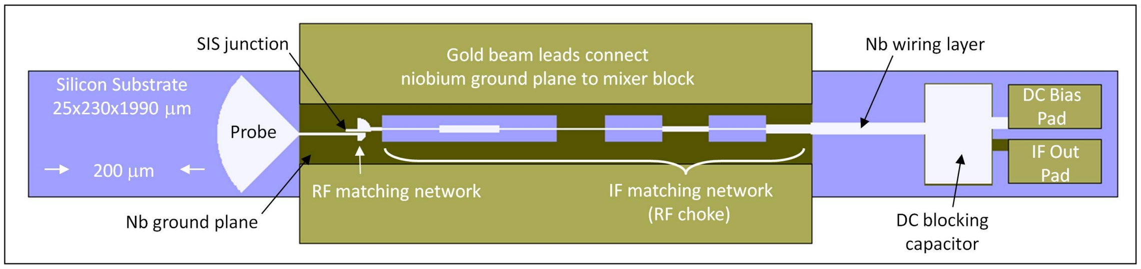

SIS junction
Gold beam leads connect niobium ground plane to mixer block
Nb wiring layer
Silicon Substrate 25x230x1990 µm
Probe
200 µm
DC Bias Pad
IF Out Pad
RF matching network
IF matching network (RF choke)
Nb ground plane
DC blocking capacitor

Figure 3.6: Layout of 230-GHz wideband mixer chip for Z-Rex. Adapted from Rice et al. [2003].

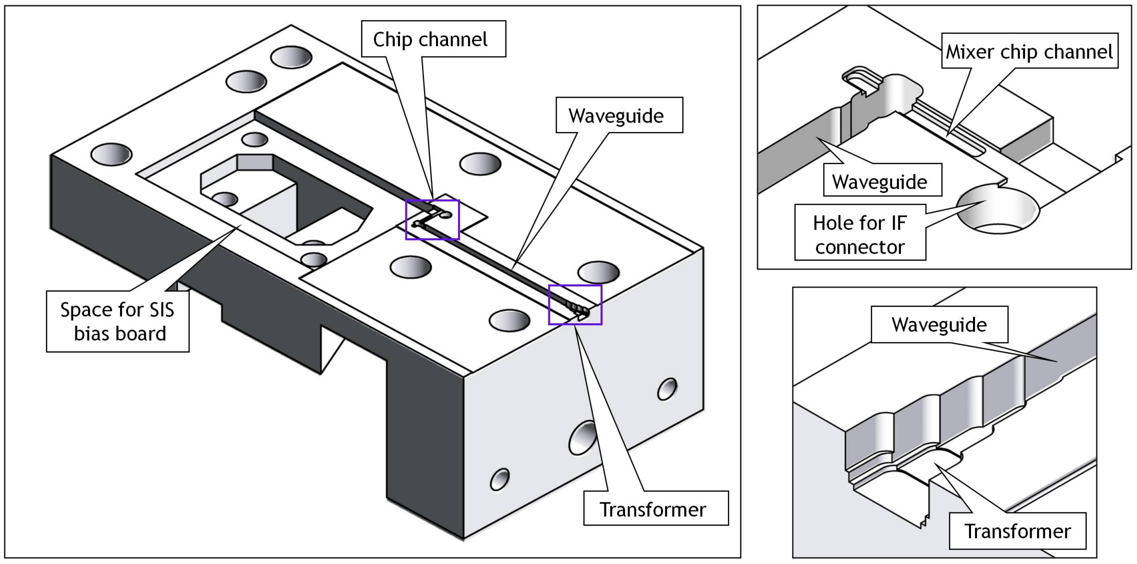


Figure 3.7: CAD rendering of top half of mixer block. Adapted from Rice et al. [2003].

An RF matching network transforms impedances appropriately so that the RF signal from the waveguide probe can be efficiently delivered to the SIS junction, where it is down-converted to the IF frequency. An IF matching network then transforms the impedance of the junction so that it matches well with the following low-noise amplifier. The IF matching network also serves as an RF choke to keep RF power from leaking away from the junction, which increases the receiver's sensitivity.

The mixer chip sits in a waveguide block on the 4-K stage of the cryostat. The front side of the mixer block was designed to interface with an existing horn; a transformer at the input to the mixer block matches the circular horn to the rectangular waveguide used within the block. The RF signal (including both sky and LO signals) travels down a short length of waveguide, past a small tuning step machined into the block, and into the probe of the mixer chip, as shown in Figure 3.8. Two soft-iron pole pieces concentrate magnetic fields on the SIS junction to minimize Josephson currents [Wengler, 1992]. A glass bead, part of the interface to a 2.9-mm coaxial connector, sits immediately next to the IF output pad of the mixer chip. A small area behind the mixer chip is reserved for the DC bias board, as shown in Figure 3.7. To ensure that slight unevenness in the mating surfaces of the block would not interfere with the tight fit of the waveguide walls, a shallow depression was cut into the top half the block, as can be seen in Figure 3.10.

A significant amount of simulation went into designing and optimizing the chip and mixer block. Much of the circuit modeling was done in *SuperMix*, a custom-built software library designed for the simulation and optimization of superconducting submillimeter receivers [Ward et al., 1999]. *SuperMix* was particularly important for modeling the behavior of the SIS junction and determining the circuit parameters that would optimize the receiver's overall performance [Rice et al., 2003]. The Ansoft *HFSS* 3-D electromagnetic simulator was used extensively to model aspects of the mixer chip circuitry; the results could then be integrated into the *SuperMix* model.

HFSS also played a key role in optimizing the design of the waveguide probe and testing the expected performance of the probe within the context of the mixer block. Simulations included real-world effects, such as the machining fillets at the end of the waveguide and on either side of the tuning step, and calculations were performed to ensure that the proposed design could tolerate typical machining errors in the final block. Because these effects were all considered in the simulation, we were able to design a high-performance mixer without requiring any movable tuning elements within the waveguide.

As shown in Figure 3.5, the mixer block was designed so that the coaxial connector on its top could be mated directly to the low-noise amplifier (LNA). The coaxial connector includes the glass bead mentioned above, and the tight connection provides a controlled path between the mixer chip and the amplifier. By using a well-understood IF path, the input impedance of the LNA could be included in simulations of the chip's performance and the on-chip matching network could be designed to maximize the IF bandwidth of the receiver.

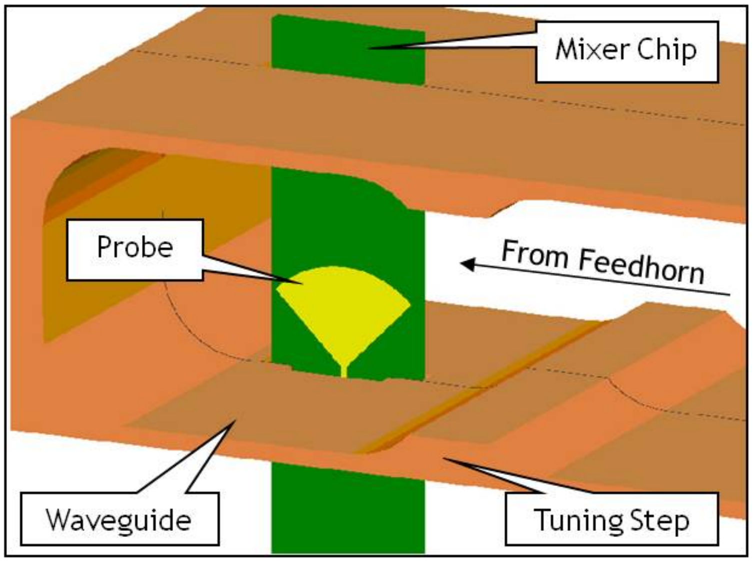


Figure 3.8: Probe and mixer chip inserted into the waveguide block. Adapted from Rice et al. [2003].

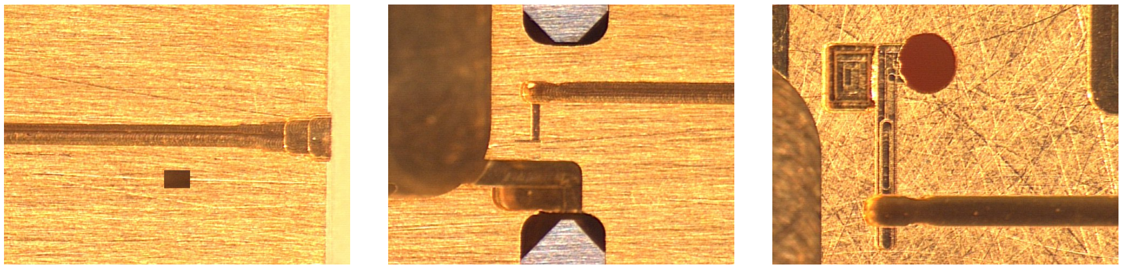
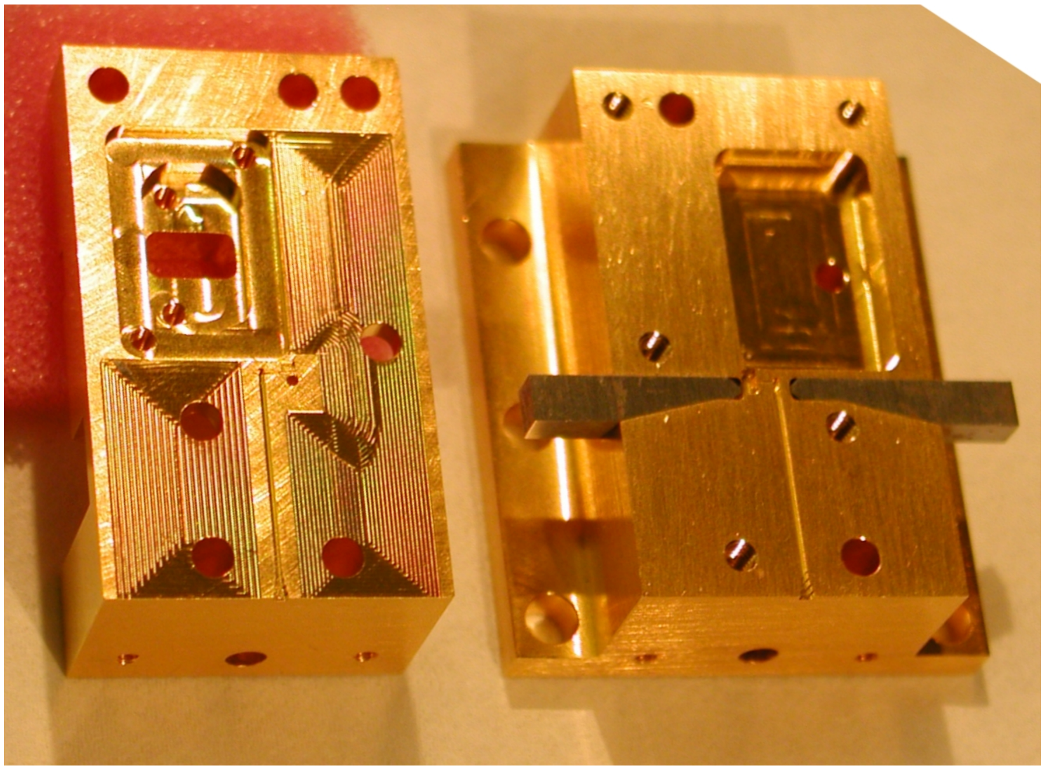


Figure 3.9: Microscope image showing the input transformer (left), the proximity of the pole pieces to the mixer chip (center), and the channel for the mixer chip (right). (The rectangle that can be seen near the center of some photos is an imaging artifact.)



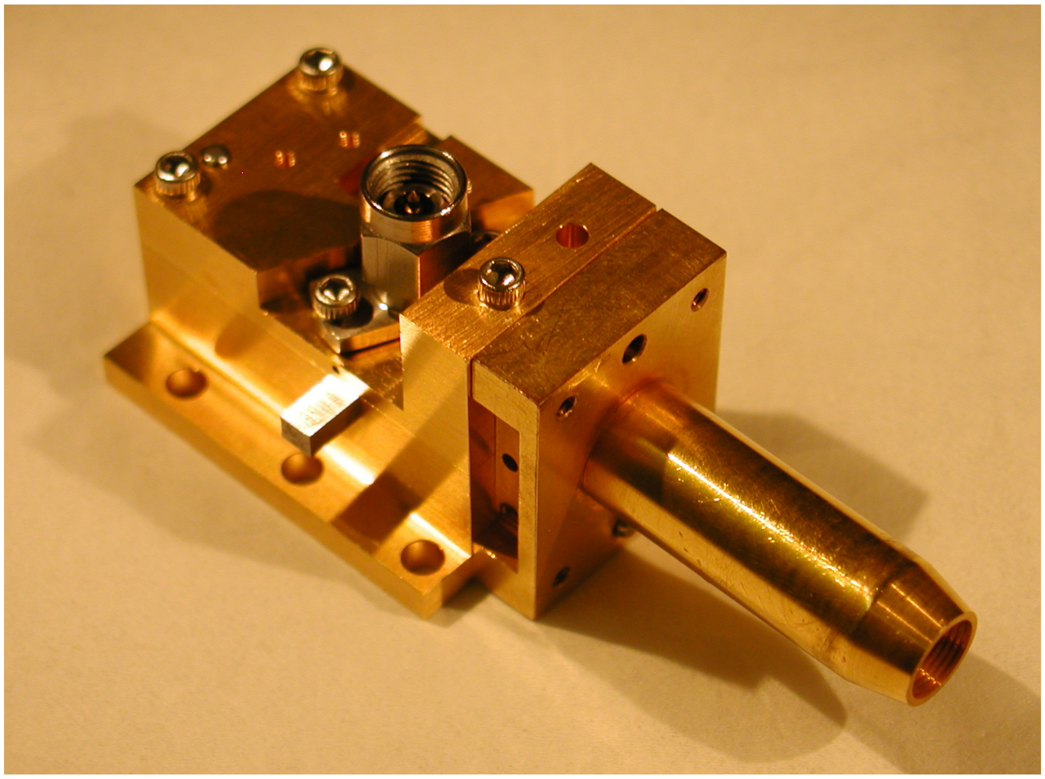


Figure 3.10: Split block (top) and mixer block assembled with horn and coaxial connector for LNA (bottom). The "spiral" pattern in the top half of the block (shown at the left of the top photo) corresponds to a shallow depression that was cut to ensure that critical surfaces along the edges of the waveguide could mate tightly.

3.4 Full-Bandwidth Spectroscopy

Originally, Z-Rex was intended to be a “z-machine,” designed to allow rapid redshift determinations of the ultra-luminous galaxies that had been discovered using submillimeter imaging arrays (e.g., Blain et al., 2002). Because of the relatively large positional error bars produced by the imaging arrays, and because the sources were highly obscured by dust, determining the redshift of these objects using follow-up observations at other frequencies proved to be difficult. In contrast, searching for molecular emission lines in the submillimeter, using some of the same telescopes responsible for the original discoveries, looked much more promising. Since the redshift of these sources was unknown, broad swaths of frequency would need to be searched; however, the limited IF bandwidth of existing receivers meant that such a search would be highly time-consuming, particularly since each individual observation would require a long integration time to detect the faint lines. The large IF bandwidth of Z-Rex was intended to solve this problem by offering 12 GHz of frequency coverage per sideband, for a total instantaneous bandwidth of 24 GHz. Using such a receiver, the molecular lines from a given source could be found using only a few different LO settings.

Because the distant galaxies are not resolved by the CSO beam, linewidths are expected to be $\sim$ 300 km/s [Blain et al., 2002]. At 300 GHz, a velocity of 1 km/s corresponds to a Doppler shift of 1 MHz; thus, Z-Rex, operating in the 230-GHz atmospheric window, would see line widths $\gtrsim$ 300 MHz. Given the inherently broad linewidths, a relatively low-resolution spectrometer would be well suited to this task. Consequently, the CSO installed four of the WASP2 analog autocorrelation spectrometers developed by Andy Harris at the University of Maryland [Harris and Zmuidzinas, 2001]. Each WASP2 unit delivers 3.5 GHz of spectral coverage over 128 channels, offering a frequency resolution of approximately 33 MHz per channel. By running four units in parallel, we cover the entire 12-GHz IF bandwidth of Z-Rex. To distribute the signal to the four units, an IF processor divides the Z-Rex output signal into four bands, downconverting and amplifying each band to deliver the appropriate frequency range and power level to the WASP2 unit.

3.4.1 Wideband IF Processor

The IF processor (see Figure 3.15) accomplishes three basic functions: separation into four bands, downconversion to the WASP2 input frequencies, and amplification. Each of the functions is relatively straightforward in principle, but the large bandwidth creates special design concerns. Each WASP2 unit accepts a 3.5-GHz wide input, centered on 6 GHz. The total bandwidth available from the four WASP2 units is 14 GHz, which slightly exceeds the intended IF bandwidth of the receiver. Z-Rex was designed to provide a 12-GHz output from 6 - 18 GHz; however, rather than waste some of the WASP2 bandwidth, we chose to design a downconverter to accept a slightly broader input, thereby providing an easy mechanism for taking advantage of any extra output range from the receiver. The input spectrum is thus divided into four slightly overlapping bands: 5.75 - 9.75 GHz, 9.25 - 13.25 GHz, 12.75 - 16.75 GHz, and 16.25 - 20.25 GHz. Each band is downconverted to a 4 - 8 GHz output. If the WASP2 spectrometers have exactly the quoted 3.5-GHz bandwidth, these ranges will precisely tile the range from 6 - 20 GHz. Any extra input bandwidth offered by the WASP2s allows the spectra to overlap lightly. The four branches are summarized in Table 3.2.

The downconversion also requires appropriate component selection. The choice to extend the frequency coverage to 20 GHz (rather than 18 GHz) requires high-performance components. The 5.75 - 9.75 GHz band provides additional challenges, as the RF input overlaps the desired IF output. Discussions with vendors indicated that a triple-balanced mixer would be appropriate for that branch, as such mixers are particularly good at minimizing RF-to-IF leakage. Further investigation revealed that triple-balanced mixers would be needed for all four bands, as the 4 - 8 GHz output is beyond the range of double-balanced mixers designed for similar RF frequencies.

The LO sources we selected are free-running DROs (dielectric resonator oscillators), which are affordable and easy to use. The free-running DROs do not possess the same frequency stability as phase-locked models, but because the IF processor is driving a relatively low-resolution spectrometer, the free-running performance is more than adequate. Each DRO is specified to deliver at least +13 dBm of power, which is consistent with the mixers' requirements.

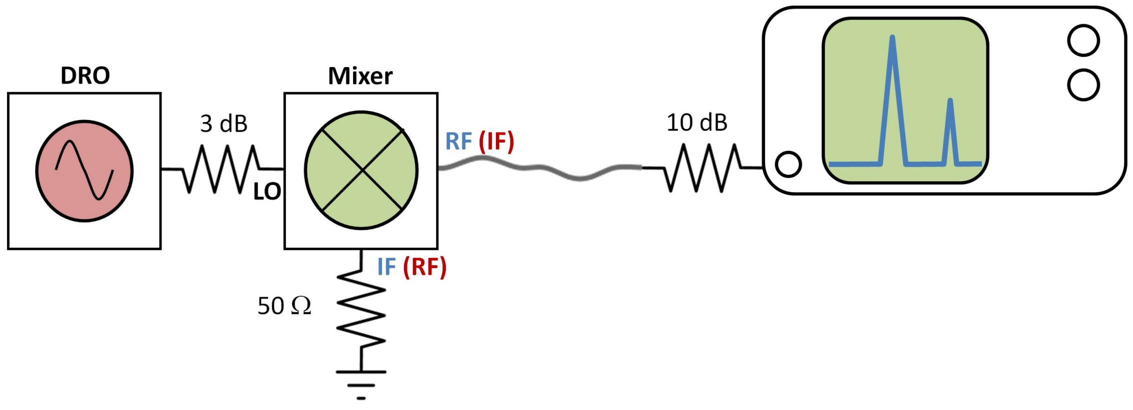


Figure 3.11: Experimental set up for measuring port-to-port mixer isolation. The DRO (left) was connected to the LO port of the mixer via a 3-dB attenuator. To measure LO-to-RF isolation, the IF port was loaded with a 50-Ω terminator, and the RF port was connected to a spectrum analyzer via a short (∼ 12″) cable with a 10-dB attenuator at the end (blue labels). LO-to-IF isolation was measured by terminating the RF port and connecting the IF port to the spectrum analyzer (red labels). Reference measurements were taken by removing the mixer and measuring the output of the DRO through the remaining elements.

Triple-balanced mixers are readily available commercially, but they typically have worse port-to-port isolation than their double-balanced counterparts. Since our design includes LO sources within the input band, isolation is also an important criterion for us. To determine whether the mixers would work for our needs, we directly tested the isolation using the experimental setup shown in Figure 3.11. We paired each mixer with all four LO sources to determine its LO-to-RF and LO-to-IF isolation at the frequencies of interest, generating the results given in Table 3.1.

To measure LO-to-RF isolation, we connected the DRO to the LO port of the mixer via a 3-dB attenuator to address potential impedance mismatch. We then terminated the IF port with a broadband terminator and measured the LO leakage using a spectrum analyzer connected to the RF port. Similarly, we measured the LO-to-IF isolation by terminating the RF port and connecting the IF port to the spectrum analyzer. The results show significant variations between different units of the same mixer model. These tests also indicated that it was particularly important to drive certain mixers at their optimum pumping levels to ensure optimal isolation.

	LO-to-RF Isolation (dB)				LO-to-IF Isolation (dB)			
	DRO Frequency (GHz)				DRO Frequency (GHz)			
Mixer	8.75	12.25	13.75	17.25	8.75	12.25	13.75	17.25
M3006L 0247A	26.5	18.8	22.9	35.6	23.8	27.3	23.1	30.1
M3006L 0247B	29.2	20.6	25.0	37.0	27.4	21.2	21.3	37.5
M3006L 0247C	22.5	29.0	23.2	28.1	27.5	16.6	15.6	32.8
M3006L 0247D	26.7	21.6	25.6	32.4	31.0	26.9	35.1	30.6
M3006L 0247E	25.3	22.4	27.4	32.5	38.8	20.9	28.2	42.7
M2H-0220LA	27.7	45.1	44.7	30.1	22.3	32.0	29.1	27.3

Table 3.1: Mixer isolation, measured using the setup shown in Figure 3.11. The M3006L 0247X lines represent different units from Advanced Microwave while the final line corresponds to Marki Microwave M2H-0220LA.

3.4.1.1 Prototype and Spur Analysis

To determine whether the in-band LO sources would generate spurious spikes in the output signal, a two-branch prototype of the IF processor was built, as shown in Figure 3.12. In the prototype, the input signal, shown by the green arrow, was amplified, passed through a coupler, and then split by a four-way power divider. Two of the outputs from the divider were terminated, while the other two outputs fed into 4-GHz-wide bandpass filters. Each of these bands was then downconverted into the 4 - 8 GHz band and amplified. All parts were the same as the ones described in the final IF processor design in Section 3.4.1.3.

These two branches were chosen because they offered particularly strong tests of the mixers' isolation; the RF input for the 5.75 - 9.75 GHz branch overlapped with the IF output, and the 8.75-GHz LO of the other branch fell near the edge of the input bandwidth for the final 4 - 8 GHz amplifier. The best mixers were picked for each branch based on the results found in Table 3.1. The 5.75 - 9.75 GHz branch also allowed us to study the high-frequency behavior of the bandpass filters, which was also a source of concern.

As shown in Figure 3.13, the filters had low loss in the desired passband and strong losses on either side. However, the the high-frequency rejection did not extend as far as expected. The filters tended to admit signals at frequencies about an octave above the lower edge of the bandpass. Some had spikes of higher-frequency transmission, with loss elsewhere while others admitted the majority of signals above a certain frequency. The

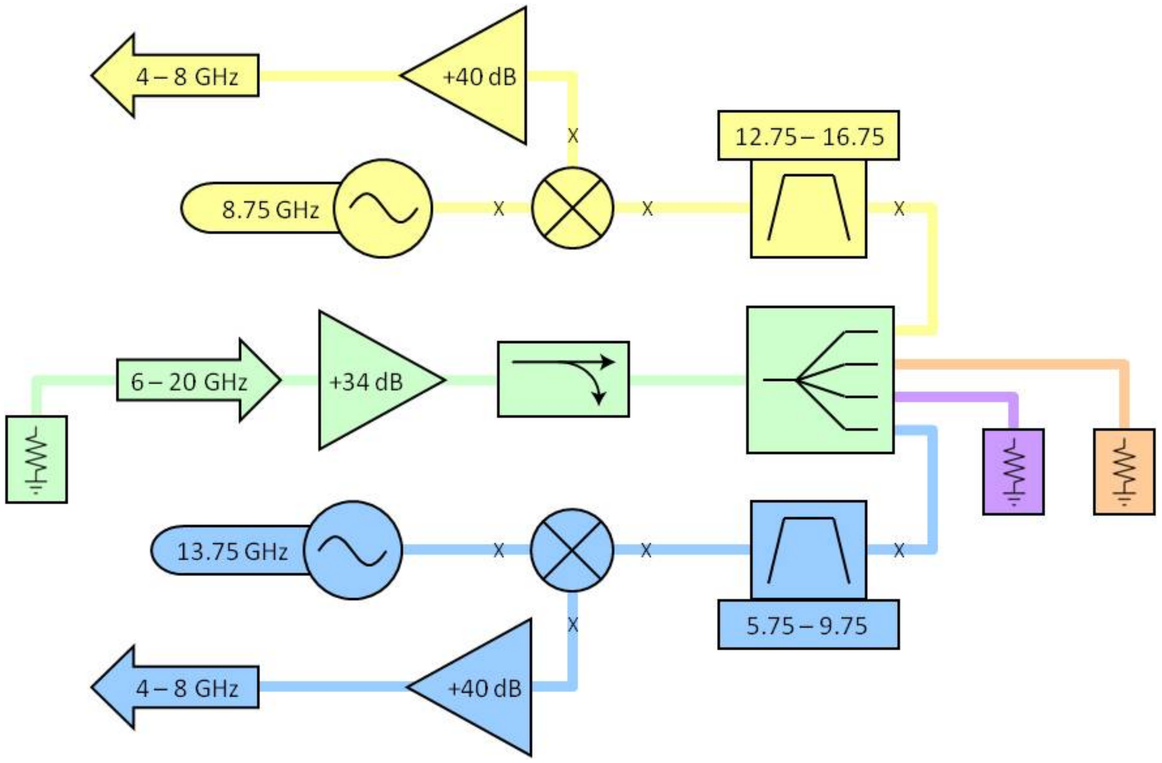


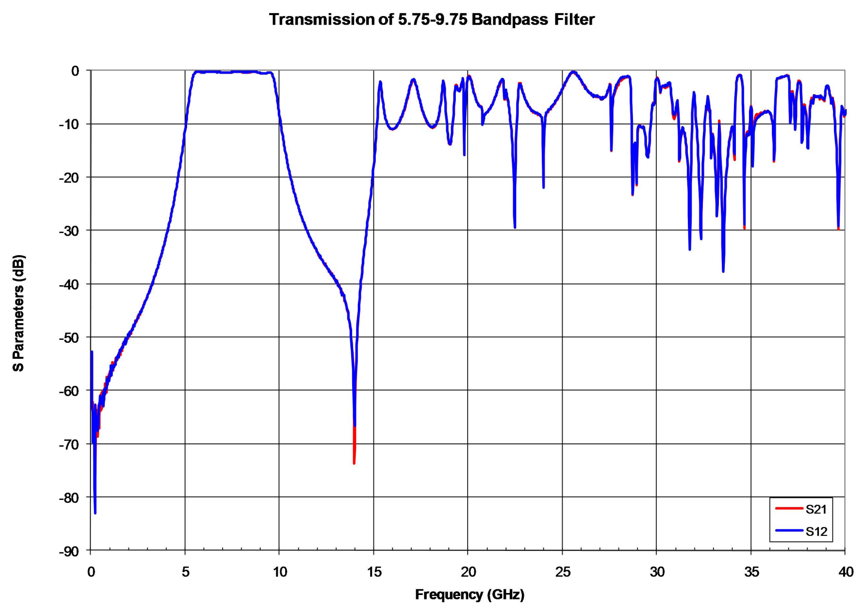
Figure 3.12: Simple two-branch prototype of the IF processor. X's indicate 3-dB attenuators.

5.75 - 9.75 GHz branch demonstrated the latter behavior, providing a good test of whether this high-frequency leakage would allow unintended interactions between bands.

The entire range of the spectrum analyzer, from 0 - 26.5 GHz, was searched for peaks; particular attention was given to 5 GHz and 22.5 GHz since mixing between the fundamental frequencies of the LO sources would be expected to yield peaks at those frequencies. Since the LOs themselves were the sources of interest, the input to the broadband amplifier was terminated, as was the output of whichever line was not connected to the spectrum analyzer.

For the 5.75 - 9.75 GHz line, peaks were found at 3.75 GHz and 13.75 GHz; no peaks were found at 5 or 22.5 GHz. The 13.75-GHz peak's power and frequency were consistent with the LO leaking through to the IF. The 3.75-GHz peak could best be explained by the second harmonic of the 8.75-GHz LO mixing with the fundamental of the 13.75-GHz LO $(2 \times 8.75 - 13.75 = 3.75 \text{ GHz})$.

For the 12.75 - 16.75 GHz branch, peaks were seen at 5, 8.75, 17.50, and 26.25 GHz, but not at 22.5 GHz. The peak at 5 GHz represented the anticipated interaction between



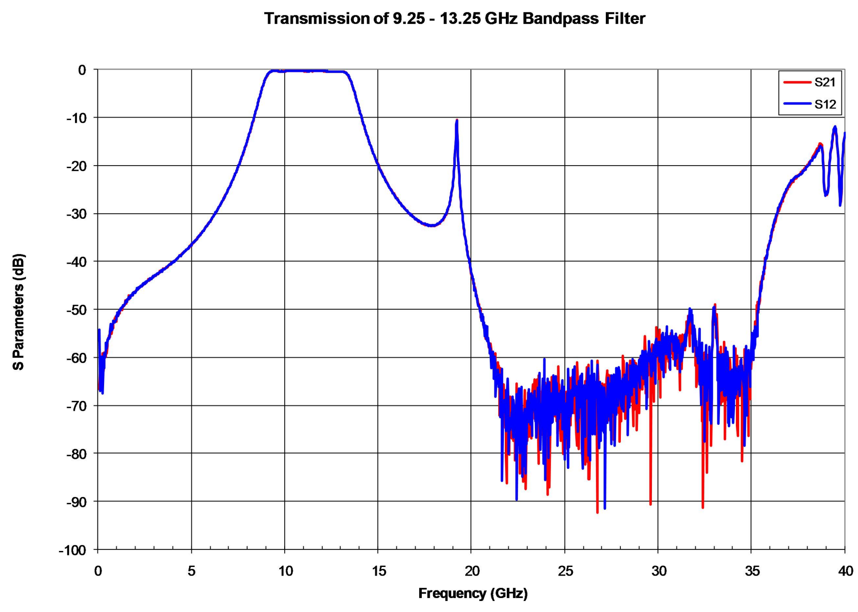


Figure 3.13: Measured responses of bandpass filters.

the fundamentals of the LO sources, and the 8.75-GHz peak was consistent with LO-to-IF leakage. The 17.50- and 26.25-GHz lines represented the second and third harmonics of the LO frequency; presumably, they were due to frequency multiplication in the mixer and/or amplifier. To determine which element was causing the multiplication, the amplifier was removed, and the spectrum analyzer was connected to the 3-dB attenuator on the IF port of the mixer. The 17.50-GHz peak dropped by only a few dB, implying that the mixer was at least partially responsible for the frequency doubling. In contrast, the 26.25-GHz peak dropped by $\sim$ 20 dB, indicating that it was most likely caused by frequency multiplication within the amplifier. This can happen when a particularly strong line saturates the amplifier, forcing it into a nonlinear regime.

Further tests were done to try to rule out other sources of interaction (such as LO frequencies leaking through the ground plane or DC wiring), but all tests were consistent with the interpretations given above. In the case of the 3.75-GHz peak, it was possible to trace the hypothesized 17.5-GHz line through much of the system. It was found at the RF port of the mixer on the 12.75 - 16.75 GHz branch, supporting the idea that it was caused by LO-to-RF leakage. The same line could be found at the output of the power splitter sending signals into the 5.75 - 9.75 GHz branch. From there, the signal disappeared below the noise floor of the spectrum analyzer.

Experiments with the prototype system yielded several important insights for moving forward on the final design. It demonstrated that interactions between LOs could generate spurs and that higher-order interactions would need to be considered as well. These interactions were not symmetric; the 5-GHz peak could be detected in the 12.75 - 16.75 GHz branch, but not in the 5.75 - 9.75 GHz branch, presumably because the two LO signals encountered different losses in the bandpass filters as they worked through the system. Finally, the frequency multiplication of the 8.75-GHz LO indicated that the output amplifiers could be saturated by strong out-of-band spurs; adding a broadband 4 - 8 GHz bandpass filter would resolve this problem.

To better understand the nature of the interactions between the LO sources, we developed a *Mathematica* program to calculate the expected mixing products. It considers all pairwise combinations of fundamental LO frequencies and harmonics (up to a specified order), and calculates the sum and difference frequencies generated by the two signals. In addition, it calculates the attenuation encountered by each signal as it works its way upstream in one branch and then back down through a different branch using laboratory measurements of the bandpass filters' performance taken with a 40-GHz network analyzer. Whenever a mixing product is found that falls within the output band of the IF processor, the program reports the frequency of the spur and the expected attenuation of the underlying signal.

The program accurately predicts the spurs that were observed with the prototype, and predicts that the 3.75-GHz spur should be significantly stronger in the 5.75 - 9.75 GHz line, as was seen. It also finds other potential spurs that were not identified with the prototype, but attaches higher attenuation values to them.

In addition to modeling the existing filters, the program also allowed us to study the effects of adding new filters. We investigated whether the high-frequency leakage of the bandpass filters was contributing to the observed spurs. The program demonstrated that adding low-pass filters to several of the lines would solve the problems seen with the prototype. By ordering additional low-pass filters that matched those used in the *Mathematica* program, we were able to eliminate the spurs.

3.4.1.2 Prototype Linearity

The linearity of the prototype system was also tested by connecting the input to a signal generator and driving the system at different power levels. At the time, the only signal generator available was limited to signals in the 10 - 18 GHz range, so only the 12.75 - 16.75 GHz branch was tested. As shown in Figure 3.14, the prototype system offered a conversion gain of $\sim$ 48 - 51 dB for input powers $\lesssim$ 45 dBm, corresponding to output powers of approximately +3 dBm. This is sufficient to drive the WASP2 spectrometers, which have a nominal input power of -10 dBm, and even leave some overhead to allow for cable losses between the IF processor and the spectrometers.

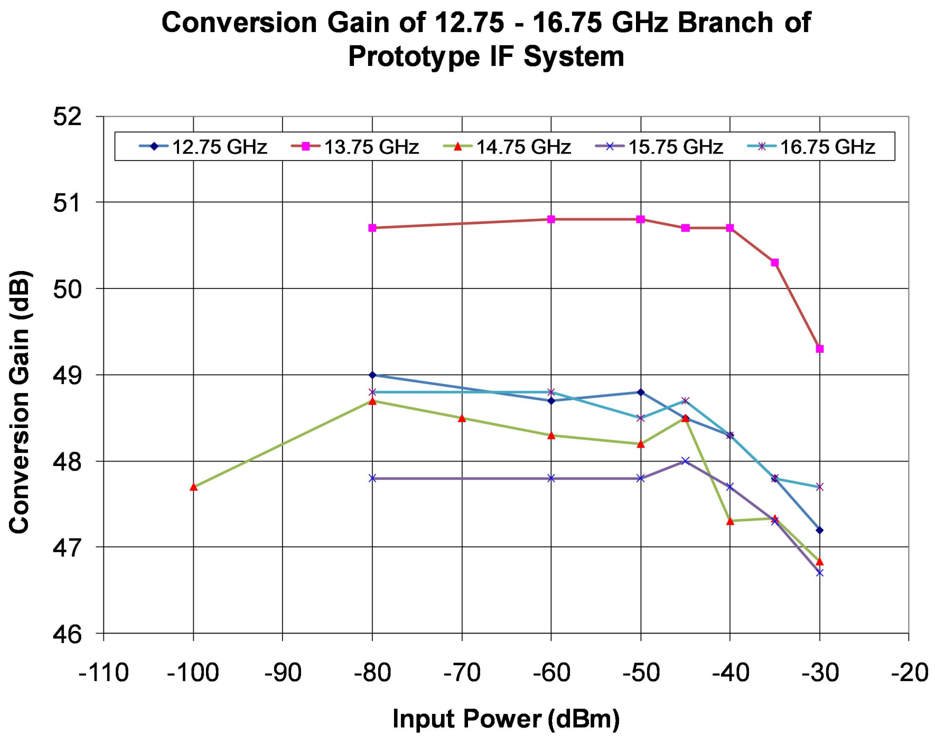


Figure 3.14: Conversion gain of prototype IF processor.

Spectral Range (GHz)	LO Frequency (GHz)	Sideband
5.75 – 9.75	13.75	Lower
9.25 – 13.25	17.25	Lower
12.75 – 16.75	8.75	Upper
16.25 – 20.25	12.25	Upper

Table 3.2: Key properties of the IF downconverter.

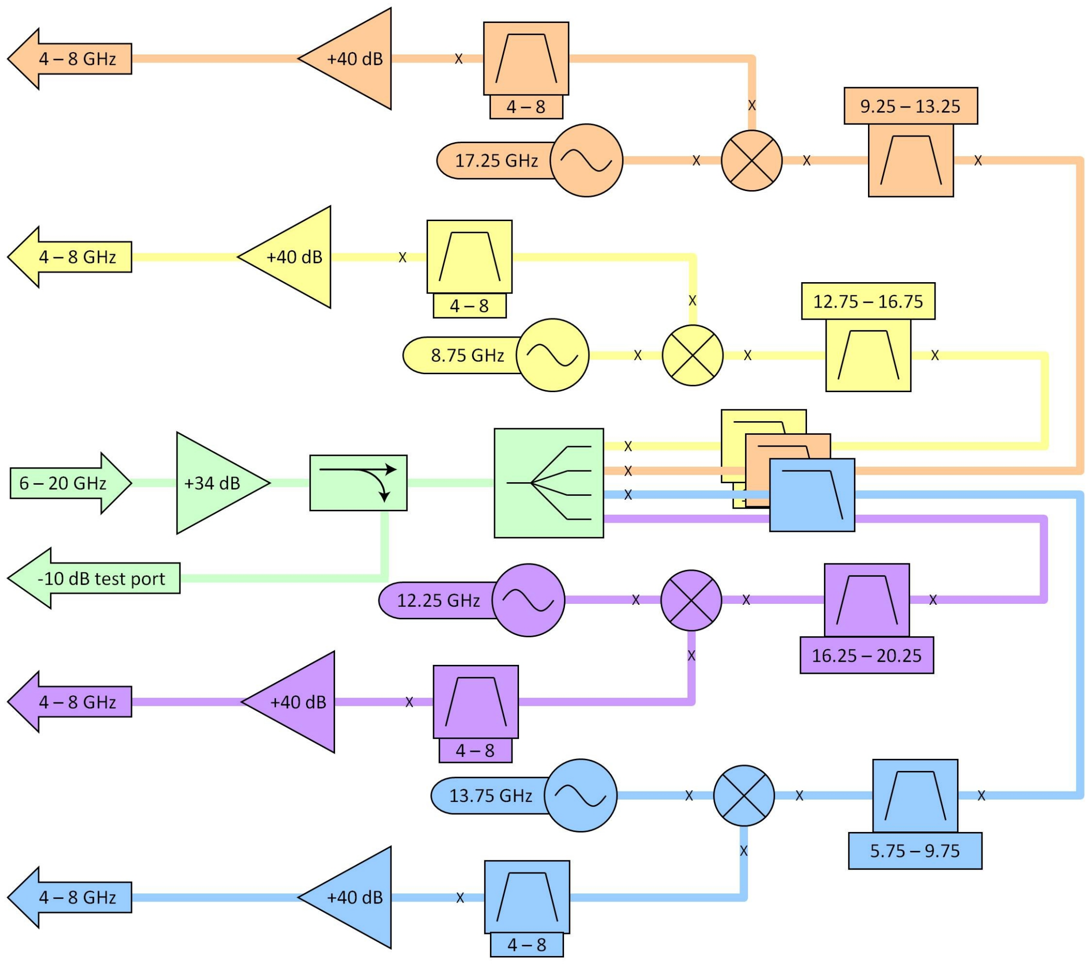


Figure 3.15: Schematic of IF downconverter; attenuators indicated by X's.

3.4.1.3 Final Design

The basic design for the IF downconverter is relatively simple, as shown in Figure 3.15. The wideband 6 - 20 GHz IF signal from Z-Rex is immediately boosted by a broadband amplifier. A small fraction of the signal is picked off by a 10-dB coupler to create a full-bandwidth test signal while the rest is split into one of four branches by a power divider. A bandpass filter selects a 4-GHz chunk of the spectrum, which is then downconverted to the appropriate input frequency for the WASP2 spectrometers. A final bandpass filter limits the output to 4 - 8 GHz, and another amplifier boosts the signal once more before it is returned to the front panel, where it can be connected to the WASP2 input. The extra low-pass filters needed to prevent crosstalk between branches are located on the output of the power divider.

Attenuators (typically 3 dB) are located between components expected to have a poor input match to help minimize reflections. Where possible, components are connected directly together or via SMA adapters. Cables are only used at the inputs and outputs of the box and to make one "fold" in each branch to conserve space; in these locations, hand-formable cables are used. To provide good thermal and mechanical stability, the entire assembly is mounted to an aluminum plate. The completed IF processor is shown in Figure 3.16, and a parts list is provided in Table C.1.

The IF processor has been modified since its construction by converting one of the lines into a straight-through 4 - 8 GHz branch. No downconversion occurs in that line, but it does provide the filters and amplifiers needed to produce an appropriate signal at the output. The modified version of the downconverter is shown in Figure 3.17. The excised parts are available, and the IF processor can easily be restored to its previous state when needed.

3.5 High-Resolution Spectroscopy

Despite its original intent, Z-Rex has rarely been used for redshift determinations. By the time it was fully deployed at the CSO, a large sample of redshifts had been determined using other means by Chapman et al. [2003]. Moreover, newly available millimeter-wave

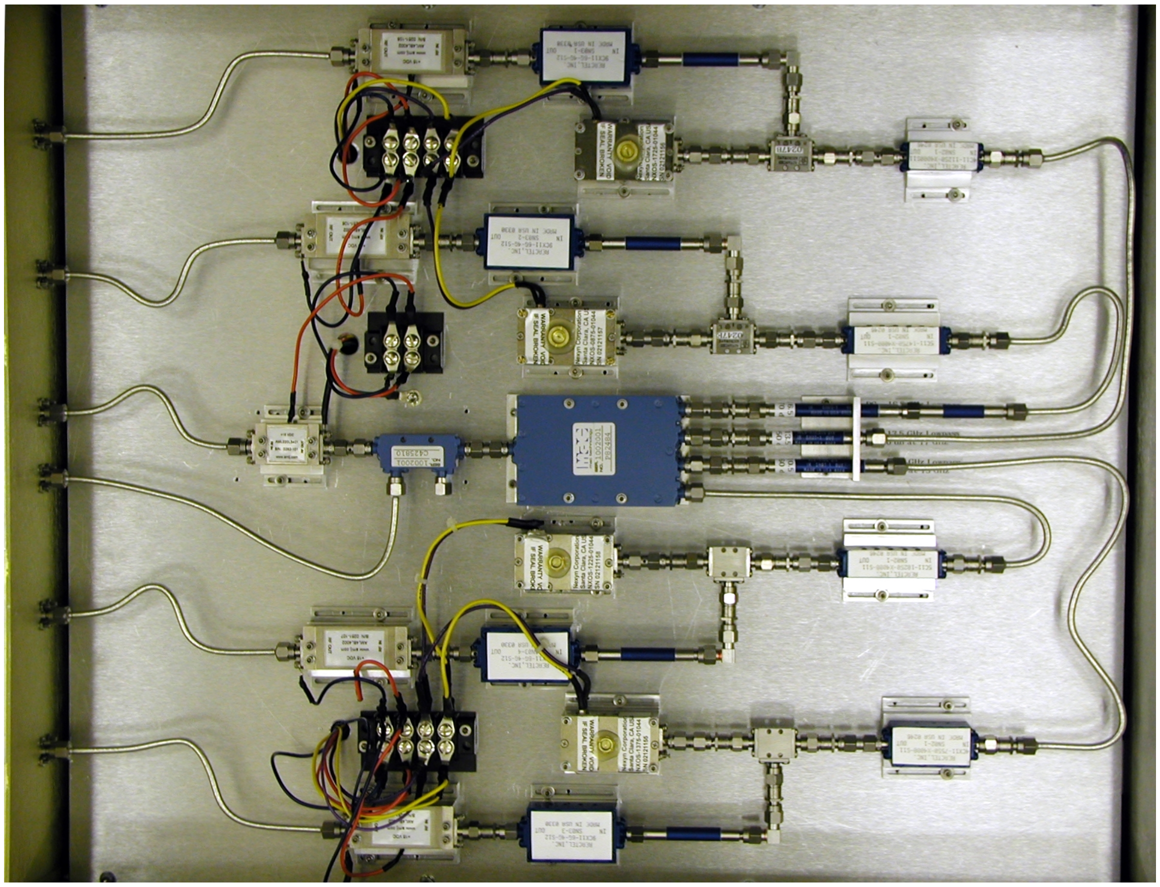


Figure 3.16: IF downconverter, as built. Hand-formable cables with SMA connectors are used when needed; otherwise, short SMA adapters are used. DC power at +12 V is provided via the yellow wires, using the purple wires for return current, and +15 V DC power is provided through the orange wires, using the blue wires for return current.

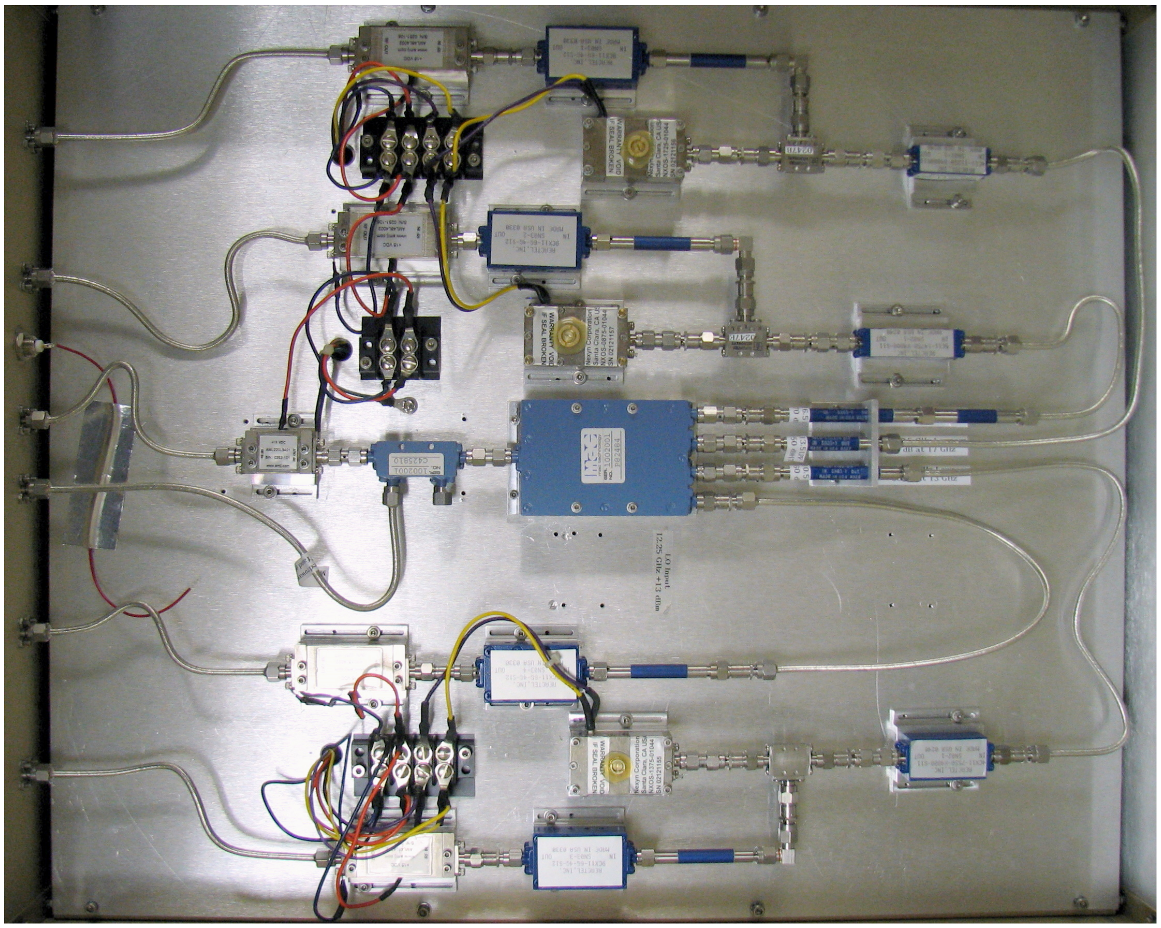


Figure 3.17: IF downconverter (current version, with direct 4 - 8 GHz line).

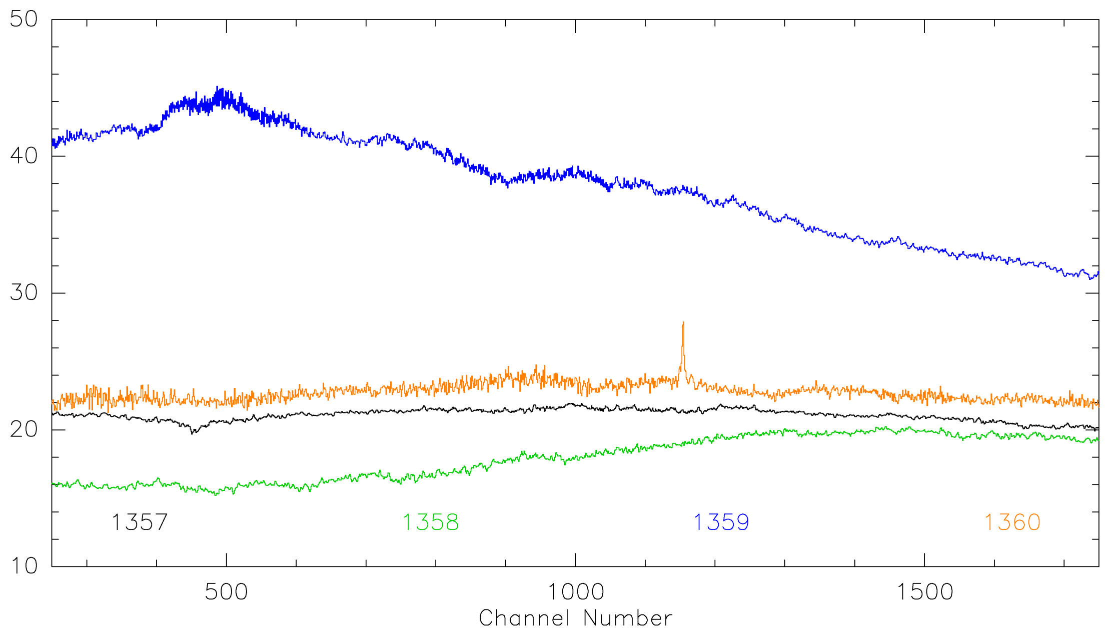
grating spectrometers, such as Z-spec [Naylor, 2008], permitted even faster low-resolution searches than Z-Rex could offer. Equally relevant was the fact that Z-Rex had been enthusiastically embraced by several groups interested in performing molecular-line surveys of nearby objects (such as star-forming regions). The surveys required much higher-resolution spectra, and ongoing development efforts for Z-Rex focused on bringing these capabilities online.

Due to the source's proximity, emission lines from galactic objects tend to be much narrower. For instance, lines from star-forming regions are often only a few km/s wide, so a higher-resolution spectrometer is needed than would be required for extra-galactic redshift follow-up. To enable these observations, the CSO uses a 4×1-GHz array acousto-optical spectrometer (AOS) from the University of Köln [Horn et al., 1999]. As with the WASP2 spectrometers, an IF processor is needed to divide the signal into four bands at the correct frequency and power level for the AOS. The requirements for this IF processor are considerably more stringent than for the one discussed in Section 3.4.1; therefore, it was designed and constructed by CSO staff and maintained at the telescope as a facility instrument.

3.5.1 High-Resolution IF Processor

A great deal of effort has been put into determining and providing the optimal signal characteristics at the interface between Z-Rex and the high-resolution IF processor. In particular, the IF processor can actively adjust the output power it provides to the AOS via computer-controlled attenuators; however, it has no means to control the input power provided to it by the receiver. Therefore, it is critically important to insure that the right power levels are provided at the interface by manually inserting attenuators. Before determining the ideal power levels, we experienced frustrating observing attempts in which brightness temperatures (and hence line strengths) were unreliable, apparently due to saturation in the IF processor. (See Figures 3.18 and 3.19.)

Based on measurements like those shown in Figure 3.19, it was determined that the IF processor works best when the input attenuation is ~ 25 dB. To further test the IF processor's performance, measurements were taken of its end-to-end conversion gain at different



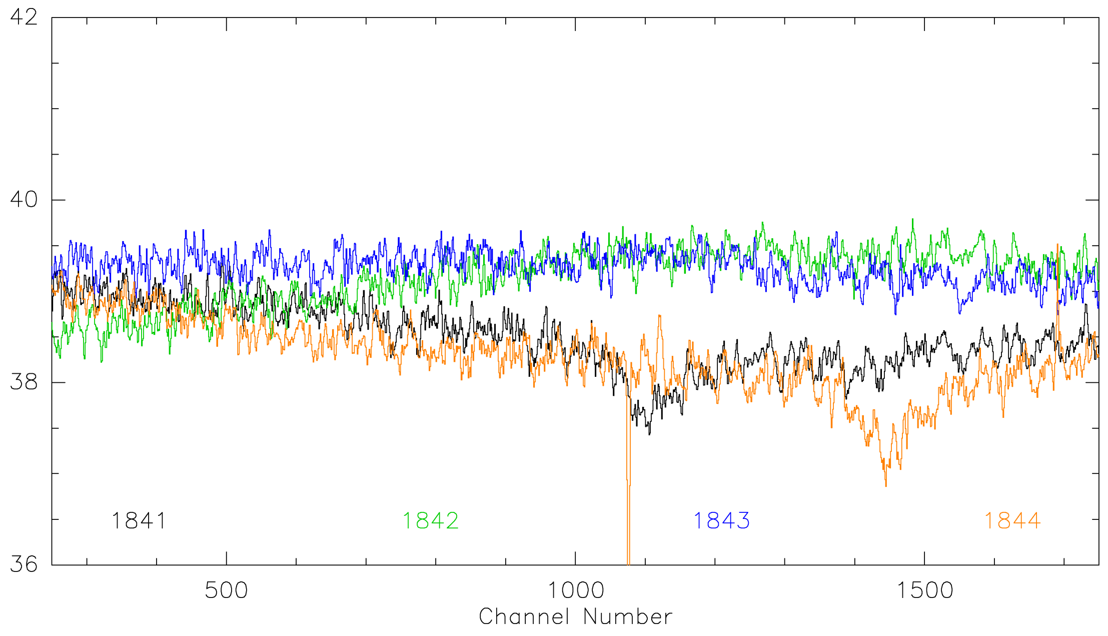


Figure 3.18: Spectra taken on Jupiter using incorrect IF attenuation (10 dB, top) and improved IF attenuation (28 dB) with equalizer (bottom). The vertical axis corresponds to brightness temperature, T_A^*, in K. Note the significantly expanded scale in the top plot. Scan numbers (across the bottom) correspond to AOS bands 1 through 4, respectively. Since these data were taken during two different runs (May 2007 and November 2007, respectively), the IF processor also might have had some internal optimization.

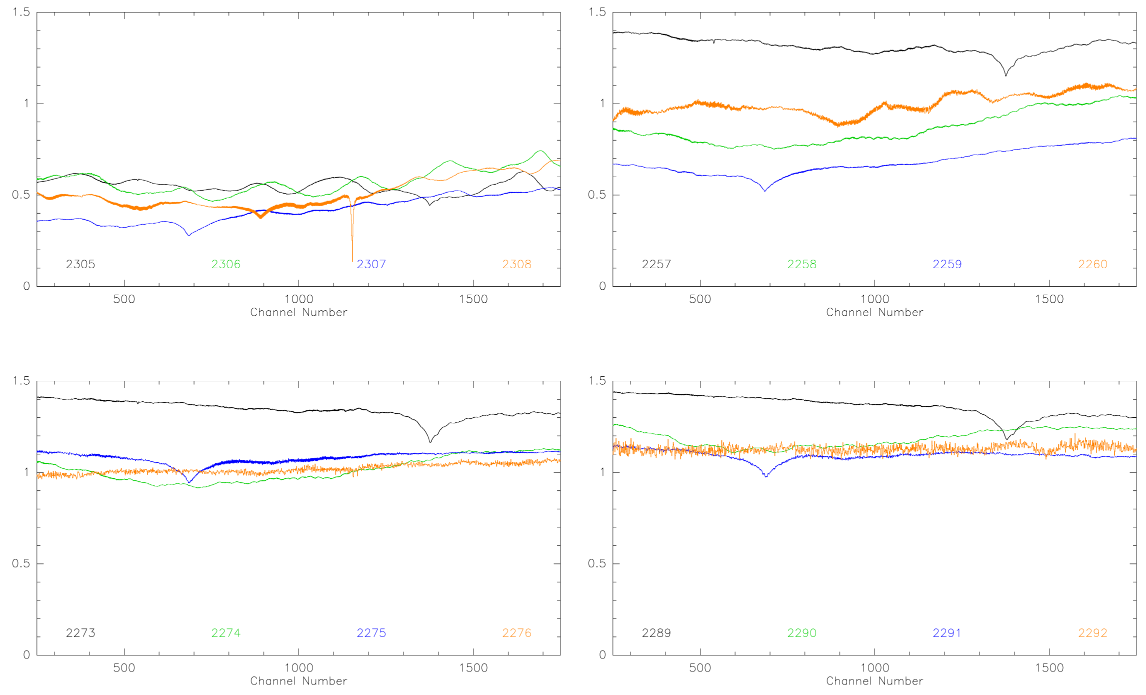


Figure 3.19: Effects of increasing IF attenuation in the absence of an equalizer. The vertical axis of each plot indicates the value of the telescope's calibration scan, which corresponds to $(Y_{sky} - 1) = \frac{P_{hot}}{P_{sky}}$. The four scans (indicated by the legend at the bottom of the plot) correspond to IF frequencies of 4 - 5, 5 - 6, 6 - 7, and 7 - 8 GHz, respectively. At 0 dB of IF attenuation (upper left), all bands suffer from gain compression, yielding poor Y factors. When the attenuation is increased to 12 dB (upper right), the Y factors spread out, as some bands receive appropriate power while others do not. At 20 dB (lower left), the four bands offer similar Y factors, but the bands corresponding to larger IF frequencies, and hence greater attenuation in the IF cable, begin to show increased noise levels. At 25 dB (lower right), the Y factors are still similar, but the noise level in band 4 continues to degrade, and band 3 shows some increased noise. All scans have $Y_{sky} \approx 2.4$, measured at the 4 - 8 GHz output of the receiver's warm IF amplifiers and were taken during the May 2007 run.

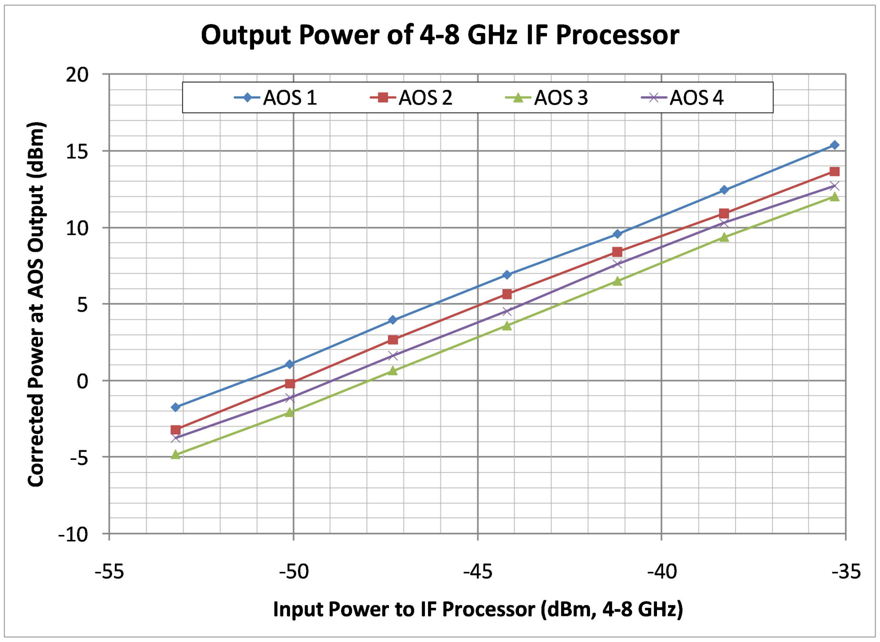
input attenuation levels. Based on measurements taken in August 2007 (Figure 3.5.1), the IF processor was found to have good linearity for inputs from -39 dBm to its nominal power level of -45 dBm.[3] We also determined that an equalizer is required to correct the frequency slope introduced by the cable from the receiver to the IF processor. Otherwise, when the low-frequency band is properly powered, the high-frequency band is badly underpowered, greatly increasing the noise levels seen in that band.

As discussed in Section 3.4.1 (and shown in Figure 3.17), the IF processor designed for use with the WASP2 spectrometers has been modified to provide one branch that amplifies a 4 - 8 GHz signal without doing a downconversion. This modified branch has been serving as a warm IF amplifier to boost the signal before it travels through the long cable from the receiver to the high-resolution IF processor. While this arrangement is sufficient, it is not ideal, as the WASP2 IF processor contains several other strong LO sources that should not be coupled into the IF path unnecessarily. Therefore, the development of a dedicated 4 - 8 GHz warm IF amplifier box for Z-Rex is recommended.

3.5.2 Acousto-Optical Spectrometer (AOS)

Much like the array of WASP2 units, the high-resolution spectrometer provides 4 GHz of bandwidth by running four 1-GHz AOSs in parallel. Within each AOS, the RF signal is coupled to a crystal via a piezoelectric transducer, creating modulations in the index of refraction of the crystal that act like a diffraction grating. The crystal is illuminated by a laser beam, casting the diffraction pattern onto a linear CCD. The intensity of the light on each pixel of the CCD corresponds to the input power of the RF signal at that pixel's characteristic frequency. Spectra are calibrated by injecting a known frequency comb into the AOS and studying the resulting peaks. The calibration allows CCD positions to be converted into the corresponding frequencies and reveals the frequency resolution of the system.

[3]When measuring the output power of the IF processor, readings were corrected for internal attenuation using the nominal settings of the computer-controlled variable attenuators, as displayed by the control software. For instance, if the computer indicated it was using 3 dB of attenuation, the readings were increased by 3 dB. There could be small discrepancies between the nominal and actual values of the attenuation that could lead to minor inaccuracies in that data. If present, such errors would only occur in the two highest-frequency points for AOS 1 and the highest-frequency point for AOS 2, as all other measurements correspond to 0 dB of internal attenuation.



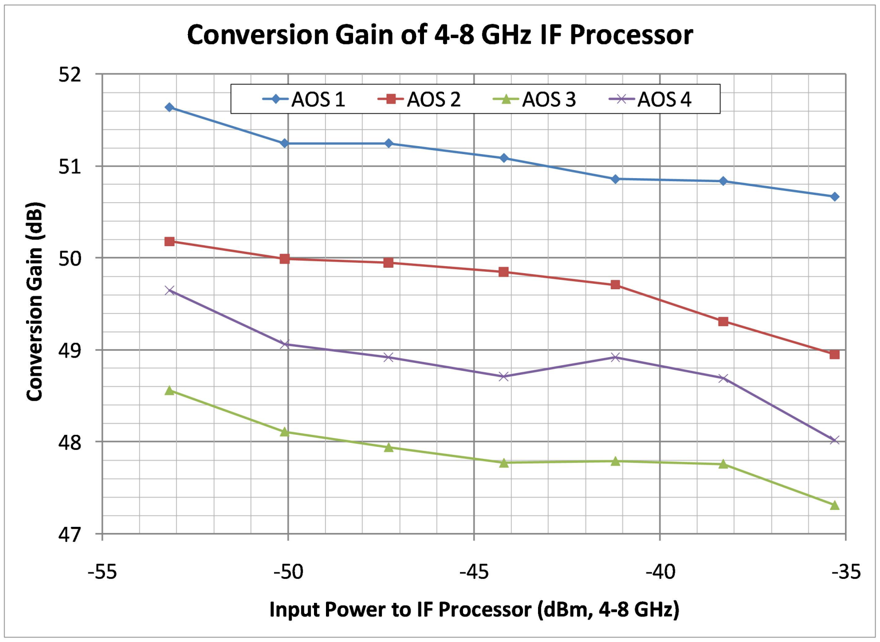


Figure 3.20: Output power (top) and conversion gain (bottom) of the IF processor, measured in August 2007. The IF processor has an optimal input power level of -45 dBm, with a 6-dB margin below that, and it can be seen to offer excellent linearity across that range. These results indicate that $\approx$ 29 dB of input attenuation is appropriate for the IF processor, consistent with Figures 3.18 and 3.19. The power levels coming into the IF processor were -15 dBm, so 29 dB of input attenuation corresponds almost exactly to its nominal -45 dBm input power. (Also see Footnote 3.)

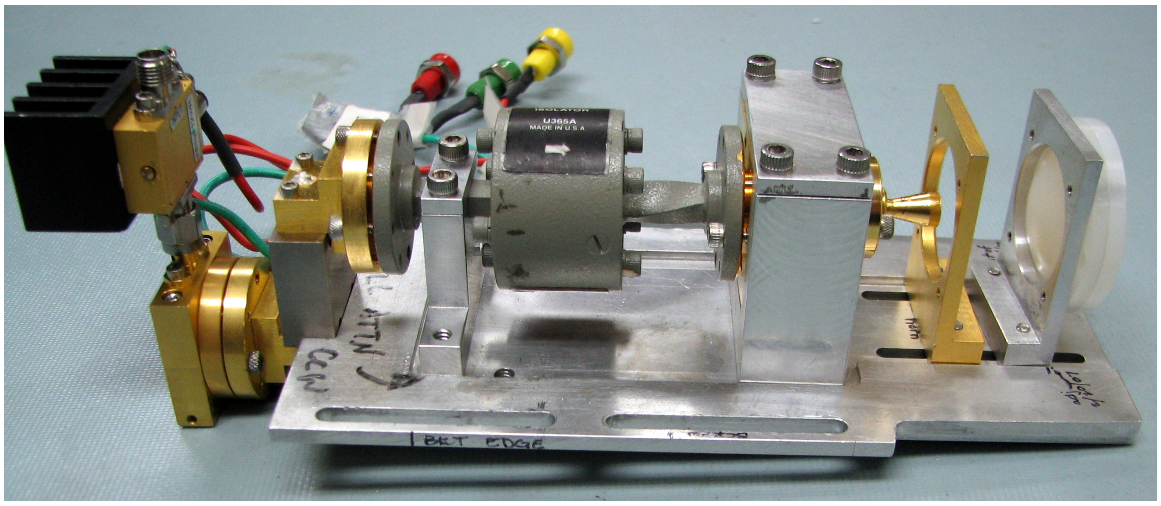


Figure 3.21: Z-Rex synthesized LO from Virginia Diodes.

The array AOS at the CSO generates four spectra, each containing 2048 channels and covering 1 GHz of spectral bandwidth. By design, some channels at either end of each spectrum are under illuminated, so spectra usually have $\sim$ 1700 valid channels. Although it varies with time, the resolution of the system is typically 1.2 MHz or slightly better, which means that peaks with linewidths of a few km/s still span several channels, allowing an accurate determination of peak shape.

3.6 Synthesized LO

In recent years, synthesizer-driven LO sources have become readily available at submillimeter frequencies. They are commercially available from companies such as Virginia Diodes,[4] who constructed the unit shown in Figure 3.21 for Z-Rex. At the left of Figure 3.21 is a coaxial connector that attaches to a standard microwave generator to provide an input signal with frequencies of $\sim$ 13 - 19 GHz at power levels of ~ -5 dBm.

The synthesized LO chain consists of an amplifier followed by a frequency tripler. The signal is again amplified and sent through an isolator, before going into the final quintupler and the output horn. Overall, the LO chain multiplies the input frequency by a factor of 15 to create output frequencies of $\sim$ 200 - 285 GHz at power levels of $\sim$ 60 - 250 μW, providing ample power to the SIS junction.

[4]Virginia Diodes, Inc., Charlottesville, VA; www.virginiadiodes.com

The other type of commonly used LO is the Gunn-based LO chain, an example of which is shown in Figure 3.22. In this LO chain, the fundamental frequency of $\sim$ 70 - 90 GHz is produced by a Gunn diode in a resonant waveguide cavity (top of Figure 3.22). The frequency of the oscillations is set by the tuning knob on the left, and the knob on the top can be used to optimize the output power. The signal then travels through an isolator, a harmonic mixer, and a waveguide attenuator before going through a waveguide bend, into the final tripler and out the horn.

The tuning control provided by the micrometer on the Gunn is not sufficient to maintain the stability needed for high-resolution spectra. Instead, coarse frequency adjustments are made using a calibration table to generate approximate settings for the tuning and power knobs. Precise frequency control relies on a lock frequency generated by a microwave synthesizer; a phase-locked loop (PLL) uses a harmonic mixer to compare a multiple of this lock frequency to the current frequency of the Gunn. By modulating the DC bias provided to the Gunn, the PLL can stabilize the LO frequency to the needed accuracy. While the signal from the Gunn diode is relatively clean, the PLL can introduce significant spurs into the spectrum. However, as long as the cause of these spurs is recognized, they can easily be eliminated by adjusting the PLL parameters.[5]

Although accurate, a PLL-controlled Gunn can be finicky; at times, it can be very difficult to get the PLL to lock onto the desired frequency, requiring careful adjustments of the tuning knob, experimentation with different power levels for the signal carrying the lock frequency, and a good dose of patience. There have been nights at the CSO when we have literally spent hours trying to get a Gunn and PLL locked to a needed frequency; on other occasions, the Gunn has locked to the incorrect frequency, generating invalid spectra. Clearly these types of problems can impact any observing project, not just line surveys.

Much of the interest in synthesized LOs is driven by the higher level of convenience they offer relative to Gunn-based LOs. The tunerless design means that changing frequencies is as simple as setting the frequency (and perhaps power) on a microwave synthesizer. In concert with the tunerless mixer design of Z-Rex, this gives the overall receiver system

[5]In particular, the multiple of the lock frequency being used by the PLL should be changed to move the lock signal out of the 4 - 8 GHz IF band of the receiver. At the CSO, this can be accomplished with the UIP command `LO /LOOP=N`, with a suitable choice for N. For most LO frequencies, $N = 9$ works well.

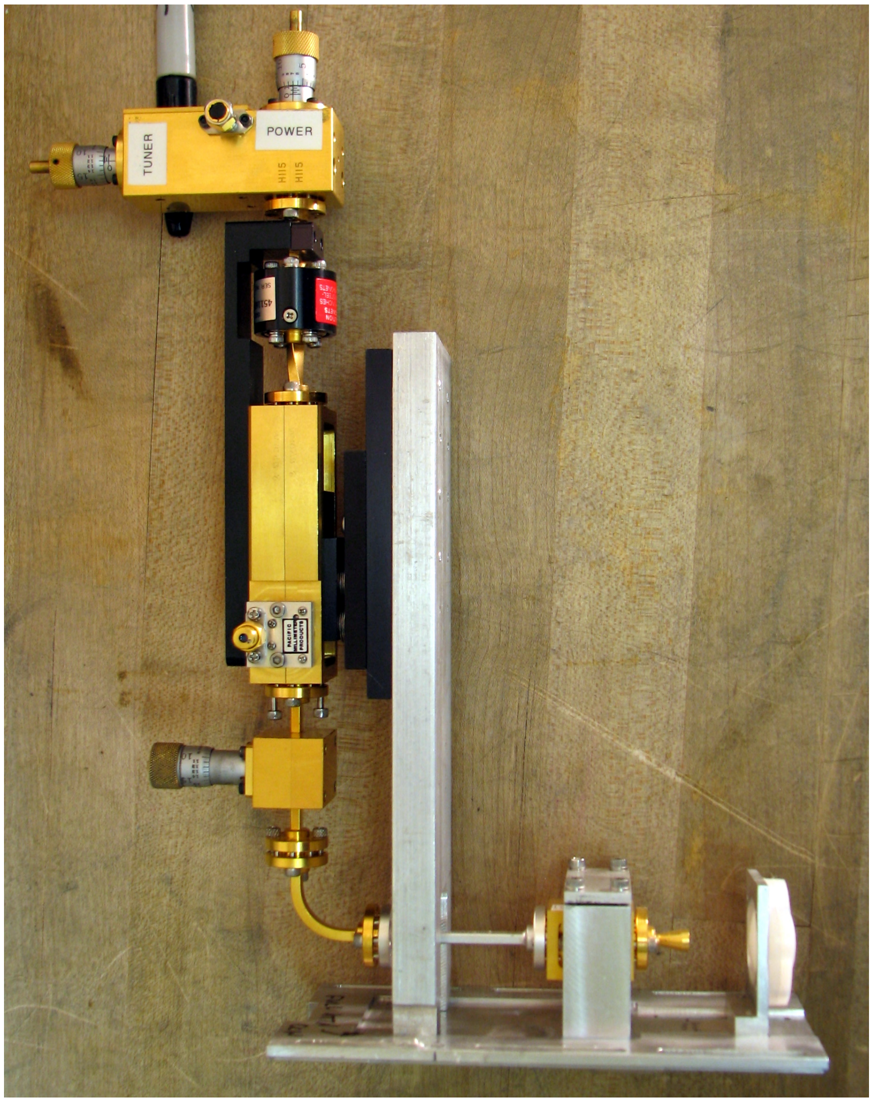


Figure 3.22: 230-GHz Gunn-based LO chain.

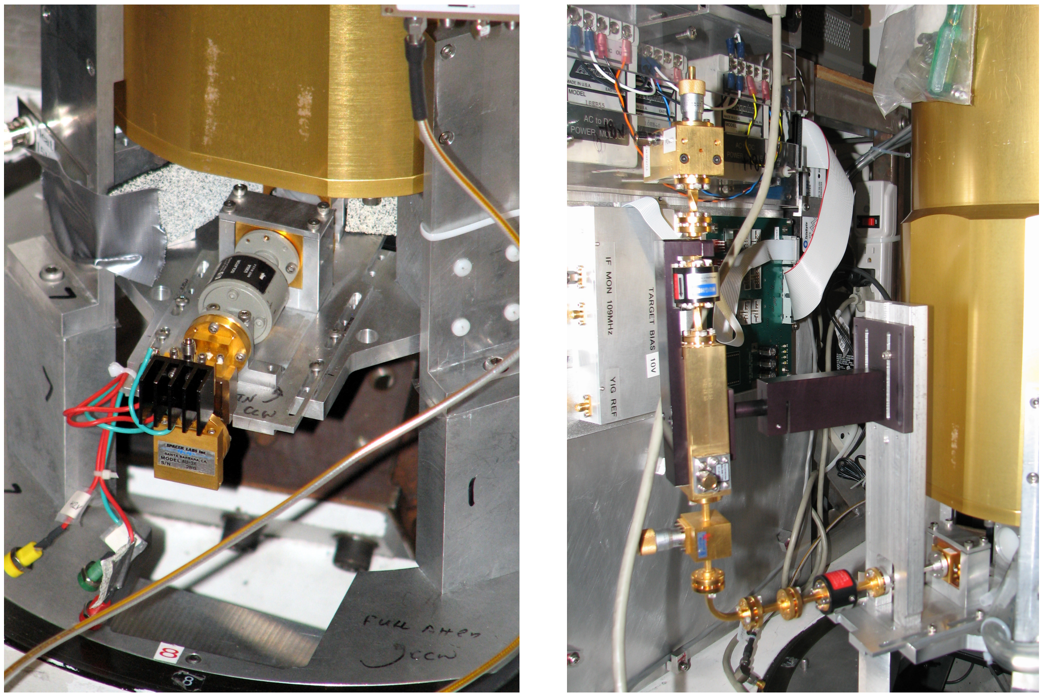


Figure 3.23: Close-up of synthesized LO mounted on Z-Rex (left) and Gunn-based LO chain mounted on Z-Rex (right). The plate containing electronics for the phase-locked loop (PLL) can be seen in the background of the right-hand figure.

impressive frequency agility, making an important difference in the time required for an unbiased line survey, which might include hundreds of LO settings.

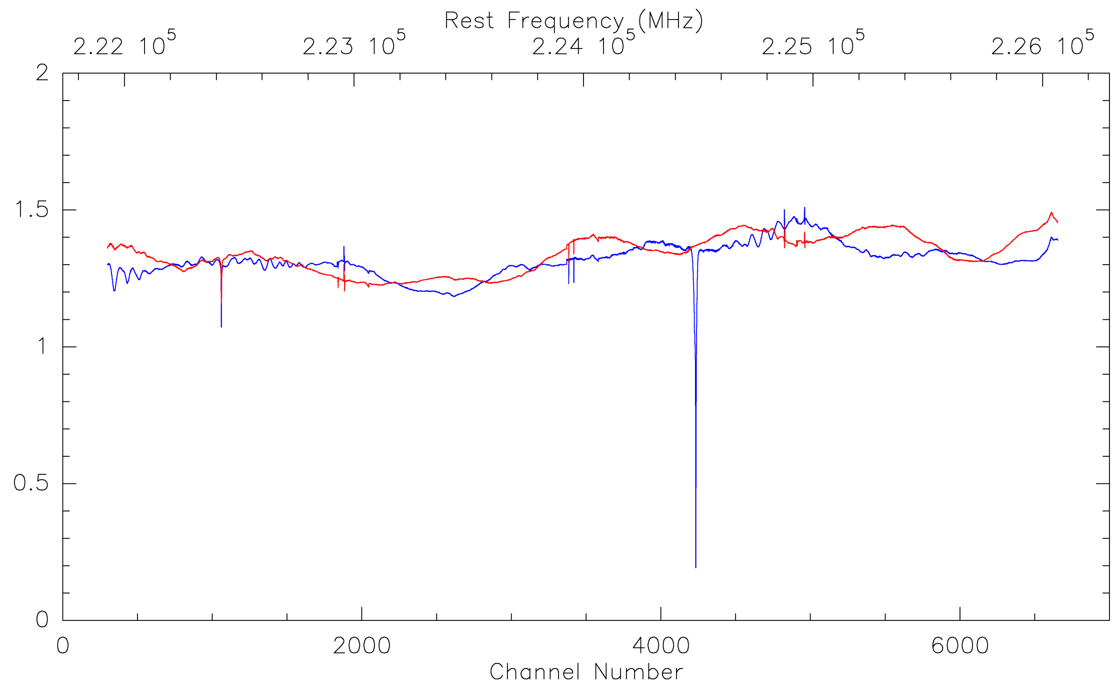
In addition to fast tuning, a synthesized LO can build on all of the other technologies that go into a commercial microwave synthesizer. For instance, by connecting the synthesizer to the telescope's computer network via GPIB, we were easily able to implement automatic tuning; in contrast, installing a computer interface board for a Gunn oscillator would have required significant effort. Similarly, we have been able to build on existing software libraries to construct automated calibration and testing routines for the synthesized LO by making use of a GPIB-connected synthesizer and power meter.

Finally, the synthesized LO offers better robustness under realistic usage scenarios in an operational telescope. The simplified user interface makes error less likely, and the fact that observers do not need to tune the sensitive RF components greatly decreases the likelihood of damage by electrostatic discharge (ESD). When placed under appropriately designed computer control, the dangers of inadvertently damaging the multiplier chain are nearly eliminated. When using a Gunn diode, observers must work carefully to avoid ruining a component with ESD or overpowering the final multiplier.

Despite these benefits, the synthesized LO has some significant disadvantages. Perhaps the most important is the introduction of spurious signals ("spurs") into the spectrum. After observing large spurs in our astrophysical data, we began tracing signals back through the system, and eventually found the source of the spurs hiding in the output from the microwave synthesizer. The LO chain was dutifully multiplying these by a factor of 15, producing errant lines in the spectra. In principle, the on-source/off-source differencing inherent in submillimeter observations could remove such signals. However, they were sufficiently strong that even tiny fractional fluctuations translated into large lines in the spectra.

Figure 3.24 shows an example of the spurs introduced into the telescope's calibration scans (blue lines). The figure includes two different models of synthesizers, but even the cleaner synthesizer still introduces significant noise spurs into the spectrum. To eliminate the spurs, we use a YIG tracking filter, inserted between the synthesizer and the multiplier chain. The YIG filter is a very high-Q filter with an approximately 50-MHz bandpass that can be electrically tuned to any frequency from 2 - 18 GHz. By setting the frequency of the filter to match that of the synthesizer, we eliminate most of the undesired signals, significantly improving the observed spectra (red lines in Figure 3.24). The YIG filter is digitally tuned, making it amenable to computer control; Z-Rex uses a dedicated control computer for the YIG filter, but that functionality could be incorporated into the telescope's control system relatively easily.

In general, the YIG filter has worked extremely well, except that it sometimes spontaneously detunes, causing the LO signal to shut off in the middle of an observation. No definitive cause has been determined, although there are hints that the problem might stem from minute amounts of ESD. Based on these experiences, designs for the next generation of synthesized LO chains at the CSO call for real-time monitoring of LO output power. If LO power drops suddenly, receiver electronics will flip a status bit that instructs the spectrometers to pause the integration until the problem is resolved, much like the "PLL Lock" bit does for the Gunn-based LO on the current facility receivers.



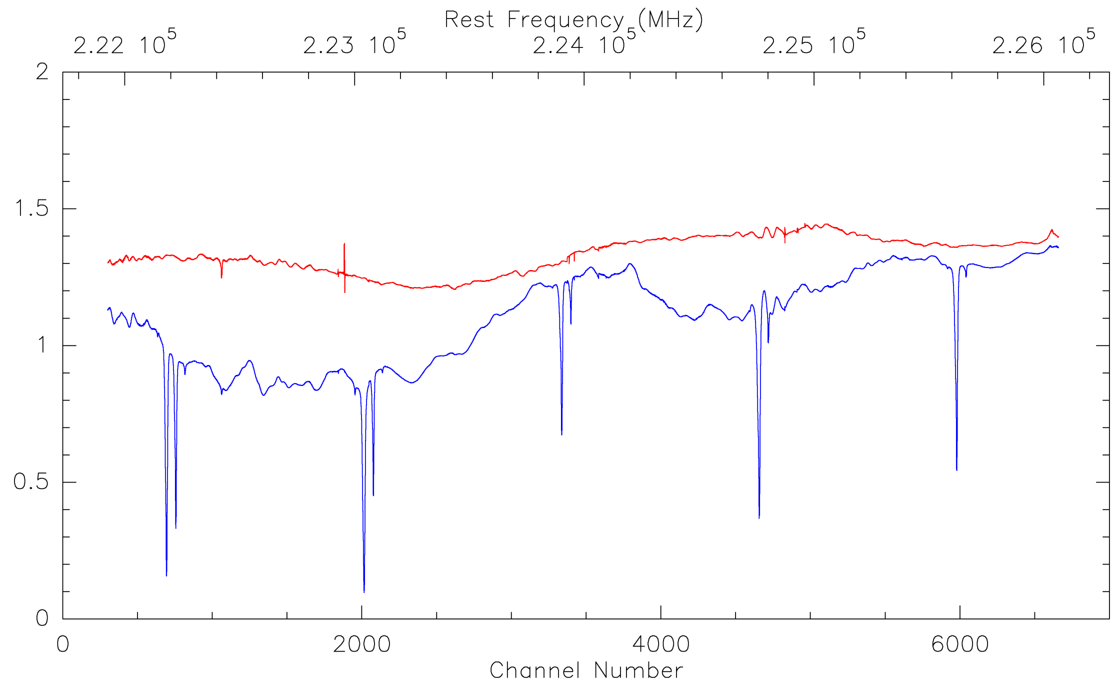


Figure 3.24: CSO calibration scans taken using the synthesized LO chain with an HP 83620B synthesizer (top) and an Anritsu MG3694A synthesizer (bottom). The blue scan shows the spurs that occur using the unfiltered synthesizer output; the red scan demonstrates that inserting a tracking YIG filter between the synthesizer and LO chain removes many of the spurs. Room-temperature and liquid-nitrogen-dipped Eccosorb paddles were used as sources for the calibration scan.

Another significant issue with the synthesized LO is that it can introduce excess noise into the spectrum. Our initial tests of Z-Rex using the synthesized LO produced unacceptably high system noise temperatures. After some experimentation, we determined that the noise could be greatly reduced if we decreased the bias voltage to the final amplifier in the LO chain, thereby forcing it into saturation. We concluded that the LO signal going into the receiver was carrying a significant amount of noise in the form of high-frequency amplitude variations; by clipping the signal in the final amplifier, we could compress that noise to a manageable level.

Figure 3.25 shows a comparison of on-telescope receiver performance when using either the synthesized LO chain or a Gunn oscillator. Y-factor measurements were made using room-temperature and liquid-nitrogen-cooled absorbers that filled the beam of the receiver. For LO frequencies above 245 GHz, both systems behave roughly equivalently; however, at lower frequencies the synthesized LO appears to inject excess noise.

One of the challenges in designing a submillimeter receiver is coupling enough LO power to the junction at low frequencies. With both LO sources, the junction current, I_{SIS}, which measures the degree of LO pumping, was less than its optimal value at lower frequencies. The Gunn provided almost no power at 210 GHz and too little power at 215 GHz, but worked fine for higher frequencies. With the synthesized LO, the junction appeared to be LO-starved for frequencies of 210 - 225 GHz; to boost LO power, we had to increase the bias applied to the final amplifier of the LO chain, allowing more amplitude noise to leak through. Thus, the increased receiver noise with the synthesized LO can probably be explained by a combination of insufficient LO power and increased noise on the LO signal.

The cause of the decreased LO power at low frequencies is somewhat uncertain. The obvious explanation would be that the LO sources are running out of power. This is probably true for the Gunn, as the device used on Z-Rex for these measurements has a low-frequency limit of $\sim$ 70 GHz, corresponding to an LO frequency of 210 GHz. It is less clear how to interpret the results with the synthesized LO. While low output power could be one explanation, there is a compounding factor. Rather than building a new horn for the Z-Rex mixer block, we used an existing horn. Unfortunately, the response of the recycled

Scan	f_{LO} (GHz)	f_{IF} (GHz)	Sideband	AOS Band	Amplitude (K)	Peak Area (K chan)
1532	238	7.5	Lower	4	3.1	112
1538	236	5.5	Lower	2	2.9	108
1546	225	5.5	Upper	2	3.0	118
1556	223	7.5	Upper	4	3.9	141
1493	235	4.5	Lower	1	0.63	21.5
1502	236	5.5	Lower	2	0.67	19.7
1511	237	6.5	Lower	3	0.78	21.6
1520	238	7.5	Lower	4	0.73	21.0

Table 3.3: Scan parameters and peak-fitting results for the spectra shown in Figure 3.26, including the LO frequency, f_{LO}, the IF frequency at which the CO line appears, f_{IF}, and the results of Gaussian fits to the various peaks.

horn falls off at low frequencies ($\lesssim$ 220 GHz), limiting the receiver's RF bandwidth. This effect undoubtedly contributes to the difficulty in coupling LO power to the junction at low frequencies. To test this hypothesis, we attempted to make a rough measurement of the receiver's performance at low frequencies. As shown in the top plot in Figure 3.26, we measured the CO line of CRL2668 in both the upper and lower sidebands. The larger peak strength in scan 1556 most likely derives from the sideband imbalance of the receiver, caused by the lack of sensitivity of the horn at lower frequencies. The bottom plot shows similar data for CRL2155, except that all scans are in the lower sideband. Without the low-frequency sideband imbalance, all peaks are approximately the same strength.

Overall, the synthesized LO chain is very convenient, although the extra components required by the YIG filter decrease that convenience. The excess noise represents a problem, particularly at lower frequencies, and we continue to be concerned that the LO chain might be introducing spurs to the spectrum. The disadvantages might be worthwhile in some circumstances, such as an observing project in which frequency agility is critical; however, for the line surveys discussed in this thesis, we chose to use the Gunn, despite the inconvenience of tuning.

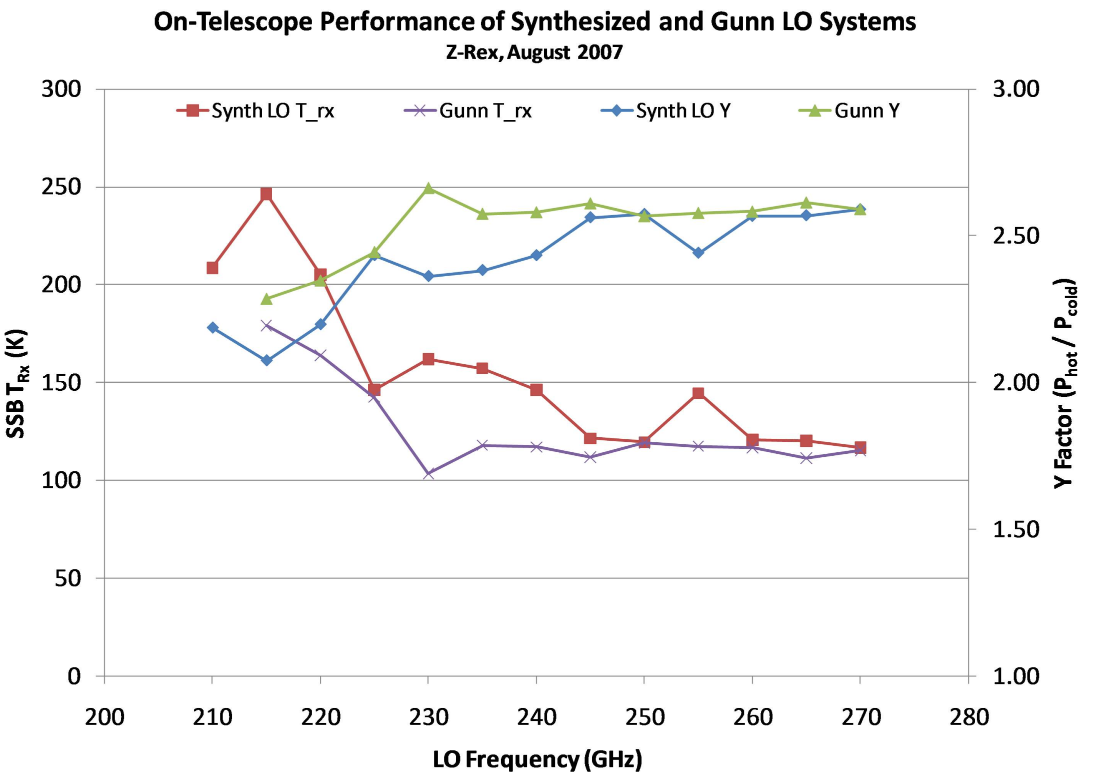
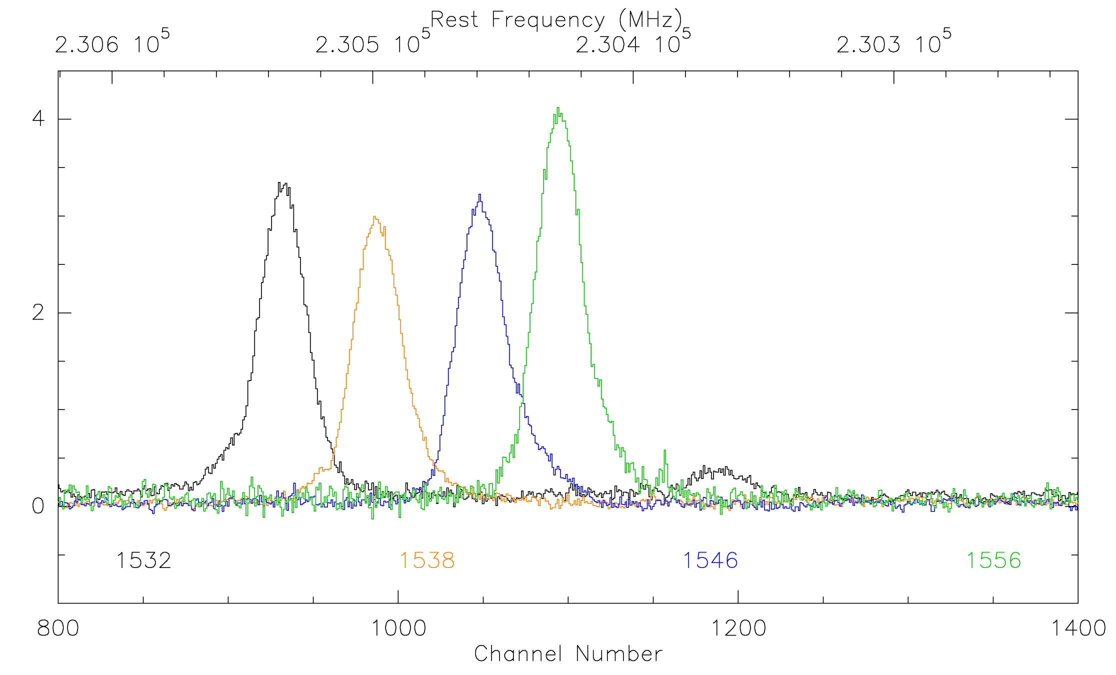


Figure 3.25: On-telescope performance of Z-Rex with synthesized LO and Gunn LO.



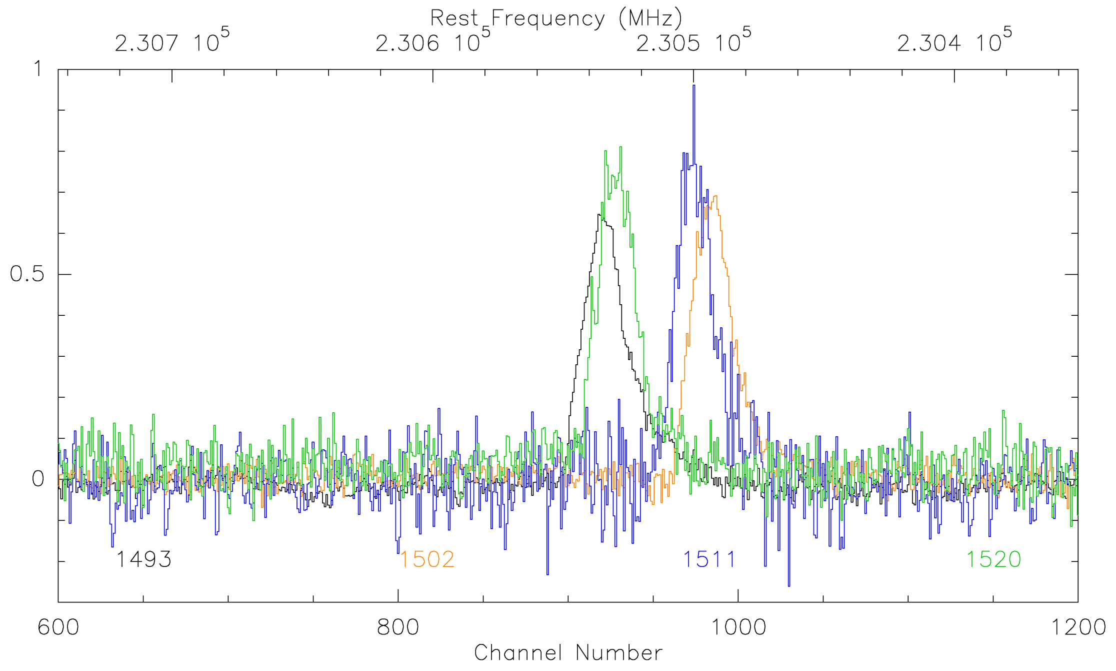


Figure 3.26: Estimating sideband imbalance. All scans were taken at an IF attenuation of 29 dB with no equalizer. Scan parameters and peak-fitting results are listed in Table 3.3.

Chapter 4

Deconvolving Double-Sideband Data

4.1 Introduction

The basics of heterodyne detection can be understood by considering a simple model in which the mixer multiplies two sinusoidal signals with frequencies f_{RF} and f_{LO}. The former represents the signal to be analyzed (such as the submillimeter photons collected by the telescope) while the latter represents the frequency of the local oscillator (LO), a reference signal generated by the receiver electronics. When these signals are multiplied by the mixer, the output contains two new frequencies, as can be demonstrated using trigonometric identities:[1]

$$\sin(\omega_{RF}) \times \sin(\omega_{LO}) = \frac{1}{2}\left[\cos(\omega_{RF} - \omega_{LO}) - \cos(\omega_{RF} + \omega_{LO})\right].$$

This equation contains the sum and difference frequencies, describing upconversion and downconversion, respectively.[2] For a heterodyne receiver, downconversion is desired, so the sum frequency is removed with a low-pass filter, leaving the difference frequency, often referred to as the intermediate frequency (IF) and represented by f_{IF}.

For molecular-line surveys, the IF signal is fed to a spectrometer, which generates a power spectrum of the data, $P(f)$. The power spectrum depends on the amplitude, but

[1]For compactness, we have used $\omega = 2\pi f$ in this equation.

[2]In terrestrial applications (such as AM radio), upconversion and downconversion represent a complementary pair. A low-frequency signal (e.g., music) is encoded onto a high-frequency carrier signal at the transmitter. The original low-frequency signal can then be recovered using downconversion at the receiving end. For an astronomical receiver, there is no upconversion process; the signal is generated directly by the source at f_{RF}. Thus, there is no low-frequency original signal to be recovered; instead, downconversion allows the signal to be processed at lower frequencies, which is significantly easier.

not the phase, of the input signal. Since a "negative" frequency is equivalent to a phase shift of π, the spectrometer cannot discriminate between power from a signal at f and one at $(-f)$. Therefore, a single value of $f_{IF} = |f_{RF} - f_{LO}|$ in the power spectrum actually contains contributions from two RF frequencies:

$$P_{IF}(f_{IF}) = P_{RF}(f_{LO} - f_{IF}) + P_{RF}(f_{LO} + f_{IF}). \tag{4.1}$$

For a given f_{IF}, these two frequencies are called the lower sideband (LSB) and the upper sideband (USB):

$$\begin{aligned} f_{LSB} &= f_{LO} - f_{IF} \\ f_{USB} &= f_{LO} + f_{IF}. \end{aligned} \tag{4.2}$$

Since the resulting power spectrum combines data from both sidebands, it is referred to as the double-sideband (DSB) spectrum.

The sideband ambiguity in the power spectrum can make it difficult to concretely identify molecular lines, particularly in crowded spectra. Therefore, some method of discriminating between the sidebands is needed. The sideband identification can be done in hardware, via a sideband-separating receiver or a sideband-rejecting receiver, or it can be implemented in software during the data-analysis phase. Including sideband rejection or separation capabilities in the receiver provides a more direct method of observation, but the receiver designs are more complex than the single-mixer construction used for Z-Rex.

One of the key advantages of Z-Rex over previous generations of receivers is its very large instantaneous bandwidth, which significantly exceeds the bandwidth of the spectrometers available to analyze the signal. In its broadest configuration, the IF bandwidth of the receiver is ~ 12 GHz, but the CSO high-resolution spectrometers, at present, can only accept 4 GHz of data. Since the spectrometers represent the bottleneck, it is important to choose a receiver design that can use them as efficiently as possible.

In this scenario, a DSB receiver offers an advantage in observing efficiency, at the expense of its inherent sideband ambiguity. Because it can observe both sidebands simultaneously, a double-sideband receiver effectively doubles the amount of data that can be extracted from the spectrometers. A single observation captures 8 GHz worth of spectral data using the spectrometers' 4-GHz bandwidth.

In contrast, a sideband-separating or sideband-rejecting receiver achieves only a one-to-one mapping between spectrometer bandwidth and the range of observed sky frequencies. Therefore, either receiver would require twice as many LO settings. For an unbiased survey, LO tuning represents a significant fraction of the overall observing time; doubling the number of tunings would adversely impact the overall survey efficiency. Moreover, even sideband-separating or sideband-rejecting receivers have some amount of sideband leakage, making it desirable to perform further sideband deconvolution during the data-analysis phase.

Fortunately, it is possible to do sideband deconvolution after the fact, as can be seen with a simple example.[3] Consider a molecular line that appears at an IF frequency of $f_{IF} = 5.0$ GHz with an LO frequency of $f_{LO} = 242$ GHz, indicating that the line's RF frequency could be either $f_{LSB} = 237$ GHz or $f_{USB} = 247$ GHz. If the LO frequency is then changed to $f_{LO} = 242.1$ GHz, the line's IF frequency will change. A line in the upper sideband would move to a lower IF frequency of $f_{IF} = 4.9$ GHz while a line in the lower sideband would move in the opposite direction to $f_{IF} = 5.1$ GHz.

At an intuitive level, at least, this example demonstrates that the sideband ambiguity can be removed by observing each line at multiple LO frequencies. In practice, the process described in the previous paragraph can be used to manually deconvolve relatively simple spectra. For complicated spectra, an automated process is needed, but it relies on the same principles.

4.2 Overview

The goal of sideband deconvolution is to take a series of double-sideband spectra and undo the effects of the spectral "folding" that occurs during RF downconversion, separating the power in the lower sideband from that in the upper sideband. This produces a single-sideband spectrum in which the frequency ambiguity has been removed and every channel can be assigned a unique frequency.

[3] Also see Figure 1.9.

One frequently used method of deconvolving data involves a forward-folding process in which a parameter-dependent model is used to simulate the observed spectrum. A computer algorithm can then generate a trial set of model parameters and produce a simulated data set corresponding to those parameters. By comparing the predicted data set to the actual data, the algorithm can iteratively improve the values of the model parameters. Typically, this process relies on minimizing some quantitative measure of the difference between actual and modeled data, such as χ^2.

In this work, we describe such a method for recovering the single-sideband spectrum by minimizing a χ^2-like quantity. However, this is not the only approach that can be used to perform the sideband deconvolution. For example, an early survey of Orion KL used a technique similar to the CLEAN algorithm common in aperture-synthesis imaging [Sutton et al., 1985]. This method is relatively straightforward, but it requires the relative sideband gains to be specified at the outset. Sutton et al. [1985] find that the deconvolution leaves false peaks as strong as 0.5 K in their single-sideband spectrum, which they attribute to a sideband-rejection ratio for the algorithm of roughly 15 - 20 dB. They conclude that the SSB results are trustworthy down to $\sim$ 0.3 K across the spectrum.

Later researchers have used a maximum-entropy method (MEM), another technique common in radio aperture-synthesis imaging. This method has the advantage that it inherently emphasizes smoothness in the final image. As described in Sutton et al. [1995], the MEM deconvolution algorithm is based on χ^2, with an additional term that represents the deviations from a model spectrum:

$$\chi^2_{MEM} = \frac{1}{N_{DOF}} \sum_k \left(\frac{d^o_k - d_k}{\sigma_k} \right)^2 - \lambda_{MEM} \sum_i \frac{s_i}{s_s} \log \left(\frac{s_i/s_s}{m_i/m_s} \right),$$

where d^o_k and d_k represent the channel-by-channel spectral values in the measured and predicted DSB spectra, respectively, and σ_k represents the uncertainty in the k^{th} channel of the observed DSB spectrum. N_{DOF} represents the number of degrees of freedom. s_i and m_i represent the i^{th}value of the SSB data and the SSB model spectrum, respectively, with $s_s = \sum_i s_i$ and $m_s = \sum_i m_i$. In this equation, the first term represents χ^2_ν, the reduced chi-square, which emphasizes fidelity to the measured data. The second term represents the entropy, a measure of the structure present in the spectrum, relative to the model spectrum; increasing entropy corresponds to an increasingly featureless (smooth) spectrum. The multiplier λ_{MEM} allows the strength of the entropy term to be adjusted; Sutton et al. [1995] found a value of $\lambda_{MEM} \sim 0.1$ - 0.2 worked well. As discussed in Comito and Schilke [2002], a MEM approach to sideband deconvolution has been included in the XCLASS data-analysis software, an extended version of the commonly used GILDAS CLASS package.

One key advantage of MEM is that it allows the sideband gains to be determined as part of the deconvolution, rather than requiring that they be specified in advance, as with the CLEAN-type deconvolution. As Schilke et al. [2001] point out, the MEM approach described here still requires the baselines to be determined manually, which can be difficult in a crowded spectrum.

It is worth noting that the logarithmic function in the entropy term requires that all the data points in the SSB spectrum be greater than zero. This constraint poses no problem for aperture-synthesis imaging, as it is assumed that each point in the sky must produce some amount of positive radio flux; however, the same cannot be said for the chopped observing mode used for this work. (See Chapter 2.) In our observations, it is not unusual for the single-sideband spectrum to have negative values, either from absorption lines in intervening material, or from random noise around a nearly zero baseline. One can work around this limitation by adding a constant offset to the DSB data, and removing that offset from the final spectrum, but this extra step is indicative of the fact that this statistical method is not a natural fit to the underlying problem.

In this work, we take a χ^2-based approach; extra terms are added to χ^2 to represent additional information we believe to be true about the spectrum. One of the new features provided by our methodology is the ability to fit baselines during the deconvolution, allowing the algorithm to determine the set of baselines that best optimizes overall spectral fit. We have also invested significant effort in modeling the sensitivity with which the deconvolution algorithm can recover the single-sideband spectrum, allowing us to make concrete statements about the quality of the results.

We do not feel that one of these methods is necessarily superior to the others; rather, we believe it is inherently useful to have a variety of tools available, particularly given the high-sensitivity surveys currently being performed. In this way, single-sideband spectra can be generated using entirely independent methodologies, thereby providing more insight into the reliability of weak lines detected in the data.

4.2.1 Implementation Overview

The overall process for accomplishing the deconvolution is sketched out in Figure 4.1. It assumes that we have a series of N_{obs} DSB spectra representing a set of molecular-line observations. The individual spectra can represent any combination of different LO frequencies and can include data from different spectrometers with differing channel sizes. We wish to recover the underlying single-sideband spectrum as it would appear on a set of defined spectral channels.

Figure 4.1 shows a high-level sketch of the optimization method used in this work. We break the process into two steps, partly for the sake of conceptual simplicity, and partly to minimize the computation needed in the innermost iteration loop. In the first step, the DSB spectra and the SSB spectrum are resampled onto a set of frequency-aligned channels. The DSB resampling only needs to be done once during analysis; however, the SSB spectrum must be resampled during each iteration. This can be achieved efficiently by creating a resampling matrix that is passed to the iterative loop. The matrix only needs to be calculated once, greatly speeding the program.

The second step of the model represents the sideband mixing that occurs during RF downconversion. The majority of the information about this process can be contained in two convolution matrices that describe how the SSB and DSB channels couple to one another. The coupling depends only on the channel frequencies and the LO settings used for the observations. Since these are known at the outset, these matrices can also be calculated once during the setup phase and then simply passed to the iterative loop. Both the resampling and sideband-convolution matrices are extremely sparse, containing relatively few non-zero entries; performing the deconvolution in a programming environment that supports sparse-matrix calculations greatly reduces memory requirements and increases execution speed.

During the setup phase, a series of matrices representing the effects of the convolution model are created. The DSB data produced by the telescope are contained in a series of N_{obs} spectra, which are assembled into a single long vector by stacking them end to end. The desired SSB spectrum is represented by a set of frequency bins corresponding to the channels that should be used for the recovered spectrum.

After the setup phase is complete, the deconvolution proceeds as an iterative optimization process. A model is used to generate a comparison set of DSB spectra based on a set of model parameters, including the brightness temperature of the SSB spectrum in each of the supplied channels, the receiver sideband-gain imbalance as a function of frequency, and the baselines underlying each of the DSB spectra.

The model can be represented by a large system of equations containing one equation for each channel in the set of observed DSB spectra. Generally, the number of independent DSB channels exceeds the number of free parameters by a factor of ~ 10, indicating that the system is highly overconstrained. Since the measured values are statistical quantities that include random noise, it is nonsensical to look for a solution that exactly satisfies all of the equations; instead, we use a χ^2-like figure of merit that reflects the difference between the DSB spectra predicted by the model and those measured at the telescope. The "optimal" (most probable) set of parameters, corresponds to the minimum value of the figure of merit.

As long as our figure of merit does not deviate significantly from true χ^2, we can also use the final set of parameters in a goodness-of-fit test to determine whether the model is consistent with the observed data.

4.3 Simple Model of Convolution

The optimization routine described in the previous section requires an accurate model of the heterodyne detection process. In the following sections, the model is developed piece

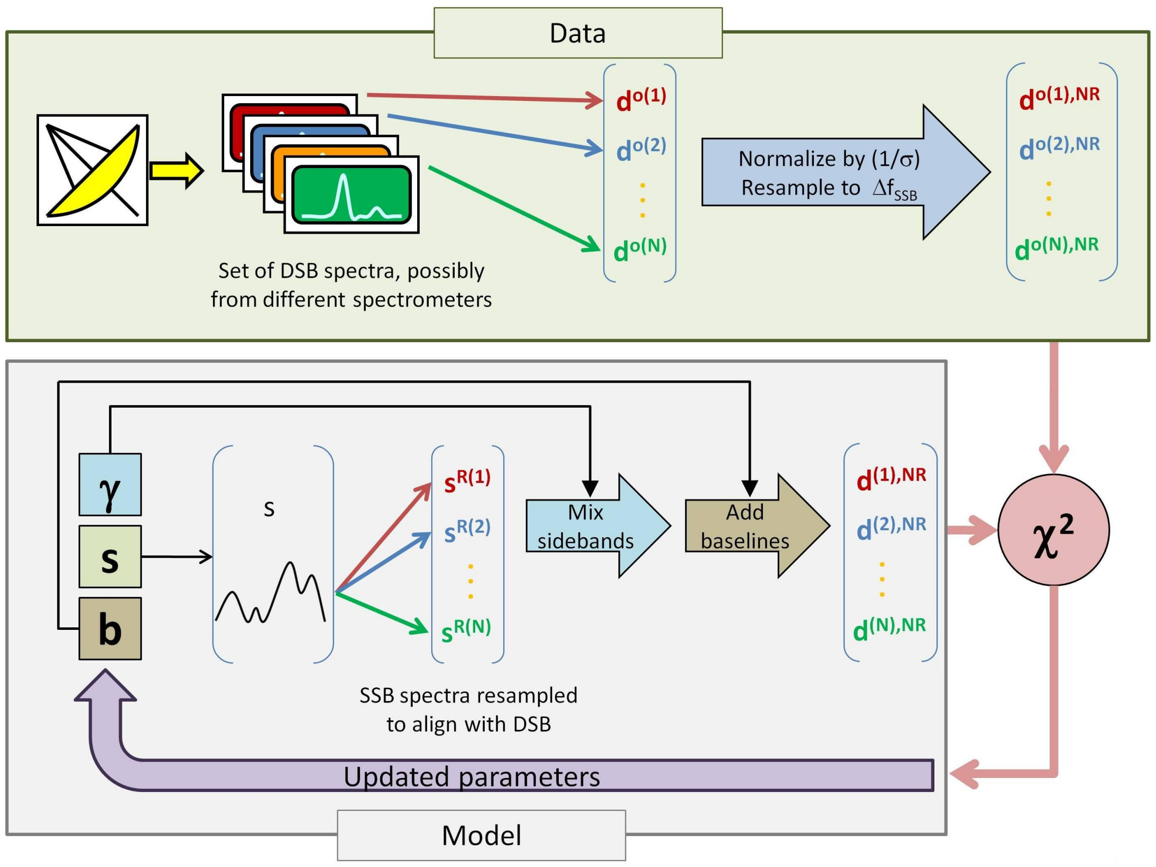


Figure 4.1: Deconvolution Overview

by piece in order to make it as intuitive as possible. We start by considering a simple example that ignores instrumental effects, such as uneven sideband responses and imperfect spectral baselines.

Let $\boldsymbol{d^o}$ represent a vector containing all of the raw DSB spectra stacked end to end, and let $\boldsymbol{s}$ represent the (unknown) SSB spectrum that we would like to find. The mixing of sidebands can be represented by a matrix $\boldsymbol{M}$ in which each row corresponds to one channel in the set of DSB spectra and each column represents a channel of the SSB spectrum. Further assume that the DSB channels are frequency-aligned with the SSB channels so that every channel in $\boldsymbol{d^o}$ corresponds exactly to one USB channel and one LSB channel in $\boldsymbol{s}$ and that the DSB and SSB channel widths are the same. (See Figure 4.2 and Section 4.4.3.) Under these assumptions, the convolution matrix has a simple form

$$M_{ij} = \begin{cases} 1 & \text{if } f_{SSB(j)} = f_{LSB(i)} \text{ or } f_{SSB(j)} = f_{USB(i)} \\ 0 & \text{otherwise} \end{cases}, \tag{4.3}$$

where $f_{SSB(j)}$ represents the center frequency of the j^{th} SSB channel, and $f_{LSB(i)}$ and $f_{USB(i)}$ represent the LSB and USB frequencies corresponding to the i^{th} DSB channel.

In this simplified scenario, we can model the convolution process using the equation

$$\boldsymbol{d} = \boldsymbol{M} \cdot \boldsymbol{s}, \tag{4.4}$$

where $\cdot$ represents matrix multiplication, and $\boldsymbol{d}$ represents the model's prediction for the DSB data, to be compared to $\boldsymbol{d^o}$.

4.4 Incorporating Non-Aligned Spectra

The assumptions underlying Equation 4.3 constrain the selection of LO frequencies.[4] In particular, LO frequencies must be set so that they fall on either the center frequency or an edge of an SSB channel. From a logistical perspective, this is an unwelcome limitation. It would require an observer to establish the desired SSB channel limits before taking any

[4]See Section 4.4.3 for a full discussion.

observations. In addition, receiver control systems might not offer this degree of control over the LO frequency.[5]

Beyond these practical objections, there is a deeper scientific reason not to choose such LO spacing. As demonstrated in Comito and Schilke [2002, Fig. 4], successful deconvolution requires some randomness in the LO spacings. If spacings are perfectly regular, ripples can be introduced into the deconvolved spectrum. The positive lobe of the ripple in one part of the spectrum can be exactly canceled by a negative lobe in another section when those channels are added during the convolution. Including randomness in the LO spacings ensures that this precise cancellation cannot occur.

Thus, it is neither practical nor desirable to design observations such that the DSB channels align exactly with the SSB channels. Instead, the deconvolution algorithm must be capable of properly handling spectra generated with essentially random LO settings. In this case, not only are the DSB and SSB channels mismatched, but channels from different DSB spectra are not aligned with each other.

4.4.1 Resampling Spectra

To handle arbitrary LO settings, both the DSB and SSB spectra must be resampled. In addition to resolving the issues outlined above, this provides a natural method for including DSB spectra with differing channel sizes.

Consider an arbitrary, N-channel spectrum with channel centers $\boldsymbol{f}$, channel widths $\boldsymbol{\Delta f}$, and channel limits $\boldsymbol{c}$, such that the i^{th} channel covers the frequency range c_i to c_{i+1}. Let the spectrum be represented by a set of temperatures, $\boldsymbol{T}$. We can resample this data to produce a new N^R-channel spectrum, characterized by frequency vectors $\boldsymbol{f^R}$, $\boldsymbol{\Delta f^R}$, and $\boldsymbol{c^R}$, and brightness temperatures $\boldsymbol{T^R}$. For concreteness, assume that the channel centers are sorted in ascending order $\left(f_i < f_{i+1} \text{ and } f_i^R < f_{i+1}^R\right)$and that the frequency limits of the resampled spectrum fall entirely within the frequency range of the original spectrum $\left(c_1 \leq c_1^R \text{ and } c_{N+1} \geq c_{N^R+1}^R\right)$.

[5]For instance, at the CSO, the antenna computer automatically calculates Doppler corrections and other fine shifts to the LO frequency that are much larger than a typical spectrometer channel width.

We can express both the original and resampled spectra as continuous functions of frequency, f, by introducing a set of "boxcar" functions that represent the frequency responses of the individual channels:[6]

$$
\begin{aligned}
u_i(f) &= \begin{cases} 1 & \text{if } c_i \le f < c_{i+1} \\ 0 & \text{otherwise} \end{cases} \\
u_j^R(f) &= \begin{cases} 1 & \text{if } c_j^R \le f < c_{j+1}^R \\ 0 & \text{otherwise} \end{cases} .
\end{aligned}
\tag{4.5}
$$

The inner product between the boxcar vectors can be defined in the usual fashion:

$$
\langle v_1 | v_2 \rangle \equiv \int_{-\infty}^{\infty} v_1(f) v_2(f) df, \tag{4.6}
$$

where each of the functions v_1 and v_2 can be any one of the $\{u_i(f)\}$ and/or $\{u_j^R(f)\}$ functions. Also note that the inner product is symmetric, so that

$$
\langle v_1 | v_2 \rangle = \langle v_2 | v_1 \rangle . \tag{4.7}
$$

Each boxcar function is orthogonal to other members of its set, so that

$$
\begin{aligned}
\langle u_a | u_b \rangle &= \Delta f_a \delta_{ab} \text{ and} \\
\left\langle u_c^R \middle| u_d^R \right\rangle &= \Delta f_c^R \delta_{cd},
\end{aligned}
\tag{4.8}
$$

where δ_{ij} is the Kronecker delta function:

$$
\delta_{ij} = \begin{cases} 1 & \text{if } i = j \text{ and} \\ 0 & \text{otherwise.} \end{cases}
$$

The inner product can also be used to calculate the overlap between the two sets of functions:

[6]To avoid confusion between variables, note that the set of channel centers is represented by the vector f while the center of an individual channel is represented by a component of that vector, f_i. The continuous variable representing frequency is written simply as f.

$$\left\langle u_i \middle| u_j^R \right\rangle = \Delta f_{overlap}^{(i,j)}, \tag{4.9}$$

where

$$\Delta f_{overlap}^{(i,j)} = \max\left(0,\ f_{hi}^{(i,j)} - f_{lo}^{(i,j)}\right). \tag{4.10}$$

If the two boxcar functions overlap,

$$\begin{aligned} f_{lo}^{(i,j)} &= \max\left(c_i,\ c_j^R\right) \text{ and} \\ f_{hi}^{(i,j)} &= \min\left(c_{i+1},\ c_{j+1}^R\right) \end{aligned}$$

represent the edges of the overlap region, and $\Delta f_{overlap}^{(i,j)}$ represents its size.

These definitions allow us to write the spectra as continuous functions of frequency:

$$T(f) = \sum_{i=1}^{N} T_i\ u_i\,(f) \text{ and} \tag{4.11}$$

$$T^R(f) = \sum_{j=1}^{N^R} T_j^R\ u_j^R\,(f)\,. \tag{4.12}$$

To generate the resampled spectrum (Equation 4.12), we must find the appropriate values of $\boldsymbol{T^R}$. There are several methods that could be used to find the resampled spectrum; we choose to do it by minimizing the mean-square deviation, A, between the resampled spectrum and the original one:

$$A = \int_{c_1^R}^{c_{N^R+1}^R} \left[T\,(f) - T^R\,(f)\right]^2 df \tag{4.13}$$

$$= \int_{c_1^R}^{c_{N^R+1}^R} \left[\sum_{i=1}^{N} T_i u_i\,(f) - \sum_{j=1}^{N^R} T_j^R u_j^R\,(f)\right]^2 df, \tag{4.14}$$

where we have inserted Equations 4.11 and 4.12.

By taking the derivative of Equation 4.14 with respect to T_k^R and setting the result equal to zero, we can find the value of $\boldsymbol{T^R}$ that minimizes A:

$$\begin{aligned}
0 &= \frac{\partial A}{\partial T_k^R} \\
&= \int_{c_1^R}^{c_{N^R+1}^R} 2 \left[\sum_{i=1}^{N} T_i\, u_i\,(f) - \sum_{j=1}^{N^R} T_j^R\, u_j^R\,(f) \right] \frac{\partial}{\partial T_k^R} \left[\sum_{m=1}^{N} T_m\, u_m\,(f) - \sum_{n=1}^{N^R} T_n^R\, u_n^R\,(f) \right] df,
\end{aligned} \tag{4.15}$$

where Equation 4.15 simply represents a "chain-rule" expansion of the derivative. This result can be simplified by noting that $\frac{\partial}{\partial T_k^R} T_i = 0$, eliminating the first term in brackets altogether, and that $\frac{\partial}{\partial T_k^R} T_n^R = \delta_{nk}$, reducing the final sum to a single term $u_k^R\,(f)$. Since the equation is being set equal to zero, the initial factor of two can be dropped as well:

$$\begin{aligned}
0 &= \int_{c_1^R}^{c_{N^R+1}^R} \left[\sum_{i=1}^{N} T_i u_i\,(f) - \sum_{j=1}^{N^R} T_j^R u_j^R\,(f) \right] u_k^R\,(f)\, df \\
&= \sum_{i=1}^{N} T_i \int_{c_1^R}^{c_{N^R+1}^R} u_i\,(f)\, u_k^R\,(f)\, df - \sum_{j=1}^{N^R} T_j^R \int_{c_1^R}^{c_{N^R+1}^R} u_j^R\,(f)\; u_k^R\,(f)\; df \\
&= \sum_{i}^{N} T_i \left\langle u_i \middle| u_k^R \right\rangle - \sum_{j=1}^{N^R} T_j^R \left\langle u_j^R \middle| u_k^R \right\rangle \\
&= \sum_{i}^{N} T_i F_{ik} - \sum_{j=1}^{N^R} T_j^R \delta_{jk} \Delta f_j^R \\
&= \sum_{i}^{N} T_i F_{ik} - T_k^R \Delta f_k^R.
\end{aligned}$$

Solving for T_k^R gives

$$T_k^R = \frac{1}{\Delta f_k^R} \left[\boldsymbol{F} \cdot \boldsymbol{T} \right]_k, \tag{4.16}$$

where the overlap matrix $\boldsymbol{F}$ is defined as

$$F_{ij} = \left\langle u_i \middle| u_j^R \right\rangle = \Delta f_{overlap}^{(i,j)}, \tag{4.17}$$

and we have used the fact that $\boldsymbol{F}^T = \boldsymbol{F}$ (by Equation 4.7). The brackets in the final result indicate that $\sum_i^N T_i F_{ik}$ is just the k^{th} component of the matrix multiplication $\boldsymbol{F} \cdot \boldsymbol{T}$.

In terms of the individual channel limits, $\Delta f_{overlap}^{(i,j)}$ can be expressed as

$$\Delta f_{overlap}^{(i,j)} = \max\left[0, \min\left(c_{i+1}, c_{j+1}^R\right) - \max\left(c_i, c_j^R\right)\right] \tag{4.18}$$

in accordance with Equation 4.9. Since $F_{ij} = F_{ji}$, only the upper-diagonal or lower-diagonal entries need to be calculated.

Finally, we define the frequency-normalization matrix $\mathbb{A}^{\Delta f^R}$ to be a diagonal matrix with $\frac{1}{\Delta f_i^R}$ along the diagonal:

$$\mathbb{A}_{ij}^{\Delta f^R} = \frac{\delta_{ij}}{\Delta f_i^R}.$$

Since Equation 4.16 is true for all values of k, it can be expressed as a vector equation:

$$\boldsymbol{T^R} = \mathbb{A}^{\Delta f^R} \cdot \boldsymbol{F} \cdot \boldsymbol{T}. \tag{4.19}$$

Typically, the observed spectrum has an experimental uncertainty associated with it (such as the radiometer noise from Equation 2.1). The uncertainty in the resampled spectrum can be determined using standard propagation-of-error techniques [see, e.g., Squires, 1985]. In general, multiplying a random variable by a constant simply scales the uncertainty by the same constant:

$$x' = cx \quad \Longrightarrow \quad \sigma_{x'} = c\sigma_x \text{ (where } c \text{is a constant).}$$

When adding random variables, the resulting uncertainty is the quadrature sum of the individual uncertainties:

$$x' = \sum_i x_i \quad \Longrightarrow \quad \sigma_{x'} = \sqrt{\sum_i \sigma_{x_i}^2}.$$

These relations can be combined to give

$$x' = c' \sum_i c_i x_i \quad \Longrightarrow \quad \sigma_{x'} = c' \sqrt{\sum_i \left(c_i \sigma_{x_i}\right)^2}.$$

Comparing this result to Equation 4.16 demonstrates that the uncertainty of the k^{th} channel of the resampled spectrum is

$$\sigma_k^R = \frac{1}{\Delta f_k^R} \sqrt{\sum_i \left(F_{ki}\right)^2 \left(\sigma_i\right)^2}. \tag{4.20}$$

4.4.2 Spectral Definitions

We can now apply the results of Section 4.4.1 to resample the spectra used for the deconvolution. The following discussion assumes that we have a set of DSB data, $\boldsymbol{d^{o,W}}$, the corresponding USB and LSB frequencies for each of these channels, and the frequency bins corresponding to the desired SSB spectrum.

To fully define each of these spectra, we require a set of frequencies representing the channel centers, $\boldsymbol{f}$, and another set representing the width of each channel, $\boldsymbol{\Delta f}$. The SSB, USB, and LSB spectral channels can then be represented by the vectors $\boldsymbol{f^{SSB}}$, $\boldsymbol{\Delta f^{SSB}}$, $\boldsymbol{f^{USB,W}}$, $\boldsymbol{\Delta f^{USB,W}}$, $\boldsymbol{f^{LSB,W}}$, and $\boldsymbol{\Delta f^{LSB,W}}$.

For each individual spectrum in $\boldsymbol{d^{o,W}}$, assume that the USB channel centers are sorted in ascending order while the LSB channel centers are sorted in descending order such that $f_i^{LSB,W} > f_{i+1}^{LSB,W}$. The individual sets of frequencies are stacked together end to end in the same manner as $\boldsymbol{d^{o,W}}$ to form $\boldsymbol{f^{USB,W}}$ and $\boldsymbol{f^{LSB,W}}$. Likewise, assume the SSB channel centers are sorted in ascending order. The USB and LSB channels for each spectrum are not independent; both correspond to an underlying set of IF channel spacings used in the spectrometer (Equation 4.2). Therefore, we must have $\Delta f_i^{LSB,W} = \Delta f_i^{USB,W}$.

From these vectors, we can easily calculate several vectors corresponding to the edges of each channel, such that the ith channel covers the frequency range c_i to c_{i+1}:

$$
\begin{aligned}
c_i^{SSB} &= \begin{cases} f_1^{SSB} - \frac{1}{2}\Delta f_1^{SSB} & \text{for } i = 1 \\ f_i^{SSB} + \frac{1}{2}\Delta f_{i-1}^{SSB} & \text{for } i > 1 \end{cases} \\
c_i^{USB,W} &= \begin{cases} f_1^{USB,W} - \frac{1}{2}\Delta f_1^{USB,W} & \text{for } i = 1 \\ f_i^{USB,W} + \frac{1}{2}\Delta f_{i-1}^{USB,W} & \text{for } i > 1 \end{cases} \\
c_i^{LSB,W} &= \begin{cases} f_1^{LSB,W} + \frac{1}{2}\Delta f_1^{LSB,W} & \text{for } i = 1 \\ f_i^{LSB,W} - \frac{1}{2}\Delta f_{i-1}^{LSB,W} & \text{for } i > 1. \end{cases}
\end{aligned}
\tag{4.21}
$$

As with the frequency vectors, the USB and SSB channel-edge vectors are stored in ascending order while the LSB vector is sorted in descending order such that $c_i^{LSB,W} > c_{i+1}^{LSB,W}$ within each spectrum. If there are N channels in a particular spectrum, there will be $(N + 1)$ values in the channel-edge vector.

4.4.3 Aligned SSB and DSB Spectra

Using the notation of Section 4.4.2, we can now define the concept of "frequency-aligned" spectra from Section 4.3 more precisely. In order to use Equation 4.3 for the convolution matrix, the edges of each DSB channel must match the edges of two channels in the SSB spectrum, one for the upper sideband, and one for the lower sideband. Figure 4.3 gives an overview of how we can resample the spectra to make this happen. The top plot shows the spectrum as measured in the spectrometer; a particular channel (green) is measured at an absolute IF frequency of f_0. When converted to a DSB spectrum (second diagram), there is ambiguity about the actual frequency of the observed channel. One way to represent this observation is to plot the spectrum with two frequency axes. The LSB axis, shown at the top of the plot, increases to the left while the USB axis, bottom, increases to the right. The green channel appears at a frequency offset of f_0 relative to the LO frequency, f_{LO}, shown at the left edge of the plot. When deconvolving the spectrum, the power in the green channel could be assigned to the lower sideband, to the upper sideband, or split between the two, as shown by the light green channels in the two SSB spectra at the bottom of the figure. The bottom diagram corresponds to the SSB spectrum sampled on the desired output channels, and the third plot shows that same spectrum, resampled onto channels that are aligned with the DSB spectrum. Note that the LO frequency falls on the boundary of a channel in the resampled SSB spectrum, as required by Equation 4.24. For simplicity, the step of resampling the DSB spectra to match the SSB channel width is not shown; if needed, it would represent an additional step between the first and second plots.

Converting these concepts into equations, we can say that LSB channel i aligns with SSB channel j if

$$c_i^{LSB} = c_{j+1}^{SSB} \text{ and} \tag{4.22a}$$

$$c_{i+1}^{LSB} = c_j^{SSB}, \tag{4.22b}$$

and that USB channel i aligns with SSB channel k if

$$c_i^{USB} = c_k^{SSB} \text{ and} \tag{4.22c}$$

$$c_{i+1}^{USB} = c_{k+1}^{SSB}. \tag{4.22d}$$

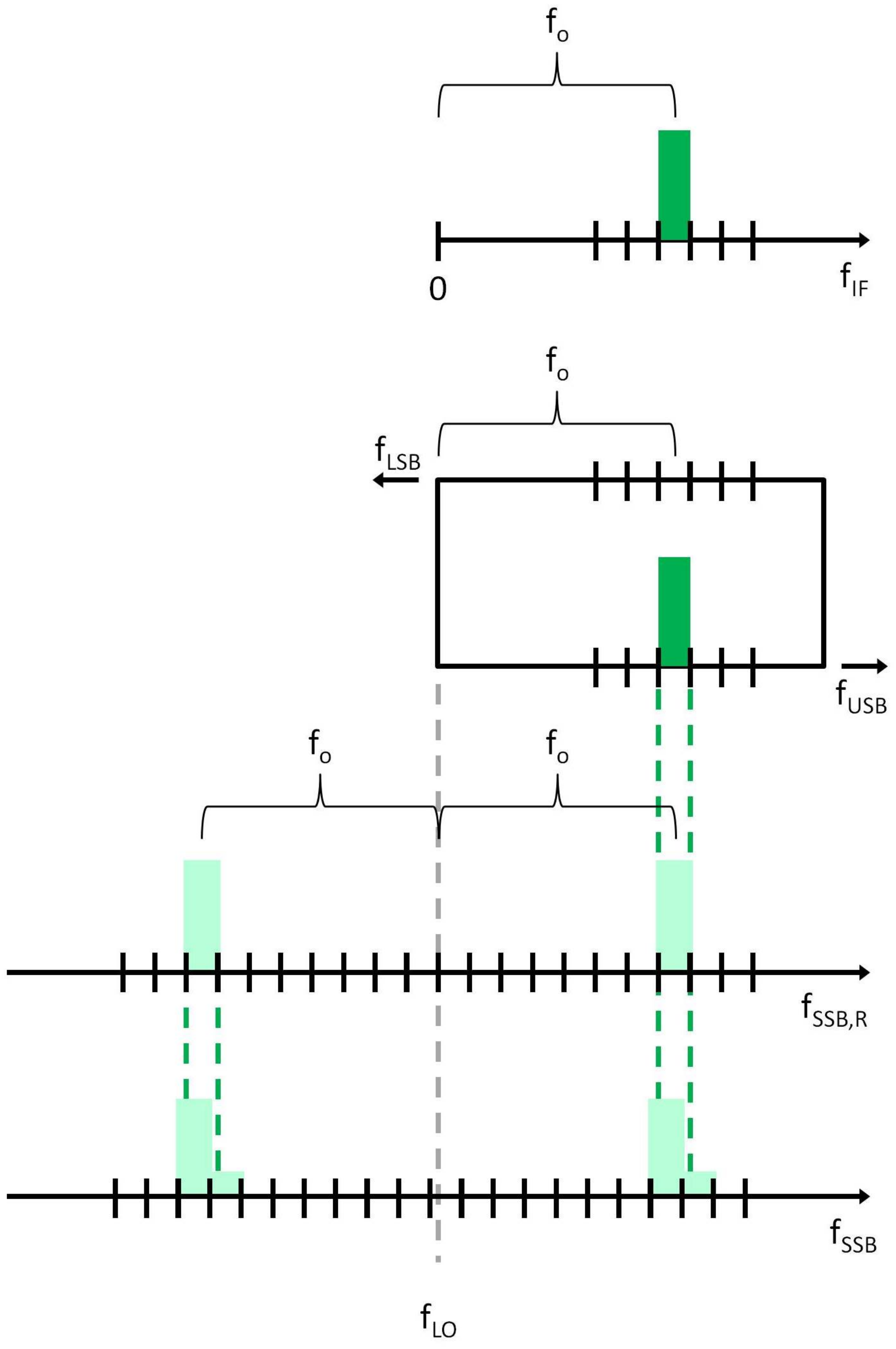


Figure 4.2: Generating frequency-aligned spectra.

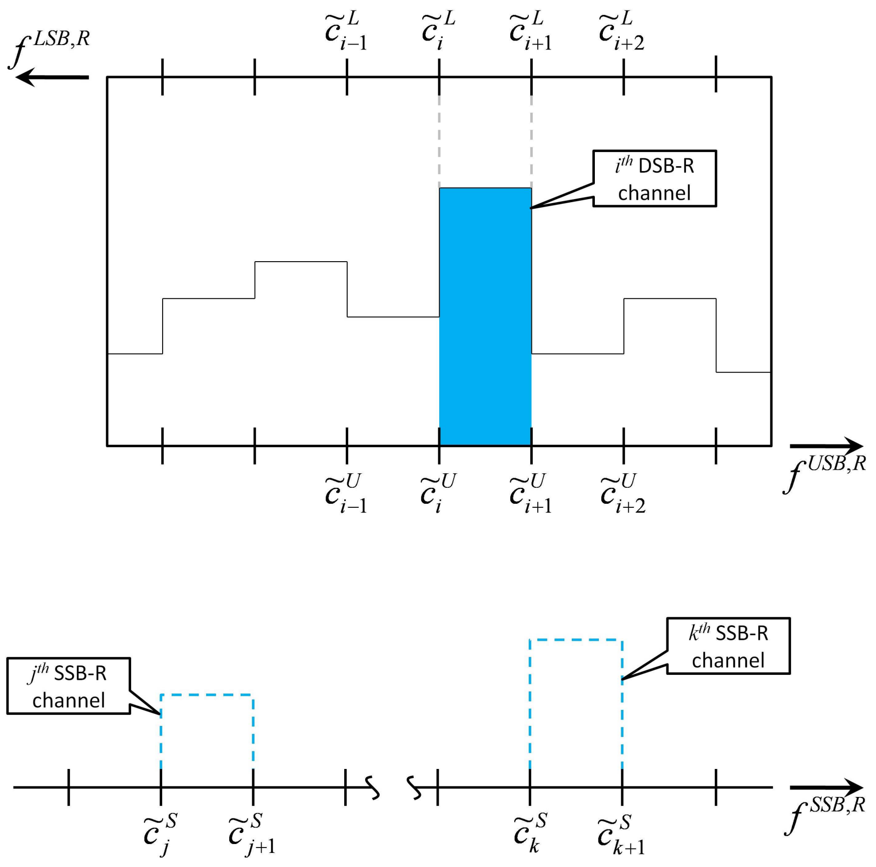


Figure 4.3: Spectral definitions. The top plot shows a DSB spectrum, with two frequency axes corresponding to the possible frequencies of the observed channel. The LSB axis, top, increases to the left while the USB axis, bottom, increases to the right. In order to be aligned with the DSB spectrum, the SSB spectrum (bottom plot) must have channels with edges that match either the LSB channel limits or the USB channel limits in the DSB spectrum.

We define the DSB and SSB spectra as frequency-aligned if values of j and k can be found to satisfy these equations for every value of i.

From Equations 4.2, we know that

$$f_{IF} = f_{LO} - f_{LSB} = f_{USB} - f_{LO},$$

so that for any frequency in the DSB spectrum, the corresponding LSB and USB frequencies are related by the equation

$$f_{LSB} = 2f_{LO} - f_{USB}.$$

In particular, this equation can be applied to Equation 4.22a to give

$$c_i^{LSB} = 2f_{LO} - c_i^{USB} = c_{j+1}^{SSB}.$$

Substituting Equation 4.22c for c_i^{USB} gives

$$2f_{LO} - c_k^{SSB} = c_{j+1}^{SSB}. \tag{4.23}$$

In order to simplify this result further, assume that the channel size is constant across the SSB spectrum $\left(\Delta f_i^{SSB} = \Delta f_o^{SSB}\right)$. In that case, any two channel boundaries in the SSB spectrum must be separated by an integral number of frequency steps

$$c_k^{SSB} - c_{j+1}^{SSB} = N\Delta f_o^{SSB},$$

where N is an integer. Combining this with equation 4.23 yields the important result

$$\begin{aligned} 2f_{LO} - c_k^{SSB} &= c_k^{SSB} - N\Delta f_o^{SSB} \text{ so that} \\ c_k^{SSB} &= f_{LO} + \frac{N}{2}\Delta f_o^{SSB}. \end{aligned} \tag{4.24}$$

In other words, in order for the SSB and DSB spectra to be frequency aligned, the LO frequency *must* either fall on a channel boundary or in the center of a channel in the SSB spectrum, as shown in the third plot in Figure 4.2.

One way of understanding this constraint is shown in Figure 4.4. RF downconversion essentially "folds" the spectrum around the LO. The two sidebands are then added together and integrated over discrete channels to form the resulting double-sideband spectrum. Deconvolution attempts to "unfold" this process to recover the original spectrum, albeit in a channelized form. Intuitively, the DSB and SSB channels can only be frequency aligned if the SSB channels are set up so that the LO frequency (the folding line) occurs at the edge or the center of a channel.

For a single observation, it is always possible to find a set of SSB channels that meets this requirement. In fact, given that N in Equation 4.24 is a free parameter, there are in principle an infinite number of sets of SSB channels that would satisfy this requirement, each with slightly different channel width. However, this freedom disappears when the

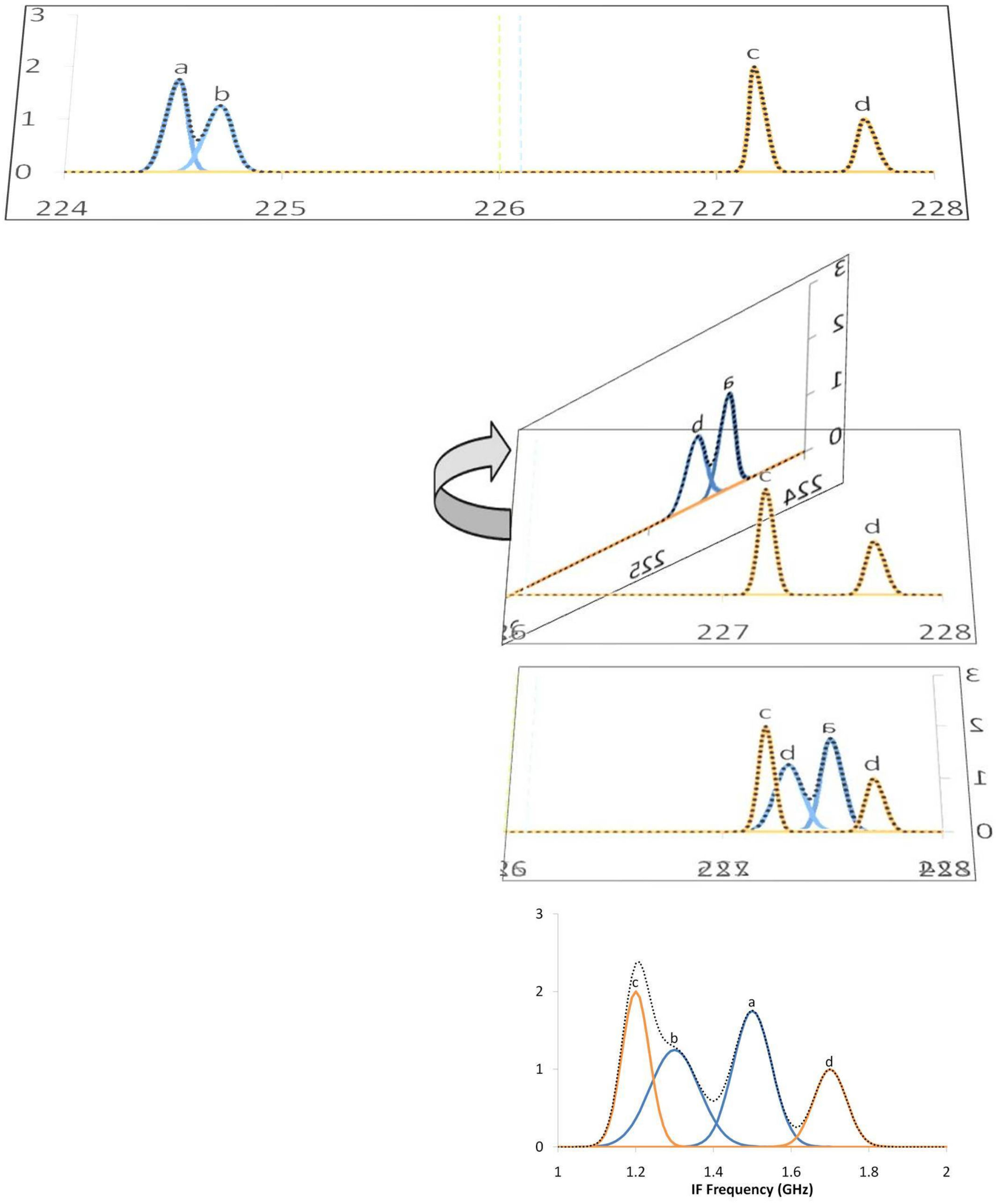


Figure 4.4: Convolution as spectral folding. The top diagram shows a notional SSB spectrum containing four peaks. If we consider an LO frequency of 226 GHZ (dashed green line), then two of the peaks are in the LSB (blue), and two are in the USB (orange). Sideband convolution can be viewed as a "folding" of the spectrum about the LO frequency such that a mirrored version of the LSB is superimposed on the USB. The spectrum taken at the IF output of a double-sideband receiver (such as Z-Rex) can be represented by the black line in the bottom plot. At each IF frequency, the power from the LSB is added to that of the USB, and the contribution of each sideband cannot be determined from a single observation. (See Figure 4.6 for an example of how this ambiguity can be resolved.)

remaining constraints (Equations 4.22b and 4.22d) are incorporated. Equations 4.22a and 4.22b can be rewritten

$$c_i^{LSB} = c_{j+1}^{SSB} = c_j^{SSB} + \Delta f_o^{SSB} \text{ and}$$
$$c_j^{SSB} = c_{i+1}^{LSB} = c_i^{LSB} - \Delta f_i^{LSB}.$$

The second equation can be substituted for c_j^{SSB} in the first, giving

$$c_i^{LSB} = c_i^{LSB} - \Delta f_i^{LSB} + \Delta f_o^{SSB} \text{ and}$$
$$\Delta f_i^{LSB} = \Delta f_o^{SSB}. \tag{4.25a}$$

Likewise, equations 4.22d can be written as

$$c_i^{USB} + \Delta f_i^{USB} = c_k^{SSB} + \Delta f_o^{SSB},$$

which can be simplified using Equation 4.22c to produce a similar results for the USB channel sizes:

$$\Delta f_i^{USB} = \Delta f_o^{SSB}. \tag{4.25b}$$

Equations 4.25 require that the DSB and SSB channel sizes be the same width, removing the flexibility introduced by the value of N in Equation 4.24. Nonetheless, for a given DSB spectrum and LO setting, there is always a set of SSB channel spacings satisfying these requirements, allowing the simple model presented in Section 4.3 to be applied.

However, while this approach works for a *single* DSB spectrum, in general no set of SSB channel spacings can be found that would satisfy these constraints for multiple DSB spectra. The only way to achieve this goal would be to choose LO settings that were multiples of $\Delta f_0^{SSB}/2$. Not only would this significantly complicate the observing process, but such a choice would adversely impact the quality of the deconvolved spectrum, as discussed at the beginning of Section 4.4.

Therefore, is necessary to add an extra step to the deconvolution. Equation 4.4 proposes a model in which the SSB and DSB spectra are linked by a simple convolution matrix M, comprised of ones and zeros. This simple form of M can be maintained if $\boldsymbol{s}$ is resampled onto frequency-aligned channels; however, there must be N_{obs} resampled SSB spectra, one for each DSB spectrum. Let the n^{th} such spectrum be represented by

$$\boldsymbol{s}^{(n),\boldsymbol{R}} = \boldsymbol{S}^{(n)} \cdot \boldsymbol{s}, \tag{4.26}$$

where $\boldsymbol{S}^{(n)}$ is a matrix that resamples $\boldsymbol{s}$ according to the method represented by Equation 4.19. Note that $\boldsymbol{s}^{\boldsymbol{R}}$ only needs to include frequencies that are part of either the upper or lower sidebands of the corresponding DSB spectrum.

Under this formulation, $\boldsymbol{s}^{\boldsymbol{R}}$ must have channel spacings equal to those in the n^{th} DSB spectrum. However, it does not make sense to use channel spacings considerably smaller than the underlying "master spectrum," $\boldsymbol{s}$. The resampling calculations are not meaningful for that case, plus it increases computing time without producing any improvements in the final spectrum. Therefore, we resample the DSB data to give it channel widths similar to those in $\boldsymbol{s}$. In an analogous fashion to Equation 4.26, this can be represented as

$$\boldsymbol{d}^{o(n),\boldsymbol{R}} = \boldsymbol{D}^{(n)} \cdot \boldsymbol{d}^{o(n),\boldsymbol{W}}, \tag{4.27}$$

where $\boldsymbol{D}^{(n)}$ resamples the DSB data according to Equation 4.19.

Each of the resampling equations can be simplified by stacking the spectra end to end to give vectors $\boldsymbol{s}^{\boldsymbol{R}}$ and $\boldsymbol{d}^{o,\boldsymbol{R}}$, containing all of the individual resampled SSB spectra and resampled DSB spectra, respectively. The individual resampling matrices, $\boldsymbol{S}^{(n)}$ and $\boldsymbol{D}^{(n)}$, can likewise be assembled into matrices, $\boldsymbol{S}$ and $\boldsymbol{D}$. (See Figure 4.5.) The resampling equations can be summarized as

$$\begin{aligned} \boldsymbol{s}^{\boldsymbol{R}} &= \boldsymbol{S} \cdot \boldsymbol{s} \text{ and} \\ \boldsymbol{d}^{o,\boldsymbol{R}} &= \boldsymbol{D} \cdot \boldsymbol{d}^{o,\boldsymbol{W}}. \end{aligned} \tag{4.28}$$

4.4.4 Convolution Model with Non-Aligned Spectra

With these changes, the model of Section 4.3 becomes

$$\boldsymbol{d}^{\boldsymbol{R}} = \boldsymbol{M}^{\boldsymbol{R}} \cdot \boldsymbol{s}^{\boldsymbol{R}}, \tag{4.31}$$

$$\begin{array}{c}N_{SSB}^{R}\\ \text{elements}\end{array}\left\{\begin{pmatrix}\uparrow\\ \boldsymbol{s^{R1}}\\ \downarrow\\ \hdashline \uparrow\\ \boldsymbol{s^{R2}}\\ \downarrow\\ \hdashline \vdots\\ \hdashline \uparrow\\ \boldsymbol{s^{RN_{obs}}}\\ \downarrow\end{pmatrix}\right. = \begin{pmatrix}\boldsymbol{S^{1}}\\ \hdashline \boldsymbol{S^{2}}\\ \hdashline \vdots\\ \hdashline \boldsymbol{S^{N_{obs}}}\end{pmatrix}\cdot\left.\begin{pmatrix}\uparrow\\ \boldsymbol{s}\\ \downarrow\end{pmatrix}\right\}\begin{array}{c}N_{SSB}\\ \text{elements}\end{array} \tag{4.29}$$

$$\begin{array}{c}N_{DSB}^{R}\\ \text{elements}\end{array}\left\{\begin{pmatrix}\uparrow\\ \boldsymbol{d^{o,R,1}}\\ \downarrow\\ \hdashline \uparrow\\ \boldsymbol{d^{o,R,2}}\\ \downarrow\\ \hdashline \vdots\\ \hdashline \uparrow\\ \boldsymbol{d^{o,R,N_{obs}}}\\ \downarrow\end{pmatrix}\right. = \begin{pmatrix}\boxed{\boldsymbol{D^{1}}} & 0 & 0 & 0\\ 0 & \boxed{\boldsymbol{D^{2}}} & 0 & 0\\ & & \ddots & \\ 0 & 0 & 0 & \boxed{\boldsymbol{D^{N_{obs}}}}\end{pmatrix}\cdot\left.\begin{pmatrix}\uparrow\\ \boldsymbol{d^{o,W,1}}\\ \downarrow\\ \hdashline \uparrow\\ \boldsymbol{d^{o,W,2}}\\ \downarrow\\ \hdashline \vdots\\ \hdashline \uparrow\\ \boldsymbol{d^{o,W,N_{obs}}}\\ \downarrow\end{pmatrix}\right\}\begin{array}{c}N_{DSB}\\ \text{elements}\end{array} \tag{4.30}$$

Figure 4.5: Assembling resampled $\boldsymbol{s^{(n),R}}$ (top) and $\boldsymbol{d^{o(n),R}}$ (bottom) into single vectors, along with the associated resampling equations.

where $\boldsymbol{M^R}$ keeps the simple zero-or-one form described in Equation 4.3, and markers have been added as a reminder that $\boldsymbol{d^R}$ should be compared to $\boldsymbol{d^{o,R}}$, not $\boldsymbol{d^{o,W}}$.

4.5 Incorporating Unequal Sideband Gains

In deriving the simple models of the previous sections, several important effects were excluded. Now that the basic aspects of the model have been described, it is time to incorporate those modifications, starting with the receiver's sideband gains. The results of Section 2.1 demonstrate that the signals from the two sidebands need to be treated separately in the presence of unequal sideband gains. We can rewrite Equation 2.14 as

$$T_A^* = (1-\gamma)\, T_A^{*(LSB)} + (1+\gamma)\, T_A^{*(USB)}, \tag{4.32}$$

where η_{src} and η_{cold} have been incorporated into each term by defining $T_A^{*(LSB)} = \eta_{src}\eta_{cold}T_{src}^{LSB}$ and $T_A^{*(USB)} = \eta_{src}\eta_{cold}T_{src}^{USB}$.

The derivation in Section 2.1 considers only the output of a single spectrometer channel. It is relatively easy to generalize the results to cover multi-channel spectra taken at a variety of LO frequencies, but it will require some additional bookkeeping. In particular, γ cannot be represented as a single, fixed scalar. Instead, the sideband-gain imbalance must be treated as a frequency-dependent function, $\gamma\left(f_{LO}, f_{IF}\right)$. Experience with these receivers indicates that γ changes relatively slowly with frequency, allowing us to approximate it as a series of flat gains represented as the components of a vector, $\boldsymbol{\gamma}$.

The formalism developed for previous models (e.g., Equation 4.4) can be applied by noting that spectra at the CSO are recorded using the T_A^* scale [Peng, 2002]. Therefore, we can associate Equation 4.32 with a single channel in the modeled DSB data, $\boldsymbol{d^R}$ while $T_A^{*(LSB)}$ and $T_A^{*(USB)}$ are just the appropriate channels of $\boldsymbol{s^R}$:

$$d_i^{(a),R} = \left[1-\gamma\left(f_{LO}, f_{IF}\right)\right] s_j^{(a),R} + \left[1+\gamma\left(f_{LO}, f_{IF}\right)\right] s_j^{(a),R}, \tag{4.33}$$

Assume that channel i falls in the a^{th} DSB spectrum and that γ changes sufficiently slowly that all of the a^{th} DSB spectrum can be modeled using a single value of the sideband gain, γ^a. Then, as before, we can use a convolution matrix to generate a set of predicted data:

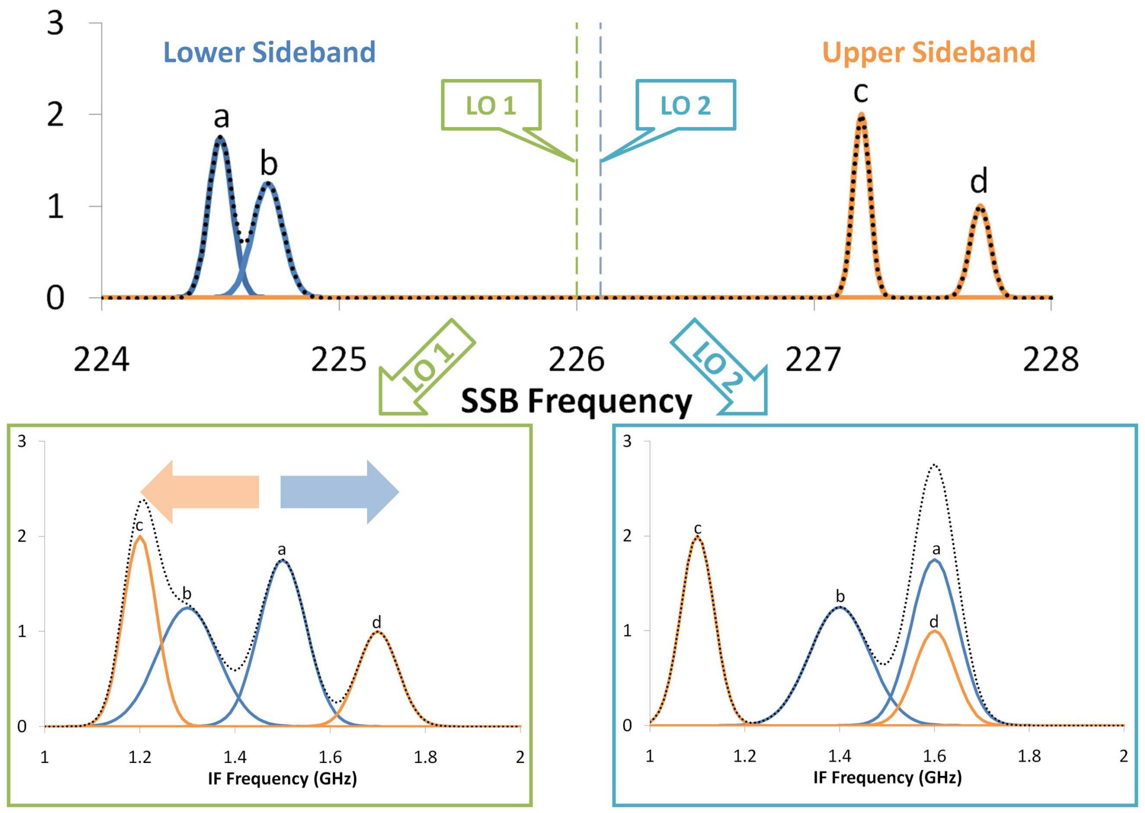


Figure 4.6: As shown in Figure 4.4, a single DSB observation cannot determine which sideband produced a given peak; however, that ambiguity can be resolved by performing additional observations at other LO frequencies. Here, the top plot shows the true SSB spectrum, and the green line (marked “LO 1”) corresponds to the LO used in Figure 4.4. The black dashed line in the lower left plot corresponds to the spectrum that would be seen using that LO. By shifting the LO to a slightly higher frequency (“LO 2”), we can resolve the sideband ambiguity. As shown by the orange arrow, peaks in the USB will move to lower IF frequencies while LSB peaks will move to higher IF frequencies. The spectrum that would be generated using the higher LO setting is represented by the black line in the lower right plot. Due to peak blending, the original sideband for each peak cannot be determined by simple inspection, but the algorithms described in this work can be used to estimate the original SSB spectrum.

$$d_i^{(a),R} = \sum_j \left[\left(1 - \gamma^a\right) M_{ij}^{LSB(a),R} + \left(1 + \gamma^a\right) M_{ij}^{USB(a),R} \right] s_j^{(a),R}. \tag{4.34}$$

This result also includes the appropriate resampling to ensure that the SSB and DSB channels are frequency-aligned (Section 4.4.1).

To accommodate the unequal sideband gains, the convolution matrix has been divided into two parts, corresponding to the downconversion from the different sidebands. Conversion from the upper sideband is represented by $\boldsymbol{M^{USB,R}}$, where

$$M_{ij}^{USB,R} = \begin{cases} 1 & \text{if } f_i^{USB,R} = f_j^{SSB,R} \\ 0 & \text{otherwise} \end{cases}. \tag{4.35a}$$

Likewise, lower-sideband conversion is represented by $\boldsymbol{M^{LSB,R}}$, where

$$M_{ij}^{LSB,R} = \begin{cases} 1 & \text{if } f_i^{LSB,R} = f_j^{SSB,R} \\ 0 & \text{otherwise} \end{cases}. \tag{4.35b}$$

Comparing these definitions to Equation 4.3 demonstrates that these matrices are just the single-sideband versions of the previous double-sideband deconvolution matrix.

Rearranging Equation 4.34 and converting to matrix form gives the slightly simpler result

$$\boldsymbol{d^{(a),R}} = \left(\boldsymbol{M^{\Sigma(a),R}} + \gamma^a \boldsymbol{M^{\Delta(a),R}} \right) \cdot \boldsymbol{s^{(a),R}}, \tag{4.36}$$

where we have defined

$$\begin{aligned} \boldsymbol{M^{\Sigma,R}} &= \boldsymbol{M^{USB,R}} + \boldsymbol{M^{LSB,R}} \text{ and} \\ \boldsymbol{M^{\Delta,R}} &= \boldsymbol{M^{USB,R}} - \boldsymbol{M^{LSB,R}}. \end{aligned} \tag{4.37}$$

To generalize this to the full set of DSB data, we can define a sideband-gain matrix, $\boldsymbol{\Gamma^R}$, with the individual sideband-gain parameters arranged along the diagonal as shown in Figure 4.7.

Equation 4.36 can then be seen to be the a^{th} component of the more general equation

$$\boldsymbol{d^R} = \left(\boldsymbol{M^{\Sigma,R}} + \boldsymbol{\Gamma^R} \cdot \boldsymbol{M^{\Delta,R}} \right) \cdot \boldsymbol{s^R}, \tag{4.39}$$

$$\boldsymbol{\Gamma^R} = \begin{pmatrix} \boxed{\begin{matrix} \gamma^1 & 0 & & 0 \\ 0 & \gamma^1 & & 0 \\ & & \ddots & \\ 0 & 0 & & \gamma^1 \end{matrix}} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boxed{\begin{matrix} \gamma^2 & 0 & & 0 \\ 0 & \gamma^2 & & 0 \\ & & \ddots & \\ 0 & 0 & & \gamma^2 \end{matrix}} & \mathbf{0} & \mathbf{0} \\ & & \ddots & \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boxed{\begin{matrix} \gamma^{N_\gamma} & 0 & & 0 \\ 0 & \gamma^{N_\gamma} & & 0 \\ & & \ddots & \\ 0 & 0 & & \gamma^{N_\gamma} \end{matrix}} \end{pmatrix}. \quad (4.38)$$

Figure 4.7: Sideband-gain matrix, $\boldsymbol{\Gamma^R}$.

which can be further simplified to

$$\boldsymbol{d}^{\boldsymbol{R}} = \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{R}} \cdot \boldsymbol{s}^{\boldsymbol{R}} \tag{4.40}$$

by using the definition

$$\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{R}} = \boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{R}} + \boldsymbol{\Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{R}}. \tag{4.41}$$

In subsequent sections, we derive methods for finding the best-fit values of the sideband-gain parameters for a given set of DSB data. In the calculation, it is important to have a compact way of representing the unique elements of $\boldsymbol{\Gamma}^{\boldsymbol{R}}$ (namely, the N_a components of γ). To assist in that calculation, we define the set of truncated identity functions, $\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}$, which are $N^R_{DSB} \times N^R_{DSB}$ matrices with ones along a section of the diagonal and zeros elsewhere. The a^{th} matrix, $\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}$, has ones along the diagonal in the positions corresponding to the channels in $\boldsymbol{d}^{\boldsymbol{R}}$ that are modeled using the a^{th} sideband gain, γ^a. (See Figure 4.8.)

Thus, the series of matrices looks like

$$\mathbb{1}^{\boldsymbol{1},\boldsymbol{R}} = \begin{pmatrix} \boxed{\mathbb{1}} & \boldsymbol{0} & & \boldsymbol{0} \\ \boldsymbol{0} & \boxed{\boldsymbol{0}} & & \boldsymbol{0} \\ & & \ddots & \\ \boldsymbol{0} & \boldsymbol{0} & & \boxed{\boldsymbol{0}} \end{pmatrix}, \quad \mathbb{1}^{\boldsymbol{2},\boldsymbol{R}} = \begin{pmatrix} \boxed{\boldsymbol{0}} & \boldsymbol{0} & & \boldsymbol{0} \\ \boldsymbol{0} & \boxed{\mathbb{1}} & & \boldsymbol{0} \\ & & \ddots & \\ \boldsymbol{0} & \boldsymbol{0} & & \boxed{\boldsymbol{0}} \end{pmatrix},$$

$$\cdots, \quad \mathbb{1}^{\boldsymbol{N_\gamma},\boldsymbol{R}} = \begin{pmatrix} \boxed{\boldsymbol{0}} & \boldsymbol{0} & & \boldsymbol{0} \\ \boldsymbol{0} & \boxed{\boldsymbol{0}} & & \boldsymbol{0} \\ & & \ddots & \\ \boldsymbol{0} & \boldsymbol{0} & & \boxed{\mathbb{1}} \end{pmatrix}. \tag{4.43}$$

For later calculations, it is useful to note that

$$\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} \cdot \mathbb{1}^{\boldsymbol{b},\boldsymbol{R}} = \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} \delta_{ab}. \tag{4.44}$$

This definition allows us to write the sideband-gain matrix as

$$\boldsymbol{\Gamma}^{\boldsymbol{R}} = \sum_{a=1}^{N_\gamma} \gamma^a \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}. \tag{4.45}$$

$$\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} = \begin{pmatrix} \boxed{\begin{matrix}0&0&&0\\0&0&&0\\&&\ddots&\\0&0&&0\end{matrix}} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \ddots & & & & & \vdots \\ \mathbf{0} & \mathbf{0} & \boxed{\begin{matrix}0&0&&0\\0&0&&0\\&&\ddots&\\0&0&&0\end{matrix}} & \mathbf{0} & \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \boxed{\begin{matrix}1&0&&0\\0&1&&0\\&&\ddots&\\0&0&&1\end{matrix}} & \mathbf{0} & & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boxed{\begin{matrix}0&0&&0\\0&0&&0\\&&\ddots&\\0&0&&0\end{matrix}} & & \mathbf{0} \\ \vdots & & & & & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \boxed{\begin{matrix}0&0&&0\\0&0&&0\\&&\ddots&\\0&0&&0\end{matrix}} \end{pmatrix}. \tag{4.42}$$

Figure 4.8: Sample truncated-identity matrix $\left(\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}\right)$. All elements are zero except for a few ones along the diagonal (shown in red). These positions correspond to the elements of $\boldsymbol{d}^{\boldsymbol{R}}$ that are affected by the value of the a^{th} sideband gain, γ^{a}.

For simplicity, our implementation of this model uses a separate sideband gain parameter for each DSB spectrum $\left(N_\gamma = N_{obs}\right)$. However, it is easy to use fewer or more sideband-gain parameters by modifying the $\mathbb{1}^{a,R}$ matrices accordingly. This might be particularly useful if one of the sidebands contains strong atmospheric lines, which can significantly alter the atmospheric opacity over a small frequency range. The impact of such lines on the final deconvolved spectrum could be minimized by assigning additional sideband gains to cover the affected frequencies.

4.6 Spectral Baselines

DSB spectra often have baselines that must be removed before further analysis. These can be caused by instrumental artifacts in the spectrometer, imperfect subtraction in the off-position, and standing waves in the optics or IF processing equipment. These are usually assumed to be well represented by low-order polynomials, although the third effect in particular can add a sinusoidal ripple to the baseline.

Previous line surveys usually removed these baselines by fitting low-order polynomials to the individual DSB spectra and then performing the deconvolution. However, it is often difficult to separate artificial baselines from actual emission, particularly in line-confused spectra. If we instead include the baseline fit as part of the deconvolution, we can take advantage of all available information to try to produce the most accurate deconvolved spectrum. Therefore, in addition to the convolution introduced by the downconversion, we now add a baseline component to the model:

$$\boldsymbol{d}^{\boldsymbol{R}} = \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{R}} \cdot \boldsymbol{s}^{\boldsymbol{R}} + \eta_B \boldsymbol{\beta}^{\boldsymbol{R}}, \tag{4.46}$$

where $\boldsymbol{\beta}^{\boldsymbol{R}}$ represents the set of baselines, sampled onto the same channels as $\boldsymbol{d}^{\boldsymbol{R}}$. The factor of η_B should be set to either 0 or 1 and merely provides an easy way of removing the baseline term from the model if desired.

Our model allows the baseline to include arbitrary functions of the channel number (or equivalently the channel frequency), but it must be linear in those functions. The baseline for the i^{th} channel of the a^{th} DSB spectrum can therefore be represented as

$$\beta_i^{R,a} = \sum_n b_n^a C_n^{R,a}(i)$$

where $C_n^{R,a}(i)$ is a channel- or frequency-dependent function, and the optimization process is used to generate the coefficients b_n^a. With an eye toward maintaining reasonable computation times, the model does not allow $C_n^{R,a}(i)$ to contain any adjustable parameters to be fit. In particular, sinusoidal components with an adjustable period cannot be included, which seems consistent with the approach taken in previous surveys. Our deconvolution software uses only a low-order polynomial for the baseline; however, in principle, any fixed function of spectrometer channel frequencies could be used.

The baseline terms can be easily incorporated into the previous model by defining a matrix, $\boldsymbol{C^{R,a}}$, that contains the result of evaluating each of the $C_n^{R,a}(i)$ over all of the channels of an individual DSB spectrum:

$$C_{ij}^{R,a} = \begin{pmatrix} \boldsymbol{C_1^{R,a}}(1) & \boldsymbol{C_2^{R,a}}(1) & \cdots \\ \boldsymbol{C_1^{R,a}}(2) & \boldsymbol{C_2^{R,a}}(2) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}. \tag{4.47}$$

As a simplistic example, consider representing the baseline as a second-order polynomial using the following functions of the channel number, i:[7]

$$\begin{gathered} C_1^{R,a}(i) = i^0, \\ C_2^{R,a}(i) = i^1, \text{ and} \\ C_3^{R,a}(i) = i^2. \end{gathered} \tag{4.48}$$

Then the the baseline for the a^{th} DSB spectrum will be equal to

[7]These functions are chosen for illustrative clarity; however, these are not actually the appropriate functions to use for a second-order fit. To enable faster convergence and more accurate fits, the baseline functions should be orthogonal to each other. (See Section 4.6.1.)

$$\boldsymbol{\beta}^{R,a} = \begin{pmatrix} 1^0 & 1^1 & 1^2 \\ 2^0 & 2^1 & 2^2 \\ 3^0 & 3^1 & 3^2 \\ & \vdots & \\ \left(N_{DSB}^{R,a}\right)^0 & \left(N_{DSB}^{R,a}\right)^1 & \left(N_{DSB}^{R,a}\right)^2 \end{pmatrix} \begin{pmatrix} b_1^a \\ b_2^a \\ b_3^a \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \\ & \vdots & \\ 1 & N_{DSB}^{R,a} & \left(N_{DSB}^{R,a}\right)^2 \end{pmatrix} \begin{pmatrix} b_1^a \\ b_2^a \\ b_3^a \end{pmatrix},$$

where b_1^a represents the magnitude of the zeroth-order term, and so forth. The full set of baselines can be generated as a single vector with the same length as $\boldsymbol{d}^{\boldsymbol{R}}$ by joining all of the baseline coefficients into one vector, and creating a block-diagonal matrix of the $\boldsymbol{C}^{R,a}$ to give

$$\begin{pmatrix} \uparrow \\ \boldsymbol{\beta}^{R,1} \\ \downarrow \\ \hdashline \uparrow \\ \boldsymbol{\beta}^{R,2} \\ \downarrow \\ \hdashline \\ \vdots \\ \end{pmatrix} = \left(\begin{array}{c:c:c} \boldsymbol{C}^{R,1} & \boldsymbol{0} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{C}^{R,2} & \boldsymbol{0} \\ \hdashline \boldsymbol{0} & \boldsymbol{0} & \ddots \end{array}\right) \cdot \begin{pmatrix} \uparrow \\ \boldsymbol{b} \text{ for spectrum 1} \\ \downarrow \\ \hdashline \uparrow \\ \boldsymbol{b} \text{ for spectrum 2} \\ \downarrow \\ \hdashline \\ \vdots \end{pmatrix}, \tag{4.49}$$

which can be expressed in compact form as

$$\boldsymbol{\beta}^{\boldsymbol{R}} = \boldsymbol{C}^{\boldsymbol{R}} \cdot \boldsymbol{b}. \tag{4.50}$$

4.6.1 Orthogonal Baseline Functions

We use second-order polynomials as the highest-order function for our baseline fits, as those are sufficient for most of the spectra from the CSO 4-GHz AOS, as long as observations are taken under reasonable conditions. While the simple set of baseline functions presented in Equation 4.48 includes polynomials up through second order, that formulation creates difficulties for the optimization routine. For instance, the mean value of each of the three functions is non-zero; therefore, the optimizer could represent a constant offset in a spectrum using any of the three functions, or some combination thereof. This equivalence makes it difficult for the optimizer to find the best set of parameters, forcing it to put considerable effort into searching through different combinations of baseline functions.

This problem can be eliminated by choosing baseline functions that are orthogonal to each other. For convenience, the equations can also be chosen to have unity normalization, resulting in a set of orthonormal baseline functions. If the spectrum under consideration spans a channel range from $i = \{i_{min}, \ldots, i_{max}\}$, the orthonormality requirement can be represented by the equation

$$\sum_{i=i_{min}}^{i_{max}} f_i^{(j)} f_i^{(k)} = \delta_{jk}, \tag{4.51}$$

where $f_i^{(0)}$, $f_i^{(1)}$, and $f_i^{(2)}$ represent the constant, linear, and quadratic baseline functions, respectively, evaluated at the i^{th} channel. These three functions can be represented in the most general terms as

$$\begin{aligned} f_i^{(0)} &= c_0 \\ f_i^{(1)} &= l_1 \left(\frac{i - i_o}{n_o}\right) + l_0 \\ f_i^{(2)} &= q_2 \left(\frac{i - i_o}{n_o}\right)^2 + q_1 \left(\frac{i - i_o}{n_o}\right) + q_0, \end{aligned} \tag{4.52}$$

where i_o and n_o represent a reference channel and normalization constant, respectively, that can be chosen to keep the factors in parentheses from getting too large. The coefficients (c_0, l_1, l_0, q_2, q_1, and q_0) can be set by requiring that Equation 4.51 be satisfied for any combination of $f_i^{(0)}$, $f_i^{(1)}$, and $f_i^{(2)}$. This represents a system of six nonlinear equations with six unknowns. There are several self-consistent sets of coefficients that satisfy these constraints; we arbitrarily choose the set in which all of the leading coefficients (c_0, l_1, and q_2) are positive, yielding a unique solution (for arbitrary i_o and n_o):

$$
\begin{aligned}
c_0 &= \frac{1}{\sqrt{\Delta i + 1}} \\
l_1 &= \frac{2\sqrt{3} n_o}{\sqrt{\Delta i\,(\Delta i + 1)\,(\Delta i + 2)}} \\
l_0 &= -\frac{\sqrt{3}(i_{max} + i_{min} - 2i_o)}{\sqrt{\Delta i\,(\Delta i + 1)\,(\Delta i + 2)}} \\
q_2 &= \frac{6\sqrt{5} n_o^2}{\sqrt{(\Delta i - 1)\,\Delta i\,(1 + \Delta i)\,(\Delta i + 2)\,(\Delta i + 3)}} \\
q_1 &= -\frac{6\sqrt{5}(i_{max} + i_{min} - 2i_o) n_o}{\sqrt{(\Delta i - 1)\,\Delta i\,(\Delta i + 1)\,(\Delta i + 2)\,(\Delta i + 3)}} \\
q_0 &= \frac{\sqrt{5}\left(-i_{max} + i_{max}^2 + i_{min} + 4 i_{max} i_{min} + i_{min}^2 - 6(i_{max} + i_{min}) i_o + 6 i_o^2\right)}{\sqrt{(\Delta i - 1)\,\Delta i\,(\Delta i + 1)\,(\Delta i + 2)\,(\Delta i + 3)}},
\end{aligned}
$$

where $\Delta i = i_{max} - i_{min}$. These equations can be (somewhat) simplified by an appropriate choice for i_o. To satisfy the orthogonality requirements, the linear and quadratic terms must have zero means across the spectrum, implying

$$
\sum_{i=i_{min}}^{i_{max}} f_i^{(1)} = \sum_{i=i_{min}}^{i_{max}} f_i^{(2)} = 0.
$$

For the linear term, this can be easily satisfied by setting i_o to be the midpoint of the spectrum, $i_o = \frac{1}{2}(i_{max} + i_{min})$. Using $n_o = \frac{1}{2}\Delta i$ helps to keep the coefficients of reasonable order even for larger spectra. With these substitutions, the coefficients for the polynomials are

$$
\begin{aligned}
c_0 &= \frac{1}{\sqrt{\Delta i + 1}} \\
l_1 &= \frac{\sqrt{3}\Delta i}{\sqrt{\Delta i\,(\Delta i + 1)\,(\Delta i + 2)}} \\
l_0 &= 0 \\
q_2 &= \frac{3\sqrt{5}(\Delta i)^2}{2\sqrt{(\Delta i - 1)\,\Delta i\,(\Delta i + 1)\,(\Delta i + 2)\,(\Delta i + 3)}} \\
q_1 &= 0 \\
q_0 &= -\frac{\sqrt{5\Delta i}\,(\Delta i + 2)}{2\sqrt{(\Delta i - 1)\,(\Delta i + 1)\,(\Delta i + 2)\,(\Delta i + 3)}} = -\frac{(\Delta i + 2)}{3\Delta i} q_2.
\end{aligned}
\tag{4.53}
$$

The set of orthonormal baselines represented by Equations 4.53 are used in the current implementation of the deconvolution algorithm.

4.7 Normalization of Spectra

In the following sections, it is convenient to normalize the predicted and measured DSB data sets by their uncertainties such that

$$\begin{aligned} d_i^{NR} &= \frac{d_i^R}{\sigma_i^R} \text{ and} \\ d_i^{o,NR} &= \frac{d_i^{o,R}}{\sigma_i^R}, \end{aligned} \tag{4.54}$$

where σ_i^R represents the uncertainty in the i^{th} channel of the resampled DSB spectrum (Equation 4.20). To simplify these equations, we define a normalization matrix, $\mathbb{A}^{\boldsymbol{\sigma,R}}$, consisting of $\frac{1}{\sigma_i^R}$ along its diagonal,

$$\mathbb{A}^{\boldsymbol{\sigma,R}} = \begin{pmatrix} \frac{1}{\sigma_1^R} & 0 & 0 & 0 \\ 0 & \frac{1}{\sigma_2^R} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \frac{1}{\sigma_{N_{DSB}}^R} \end{pmatrix}, \tag{4.55}$$

in which case $\boldsymbol{d^{NR}}$ and $\boldsymbol{d^{o,NR}}$ can be written as

$$\begin{aligned} \boldsymbol{d^{NR}} &= \mathbb{A}^{\boldsymbol{\sigma,R}} \cdot \boldsymbol{d^R} \text{ and} \\ \boldsymbol{d^{o,NR}} &= \mathbb{A}^{\boldsymbol{\sigma,R}} \cdot \boldsymbol{d^{o,R}}. \end{aligned} \tag{4.56}$$

Similarly, we can also normalize the baselines:

$$\boldsymbol{\beta^{NR}} = \mathbb{A}^{\boldsymbol{\sigma,R}} \cdot \boldsymbol{\beta^R}. \tag{4.57}$$

4.8 Full Convolution Model

Rather than explicitly including the normalization and SSB-resampling matrices in every equation, we define normalized convolution matrices that incorporate $\mathbb{A}^{\boldsymbol{\sigma,R}}$ and $\boldsymbol{S}$:

$$\begin{aligned} \boldsymbol{M}^{\Gamma,NR} &= \mathbb{A}^{\sigma,R} \cdot \boldsymbol{M}^{\Gamma,R} \cdot \boldsymbol{S}, \\ \boldsymbol{M}^{\Sigma,NR} &= \mathbb{A}^{\sigma,R} \cdot \boldsymbol{M}^{\Sigma,R} \cdot \boldsymbol{S}, \text{ and} \\ \boldsymbol{M}^{\Delta,NR} &= \mathbb{A}^{\sigma,R} \cdot \boldsymbol{M}^{\Delta,R} \cdot \boldsymbol{S}. \end{aligned} \tag{4.58}$$

Because $\boldsymbol{\Gamma}^{R}$ and $\mathbb{A}^{\sigma,R}$ are both diagonal matrices that can be safely interchanged during multiplication, $\boldsymbol{M}^{\Gamma,NR}$ may be written in terms of $\boldsymbol{M}^{\Sigma,NR}$ and $\boldsymbol{M}^{\Delta,NR}$:

$$\boldsymbol{M}^{\Gamma,NR} = \boldsymbol{M}^{\Sigma,NR} + \boldsymbol{\Gamma}^{R} \cdot \boldsymbol{M}^{\Delta,NR}. \tag{4.59}$$

To simplify equations, the normalization matrix can also be pulled into $\boldsymbol{C}^{R}$ to give

$$\boldsymbol{C}^{NR} = \mathbb{A}^{\sigma,R} \cdot \boldsymbol{C}^{R},$$

so that

$$\boldsymbol{\beta}^{NR} = \boldsymbol{C}^{NR} \cdot \boldsymbol{b} \tag{4.60}$$

The full convolution model can be written as

$$\begin{aligned} \boldsymbol{d}^{R} &= \boldsymbol{M}^{\Gamma,R} \cdot \boldsymbol{s}^{R} + \eta_B \boldsymbol{C}^{R} \cdot \boldsymbol{b}, \text{ and} \\ \boldsymbol{d}^{NR} &= \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}. \end{aligned} \tag{4.61}$$

For later reference, it is useful to expand $\boldsymbol{d}^{NR}$ in terms of the unknown parameters:

$$\boldsymbol{d}^{NR} = \left[\boldsymbol{M}^{\Sigma,NR} + \left(\sum_a \gamma^a \mathbb{1}^{a,R} \right) \cdot \boldsymbol{M}^{\Delta,NR} \right] \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}. \tag{4.62}$$

The adjustable parameters that can be used to optimize the value of $\boldsymbol{d}^{NR}$ are γ, $\boldsymbol{s}$, and $\boldsymbol{b}$. The remaining quantities are fixed matrices that can be calculated once during the setup phase of the deconvolution and do not need to be adjusted during each iteration.

4.9 Figure of Merit

As outlined in Section 4.2.1, the sideband deconvolution can be performed using an optimization routine that seeks to minimize an error function by adjusting the values of $\boldsymbol{\gamma}$, $\boldsymbol{s}$, and $\boldsymbol{b}$ in Equation 4.62. We define our figure of merit to be a slightly modified version of χ^2:

$$\chi^2_{mod} = \lambda_\chi \sum_i \frac{\left(d_i^R - d_i^{o,R}\right)^2}{\left(\sigma_i^R\right)^2} + \lambda_S \boldsymbol{s}^T \cdot \boldsymbol{s} + \eta_B \lambda_B \sum_i \frac{\left(\beta_i^R\right)^2}{\left(\sigma_i^R\right)^2} + \lambda_G \boldsymbol{\gamma}^T \cdot \boldsymbol{\gamma}, \tag{4.63}$$

where the first term represents the typical definition of χ^2, and the superscripted T indicates the matrix transpose. We can further simplify this equation by absorbing σ_i^R into the numerators of the first and third terms. By defining the normalized DSB spectra (Equation 4.56) and the normalized baselines (Equation 4.57), we can rewrite Equation 4.63 as

$$\chi^2_{mod} = \lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}^T} - \boldsymbol{d}^{\boldsymbol{o,NR}^T}\right) \cdot \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o,NR}}\right) + \lambda_S \boldsymbol{s}^T \cdot \boldsymbol{s} + \eta_B \lambda_B \boldsymbol{\beta}^{\boldsymbol{NR}^T} \cdot \boldsymbol{\beta}^{\boldsymbol{NR}} + \lambda_G \boldsymbol{\gamma}^T \cdot \boldsymbol{\gamma}. \tag{4.64}$$

The final three terms can be used to adjust the solution found by the optimization routine while the various λ factors control the strength of each of these effects.

If $\lambda_S > 0$, the second term, $\lambda_S \boldsymbol{s}^T \cdot \boldsymbol{s}$, encourages the optimizer to minimize the norm of $\boldsymbol{s}$ unless the data require otherwise. In order to increase the magnitude of a given s_i, the optimizer must determine that the corresponding decrease in the first term overcomes the penalty represented by the second term. In particular, this term forces s_i to zero if that frequency is not included in any of the DSB spectra; otherwise, such channels are unconstrained and can cause the optimization to become unstable.

Similarly, the third term in Equation 4.63, $\eta_B \lambda_B \boldsymbol{\beta}^{\boldsymbol{NR}^T} \cdot \boldsymbol{\beta}^{\boldsymbol{NR}}$, forces the routine to minimize the norm of the baseline. Using the normalized baselines, $\boldsymbol{\beta}^{\boldsymbol{NR}}$, prevents the algorithm from working too hard to fine-tune the baseline of an inherently poor spectrum. In principle, balancing the value of λ_B with respect to λ_S should encourage the algorithm to preferentially attribute flux to the single-sideband spectrum if possible. In practice, however, we have not found this necessary and usually set $\lambda_B = 0$. As discussed previously (see Equation 4.46), the η_B factor can be used to turn off the baseline term. The final term,

$\lambda_G \gamma^T \cdot \gamma$, pulls the sideband gains as close to to unity as possible, although we often use $\lambda_G = 0$ as well.

To simplify calculations, we use the following equations to indicate the individual terms of Equation 4.64:

$$\chi^2_{o,i} = \lambda_\chi \left(d_i^{NR} - d_i^{o,NR}\right)^2, \tag{4.65a}$$

$$\chi^2_{S,j} = \lambda_S \left(s_j\right)^2, \tag{4.65b}$$

$$\chi^2_{\beta,k} = \eta_B \lambda_B \left(\beta_k^{NR}\right)^2, \text{ and} \tag{4.65c}$$

$$\chi^2_{\gamma,m} = \lambda_G \left(\gamma^m\right)^2. \tag{4.65d}$$

Equation 4.64 can then be written

$$\chi^2_{mod} = \sum_{i=1}^{N_{DSB}^R} \chi^2_{o,i} + \sum_{j=1}^{N_{SSB}} \chi^2_{S,j} + \sum_{k=1}^{N_{DSB}^R} \chi^2_{\beta,k} + \sum_{m=1}^{N_\gamma} \chi^2_{\gamma,m}. \tag{4.66}$$

For use in later sections, we write each of these quantities in terms of the model parameters. Equations 4.65b and 4.65d are already expressed in that form while Equation 4.65c can be easily expanded via Equation 4.50. Using Equation 4.61, we can write Equation 4.65a as

$$\chi^2_{o,i} = \lambda_\chi \left[\left(M_{ik}^{\Sigma,NR} + \sum_{a=1}^{N_\gamma} \gamma^a \mathbb{1}_{ij}^{a,R} M_{jk}^{\Delta,NR}\right) s_k + \eta_B C_{ij}^{NR} b_j - d_i^{o,NR}\right]^2,$$

where repeated indices on matrices and vectors represent implicit sums. Since the a index on γ^a and $\mathbb{1}^a$ is non-standard, that sum is represented explicitly.

4.9.1 Continuation Solution

In early tests of the deconvolution algorithm, we found that the optimizer sometimes got stuck on a clearly incorrect solution. It appeared to be trapped in a local minimum, with no way to explore the rest of the χ^2_{mod} space to find an improved solution. To help guide the optimizer to a reasonable set of parameters, we have implemented a "continuation solution" method. The first term of χ^2_{mod} contains a multiplicative factor, λ_χ, that can be used to implement the continuation solution. Initially, λ_χ is set to some small value so that the other terms dominate χ^2_{mod}. The optimizer generates a solution for that small value of λ_χ; then λ_χ is increased and the previous solution is used as the starting point for the new optimization. The process is repeated until $\lambda_\chi = 1$.

The value of the optimization parameters at the end of the first iteration is determined by the λ_S, λ_B, and λ_G terms, presumably causing the parameters to go to zero (or at least to a well-understood value) at a point representing the global minimum of the χ^2_{mod} surface. If λ_χ is increased slowly enough, the position of the global minimum in the parameter space will move only slightly for the next iteration. The optimizer will start near this location, which should help it to find the new global minimum rather than wandering off to a different local minimum. We have found that adjusting $\lambda_\chi$in logarithmic steps

$$\lambda_\chi = \{\lambda_\chi^{min}, 10^{\Delta L}\lambda_\chi^{min}, 10^{2\Delta L}\lambda_\chi^{min}, \ldots, \lambda_\chi^{max}\} \text{ with } \Delta L = \frac{\log_{10}\lambda_\chi^{max} - \log_{10}\lambda_\chi^{min}}{N_{steps} - 1}$$

works better than linear steps

$$\lambda_\chi = \{\lambda_\chi^{min}, \lambda_\chi^{min} + \Delta\lambda_\chi, \lambda_\chi^{min} + 2\Delta\lambda_\chi, \ldots, \lambda_\chi^{max}\} \text{ with } \Delta\lambda_\chi = \frac{\lambda_\chi^{max} - \lambda_\chi^{min}}{N_{steps} - 1}.$$

In general, λ_χ^{max} should be equal to 1; setting $\lambda_\chi^{min} = 10^{-6}$ with $N_{steps} = 12$ logarithmic steps worked well in our testing.

4.10 Optimizing χ^2_{mod}

When selecting an optimization method to minimize χ^2_{mod}, there are several general features of the model to keep in mind. First, the spectral values in the single-sideband spectrum are multiplied by the sideband-gain parameters (Equation 4.61), making the problem nonlinear in its independent parameters. Because the problem is nonlinear, there is no guarantee that the minimum identified by the algorithm represents a global minimum, rather than a local minimum. Second, the problem is typically over constrained, as most line surveys have considerably more constraints (represented by the number of elements in $\boldsymbol{d}^{o,W}$) than independent parameters. This is a desirable situation that improves the deconvolution routine's ability to recover the full spectrum. Finally, when fitting such a large number of parameters, it is easy for the optimization algorithms to wander off in entirely the wrong direction.[8]

The most direct approach to minimizing χ^2_{mod} is to use an algorithm capable of finding the minimum (or maximum) value of an arbitrary multi-dimensional function. Usually this is known simply as parameter optimization, but we will refer to it as "direct optimization" to distinguish from the other optimization methods discussed below. Not surprisingly, nonlinear parameter estimation is commonly encountered in scientific computing, and there are a variety of algorithms designed for this purpose [see, for example, Press et al., 1992, Chap. 10].

Another method is to take the derivative of χ^2_{mod} with respect to each of the independent parameters and set the results equal to zero. This generates a set of equations $\frac{\partial \chi^2_{mod}}{\partial x_i} = 0$, where x_i represents the set of independent parameters. Solving for the x_i yields a set of parameters corresponding to a *local* minimum or maximum. As with direct optimization, nonlinear root finding also represents a major branch of numerical computing, and there are many algorithms devoted to this purpose [e.g., Press et al., 1992, Chap. 9].

Finally, we have investigated the viability of a hybrid model, using a mix of direct-optimization and root-finding algorithms. As seen in Equation 4.61, the equations underlying this model come tantalizingly close to being linear. If the sideband-gain parameters could somehow be known *a priori*, the remaining parameters would be related in a linear fashion. We capitalize on this fact by using a direct-optimization routine in an "outer loop" to find the values of the sideband gains. Inside that loop, we treat these parameters as fixed, allowing us to apply *linear* root-finding techniques. These have the advantage of being very fast; in addition, for a given set of sideband gains, the linear problem has a unique solution corresponding to the global minimum of the χ^2_{mod} surface.[9]

[8] If one were to imagine the chi-square surface in terms of its geographical analog, the algorithms have a particularly hard time dealing with broad, flat "plains" instead of deep, well-defined "bowls." The algorithm dutifully explores this plain, eventually finding its lowest point, rather than crossing the nearby ridge to find the even deeper valley beyond.

[9] Ultimately, however, this is still a local minimum, as the values of the sideband-gain parameters are determined by a nonlinear optimizer.

In principle, any of these methods should produce the correct answer; however, as is often the case with scientific computing, stability and efficiency are important issues. In practice, we have had the best results using the direct optimization and hybrid approaches, although the root-finding method works exceptionally well if the sideband gains are in fact known. In both the direct optimization and the hybrid method, the tendency of the direct-optimization algorithm to wander off from the desired solution can be mitigated by selecting small steps for λ_χ.

In the following sections, we discuss each of these approaches in more detail and describe the implementation of each that we have developed using MATLAB,[10] which provides libraries of general-purpose functions for optimization and root finding.

Usually, the algorithms require a set of parameters to use as the starting point for the first iteration. Setting all of the parameters to zero has a certain aesthetic appeal, as it represents an unbiased method of initiating the algorithm. However, we have found some instances in which the algorithms have difficulty moving off of zero. Therefore, we typically use a set of small random numbers for the first iteration.

Several of the MATLAB functions need information about the relevant derivatives, expressed in the form of a Jacobian matrix, to update the parameter values from one iteration to the next. MATLAB can either estimate the Jacobian numerically, or the user can provide a function capable of calculating the Jacobian. As demonstrated in the following examples, calculating an analytic Jacobian can be tedious; however, it dramatically reduces execution time. Otherwise, MATLAB is forced to generate a discrete approximation to the Jacobian. On current computing hardware, numerically estimating the Jacobian is not feasible for realistically sized data sets. However, MATLAB's ability to numerically estimate the Jacobian provides a valuable method of validating the analytic results.

[10]While this discussion focuses on MATLAB, most numerical-analysis packages contain similar capabilities, or routines can be built from scratch using the advice of references such as Press et al. [1992].

4.11 Direct Optimization of χ^2_{mod}

From an implementation perspective, directly minimizing χ^2_{mod} is simple. While MATLAB offers several general-purpose optimization routines, we have found the `lsqnonlin()` function to be particularly useful. It is part of the MATLAB Optimization Toolbox, and it is specifically designed for least-squares problems of this form.

The `lsqnonlin()` function assumes the user seeks to minimize a multivariable function F, which can be represented as the sum of the squares of a set of functions, $\{f_i\}$:

$$\text{Find } \boldsymbol{x} \text{ that minimizes } F(\boldsymbol{x}) = \sum_i |f_i(\boldsymbol{x})|^2 .$$

By comparison to Equations 4.65 and 4.66, χ^2_{mod} can be cast in the form required by `lsq-nonlin()` if we make the following identifications:

$$f_{o,i}(\boldsymbol{s},\boldsymbol{b},\boldsymbol{\gamma}) = \sqrt{\lambda_\chi}\left(d_i^{NR} - d_i^{o,NR}\right), \tag{4.67a}$$

$$f_{S,j}(\boldsymbol{s}) = \sqrt{\lambda_S}\left(s_j\right), \tag{4.67b}$$

$$f_{\beta,k}(\boldsymbol{b}) = \sqrt{\eta_B \lambda_B}\left(\beta_k^{NR}\right), \tag{4.67c}$$

$$\text{and } f_{\gamma,m}(\boldsymbol{\gamma}) = \sqrt{\lambda_G}\left(\gamma^m\right). \tag{4.67d}$$

In that case, χ^2_{mod} is equal to the function F used by `lsqnonlin()`:

$$\chi^2_{mod} = F(\boldsymbol{s},\boldsymbol{b},\boldsymbol{\gamma}) = \sum_{i=1}^{N_{DSB}^R} f_{o,i}^2 + \sum_{j=1}^{N_{SSB}} f_{S,j}^2 + \sum_{k=1}^{N_{DSB}^R} f_{\beta,k}^2 + \sum_{m=1}^{N_\gamma} f_{\gamma,m}^2.$$

One interesting feature of `lsqnonlin()`is that it optimizes $\boldsymbol{x}$ by analyzing the full vector of values comprised of $\{f_i(\boldsymbol{x})\}$, rather than just the scalar quantity $F(\boldsymbol{x})$; therefore, the vector of values that should be returned to it is

$$\left(\begin{array}{ccc:ccc:ccc:ccc} f_{o,1} & \cdots & f_{o,N_{DSB}^R} & f_{S,1} & \cdots & f_{S,N_{SSB}} & f_{\beta,1} & \cdots & f_{\beta,N_{DSB}^R} & f_{\gamma,1} & \cdots & f_{\gamma,N_\gamma} \end{array}\right). \tag{4.68}$$

For this problem, the Jacobian is defined as

$$J_{ij}^{opt} = \frac{\partial f_{\chi,i}}{\partial x_j}, \tag{4.69}$$

where $f_{\chi,i}$ represents $f_{o,i}$, $f_{S,i}$, $f_{\beta,i}$, or $f_{\gamma,i}$, and x_j represents the unknown parameters

$$\boldsymbol{x} = \left(\begin{array}{ccc:ccc:ccc} s_1 & \cdots & s_{N_{SSB}} & b_1 & \cdots & b_{N_{coeff}} & \gamma^1 & \cdots & \gamma^{N_\gamma} \end{array} \right). \tag{4.70}$$

If the unknowns are sorted in the order shown in Equation 4.70, then the full Jacobian can be written as a block matrix:

$$\boldsymbol{J^{opt}} = \left(\begin{array}{c:c:c} \dfrac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{s}} & \dfrac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{b}} & \dfrac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{\gamma}} \\ \hdashline \dfrac{\partial \boldsymbol{f_S}}{\partial \boldsymbol{s}} & \dfrac{\partial \boldsymbol{f_S}}{\partial \boldsymbol{b}} & \dfrac{\partial \boldsymbol{f_S}}{\partial \boldsymbol{\gamma}} \\ \hdashline \dfrac{\partial \boldsymbol{f_\beta}}{\partial \boldsymbol{s}} & \dfrac{\partial \boldsymbol{f_\beta}}{\partial \boldsymbol{b}} & \dfrac{\partial \boldsymbol{f_\beta}}{\partial \boldsymbol{\gamma}} \\ \hdashline \dfrac{\partial \boldsymbol{f_\gamma}}{\partial \boldsymbol{s}} & \dfrac{\partial \boldsymbol{f_\gamma}}{\partial \boldsymbol{b}} & \dfrac{\partial \boldsymbol{f_\gamma}}{\partial \boldsymbol{\gamma}} \end{array} \right), \tag{4.71}$$

where each of the entries represents a matrix in its own right. For instance, $\frac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{s}}$ is a matrix in which component $\left[\frac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{s}}\right]_{ij} = \frac{\partial f_{o,i}}{\partial s_j}$.

In the direct-optimization case, the Jacobian matrix is easy to calculate as all of the parameters $(\boldsymbol{s}, \boldsymbol{b},$ and $\boldsymbol{\gamma})$ are assumed to be independent from one another. From Equations 4.67 it is clear that many of the sub-matrices in the Jacobian are simply equal to 0 since they do not have any dependence on the differentiation variable:

$$\begin{array}{ll} \frac{\partial \boldsymbol{f_S}}{\partial \boldsymbol{b}} = 0 & \frac{\partial \boldsymbol{f_S}}{\partial \boldsymbol{\gamma}} = 0 \\ \frac{\partial \boldsymbol{f_\beta}}{\partial \boldsymbol{s}} = 0 & \frac{\partial \boldsymbol{f_\beta}}{\partial \boldsymbol{\gamma}} = 0 \\ \frac{\partial \boldsymbol{f_\gamma}}{\partial \boldsymbol{s}} = 0 & \frac{\partial \boldsymbol{f_\gamma}}{\partial \boldsymbol{b}} = 0 \end{array} . \tag{4.72}$$

The first of the remaining elements, $\frac{\partial \boldsymbol{f_o}}{\partial \boldsymbol{s}}$, can be calculated from Equations 4.67a, 4.61, and 4.28:

$$\begin{aligned}
\left[\frac{\partial f_o}{\partial \boldsymbol{s}}\right]_{ij} &= \sqrt{\lambda_\chi}\frac{\partial}{\partial s_j}\left(d_i^{NR} - d_i^{o,NR}\right) \\
&= \sqrt{\lambda_\chi}\frac{\partial}{\partial s_j}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \\
&= \sqrt{\lambda_\chi}\frac{\partial}{\partial s_j}\sum_k M_{ik}^{\Gamma,NR} s_k \\
&= \sqrt{\lambda_\chi}\sum_k M_{ik}^{\Gamma,NR}\frac{\partial s_k}{\partial s_j} \\
&= \sqrt{\lambda_\chi}\sum_k M_{ik}^{\Gamma,NR}\delta_{jk} \\
&= \sqrt{\lambda_\chi}M_{ij}^{\Gamma,NR}.
\end{aligned}$$

To avoid repeating similar steps for each calculation, common derivatives have been calculated in Sections A.1 and A.2 and tallied in Tables A.1 and A.2. The previous result can be determined directly from Equation A.13.

The expanded form of $\boldsymbol{d}^{\boldsymbol{N}}$ from Equation 4.62 can be used to determine $\frac{\partial f_o}{\partial \gamma}$:

$$\begin{aligned}
\left[\frac{\partial f_o}{\partial \gamma}\right]_{ij} &= \sqrt{\lambda_\chi}\frac{\partial}{\partial \gamma^j}\left(d_i^{NR} - d_i^{o,NR}\right) \\
&= \sqrt{\lambda_\chi}\left[\mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \quad \text{(by Eq. A.22)},
\end{aligned}$$

where $\left[\mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i$ represents the i^{th} component of the matrix multiplication $\mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}$.

From Equation 4.46, combined with Equation 4.50, it can be seen that $\boldsymbol{d}^{\boldsymbol{NR}}$ has a simple dependence on $\boldsymbol{b}$:

$$\begin{aligned}
\left[\frac{\partial f_o}{\partial \boldsymbol{b}}\right]_{ij} &= \sqrt{\lambda_\chi}\frac{\partial}{\partial b_j}\left(d_i^{NR} - d_i^{o,NR}\right) \\
&= \eta_B\sqrt{\lambda_\chi}C_{ij}^N \quad \text{(by Eq. A.14)}.
\end{aligned}$$

The derivatives for the remaining terms be calculated quite easily. Equation 4.67b gives

$$\begin{aligned}
\left[\frac{\partial f_S}{\partial \boldsymbol{s}}\right]_{ij} &= \frac{\partial f_{S,i}}{\partial s_j} \\
&= \sqrt{\lambda_S}\frac{\partial s_i}{\partial s_j} \\
&= \sqrt{\lambda_S}\delta_{ij} \\
&= \sqrt{\lambda_S}\mathbb{1}_{ij}.
\end{aligned} \tag{4.73}$$

Similarly, Equation 4.67d allows us to calculate

$$\begin{aligned}\left[\frac{\partial f_\gamma}{\partial \gamma}\right]_{ij} &= \frac{\partial f_{\gamma,i}}{\partial \gamma^j} \\ &= \sqrt{\lambda_G}\frac{\partial \gamma^i}{\partial \gamma^j}. \\ &= \sqrt{\lambda_G}\mathbb{1}_{ij}.\end{aligned} \tag{4.74}$$

Equations 4.50 and 4.67c can be combined to give

$$\begin{aligned}\left[\frac{\partial f_\beta}{\partial \boldsymbol{b}}\right]_{ij} &= \frac{\partial f_{\beta,i}}{\partial b_j} \\ &= \sqrt{\eta_B\lambda_B}\frac{\partial}{\partial b_j}\left(\sum_k C^{NR}_{ik} b_k\right) \\ &= \sqrt{\eta_B\lambda_B}C^{NR}_{ij} \quad \text{(by Eq. A.2)}.\end{aligned} \tag{4.75}$$

Combining all of these results yields the full Jacobian for direct optimization of χ^2_{mod}:

$$\boldsymbol{J}^{opt} = \left(\begin{array}{c:c:c} \sqrt{\lambda_\chi}\boldsymbol{M}^{\Gamma,NR} & \eta_B\sqrt{\lambda_\chi}\boldsymbol{C}^{NR} & \boldsymbol{J}^{opt(\gamma)} \\ \hdashline \sqrt{\lambda_S}\mathbb{1} & 0 & 0 \\ \hdashline 0 & \sqrt{\eta_B\lambda_B}\boldsymbol{C}^{NR} & 0 \\ \hdashline 0 & 0 & \sqrt{\lambda_G}\mathbb{1} \end{array}\right), \tag{4.76}$$

where

$$J^{opt(\gamma)}_{ij} = \sqrt{\lambda_\chi}\left[\mathbb{1}^{j,R}\cdot\boldsymbol{M}^{\Delta,NR}\cdot\boldsymbol{s}\right]_i. \tag{4.77}$$

4.12 Optimization of χ^2_{mod} via Nonlinear Root Finding

4.12.1 Nonlinear Equations

An alternative approach to minimizing χ^2_{mod} is to take the derivative of χ^2_{mod} with respect to each of the independent parameters and to set the result equal to zero. This generates a set of coupled, nonlinear equations; if a set of parameters $(\boldsymbol{s}, \boldsymbol{b},$ and $\gamma)$ can be found that simultaneously satisfies these equations, the solution represents a critical point (local minimum, local maximum, or saddle point) in the χ^2_{mod} surface.

Taking the partial derivatives of χ^2_{mod} with respect to s_n and setting the result equal to zero generates a set of equations that can be used to determine the values of the deconvolved spectrum:

$$\frac{\partial \chi^2_{mod}}{\partial s_n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial s_n} + \sum_j \frac{\partial \chi^2_{S,j}}{\partial s_n} = 0. \tag{4.78}$$

Similarly, the coefficients of the baseline terms can be found by taking derivatives with respect to b_n and setting the result equal to zero:

$$\frac{\partial \chi^2_{mod}}{\partial b_n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial b_n} + \sum_k \frac{\partial \chi^2_{\beta,k}}{\partial b_n} = 0. \tag{4.79}$$

Finally, the set of sideband gains can be found by taking the derivative of χ^2_{mod} with respect to γ^n and setting the result equal to zero:

$$\frac{\partial \chi^2_{mod}}{\partial \gamma^n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial \gamma^n} + \sum_m \frac{\partial \chi^2_{\gamma,m}}{\partial \gamma^n} = 0. \tag{4.80}$$

The details of these calculations are reserved for Appendix A.3. The three conditions shown above generate a set of nonlinear equations, which can be copied from Equations A.23, A.24, and A.25:

$$\lambda_\chi \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) + \lambda_S \boldsymbol{s} = 0, \tag{4.81a}$$

$$\eta_B \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \left[\lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) + \lambda_B \boldsymbol{\beta}^{\boldsymbol{NR}}\right] = 0, \text{ and} \tag{4.81b}$$

$$\lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)^T \cdot \mathbb{1}^{\boldsymbol{n},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^n = 0. \tag{4.81c}$$

Note that the first line represents a vector equation corresponding to N_{SSB} components while the second represents N_{coeff} components, and the third represents N_γ equations $(n = 1...N_\gamma)$.

4.12.2 Jacobian

As in Section 4.11, the Jacobian is relatively easy to calculate since all of the parameters are assumed to be independent from one another. If the unknown parameters are sorted in the order shown in Equation 4.70, the full Jacobian can then be written in the following form:

$$J^{nonlin} = \left(\begin{array}{c:c:c} \dfrac{\partial L_1}{\partial s_n} & \dfrac{\partial L_1}{\partial b_p} & \dfrac{\partial L_1}{\partial \gamma^q} \\ \hdashline \dfrac{\partial L_2}{\partial s_n} & \dfrac{\partial L_2}{\partial b_p} & \dfrac{\partial L_2}{\partial \gamma^q} \\ \hdashline \dfrac{\partial L_3}{\partial s_n} & \dfrac{\partial L_3}{\partial b_p} & \dfrac{\partial L_3}{\partial \gamma^q} \end{array}\right), \tag{4.82}$$

where $L_{1,i}$, $L_{2,i}$, and $L_{3,i}$ represent the i^{th} components of the left-hand sides of the first, second, and third lines in Equations 4.81, respectively. (Also see Equations A.27.) The Jacobian is derived in Appendix A.4.1, with the result contained in Equations A.27 and A.28:

$$J^{nlin} = \left(\begin{array}{c:c:c} \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} + \lambda_S \mathbb{1} & \lambda_\chi \eta_B \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} & \boldsymbol{J}^{nlin(L_1,\gamma)} \\ \hdashline \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} & \eta_B \left(\eta_B \lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} & \boldsymbol{J}^{nlin(L_2,\gamma)} \\ \hdashline \boldsymbol{J}^{nlin(L_3,s)} & \boldsymbol{J}^{nlin(L_3,b)} & \boldsymbol{J}^{nlin(L_3,\gamma)} \end{array}\right),$$

where:

$$\begin{aligned} J_{ij}^{nlin(L_1,\gamma)} &= \lambda_\chi \left[\left(\boldsymbol{M}^{\Sigma,NR^T} + 2\gamma^j \boldsymbol{M}^{\Delta,NR^T}\right) \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_i + \\ &\qquad \lambda_\chi \left[\boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{j,R} \cdot \left(\boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right)\right]_i \\ J_{ij}^{nlin(L_2,\gamma)} &= \eta_B \lambda_\chi \left[\boldsymbol{C}^{NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_i \\ J_{ij}^{nlin(L_3,s)} &= \lambda_\chi \left[\left(\boldsymbol{M}^{\Gamma,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} + \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Gamma,NR}\right) \cdot \boldsymbol{s}\right]_j \\ J_{ij}^{nlin(L_3,b)} &= \eta_B \lambda_\chi \left[\boldsymbol{C}^{NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_j \\ J_{ij}^{nlin(L_3,\gamma)} &= \left(\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s} + \lambda_G\right) \delta_{ij}. \end{aligned}$$

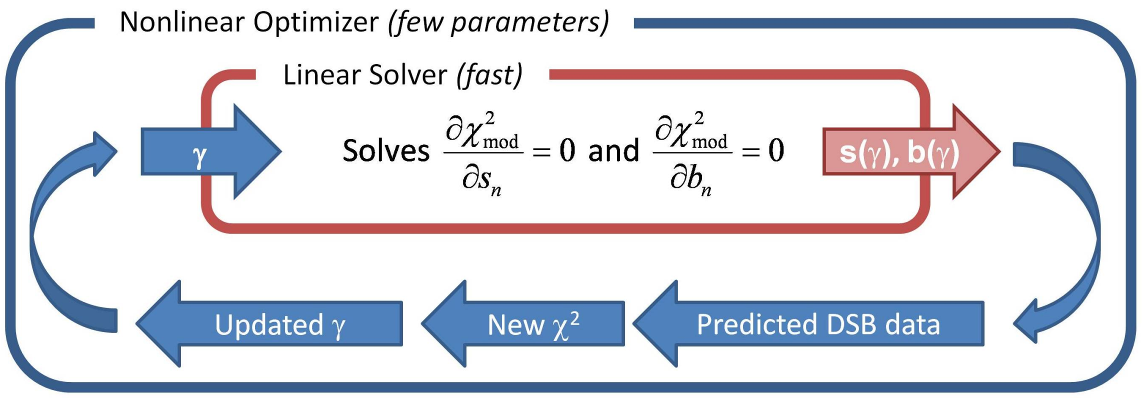

Nonlinear Optimizer *(few parameters)*
Linear Solver *(fast)*
γ
Solves $\frac{\partial \chi^2_{\rm mod}}{\partial s_n} = 0$ and $\frac{\partial \chi^2_{\rm mod}}{\partial b_n} = 0$
s(γ), b(γ)
Predicted DSB data
New χ²
Updated γ

Figure 4.9: The pseudo-linear optimization uses a hybrid approach, in which a fast, linear solver determines the single-sideband spectrum and baseline parameters associated with a given set of sideband-gain values. A slower, nonlinear loop iterates the values of the small number of sideband-gain parameters to optimize the solution.

4.13 Pseudo-Linear Optimization of χ^2_{mod}

Equation 4.61 represents a nearly linear system linking the modeled DSB spectrum $\left(\boldsymbol{d}^W\right)$ with the desired unknown quantity, the SSB spectrum $\boldsymbol{s}$ (via Equations 4.28). The inclusion of the unknown sideband gains (in $\boldsymbol{\Gamma}^R$) prevents the equations from being truly linear. However, if the gains were known, then the system of equations would be linear, making it much easier (and faster) to solve.

Unfortunately, the sideband gains for a given receiver are not usually known. The deconvolved spectrum that these algorithms produce is extremely sensitive to the precise value of the sideband gains, making it unlikely that they could be determined to sufficient precision via simulation or laboratory measurements.[11] However, this observation suggests a hybrid approach to the convolution in which the sideband-gain values are set by an "outer" loop using the direct-optimization approach of Section 4.11. Each iteration of the outer loop generates a trial set of sideband gains, $\gamma_{\boldsymbol{trial}}$, which can then be fed to a linear "inner" loop as constants. (See Figure 4.9.)

The advantage of this approach is that from the perspective of the outer loop, χ^2_{mod} is only a function of γ, greatly reducing the dimensionality of the parameter space that

[11]There is a small chance that a receiver might be stable enough for the gains derived in one deconvolution to be used in subsequent deconvolution attempts. Given that the gains must be known to high accuracy, this seems unlikely; however, since the linear method represents a radical improvement in deconvolution time, it would be worthwhile to study this possibility. This is particularly true for the Herschel mission's HIFI instrument, since it could reasonably be expected to have a high degree of stability.

must be searched by the nonlinear optimizer. In general, a survey contains tens, or at most hundreds, of individual observations, so that the outer loop would only need to deal with $\sim 10^2$ - 10^3 free parameters in $\boldsymbol{\gamma}$. In contrast, the requested SSB spectrum typically contains $\sim 10^5$ channels, each representing an independent parameter in the direct-optimization method. Thus, a hybrid approach reduces the dimensionality of the parameter space of the relatively slower non-linear optimizer by two or three orders of magnitude, presumably with a corresponding increase in execution speed.

Unfortunately, this approach does increase the complexity of the analytic calculations needed for the algorithms. In particular, the parameters returned by the inner loop are entirely defined by the value of $\boldsymbol{\gamma_{trial}}$ set by the outer loop; therefore, the remaining parameters need to be viewed as functions of $\boldsymbol{\gamma}$: $\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{\gamma})$ and $\boldsymbol{b} = \boldsymbol{b}(\boldsymbol{\gamma})$. The Jacobian for the outer loop will not only include terms representing $\frac{\partial \chi^2_{mod,i}}{\partial \gamma^j}$, but also terms of the form $\frac{\partial \chi^2_{mod,i}}{\partial s_k} \frac{\partial s_k}{\partial \gamma^j}$ and $\frac{\partial \chi^2_{mod,i}}{\partial b_k} \frac{\partial b_k}{\partial \gamma^j}$. Since the functions $\boldsymbol{s}(\boldsymbol{\gamma})$ and $\boldsymbol{b}(\boldsymbol{\gamma})$ cannot be found explicitly, a new approach to deriving the Jacobian is required.

4.13.1 Inner Loop (Linear)

For fixed $\boldsymbol{\gamma}$, the convolution model represented by Equation 4.61 is a linear system in the unknown parameters ($\boldsymbol{s}$ and $\boldsymbol{b}$). The inner loop minimizes χ^2 by taking derivatives with respect to each of the the free parameters and setting those derivatives equal to zero (an approach similar to Section 4.12).

Since the inner loop has no control over the values of $\boldsymbol{\gamma}$, we drop the χ^2_γ term from χ^2_{mod} to generate the figure of merit used for this calculation:

$$\chi^2_{lin} = \sum_{i=1}^{N^R_{DSB}} \chi^2_{o,i} + \sum_{j=1}^{N_{SSB}} \chi^2_{S,j} + \sum_{k=1}^{N^R_{DSB}} \chi^2_{\beta,k}, \tag{4.83}$$

where χ^2_o, χ^2_S, and χ^2_β have the definitions shown in Equations 4.65. As in Section 4.12, the spectral value of each channel in the deconvolved spectrum can be found by taking the derivative of χ^2_{lin} with respect to $\boldsymbol{s}$ and setting the result equal to zero (compare to Equation 4.78):

$$\frac{\partial \chi^2_{lin}}{\partial s_n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial s_n} + \sum_j \frac{\partial \chi^2_{S,j}}{\partial s_n} = 0.$$

The details of the calculation are shown in Section A.8.1.2, with the results contained in Equations A.47:

$$\lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_S \boldsymbol{s} = 0 \tag{A.47a}$$

$$\eta_B \boldsymbol{C}^{NR^T} \cdot \left(\lambda_\chi \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) = 0. \tag{A.47b}$$

These equations can be regrouped to gather like terms

$$\left(\lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} + \lambda_S \mathbb{1}\right) \cdot \boldsymbol{s} + \eta_B \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} \cdot \boldsymbol{b} \;=\; \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{d}^{o,NR} \tag{4.85a}$$

$$\eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \cdot \boldsymbol{b} \;=\; \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{d}^{o,NR}, \tag{4.85b}$$

allowing them to be represented in a more convenient block-matrix form:

$$\left(\begin{array}{c:c} \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} + \lambda_S \mathbb{1} & \eta_B \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} \\ \hdashline \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} & \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \end{array}\right) \left(\begin{array}{c} \uparrow \\ \boldsymbol{s} \\ \downarrow \\ \hdashline \uparrow \\ \boldsymbol{b} \\ \downarrow \end{array}\right) = \left(\begin{array}{c} \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{d}^{o,NR} \\ \hdashline \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{d}^{o,NR} \end{array}\right). \tag{4.86}$$

Solving this system of equations for $\boldsymbol{s}$ and $\boldsymbol{b}$ provides the optimal deconvolved spectrum for the given values of γ_{trial}.

4.13.2 Outer Loop

The outer loop seeks to minimize χ^2_{mod} as given in Equation 4.64; however, rather than being independent variables, $\boldsymbol{s}$ and $\boldsymbol{b}$ are viewed as functions of γ by the outer loop. Since it uses `lsqnonlin()` to find the optimal values of γ, the outer loop returns data in the structure indicated by Equations 4.67 and 4.68. The form of the Jacobian is still given by Equation 4.69, but now the only independent parameters are the components of γ. We can rewrite the four functions defined in Equations 4.67 to clearly show their dependence on γ:

$$
\begin{aligned}
f_{o,i}\left(\boldsymbol{s}\left(\gamma\right),\boldsymbol{b}\left(\gamma\right),\gamma\right) &= \sqrt{\lambda_\chi}\left(d_i^{NR} - d_i^{o,NR}\right), && (4.87a)\\
f_{S,j}\left(\boldsymbol{s}\left(\gamma\right)\right) &= \sqrt{\lambda_S}\left(s_j\right), && (4.87b)\\
f_{\beta,k}\left(\boldsymbol{b}\left(\gamma\right)\right) &= \sqrt{\eta_B\lambda_B}\left(\beta_k^{NR}\right), \text{ and} && (4.87c)\\
\text{and } f_{\gamma,m}\left(\gamma\right) &= \sqrt{\lambda_G}\left(\gamma^m\right). && (4.87d)
\end{aligned}
$$

We can use a generalized form of the "chain rule" to find the elements of the Jacobian:

$$
\begin{aligned}
J_{ij}^{outer,o} &= \frac{\partial f_{o,i}}{\partial \gamma^j} + \sum_k \frac{\partial f_{o,i}}{\partial s_k}\frac{\partial s_k}{\partial \gamma^j} + \sum_k \frac{\partial f_{o,i}}{\partial b_k}\frac{\partial b_k}{\partial \gamma^j},\\
J_{ij}^{outer,S} &= \sum_k \frac{\partial f_{S,i}}{\partial s_k}\frac{\partial s_k}{\partial \gamma^j},\\
J_{ij}^{outer,\beta} &= \sum_k \frac{\partial f_{\beta,i}}{\partial b_k}\frac{\partial b_k}{\partial \gamma^j}, \text{ and}\\
J_{ij}^{outer,\gamma} &= \frac{\partial f_{\gamma,i}}{\partial \gamma^j}.
\end{aligned}
$$

The overall Jacobian for the outer loop has the form

$$
\boldsymbol{J}^{outer} = \begin{pmatrix} \boldsymbol{J}^{outer,o} \\ \cdots\cdots\cdots\cdots \\ \boldsymbol{J}^{outer,S} \\ \cdots\cdots\cdots\cdots \\ \boldsymbol{J}^{outer,\beta} \\ \cdots\cdots\cdots\cdots \\ \boldsymbol{J}^{outer,\gamma} \end{pmatrix}. \qquad (4.88)
$$

The functions $\boldsymbol{s}\left(\boldsymbol{\gamma}\right)$ and $\boldsymbol{b}\left(\boldsymbol{\gamma}\right)$ cannot be found explicitly, but they are defined implicitly via Equations 4.85. These equations also contain the information needed for the Jacobian of the outer loop, which can be extracted using a first-order perturbation analysis. If we let $\boldsymbol{\gamma}_{\boldsymbol{o}}$, $\boldsymbol{s}_{\boldsymbol{o}}$, and $\boldsymbol{b}_{\boldsymbol{o}}$ represent a solution to Equations 4.85, we can then approximate the derivative by calculating the changes in $\boldsymbol{s}$ and $\boldsymbol{b}$ that would be generated by a small deviation from $\boldsymbol{\gamma}_{\boldsymbol{o}}$. Mathematically, we represent this by rewriting Equations 4.85 with the following substitutions:

$$\begin{aligned}\boldsymbol{\gamma} &\rightarrow \boldsymbol{\gamma}_{\boldsymbol{o}} + \delta\boldsymbol{\gamma}\\ \boldsymbol{s} &\rightarrow \boldsymbol{s}_{\boldsymbol{o}} + \delta\boldsymbol{s} \qquad (4.89)\\ \boldsymbol{b} &\rightarrow \boldsymbol{b}_{\boldsymbol{o}} + \delta\boldsymbol{b}.\end{aligned}$$

As described in Appendix A.5, we can then simplify the results by gathering the coefficients of like powers of $\delta\boldsymbol{\gamma}$, $\delta\boldsymbol{s}$, and $\delta\boldsymbol{b}$. The zeroth order terms (with no dependence on $\delta\boldsymbol{\gamma}$, $\delta\boldsymbol{s}$, or $\delta\boldsymbol{b}$) can be set to zero since they represent a solution to Equations 4.85. Since we are looking for the first-order change in $\delta\boldsymbol{s}$ and $\delta\boldsymbol{b}$, we can ignore any terms that have second-order or higher dependence on the variations. This leaves a linear set of equations that can be solved to find the changes in $\boldsymbol{s}$ and $\boldsymbol{b}$ that would result from small change in $\boldsymbol{\gamma}$ around $\boldsymbol{\gamma}_{\boldsymbol{o}}$. The calculation is carried out in Section A.5, with the results given in Equation A.36. If we convert those equations into a block-matrix format, we find

$$\left(\begin{array}{c:c}\lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot\boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}}+\lambda_S\mathbb{1} & \eta_B\lambda_\chi\boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot\boldsymbol{C}^{\boldsymbol{NR}}\\ \hdashline \eta_B\lambda_\chi\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot\boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}} & \eta_B\left(\lambda_\chi+\lambda_B\right)\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot\boldsymbol{C}^{\boldsymbol{NR}}\end{array}\right)\cdot\left(\begin{array}{c}\delta\boldsymbol{s}\\ \hdashline \delta\boldsymbol{b}\end{array}\right) = \left(\begin{array}{c}r_1\\ \hdashline -\eta_B\lambda_\chi\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}\cdot\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\cdot\boldsymbol{s}_{\boldsymbol{o}}\end{array}\right), \qquad (4.90)$$

where

$$r_1 = -\lambda_\chi\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}\cdot\left(\boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}}\cdot\boldsymbol{s}_{\boldsymbol{o}}+\eta_B\boldsymbol{C}^{\boldsymbol{NR}}\cdot\boldsymbol{b}_{\boldsymbol{o}}-\boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)-\lambda_\chi\boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}\cdot\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\cdot\boldsymbol{s}_{\boldsymbol{o}}.$$

While this result does represent the information needed to estimate the Jacobian, it is not equivalent to having an analytic Jacobian since the system of equations must be solved again any time a new value of $\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}$ is provided. Fortunately, MATLAB provides a mechanism for cases in which an analytic Jacobian is not available. The `JacobMult` option of `lsqnonlin` can be used in lieu of an analytic Jacobian; it allows the user to provide a function that can return $\boldsymbol{J} \cdot \boldsymbol{y}$, $\boldsymbol{J}^T \cdot \boldsymbol{y}$, and $\boldsymbol{J}^T \cdot \boldsymbol{J} \cdot \boldsymbol{y}$, where $\boldsymbol{J}$ is the Jacobian at the current solution point, and $\boldsymbol{y}$ is an arbitrary vector of appropriate dimension.

By solving Equation 4.90, we can generate values for $\boldsymbol{\delta s}\,(\delta\gamma)$ and $\boldsymbol{\delta b}\,(\delta\gamma)$ representing the change from $\boldsymbol{s_o}$ and $\boldsymbol{b_o}$ caused by changing $\boldsymbol{\Gamma}^{\boldsymbol{R}}$ by $\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}$. As the size of the variation becomes smaller $\left(\delta\boldsymbol{\Gamma}^{\boldsymbol{R}} \to 0\right)$, we expect

$$\begin{aligned} \lim_{\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}\to 0} \delta s_i\,(\delta\gamma) &= \sum_j \frac{\partial s_i}{\partial\gamma^j}\delta\gamma^j \text{ and} \\ \lim_{\delta\boldsymbol{\Gamma}^{\boldsymbol{R}}\to 0} \delta b_i\,(\delta\gamma) &= \sum_j \frac{\partial b_i}{\partial\gamma^j}\delta\gamma^j. \end{aligned} \tag{4.91}$$

We can use this approximation to estimate the value of $\boldsymbol{J} \cdot \boldsymbol{Y}$. Letting $f_{\chi,i}$ represent one of the functions from Equation 4.87 $\left(f_{o,i}, f_{S,i}, f_{\beta,i}, \text{or } f_{\gamma,i}\right)$ gives

$$\begin{aligned} [\boldsymbol{J} \cdot \boldsymbol{y}]_i &= \sum_j J_{ij} y_j \\ &= \sum_j \left(\frac{\partial f_{\chi,i}}{\partial\gamma^j} + \sum_k \frac{\partial f_{\chi,i}}{\partial s_k}\frac{\partial s_k}{\partial\gamma^j} + \sum_k \frac{\partial f_{\chi,i}}{\partial b_k}\frac{\partial b_k}{\partial\gamma^j} \right) y_j \\ &= \sum_j f^{(\gamma)}_{\chi,ij} y_j + \sum_k f^{(s)}_{\chi,ik} \left(\sum_j \frac{\partial s_k}{\partial\gamma^j} y_j \right) + \sum_k f^{(b)}_{\chi,ik} \left(\sum_j \frac{\partial b_k}{\partial\gamma^j} y_j \right) \\ &= \sum_j f^{(\gamma)}_{\chi,ij} y_j + \sum_k f^{(s)}_{\chi,ik}\, \delta s_k\,(\boldsymbol{y}) + \sum_k f^{(b)}_{\chi,ik}\, \delta b_k\,(\boldsymbol{y})\,. \end{aligned}$$

The final step relies on Equation 4.91 to yield the values of $\sum_j \frac{\partial s_k}{\partial\gamma^j} y_j$ and $\sum_j \frac{\partial b_k}{\partial\gamma^j} y_j$, which can be found by letting $\delta\boldsymbol{\Gamma}^{\boldsymbol{R}} = \boldsymbol{y}$ and solving for $\delta s_i\,(\boldsymbol{y})$ and $\delta b_i\,(\boldsymbol{y})$. We have also defined the matrices

$$\begin{aligned} f^{(\gamma)}_{\chi,ij} &= \frac{\partial f_{\chi,i}}{\partial\gamma^j}, \\ f^{(s)}_{\chi,ik} &= \frac{\partial f_{\chi,i}}{\partial s_k}, \text{ and} \\ f^{(b)}_{\chi,ik} &= \frac{\partial f_{\chi,i}}{\partial b_k}, \end{aligned}$$

which can be calculated directly from Equation 4.87. Since this result is true for each of the components of $[\boldsymbol{J} \cdot \boldsymbol{y}]_i$, it can be written as a simple matrix multiplication:

$$\boldsymbol{J} \cdot \boldsymbol{y} = \boldsymbol{f}_{\chi}^{(\gamma)} \cdot \boldsymbol{y} + \boldsymbol{f}_{\chi}^{(s)} \cdot \delta \boldsymbol{s}\left(\boldsymbol{y}\right) + \boldsymbol{f}_{\chi}^{(b)} \cdot \delta \boldsymbol{b}\left(\boldsymbol{y}\right). \tag{4.92}$$

The algorithm representing the outer loop must also be able to calculate $\boldsymbol{J}^T \cdot \boldsymbol{y}$, where $\boldsymbol{y}$ is another arbitrary vector of appropriate length. However, we cannot follow the same approach used for Equation 4.92:

$$\begin{aligned}
\left[\boldsymbol{J}^T \cdot \boldsymbol{y}\right]_i &= \sum_j {J_{ij}}^T y_j \\
&= \sum_j J_{ji} y_j \\
&= \sum_j \left(\frac{\partial \chi^2_{mod,j}}{\partial \gamma^i} + \sum_k \frac{\partial \chi^2_{mod,j}}{\partial s_k} \frac{\partial s_k}{\partial \gamma^i} + \sum_k \frac{\partial \chi^2_{mod,j}}{\partial b_k} \frac{\partial b_k}{\partial \gamma^i} \right) y_j \\
&= \sum_j y_j f^{(\gamma)}_{\chi,ji} + \sum_{j,k} y_j f^{(s)}_{\chi,jk} \frac{\partial s_k}{\partial \gamma^i} + \sum_{j,k} y_j f^{(b)}_{\chi,jk} \frac{\partial b_k}{\partial \gamma^i}.
\end{aligned}$$

In this case, the two unknown derivatives $\left(\frac{\partial s_k}{\partial \gamma^i} \text{ and } \frac{\partial b_k}{\partial \gamma^i}\right)$ have a free index (i), whereas Equation 4.91 sums over both of the derivatives' indices. In order to use Equation 4.91, we would have to be able to pull individual terms out of those sums.

Instead, we can use Equation 4.92 to find individual components of $\left[\boldsymbol{J}^T \cdot \boldsymbol{y}\right]_i$. We start by defining a set of unit vectors,

$$\theta^a_i = \delta_{ai}, \tag{4.93}$$

such that there are N_γ vectors, each with N_γ components, and $\boldsymbol{\theta^a}$ has a one in the a^{th} component, with zeros elsewhere. We can then use these unit vectors to pull out individual columns of the Jacobian:

$$\begin{aligned}
[\boldsymbol{J} \cdot \boldsymbol{\theta^a}]_i &= \sum_j J_{ij} \theta^a_j \\
&= \sum_j J_{ij} \delta_{aj} \\
&= J_{ia}.
\end{aligned}$$

We can substitute this into our earlier equation for $\left[\boldsymbol{J}^T \cdot \boldsymbol{y}\right]_i$ to find

$$
\begin{aligned}
\left[\boldsymbol{J}^T \cdot \boldsymbol{y}\right]_i &= \sum_j {J_{ij}}^T y_j \\
&= \sum_j J_{ji} y_j \\
&= \sum_j \left[\boldsymbol{J} \cdot \boldsymbol{\theta^i}\right]_j y_j \\
&= \boldsymbol{J} \cdot \boldsymbol{\theta^i} \cdot \boldsymbol{y}.
\end{aligned} \tag{4.94}
$$

Thus, by applying Equation 4.92 once for each of the unit vectors $\boldsymbol{\theta^a}$, we can build up the value of $\boldsymbol{J}^T \cdot \boldsymbol{y}$, one component at a time.

Implementing this calculation in MATLAB can be done relatively easily. MATLAB provides the `mldivide` function, represented in calculations by a backslash, \, for solving linear equations of the form $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{b}$. If matrix $\boldsymbol{A}$ and vector $\boldsymbol{b}$ are known, the unknown vector $\boldsymbol{x}$ can be found using the command `x = A\b`.

Conceptually, $\boldsymbol{A} \backslash \boldsymbol{b}$ is analogous to $\boldsymbol{A}^{-1} \cdot \boldsymbol{b}$, except that it is implemented in a more numerically stable fashion. In addition, `mldivide` can be used to solve $\boldsymbol{A} \cdot \boldsymbol{X} = \boldsymbol{B}$, in which $\boldsymbol{B}$ (and hence $\boldsymbol{X}$) are both matrices. The latter capability allows us to solve for all of the $\delta \boldsymbol{s}\left(\boldsymbol{\theta^a}\right)$ and $\delta \boldsymbol{b}\left(\boldsymbol{\theta^a}\right)$ in one step.

Let $\boldsymbol{r^a}$ represent the right-hand side of Equation 4.90, evaluated for $\boldsymbol{\delta\gamma} = \boldsymbol{\theta^a}$. In that case, we have $\boldsymbol{\delta\Gamma} = \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}$, giving

$$
\boldsymbol{r^a} = \begin{pmatrix} r_1^a \\ \hdashline -\eta_B \lambda_\chi \boldsymbol{C^{NR}}^T \cdot \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} \cdot \boldsymbol{M^{\Delta,NR}} \cdot \boldsymbol{s_o} \end{pmatrix}, \tag{4.95}
$$

where

$$
r_1^a = -\lambda_\chi \boldsymbol{M^{\Delta,NR}}^T \cdot \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} \cdot \left(\boldsymbol{M_o^{\Gamma,NR}} \cdot \boldsymbol{s_o} + \eta_B \boldsymbol{C^{NR}} \cdot \boldsymbol{b_o} - \boldsymbol{d^{o,NR}}\right) - \\
\lambda_\chi \boldsymbol{M_o^{\Gamma,NR}}^T \cdot \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} \cdot \boldsymbol{M^{\Delta,NR}} \cdot \boldsymbol{s_o}. \tag{4.96}
$$

We define matrix $\boldsymbol{R}$ to consist of columns consisting of the individual $\boldsymbol{r^a}$,

$$\boldsymbol{R} = \left(\begin{array}{c:c:c:c} \uparrow & \uparrow & & \uparrow \\ \boldsymbol{r}^1 & \boldsymbol{r}^2 & \cdots & \boldsymbol{r}^{N_\gamma} \\ \downarrow & \downarrow & & \downarrow \end{array} \right), \tag{4.97}$$

so that $R_{ia} = r_i^a$. If we let matrix $\boldsymbol{L}$ represent the left-hand side of Equation 4.90,

$$\boldsymbol{L} = \left(\begin{array}{c:c} \lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}} + \lambda_S \mathbb{1} & \eta_B \lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \\ \hdashline \eta_B \lambda_\chi \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{M}_{\boldsymbol{o}}^{\boldsymbol{\Gamma},\boldsymbol{NR}} & \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \end{array} \right), \tag{4.98}$$

then we can use `mldivide` to solve the matrix equation

$$\boldsymbol{L} \cdot \boldsymbol{X} = \boldsymbol{R} \tag{4.99}$$

for $\boldsymbol{X}$, which contains the values of $\delta\boldsymbol{s}\left(\boldsymbol{\theta}^a\right)$ and $\delta\boldsymbol{b}\left(\boldsymbol{\theta}^a\right)$ for all of the $\boldsymbol{\theta}^a$ vectors:

$$\left(\begin{array}{c:c:c:c} \uparrow & \uparrow & & \uparrow \\ \delta\boldsymbol{s}\left(\boldsymbol{\theta}^1\right) & \delta\boldsymbol{s}\left(\boldsymbol{\theta}^2\right) & \cdots & \delta\boldsymbol{s}\left(\boldsymbol{\theta}^{N_\gamma}\right) \\ \downarrow & \downarrow & & \downarrow \\ \hdashline \uparrow & \uparrow & & \uparrow \\ \delta\boldsymbol{b}\left(\boldsymbol{\theta}^1\right) & \delta\boldsymbol{b}\left(\boldsymbol{\theta}^2\right) & \cdots & \delta\boldsymbol{b}\left(\boldsymbol{\theta}^{N_\gamma}\right) \\ \downarrow & \downarrow & & \downarrow \end{array} \right). \tag{4.100}$$

Each of the $\delta\boldsymbol{s}$ and $\delta\boldsymbol{b}$ can then can be used in Equation 4.92 to produce the values of the corresponding $\boldsymbol{J} \cdot \boldsymbol{\theta}^a$. These can be used in turn in Equation 4.94 to produce the desired result of $\boldsymbol{J}^T \cdot \boldsymbol{y}$.

The remaining multiplication, $\boldsymbol{J}^T \cdot \boldsymbol{J} \cdot \boldsymbol{y}$ can be generated by calling the algorithms for $\boldsymbol{J} \cdot \boldsymbol{y}$ and $\boldsymbol{J}^T \cdot \boldsymbol{y}$ in sequence to give $\boldsymbol{J}^T \cdot (\boldsymbol{J} \cdot \boldsymbol{y})$.

4.14 Optimization of χ^2_{mod} via Iterative Linear Root Finding

Although not studied to a significant degree in this work, another potential approach would be to break the problem into two coupled sets of linear equations, as shown in Figure 4.10. To start, a reasonable value for γ is chosen (e.g., $\gamma = 0$). The linear equations

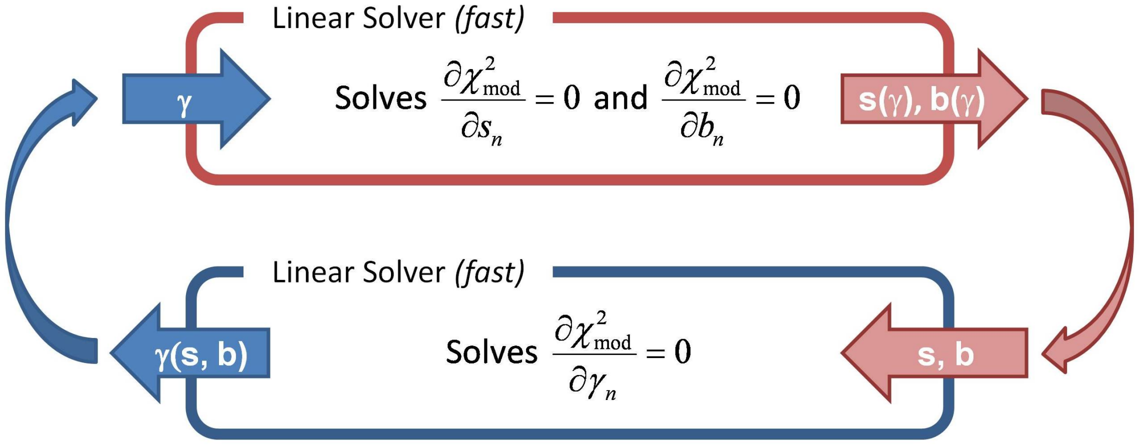

Linear Solver *(fast)*
Solves $\frac{\partial \chi^2_{\text{mod}}}{\partial s_n} = 0$ and $\frac{\partial \chi^2_{\text{mod}}}{\partial b_n} = 0$
γ
s(γ), b(γ)
Linear Solver *(fast)*
Solves $\frac{\partial \chi^2_{\text{mod}}}{\partial \gamma_n} = 0$
s, b
γ(s, b)

Figure 4.10: Iterative linear root finding. In the top part of the cycle, γ is treated as a fixed quantity, and a linear solver is used to find $\boldsymbol{s}$ and $\boldsymbol{b}$. These values are then used in another linear optimizer that finds γ, assuming the provided values of $\boldsymbol{s}$ and $\boldsymbol{b}$ are fixed. Iteration stops if a self-consistent solution is found.

$$\frac{\partial \chi^2_{mod}}{\partial s_n} = 0 \text{ and}$$
$$\frac{\partial \chi^2_{mod}}{\partial b_n} = 0$$

are solved, treating γ as constant, to produce values of $\boldsymbol{s}$ and $\boldsymbol{b}$. Next, a new value for γ is determined by solving the linear equations

$$\frac{\partial \chi^2_{mod}}{\partial \gamma^n} = 0,$$

in which $\boldsymbol{s}$ and $\boldsymbol{b}$ are treated as constants with the values found in the first step. The cycle can then be repeated, using the new value of γ. With a bit of luck, the iterative process might converge to a self-consistent set of $\boldsymbol{s}$, $\boldsymbol{b}$, and γ.

Chapter 5

Testing the Deconvolution Algorithm

One of the primary goals of the work presented in this thesis is to develop data-analysis software that can rapidly deconvolve the spectra produced by the broad-bandwidth receivers. As discussed in Section 6.2, by making line surveys more efficient, the new receivers also permit more sensitive surveys to be done in a realistic amount of observing time. Therefore, the deconvolution software must also be capable of preserving the high sensitivity of the observations, some of which are expected to push noise levels down by an order of magnitude relative to previous results.

When pursuing such a goal, the software cannot operate as a "black box." It must contain facilities for validating both the final results and the intervening steps. From its inception, the deconvolution software has included a powerful simulation engine to provide synthetic spectra for testing. In this way, the final results can be tested by comparing them to the "true" values used in simulation, both validating the algorithm's performance and quantifying the amount of error introduced by the process. In addition, analysis and display routines have been developed to probe the internal quantities used by the routine. For instance, the resampling algorithms presented in Section 4.4.1 have been validated extensively by comparing resampled SSB and DSB spectra to the original values.

To minimize "crosstalk," the code base that creates the simulated spectrum is entirely separate from the deconvolution portion of the code,[1] and there are multiple safeguards in

[1] An important exception to this statement is that the function responsible for converting a set of baseline coefficients, $\boldsymbol{b}$, into the corresponding baseline, $\boldsymbol{\beta}$, is used by both portions of the code. This code primarily implements Equations 4.52 and 4.53. Ideally, baseline calculation should follow an approach similar to the one used for DSB and SSB spectra, in which an entirely independent method of calculating the baselines is used. Initially the code was designed this way; however, baseline fitting has proven somewhat problematic, so we decided to use the same method on the simulation and deconvolution sides of the code to make troubleshooting easier.

place to ensure that the deconvolution algorithms do not have access to the "true" parameters from the simulation. Whenever possible, different methods of calculating equivalent quantities are used to check for internal consistency. For instance, the simulation side of the code does not use any type of convolution matrix, $\boldsymbol{M}$; instead, it creates a list of peaks, defining their center frequencies, amplitudes, and widths. When creating DSB data, it searches through all the peaks to find any that could contribute to either the lower sideband or upper sideband and integrates those peaks over the specified channels. Similarly, to produce the true SSB spectrum, it uses the same process, accepting a set of frequency channels over which it integrates the known peaks.

Another advantage of incorporating simulation is that it provides opportunities to gain an intuitive understanding of how the deconvolution algorithms work.

5.1 Status of Software

Chapter 4 presents multiple ways of approaching the optimization problem. The pseudo-linear method described in Section 4.13 seems particularly promising, as it separates out the nonlinear elements of the problem, leaving the majority of the parameters to be found through fast and reliable linear methods. However, this is expected to be the hardest approach to implement. From a practical point of view, it requires three separate critical functions: the outer, nonlinear optimizer, the inner, linear solver, and a second, linear solver to estimate the Jacobian. The results are very sensitive to the values of the sideband-gain factors, so significant infrastructure for troubleshooting and debugging is needed to develop this approach. In contrast, the direct optimization (Section 4.11) and nonlinear root finding (Section 4.12) techniques should be easier to implement since they rely on only a single-stage approach in which all parameters are found simultaneously.[2]

Early prototypes of the deconvolution software included all three approaches to allow for comparison of results and execution time. However, as the code expanded, adding

[2]One of the concerns with the single-stage approach is that the parameters have significantly different scales. Whereas a relative error of 10^{-3} might be sufficient for the parameters describing the single-sideband spectrum, the baseline parameters and sideband-gain values are likely to require higher precision. The MATLAB routines perform quite well, even with this difference in scaling, but it could be advantageous to consider reparameterizing the problem so that all parameters share a similar scale.

the integrated baseline fitting and the resampling of non-aligned spectra, supporting the three branches of simultaneous development became too complex. Therefore, we decided to focus on one of the three methods and develop it to a fully functional state, both as a proof of concept and to provide immediate capabilities for analyzing existing data sets. The other branches can be added when, and if, they are needed. This has proven to be an important decision, as it allowed us to develop debugging utilities and perform significant troubleshooting on the existing branch of the code.

In choosing which of the three methods to fully implement, we felt the direct optimization approach offered the most immediate promise. The Jacobians are simple to calculate, minimizing opportunities for error. Since the code works directly with χ^2_{mod} ,troubleshooting and debugging are easier. The current code therefore only offers the direct-optimization algorithm, and the results to be discussed in this chapter and Chapter 6 are based on that approach. However, the code has been designed with the intention of adding the other methods, so the necessary control structures and interfaces are in place to implement them.

5.2 Simulations

5.2.1 Deconvolution Without Baselines

The simplest case for the deconvolution routine should be one in which the data are simulated without any baselines, and the deconvolution is performed without fitting baselines ($\eta_B = 0$). Under those circumstances, the deconvolution algorithm performs extremely well.

A series of DSB spectra were simulated using the same LO-selection scheme used for the CSO line survey of L1157 (described in Section 6.1.1). The simulated survey covered frequencies from 210 to 280 GHz, and Gaussian noise was added to each DSB spectrum to simulate $T_{RMS} \approx 35$ mK. A set of sideband-gain factors was randomly chosen in the range $-0.15 < \gamma^a < 0.15$, and 1400 Gaussian-shaped peaks with randomly selected frequency, amplitude, and width were added to the spectrum. DSB spectra were simulated using spectrometer characteristics similar to CSO's AOS. No baseline was added to the DSB data,

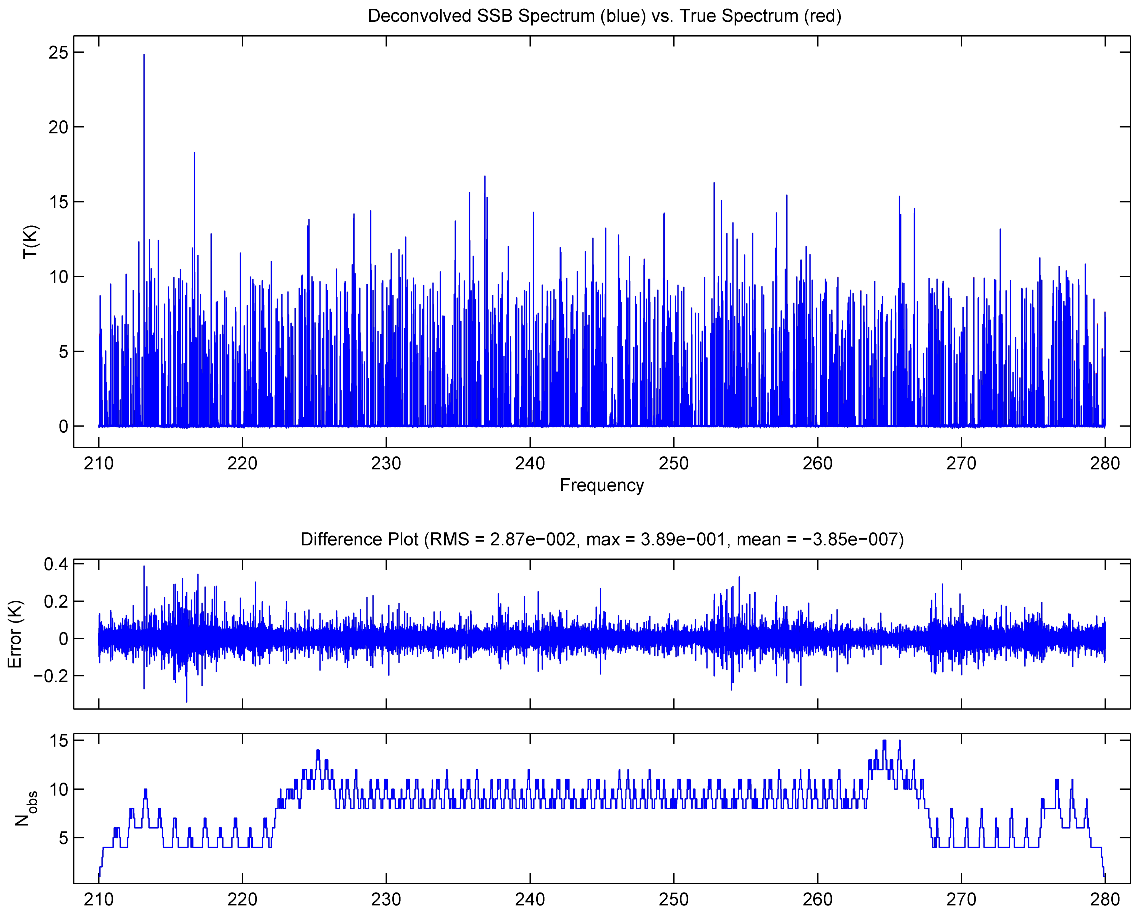


Figure 5.1: The top panel shows the recovered SSB spectrum compared to the true SSB spectrum from the baseline-free simulation (Section 5.2.1), with a difference plot below. The bottom plot indicates the number of times each channel in the SSB spectrum was included in a DSB spectrum.

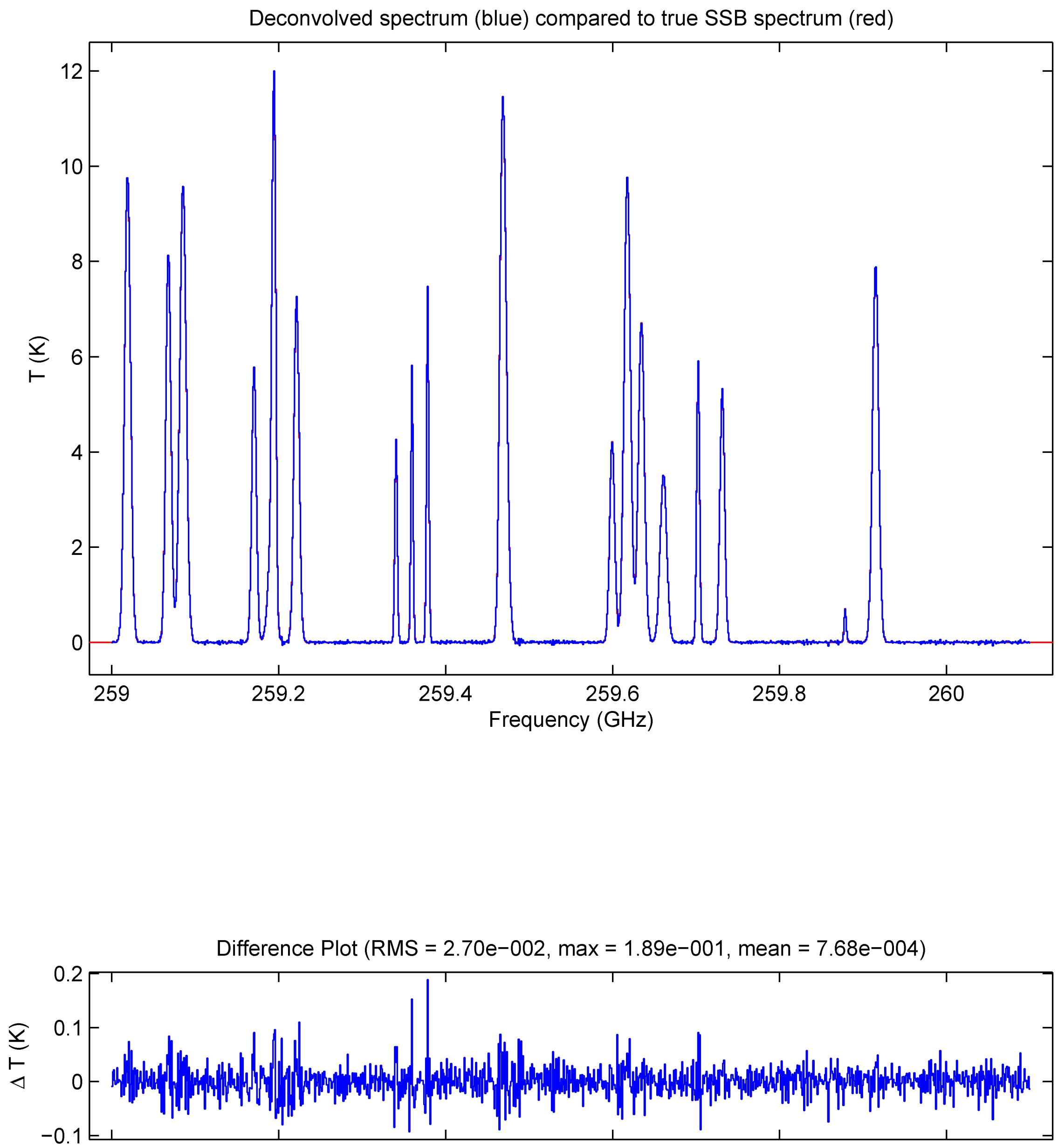


Figure 5.2: Enlarged view of the results generated by the baseline-free simulation. The top plot shows the deconvolved spectrum (blue) compared to the true SSB spectrum (red) for a single observation, with a difference plot at the bottom.

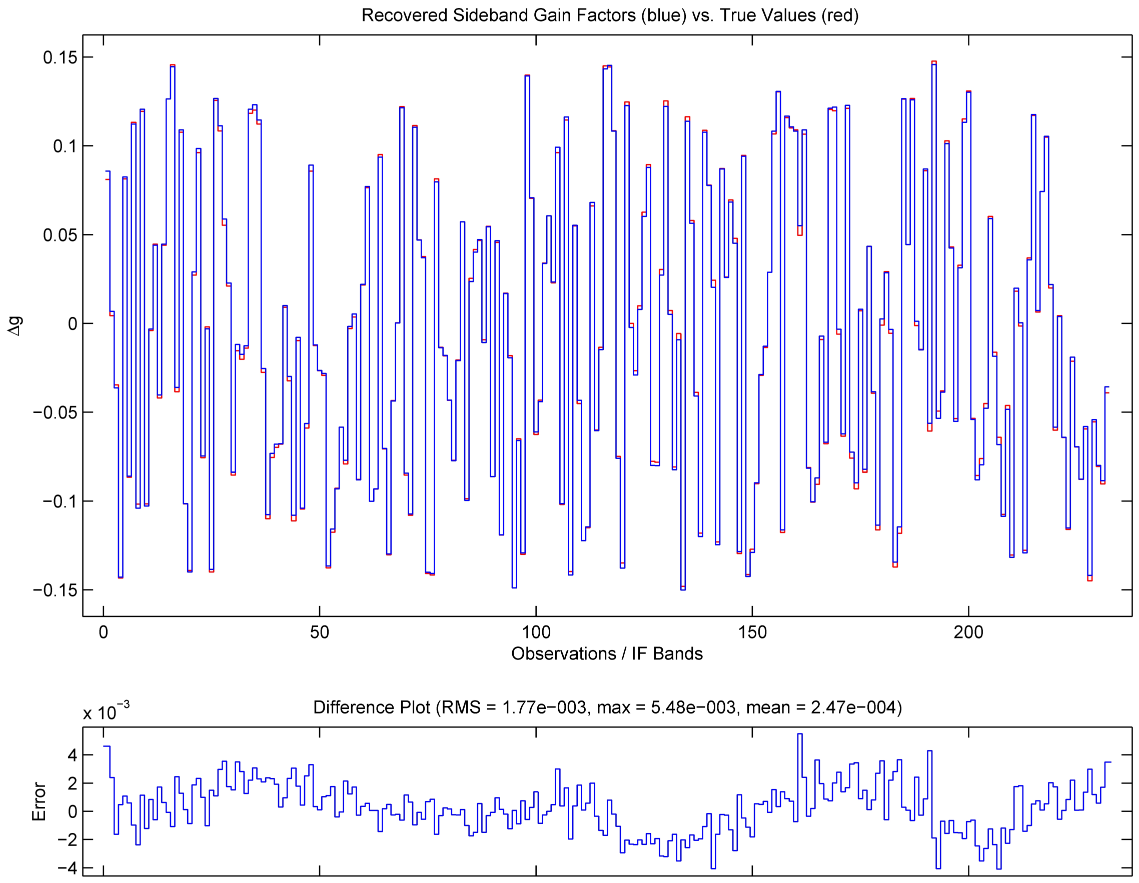


Figure 5.3: Sideband-gain factors, γ, recovered during the baseline-free deconvolution (blue) compared to the true values (red), with a difference plot at the bottom.

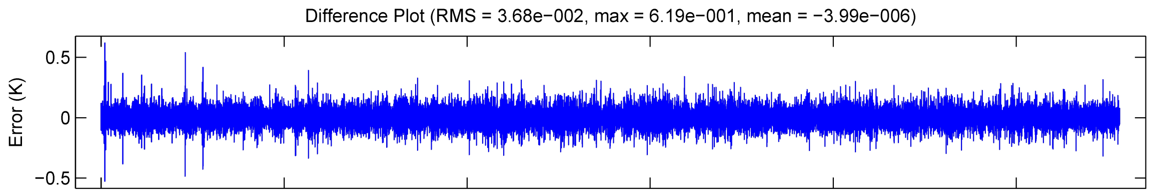


Figure 5.4: Additional results from the baseline-free simulation showing the difference between the simulated DSB data and the DSB spectra predicted by the model using the recovered parameters. The error has an RMS value of 37 mK, which compares will with the 35 mK of noise added to the simulated data.

so this simulation only tested the deconvolution routine's ability to recover the correct SSB spectrum and gain factors. A reference version of the SSB spectrum was calculated by integrating the Gaussian peaks onto the SSB channels used for the deconvolution.

Because this was such a simple case, the deconvolution ran in a single step with $\lambda_\chi = 1$, rather than using the multi-step continuation approach described in Section 4.9.1. The constraint terms were set to $\lambda_S = 0.085$ and $\lambda_G = 10,304$ to encourage the optimizer to minimize the values of s and γ, respectively.[3] The optimizer found a solution offering the following goodness-of-fit characteristics (for 289,194 degrees of freedom): $\chi_o^2 = 507,644$, $\chi_S^2 = 23,616$, $\chi_\gamma^2 = 17,036$, and $\chi_{mod}^2 = 548,295$.[4] The value of χ_o^2 corresponds to a reduced chi-square of $\chi_\nu^2 = 1.76$. Since the usual rule-of-thumb value for an acceptable fit is $\chi_\nu^2 \sim 1$, this implies that the results are not actually a good fit to the data. The probability of getting a value of χ_o^2 this large for the given number of degrees of freedom is near zero, formally confirming the conclusion that this is a poor fit, statistically speaking.

For comparison, the true values of the parameters produce $\chi_o^2 = 788,492$ (corresponding to $\chi_\nu^2 = 2.73$) and $\chi_{mod}^2 = 829,037$. The fact that the true values used for the simulation produce a higher χ_o^2 than the parameters found by the optimization is not, in its own right, alarming. Nothing in the χ^2 formalism guarantees that the best-fit values will actually be the correct values. To the contrary, fit parameters from a χ^2 optimization have uncertainties associated with them, since they are also random variables. However, the true values should be a good enough fit that the χ^2 results do not rule them out as being inconsistent with the data.

In the current case, the high value of χ_o^2 produced by the true values indicates that those values, when used in the specified model, cannot reproduce the "observed" data.

[3]The deconvolution code does not provide a mechanism for directly setting λ_S and λ_G, as the actual value of those parameters have little meaning. The relevant quantities are the relative magnitudes of the various terms that go into χ_{mod}^2. To provide an estimate of the effect of different λ values, the user provides values that are intended to represent the constraint term's likely size with respect to χ_o^2. For λ_S, λ_G, and λ_B, these are named `lambda_s_fraction`, `lambda_g_fraction`, and `lambda_b_fraction`. This is not a precise conversion, and it depends strongly on the exact nature of the spectra being studied, but it does typically offer an order-of-magnitude estimate. For a particular deconvolution, it is usually necessary to fine-tune these parameters to produce the desired result. The values shown above for λ_S and λ_G correspond to `lambda_s_fraction = 0.01` and `lambda_g_fraction = 0.01`. The disadvantage of this formulation is that the actual values of λ_S, λ_G, and λ_B are produced by the code, rather than being set by the user.

[4]In this case, χ_S^2 and χ_γ^2 were approximately 4.6% and 3.4% of χ_o^2, indicating that `lambda_s_fraction = 0.01` and `lambda_g_fraction = 0.01` are appropriate for providing order-of-magnitude control over λ_S and λ_G.

Most likely, this is a problem with the model, more than the particular values. The fact that the simulation data are produced using independent methodology provides a good check on the deconvolution method. In this example, the model appears to have failed the test; it does not capture all of the features produced by the simulation engine. The source of these discrepancies is an interesting question deserving of further study.

There is a different way to look at these results, however. We can instead think of the chi-square optimization as producing the best results that can be obtained by this model. In this view, the goodness-of-fit tests are not relevant, as we already know that the model does not accurately explain the simulated data, but we can see whether the recovered data is sufficiently good for our needs.

From this pragmatic perspective, the news is good. The recovered SSB spectrum matches the true spectrum quite well, with an RMS error of only 28.7 mK. The sideband-gain factors are also recovered accurately, with RMS error of 1.8×10^{-3}. The deconvolution results are shown in Figures 5.1, 5.2, 5.3, and 5.4. It can be seen that the distribution of peak amplitudes used in the test data is not representative of astrophysical spectra; however, this makes no difference to the final results. The deconvolution algorithm operates on a channel-by-channel basis, without any knowledge of whether that channel is part of a peak. These results show that the algorithm can reconstruct the channels of the SSB spectrum to an accuracy of ~ 29 mK, and that would be true regardless of the contents of the spectrum. It could consist of large peaks, small peaks, no peaks, or randomly selected values.

5.2.2 Deconvolution with Baseline Fitting on Baseline-Free Data

The example in Section 5.1 is a simple case, as the simulated data had been created without a baseline, and the baseline-fitting aspect of the deconvolution algorithm was turned off. To see what the deconvolution algorithm would do without this extra knowledge, the same data were reanalyzed in a single-step deconvolution with baseline fitting enabled ($\eta_B = 1$) and $\lambda_B = 0$.[5] If the baseline-fitting portion of the code is working reliably, it

[5]As discussed in Section 5.3.1, non-zero values of λ_B cannot be simulated on large data sets at this time.

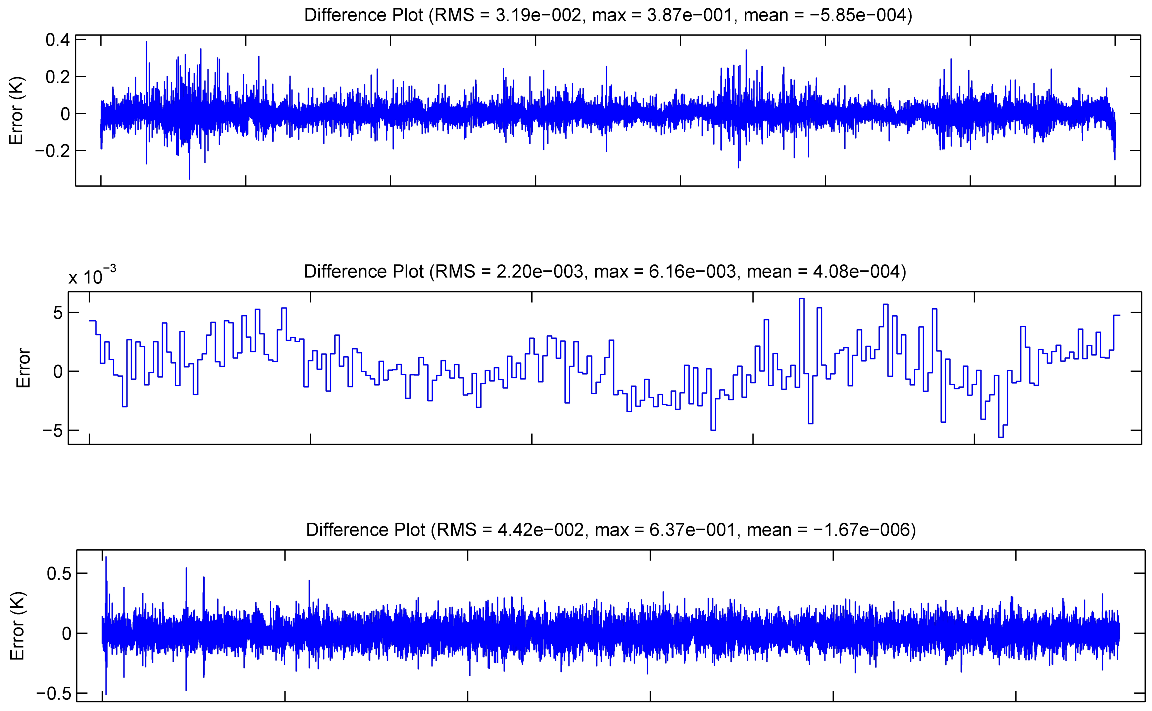
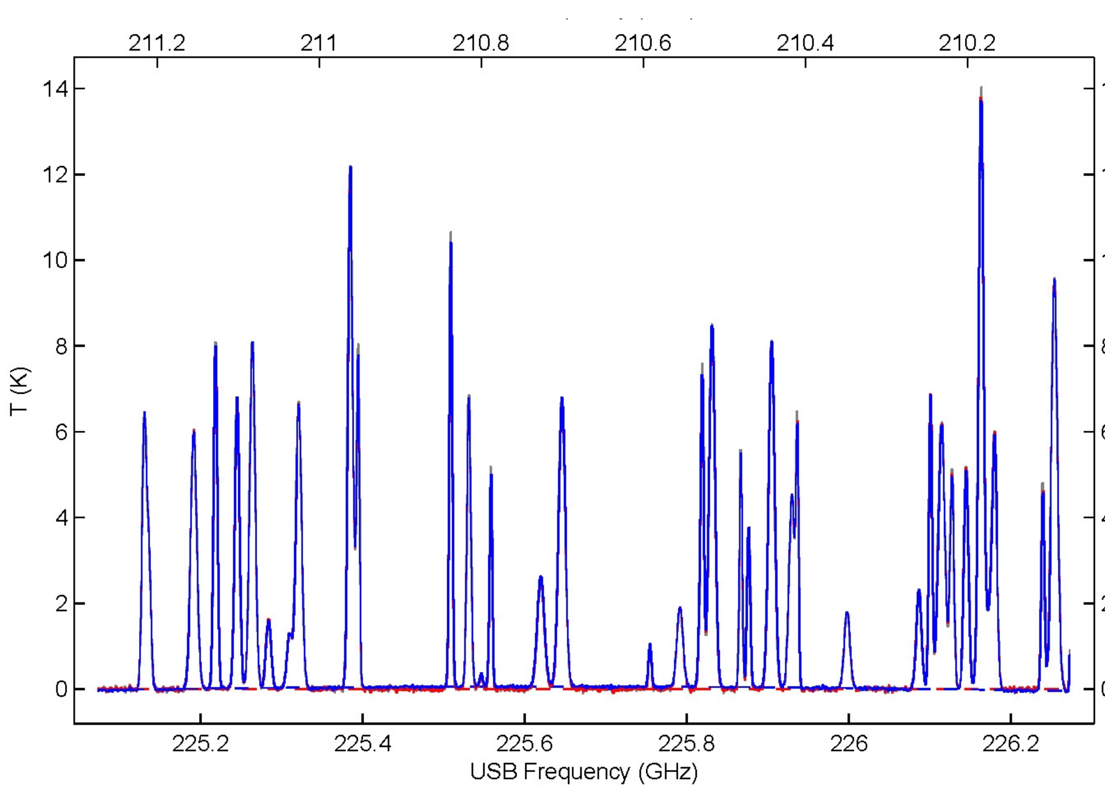


Figure 5.5: Difference plots for the recovered SSB spectrum (top, the scale of x-axis is same as the SSB difference plot in Figure 5.1), sideband-gain factors (middle), and DSB spectra (bottom) for the simulation in which the deconvolution algorithm was allowed to fit baselines to baseline-free data (Section 5.2.2).



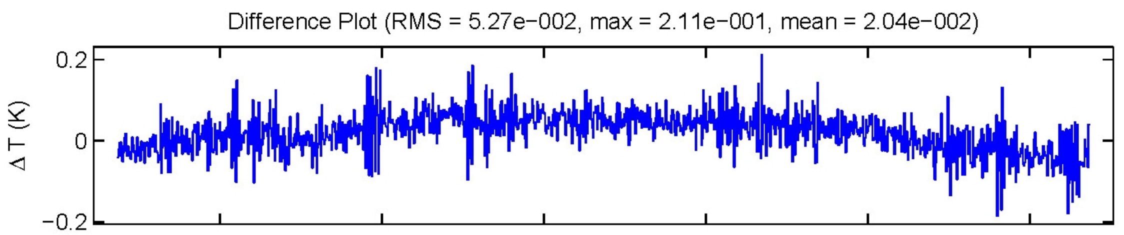


Figure 5.6: A sample DSB scan from the simulation in which the deconvolution algorithm was allowed to fit baselines to baseline-free data. The simulated data is shown in red while the DSB spectrum predicted by the model is shown in blue. Near the bottom, dashed lines can be seen corresponding to the model's baseline fit (blue) and the actual baseline contained in the data (red), which in this case is equal to zero. The fit baseline is nearly zero, but as the difference plot shows, baseline fitting introduces errors into the model's predicted data.

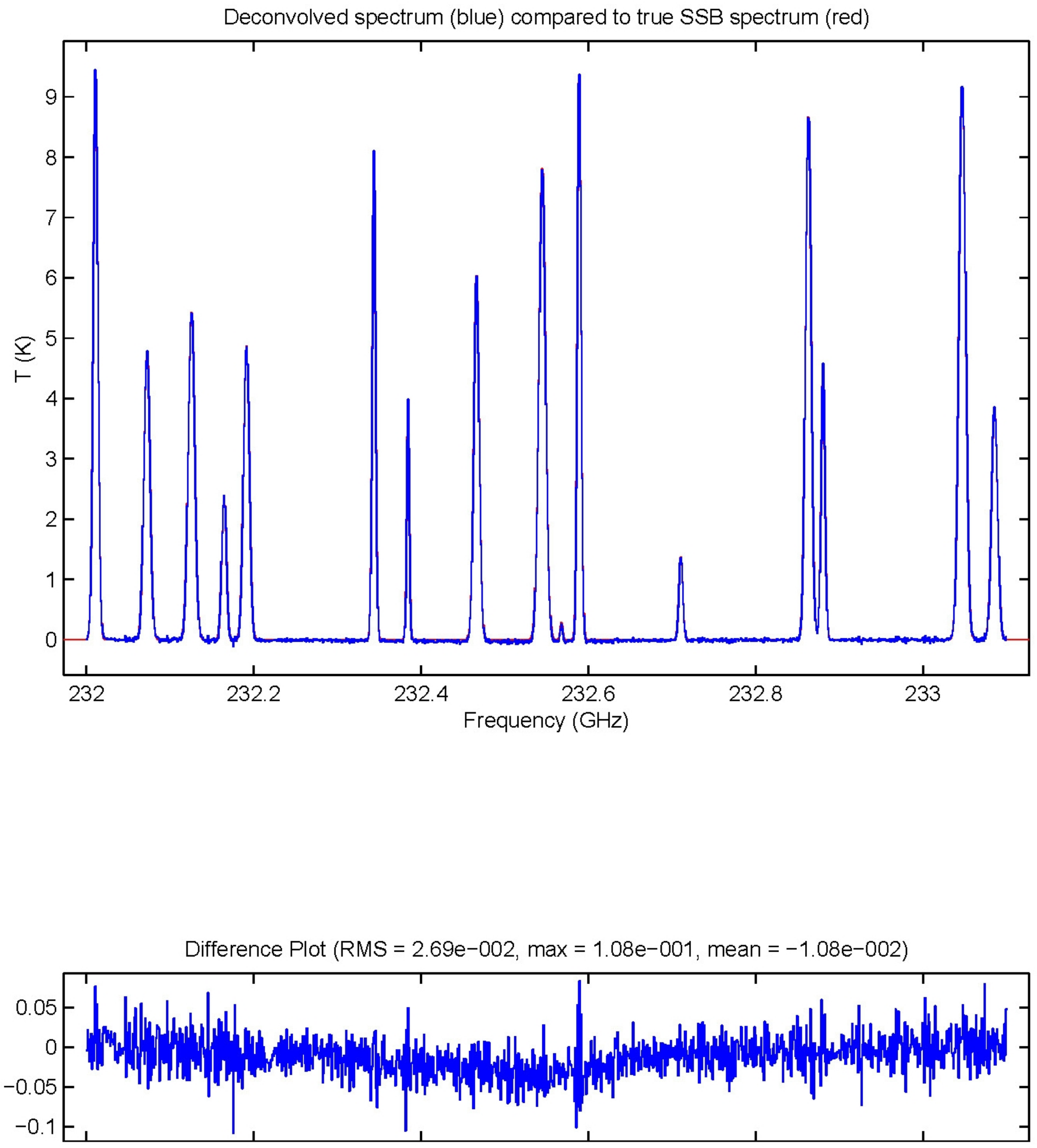


Figure 5.7: Portion of the deconvolved SSB spectrum from the simulation in which the deconvolution algorithm was allowed to fit baselines to baseline-free data (Section 5.2.2). Since each channel in the SSB spectrum draws from many DSB spectra, the baseline curvature seen in Figure 5.6 is reduced.

should produce near-zero baselines for this data set. All other aspects of the deconvolution remain the same.

The optimizer converged to a solution with $\chi_o^2 = 737,704$ (corresponding to $\chi_\nu^2 = 2.6$), $\chi_S^2 = 23,617$, $\chi_\gamma^2 = 17,044$, and $\chi_{mod}^2 = 778,365$ for 288,498 degrees of freedom. As before, the true parameters generate $\chi_o^2 = 788,492$ ($\chi_\nu^2 = 2.73$) and $\chi_{mod}^2 = 829,037$. This time, the solution identified by the optimizer and the true parameters produce values of χ_o^2 that are much closer than before; however, both are still statistically poor fits. Adding the baselines slightly degraded the quality of the resulting fit, with the recovered SSB spectrum possessing RMS errors of 32 mK and the sideband-gain factors having RMS error of 2.2×10^{-3} with respect to the true values used for the simulation. While these values are somewhat worse than before, they demonstrate that the deconvolution algorithm is capable of producing high-sensitivity results, even without the *a priori* knowledge that the data set does not contain a baseline. The RMS error between the simulated and modeled DSB spectra shows some of the difficulties encountered by the optimizer, however; it jumped from 37 mK in the previous simulation to 44 mK in the current one.

The difference plots from the current result are shown in Figure 5.5. A sample DSB spectrum is shown in Figure 5.6, where it can be seen the deconvolution algorithm fit a baseline that was nearly, but not quite, zero. This indicates that the algorithm did keep control of the baseline fitting to produce reasonable results; however, it also suggests that baseline fitting introduces some error into the predicted DSB spectra. The effects of this are mitigated somewhat in the SSB spectrum, since each of its channels depends on multiple DSB spectra, as seen in Figure 5.7.

5.2.3 Deconvolution with Baselines

Finally, the full capabilities of the deconvolution algorithm were tested by simulating data with nonzero baselines and performing the deconvolution with baseline fitting enabled. The simulation parameters were identical to those described in Section 5.2.1, except that a random, second-order baseline was added to each DSB spectrum. The single-step deconvolution used the same settings as described in Section 5.2.2. In this simulation, the optimizer settled into a minimum that was far from ideal with $\chi_o^2 = 4,054,340$ $\left(\chi_\nu^2 = 14\right)$

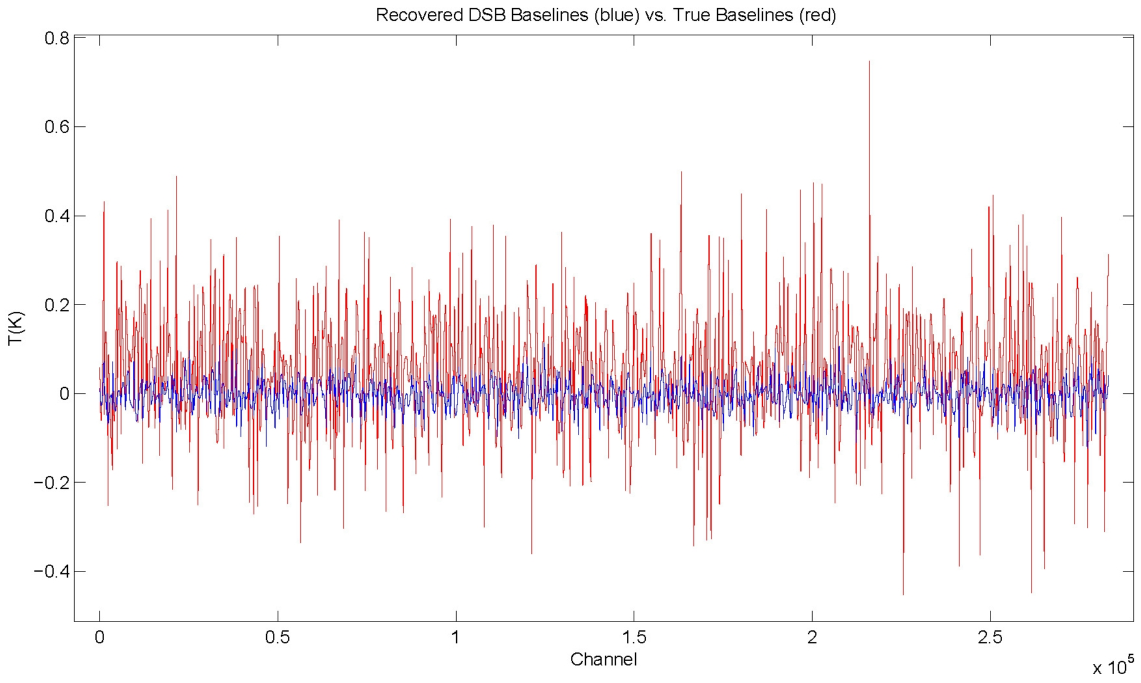


Figure 5.8: Recovered baselines from the deconvolution described in Section 5.2.3, in which the simulated data include baselines, and the baseline-fitting capabilities of the algorithm are enabled. The simulated baselines are shown in red while the baselines generated by the deconvolution algorithm are shown in blue. Clearly, the fit baselines are systematically underestimated. Although the scale used in this plot makes the baselines appear spiky, the individual baselines (both fit and simulated) are smooth functions, as shown in Figure 5.6.

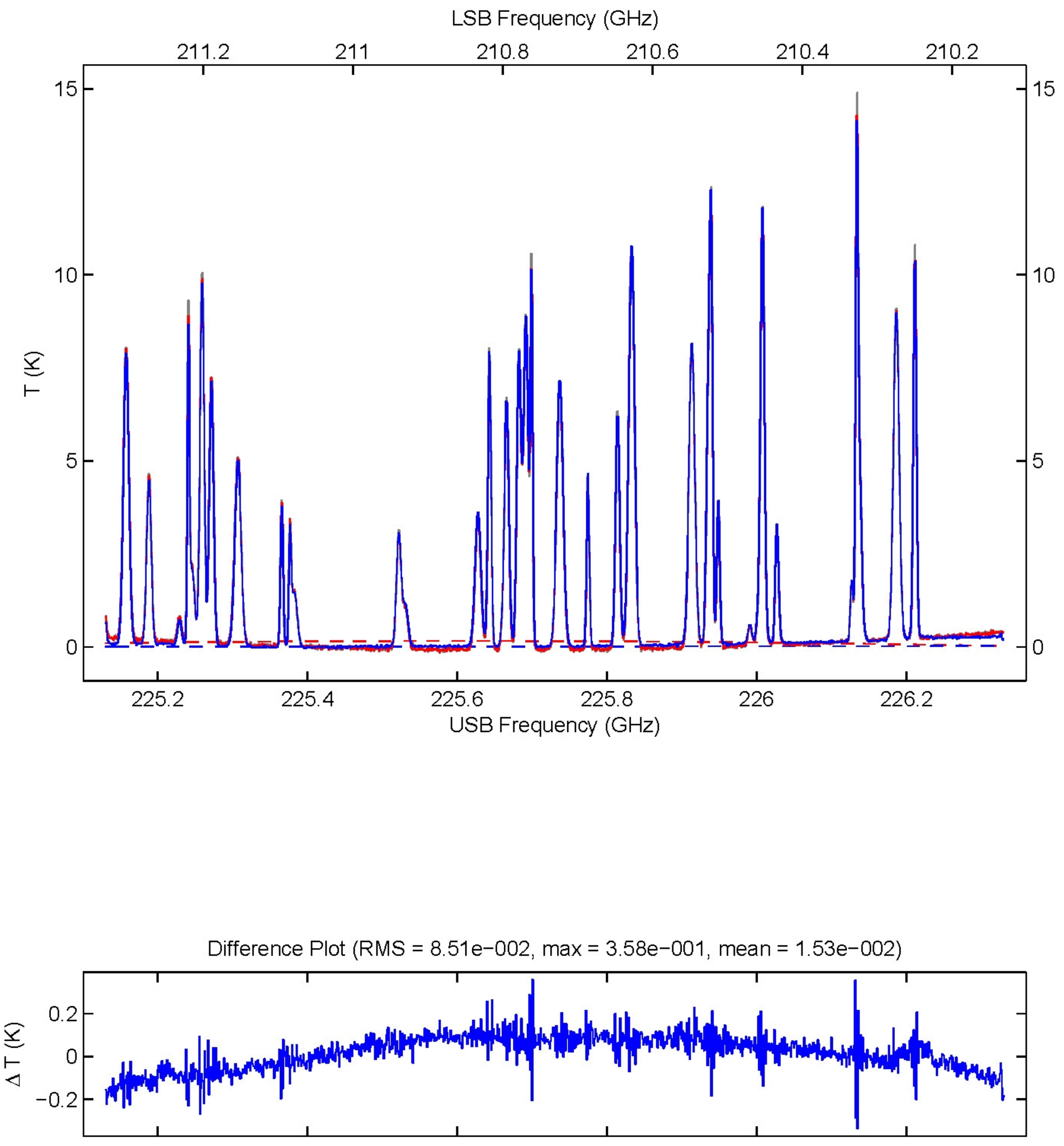


Figure 5.9: A sample DSB spectrum from the simulation with baselines. The red spectrum corresponds to the simulated DSB data, and the red dotted line corresponds to the baseline added during simulation. Blue lines represent the DSB spectrum and baseline predicted by the model. The model baseline underestimates the actual baseline, leading to the strong curvature seen in the difference plot.

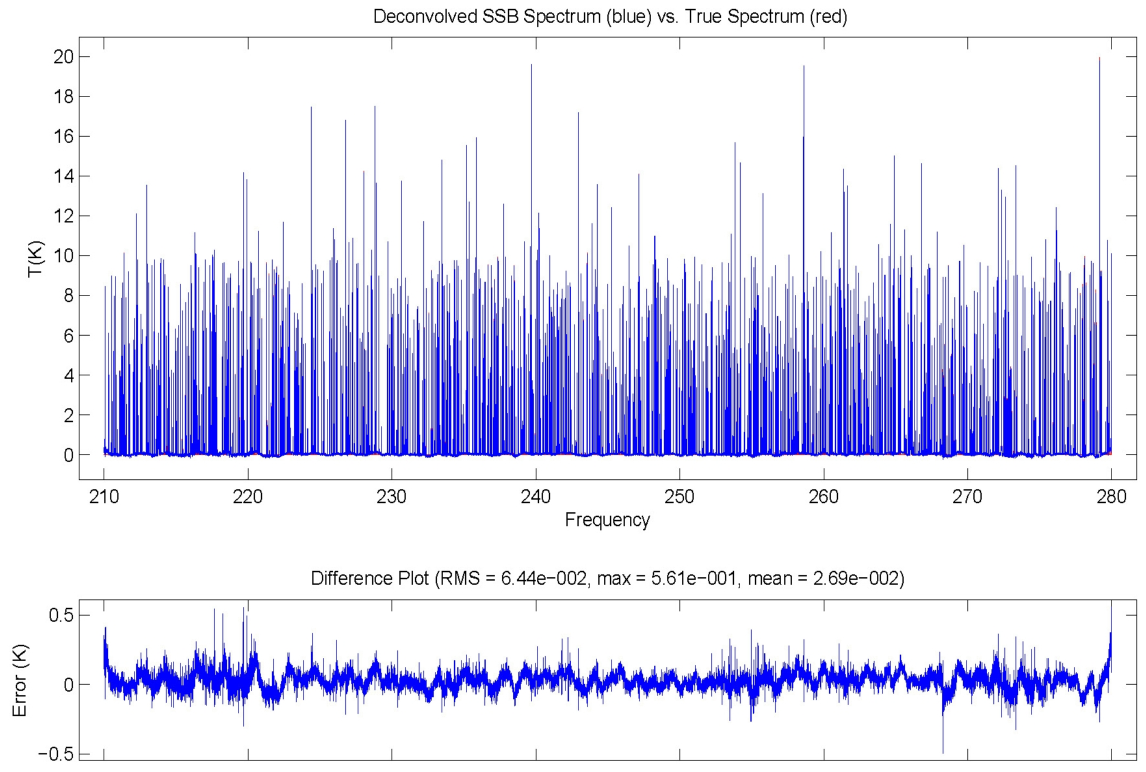


Figure 5.10: The full SSB spectrum from the simulation with baselines. The poorly fit baselines in individual DSB spectra contribute to considerable ripple in the deconvolved spectrum, producing RMS errors twice as large as the previous two deconvolution tests.

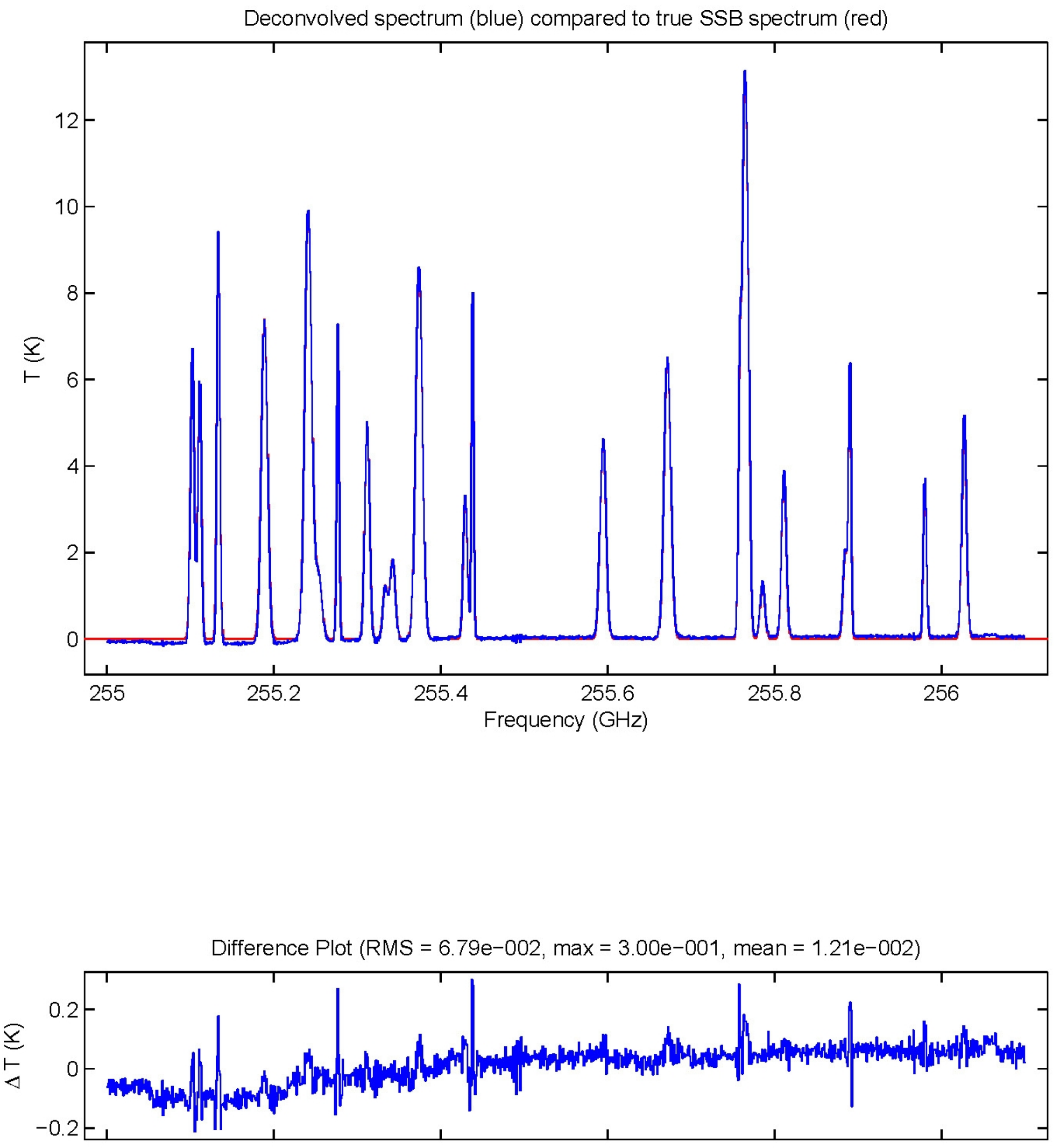


Figure 5.11: An expanded version of the SSB spectrum from the simulation with baselines. The baseline fitting mostly introduces low-frequency ripple into the SSB spectrum. Thus the peaks in the spectrum could be fit reasonable well, although the peak amplitudes would not be quite right.

and $\chi^2_{mod} = 4,098,211$ for 294,679 degrees of freedom. The true parameters offered a substantially better solution of $\chi^2_o = 796,507$ $\left(\chi^2_\nu = 2.7\right)$ and $\chi^2_{mod} = 839,789$, but the optimizer was unable to locate a parameter set of similar quality. Unlike previous runs, the SSB spectrum recovered from this deconvolution is also poor. The RMS error in the SSB spectrum is 64 mK, twice the previous results while the gain parameters have RMS error of 5.2×10^{-3}, worse by more than a factor of two.

The problem in this deconvolution can be seen clearly in Figure 5.8, which shows that the baselines are systematically underestimated across the entire set of DSB spectra. A sample DSB spectrum is shown in Figure 5.9, where the fit baseline is well below the true baseline added in the simulations. The poorly fit baselines feed through into the SSB spectrum (Figure 5.10), which suffers from significant ripple. It is worth noting that most of the error in the deconvolved spectrum takes the form of low-frequency ripples. As shown in Figure 5.11, it is still possible to fit narrower spectral peaks built on top of that ripple, although the situation is less than ideal.

5.3 Future Upgrades

5.3.1 Modification of χ^2_β

Deconvolving realistically sized simulations challenges the capabilities of cutting-edge computer hardware. The results shown in this thesis were computed using an eight-core CPU with 48 GB of memory. Successful deconvolution of a large data set typically takes $\sim$ 10 hours, which is longer than desired, and some scenarios cannot be tested at all because they exceed available memory. In particular, we have not been able to complete tests in which baseline fitting is performed using $\lambda_B \neq 0$ for a simulated data set with a size comparable to the CSO surveys. Given the current rate of advances in computing power, neither of these should be viewed as a fundamental limitation. In addition, the commercial availability of servers-for-hire provides an economical way to access powerful computing resources.[6] However, rather than relying solely on brute-force techniques, such

[6]As an example, the Amazon Elastic Compute Cloud (EC2) currently offers a 64-bit virtual platform with the equivalent of an eight-core CPU and 68 GB of memory that can be rented on an on-demand basis for approximately $2.00 per hour. See `http://aws.amazon.com/ec2/`.

as increased computing power, to solve these problems, we would also like to examine the algorithm to see whether its efficiency could be improved. Not only would that allow us to complete the tests that currently run out of memory, but it would also prepare the software to address larger data sets, like those that could be generated by the CSO's new FFT spectrometers, which offer higher resolution than the current AOS.

From this perspective, the choice of χ^2_β in Equation 4.65c is unfortunate, particularly for the direct-optimization case. As discussed in Section 4.11, `lsqnonlin()`, the MATLAB function chosen to perform the optimization, is designed to operate on each of the terms of χ^2_{mod} individually. Therefore, it expects to receive a vector of data containing terms of the form shown in Equation 4.67. With the current definition of χ^2_β, it expects to receive channel-by-channel values of the baseline. Since the $\sim 2,000$ channels of baseline data for each AOS scan depend on only a few underlying parameters, it should be possible to significantly reduce the number of values returned to `lsqnonlin()`. Since the purpose of including the χ^2_β term is to minimize the area under the baselines, it would make more sense to provide the area incorporated by each of the polynomials in Equation 4.52. Moreover, since these are orthonormal functions, it should be quite easy to do so.

Reformulating χ^2_β in this way would significantly reduce the dimensionality of the results vector and the Jacobian provided to `lsqnonlin()`, presumably with a corresponding improvements in the dimensionality of its internal operations. With this change, it is likely that the $\lambda_B \neq 0$ simulations could be completed using current computing hardware.

5.3.2 Validating Jacobian Matrices

Providing accurate Jacobians remains one of the most vexatious aspects of adding new functionality to the deconvolution code. The *Mathematica* notebook described in Section A.6 greatly simplifies the process by allowing automatic calculation of analytic Jacobians for new forms of χ^2_{mod}. However, implementing those in MATLAB provides ample opportunity for error, and ultimately, a reliable method is needed for validating the Jacobian that is created.

Several MATLAB functions include an option to verify the Jacobian using discrete approximations. This functionality has been extremely useful, but it has largely served in a binary sense; either the Jacobian passes the comparison, or it does not. When a discrepancy is found, the differences between the discrete and analytic Jacobians are simply printed to the screen. Given the dimensionality of the matrices involved, it is difficult to interpret these results. It would be quite useful to have a set of debugging functions that would allow the user to form discrete approximations of the Jacobian, compare them to the analytic form, and interpret the results. By designing the functions to be sensitive to the structure of the current problem, it would be possible to probe individual pieces of the Jacobian, making it much easier to track down the individual errors. More meaningful outputs, such as plots of the RMS error as a function of discrete step size, would also make it easier to diagnose whether a problem actually exists. Such functionality will be particularly important when trying to implement the pseudo-linear mode.

Currently, baseline fitting is not performing as expected, particularly for tests in which $\lambda_B = 0$. There have been some hints that these problems might be caused by an error in the Jacobian that leads the optimizer to search in the wrong direction, but we have not been able to investigate this possibility adequately due to the lack of appropriate diagnostic tools. This is particularly unfortunate in light of the difficulties described in Section 5.3.1 with running simulations on cases with $\lambda_B \neq 0$, as it means that we have not been able to fully validate any form of baseline fitting on simulated data sets with sizes similar to those of the CSO surveys.[7]

5.3.3 Identifying and Preventing Ghosts

One of the challenges of interpreting a deconvolved spectrum is to determine whether particular spectral features are real, or whether they represent artifacts of the deconvolution process. It can be difficult for the deconvolution algorithm to reconstruct the SSB spectrum for channels in which the DSB data include a particularly strong line from the other sideband. One way of combating this problem is to simply remove any channels with strong

[7] An obvious solution to this is to validate the baseline-fitting methods on smaller data sets, but that produces its own problems. For undetermined reasons, very small data sets cause the optimizer to run off to undesirable portions of parameter space. The optimizer is much more stable on large data sets. Therefore, the challenge is to find a data set large enough to allow stability in the optimizer, but still small enough to fit in memory.

lines from the DSB data and rely on the redundancy of the observations to fill in the missing data in the deconvolved spectrum. The deconvolution code does not currently contain this capability, but it would be worthwhile, and relatively simple, to add it to the future. This is particularly important for recovering high-sensitivity spectra in which the small peaks are of greater importance than the large ones. [8]

It is equally important to be able to identify any such ghosts that remain in the deconvolved spectrum. One way to do this is by looking at each of the underlying DSB spectra for a given SSB channel to see if it contains strong lines that could contribute to a ghost. To simplify this process, we would like to add a special plotting mode that would show the SSB spectrum at the top of each page with the contributing DSB spectra underneath. While not difficult in principle, developing the software to support such informative displays can be time-consuming, so it has not been implemented in the initial version of the deconvolution software.

5.4 Summary

The results of these simulations imply that the deconvolution algorithm works quite well, as long as there are no baselines in the DSB data. The algorithm successfully recovered deconvolved spectra with sensitivities of $\sim$ 30 mK in the two examples in which the underlying data were simulated without baselines.

At present, the deconvolution software does a poorer job of recovering spectra with baselines. The data set with baselines had the same underlying radiometric noise as the previous simulations, but the resulting SSB spectrum had twice the RMS error compared to the true SSB spectrum. This implies that the deconvolution algorithm was unable to preserve the high quality of input data. Even so, the baseline-fitting problems are unlikely to significantly affect parameters derived from peak fits for narrow-line sources.

All of the simulations had improbably high values of χ_o^2, even when the true parameters used for the simulation were inserted into the model. This most likely indicates that

[8] As an intermediate measure, an experimental feature has been included in the code to increase the uncertainty on large peaks by setting the uncertainty in each channel to the greater of its radiometric uncertainty (Equation 2.1) or a fraction of its brightness temperature. We have not developed enough experience with this method to determine its usefulness.

the model does not accurately reproduce all the features of the simulation; however, this does not necessarily imply that anything is wrong with the model. Rather, it indicates that the simulation engine and deconvolution model produce different, inconsistent results. Since the derivation of the deconvolution model appears sound (Chapter 4), it is reasonable to continue using it as the basis for analyzing astronomical data. Given the low RMS errors in the SSB spectra, it appears that the deconvolution algorithm can accurately recover the single sideband spectrum, even when χ_o^2 is high. Unfortunately, this situation does prevent us from using χ_o^2 as a goodness-of-fit test.

Chapter 6

Molecular-Line Surveys

In September and November of 2007, two observing runs were completed using Z-Rex with the high-resolution IF processor and 4-GHz AOS array described in Section 3.5. Orion KL and Sgr B2, two frequently studied high-mass star-forming regions, were observed, as was the low-mass Class-0 protostar L1157. The Orion survey achieved good coverage from $\sim$ 220 - 276 GHz, with expected post-deconvolution sensitivity of $\sim$ 10 - 15 mK. The L1157 survey covered a similar frequency range, although the integrations were not quite as deep, with expected sensitivity of $\sim$ 20 - 30 mK. Sgr B2 was only observable during the September run, leading to narrower coverage of $\sim$ 220 - 250 GHz with expected sensitivity of $\sim$ 15 - 20 mK. The sensitivity of each survey tends to be best near the middle of the frequency range, which is heavily oversampled, and drops off towards the ends.

6.1 L1157 Survey

L1157 consists of a dark cloud containing an isolated Class-0 protostellar core about 440 pc from Earth. L1157 is particularly well known for its powerful bipolar outflows, which propagate along an axis that is nearly perpendicular to the line of sight. Such favorable orientation in an isolated core makes this an ideal location to study outflow-driven shock chemistry. The geometry can be inferred from the fact that there is very little overlap between the redshifted (northern) lobe and the blueshifted (southern) lobe of the outflow. The redshifted lobe has been found to move at higher velocity and have a larger size while the blueshifted lobe generates stronger molecular emission; these facts are usually interpreted as signs that the southern lobe is pushing into denser material than the northern one [Umemoto et al., 1992, Bachiller et al., 2001].

The central source, L1157-mm (IRAS 20386+6751), has been studied using the Plateau de Bure interferometer by Gueth et al. [1997], who find that it meets all of the attributes of a Class-0 protostar as defined by Andre et al. [1993]. Their observations reveal a compact source $\sim 1''$ ($\sim$ 500 AU) in diameter at the center of the outflow. They fit a simple three-part model to their observations and find that the core consists of $\sim 0.2\ M_{sun}$ of material at a temperature of 36 K. The model also indicates that the surrounding envelope has a temperature $\lesssim$ 20 K, a radius $\sim$ 5,000 AU, and contains at least 3 M_{sun} of material. The tightly collimated outflows and the fact that the envelope mass is much greater than the mass in the central core imply that this is a relatively young protostar that is actively accreting material.

The geometry of L1157 makes it ideal for studying shock-driven chemistry. For instance, Bachiller and Gutierrez [1997] identify many molecules with strong enhancements in the outflows, particularly SiO and CH_3OH. This is consistent with a model in which the shock fronts sublimate ice mantles off dust grains and even drive some of the Si and/or SiO from the grains themselves into the gas phase. Conversely, they also identify molecules that are not detected in the outflows and therefore represent good tracers of the quiescent envelope, including C_3H_2, N_2H^+, $H^{13}CO^+$, and DCO^+.

Bachiller et al. [1993] map the L1157 outflow in ammonia, NH_3, which can be used as a particularly sensitive indicator of kinetic temperature. They find a cool, quiescent disk of gas that sits perpendicular to the outflows with a temperature of 13 K. They also observe broad-line emission from hotter NH_3 ($T \sim 50$ - 100 K) that traces the regions in which SiO abundance is highly enhanced. These results are consistent with Bachiller et al. [2001], who use a large-velocity-gradient (LVG) model to estimate temperatures of $T \approx 13$ K in the envelope and $T \approx 40$ - 100 K in the outflow. Density estimates vary by an order of magnitude across different studies, but typically suggest that assigning the quiescent gas a density of $n\left(H_2\right) \sim 10^6$ cm^{-3} and the shocked regions a density of $n\left(H_2\right) \sim 10^5$ cm^{-3} would not be unreasonable.

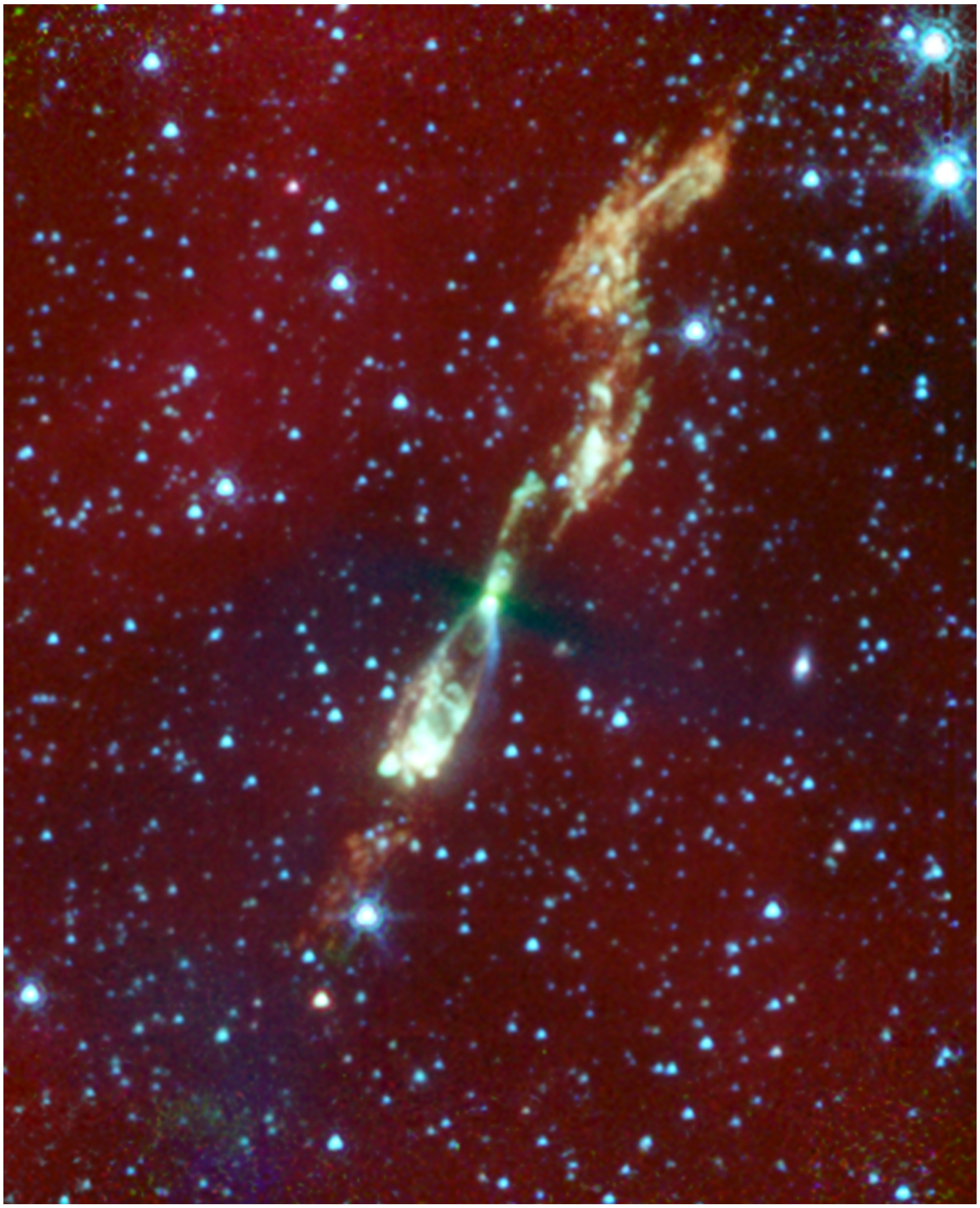


Figure 6.1: Spitzer image of L1157. (Image courtesy of NASA/JPL-Caltech/UIUC, Caltech/SSC. Also see Looney et al. [2007].)

A point in the southern lobe of the outflow, offset from L1157 by (20$''$, $-60''$), is frequently studied. Bachiller and Gutierrez [1997] identify it as location B1, one of three points in L1157 at which they search for a variety of molecular lines. It is very close to the location of peak CO emission identified by Umemoto et al. [1992], as well as the peak of the SiO emission found by Mikami et al. [1992], corresponding to point B in their study. Avery and Chiao [1996] observe many lines of CH_3OH, SiO, SO, and SO_2 at that point; based on an LVG analysis, their results imply a temperature of $T \approx 100$ - 140 K and density of $n\,(H_2) \sim 1.0 \times 10^5$ cm^{-3} at B1. This is reasonably consistent with the ammonia measurement of $T \approx 84$ K made by Bachiller et al. [1993] at B1. Point B1 is also included in the extensive molecular-line maps of Bachiller et al. [2001], in which it is identified as one of three "clumps" corresponding to periods of increased outflow activity. They estimate that the clump at B1 has a dynamical age of $\sim$ 7,000 years, compared to $\sim$ 15,000 years for the outer reaches of the outflow. Although not terribly precise, dynamical estimates of age provide an interesting method of testing the predictions of time-dependent chemical models.

Based on previous work, we identified point B1 as an interesting source for a line survey. A significant amount of study has already been performed on that location, offering a good deal of background for interpreting the results. In addition, strong molecular-line emission should be assured since it is at or near the peaks of the CO, SiO, and CH_3OH emission distributions. Line profiles are likely to be unusual in this region; Avery and Chiao [1996] find that some lines, such as CO, are strongly asymmetric, with long blueshifted wings while others, such as SiO, consist of a narrow central peak and a broad, blueshifted wing.

6.1.1 Observations

Observations of the B1 point in the L1157 outflow were made in September and November of 2007 using the Caltech Submillimeter Observatory (CSO). The coordinates used for L1157 were $20^h39^m06\overset{s}{.}19$ and $68^\circ02'15\overset{''}{.}9$ (J2000), matching the position of the compact source found Gueth et al. [1997] and the position used by Bachiller et al. [2001]. Point B1 corresponds to an offset of $(\Delta RA = 20'', \Delta dec = -60'')$ from L1157. Following Bachiller et al. [2001], we assume $v_{LSR} = 2.7$ km/s, which appears to be the most commonly used value in the literature, although the interferometric studies of Gueth et al. [1997] find a slightly lower value of $v_{LSR} = 2.6 \pm 0.5$ km/s. Based on molecular maps of the outflow from Bachiller et al. [2001], we chose a chopping throw of 300″, sufficient to clear all of the molecular emission from the protostar and the outflows, even in the worst-case orientation in which the off position is in the direction of the redshifted lobe. Chopping frequency was 1.012 Hz, representing a trade-off between integration efficiency and sky stability. A few initial scans were taken at a different chopping throw (240″ instead of 300″); after comparing them to scans at the same frequencies taken with the larger throw, it was determined that they could be safely included in the data set.

Pointing was checked every 2 - 3 hours using planets and CO line sources. Results were generally consistent, although there were occasions on which pointing corrections appeared to shift by $\sim$ 6 - 8 ″ during a set of integrations. Additionally, in order to complete the observations, it was necessary to do some observations at large zenith angles (ZA $\sim$ 60° - 70°), which can lead to pointing drift. Given that the outflow is a large, diffuse source, such shifts are unlikely to significantly affect the observations; however, pointing error is likely to represent the dominant source of uncertainty in determining line strengths. As an example, scans taken at high zenith angle under moderate skies ($\tau \approx 0.15$) indicate that some line flux was lost ($\sim$ 10 - 20 %) compared to reference scans taken previously at the same frequencies.

A set of LO frequencies was chosen to optimize deconvolution accuracy. LO settings were selected so that every frequency would be observed at least four times in each sideband, for a total of eight samples, except at the band edges, in which case extra measurements would be taken within the single available sideband.[1] The LO frequency was changed in alternating steps of 2 GHz and 100 MHz, for example, 234 GHz, 234.1 GHz, 236

[1]Comito and Schilke [2002] study how redundancy affects the quality of the deconvolved spectrum. For a perfect receiver, they find that observing every frequency once in each sideband (two samples total) is sufficient. However, once pointing errors and sideband imbalances are introduced, higher redundancy (at least four samples per frequency) improves the quality of the reconstructed spectrum and reduces the number of "ghost" lines. The optimal solution that minimizes noise in the final deconvolved spectrum depends on the amount of overhead involved in each retuning and therefore varies with telescope and receiver configuration. Since ground-based observations often produce uneven baselines for the spectra, we chose a relatively high value of eight samples per frequency.

GHz, 236.1 GHz, ... To avoid producing "ripple" artifacts in the deconvolved spectrum [Comito and Schilke, 2002, Fig. 4], a small, random offset of -50 MHz $\leq \Delta f \leq +50$ MHz was added to each LO frequency. The lowest edge of the band was chosen to provide sufficient LO power and avoid the sideband imbalance shown in Figure 3.26. The upper limit was 270 GHz, corresponding to the range of the Gunn-oscillator LO. Unfortunately, due to time constraints, we were not able to observe all of the planned LO frequencies, and coverage was somewhat limited at higher frequencies. (See Figure 6.3.)

Observations of L1157 occurred on five different nights, requiring about 20 hours of observatory (clock) time. Night-to-night compatibility of data was checked by comparing reference scans at the same frequency across multiple nights. Three nights offered $\tau \approx 0.08$ - 0.1, with single-sideband system temperatures $T_{sys} \sim 400$ - 500 K. One night was slightly wetter, producing $\tau \sim 0.15$ and $T_{sys} \sim 500$ - 750 K, with the higher values corresponding to larger zenith angles. The worst of the five nights had $\tau \sim 0.25$, generating $T_{sys} \sim 900$ - 1000 K. Reference scans on that night showed a noticeable, but acceptable, loss in line intensity at large zenith angles. Integration times were chosen according to Equation 2.1 to provide consistent $T_{RMS} \sim 70$ mK across the survey for each DSB scan, equivalent to a DSB scan with $T_{RMS} \sim 25$ mK once the eightfold sampling is taken into account.

Calibration relied on measurements of P_{hot} and P_{sky} using a chopper wheel, as described in Section 2.1, generating spectra calibrated according to the T_A^* scale. Unless otherwise noted, spectra in this thesis can be assumed to be measured on that scale. The 4 - 8 GHz IF processor and 4-GHz AOS array described in Section 3.4.1 were used for the backend, and a phase-locked Gunn oscillator was used as the LO source.

To determine data quality, data and calibration scans were reviewed to look for anomalies. Sharp spikes (spurs) were found in many scans; usually, they occurred at fixed channel numbers, indicating that they were leaking into the IF signal (rather than the RF signal). Spurs could be identified most easily in the calibration scans. A few spurs could be identified with corresponding defects in the data scan, although most of them appeared to be harmless. Any channels corresponding to a spur in the calibration scan were marked as invalid in the data scan.

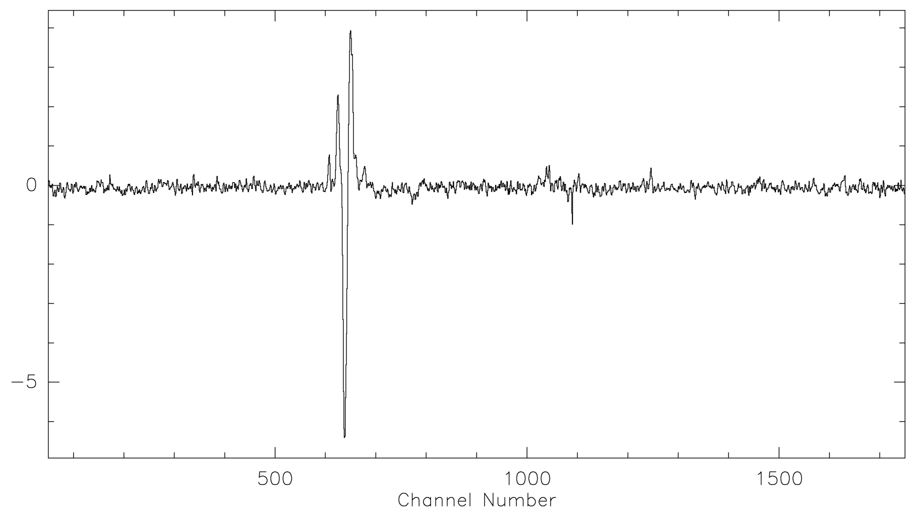


Figure 6.2: The spectral defect around channel 600 was caused by interference from the CSO's microwave oven.

In the September run, we noticed intermittent interference in band 4 (Figure 6.2); eventually, we traced it to the microwave oven in the CSO kitchen. It was difficult to make the interference occur on demand, but it was clearly correlated with microwave use and likely related to the large zenith angle used for the observations of L1157. While reviewing the data, any scans that contained interference from the microwave were flagged. By comparing those data to other scans taken at the same frequency, we determined that bands 1 - 3 were unaffected by the microwave interference, so only the data from band 4 were thrown out.

The vast majority of scans had excellent baselines; there were only a few with a ~ 0.5 K offset. Given the relative sparsity of lines from the source, those lines could easily be fit and removed by hand using second-order baselines in GILDAS CLASS.

6.1.2 Deconvolution

The L1157 data set was deconvolved using the algorithm described in Chapter 4, as implemented in MATLAB. Removing baselines in the deconvolution software slows the process considerably and adds additional uncertainty to the results; given the overall excellent condition of the baselines in these scans, the baseline-fitting elements of the code were disabled (by setting $\eta_B = 0$). A continuation solution (as described in Section 4.9.1) was used and constraints were set to $\lambda_S = 0.11$ and $\lambda_G = 1.3 \times 10^4$.[2]

Using an 11-step continuation method, the deconvolution algorithm converged to a reasonable solution in a little over 7 hours, with the following goodness-of-fit measures (for 362,193 degrees of freedom):

$$\begin{aligned}
\chi_o^2 &= 114,313 \\
\chi_S^2 &= 46.5 \\
\chi_\gamma^2 &= 983 \\
\chi_{mod}^2 &= 115,343.
\end{aligned}$$

The χ_o^2 value corresponds to a reduced chi-square of $\chi_\nu^2 \approx 0.316$, which is improbably low. Presumably, this indicates that the uncertainties assigned to the DSB data were too large, although it is not clear why that should be the case since they are based on Equation 2.1. The deconvolved spectrum, however, looks reasonable, as do the comparisons between the predicted and observed DSB spectra ($\boldsymbol{d^{o,R}}$ and $\boldsymbol{d^R}$). An overview of the single-sideband spectrum is shown in Figure 6.3, with detailed plots in Appendix D. An overview of the comparison between DSB observations and predictions is shown in Figure 6.4 and indicates good agreement, with an RMS error of $\sim$ 37 mK, averaged across the spectrum. Finally, the sideband-gain factors, γ, are shown in Figure 6.5; the relatively small values found by the optimizer are reassuring.

6.2 Orion KL

While much of this work has focused on low-mass star formation, the birth of massive stars is also an area of active research. Low-mass star formation has the advantage of being better understood theoretically; high-mass star formation, in contrast, provides a rich laboratory for studying astrochemistry. The increased temperatures and larger volumes

[2]These values correspond to `lambda_s_fraction = 0.01` and `lambda_g_fraction = 0.01`. (See Footnote 3 in Chapter 5.)

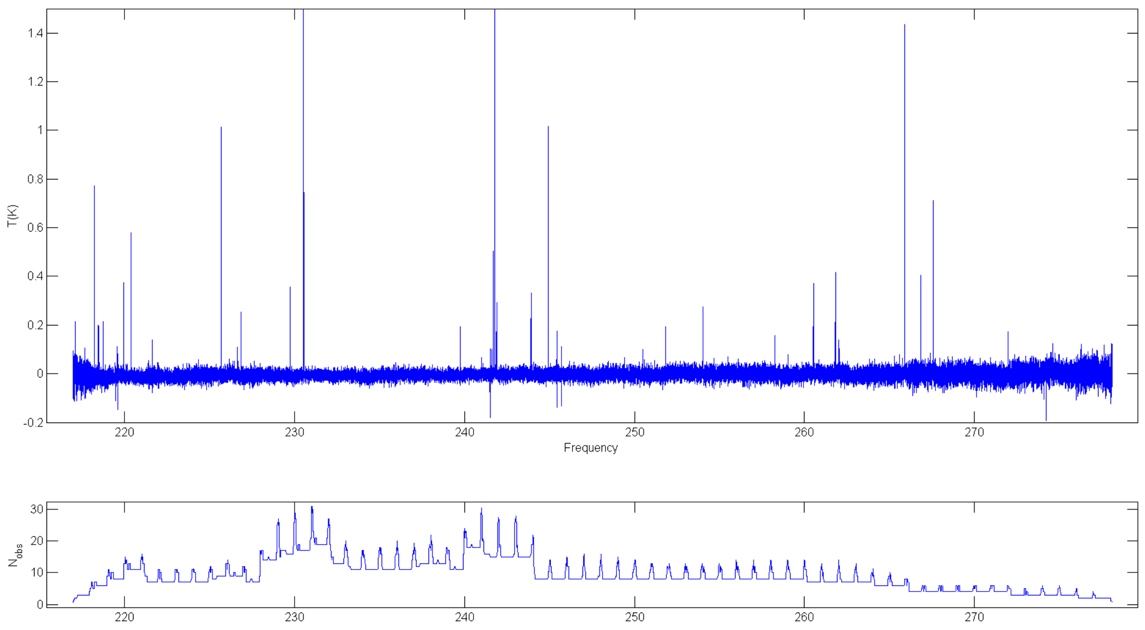


Figure 6.3: Overview of deconvolved spectrum for L1157 at position B1 in the outflow. The vertical scale has been expanded to show detail, truncating several peaks. The full extent of those peaks can be seen in Appendix D. The bottom plot shows how often each channel in the deconvolved spectrum was observed.

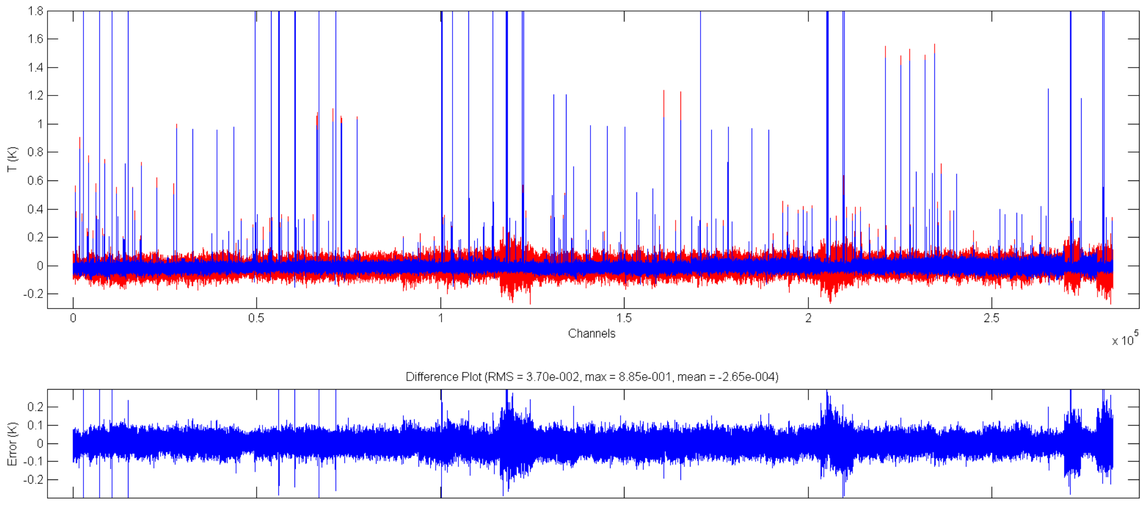


Figure 6.4: Overview of comparison between observed DSB data (red) and predicted DSB spectra (blue) for the L1157 spectra. This plot represents a vector of all DSB data, with individual observations stacked end to end. To emphasize details, the vertical scale has been expanded, truncating larger peaks.

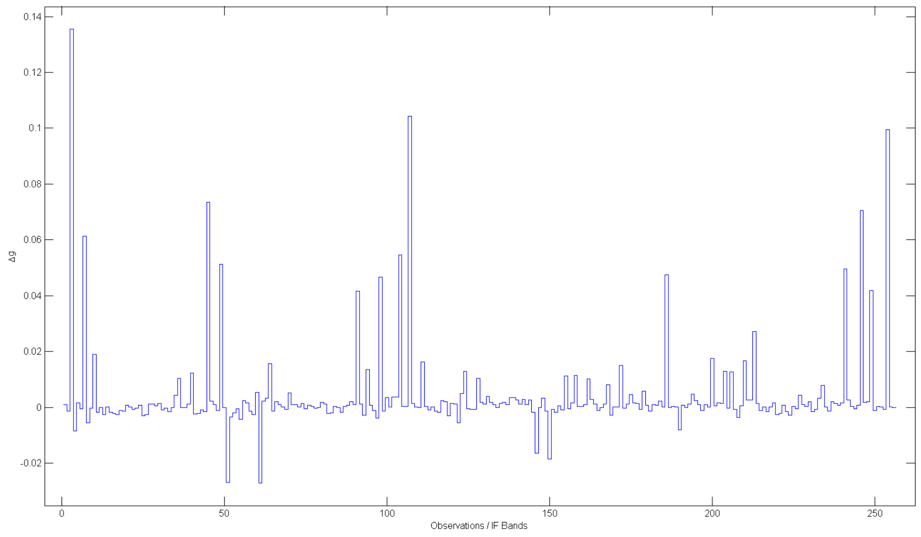


Figure 6.5: Sideband-gain factors found by the deconvolution algorithm for the L1157 observations. A single gain value was used for each AOS scan.

involved in the formation of a massive protostar provide more opportunities for chemical interaction and create a richer, more easily observable line source. The work covered by this thesis includes a collaboration with Susanna Widicus Weaver from Emory University to perform a deep, unbiased line survey of the Orion KL region at 230 GHz. In addition to enabling rapid line surveys, such as the L1157 study, the broad bandwidth of Z-Rex makes it feasible to spend longer integration times on each frequency, permitting a very sensitive survey to be completed in a reasonable amount of time.

Because of its proximity, strong millimeter and submillimeter emission, and relatively narrow line widths, the Orion KL region has been extensively studied and has been the subject of several unbiased line surveys, including one of the first surveys to be completed in the 230-GHz window [Sutton et al., 1985, Blake et al., 1986, 1987], as well as more recent work that extends spectral coverage to 900 GHz [Comito et al., 2005], with several other surveys in between. Very recently, a broad and highly sensitive survey of Orion KL has been completed by Tercero et al. [2010], with further analysis in Tercero et al. [2011]. These surveys use the IRAM 30-m telescope, along with sideband-rejecting receivers, to achieve a detection limit of ~ 20 mK.[3]

[3]The Tercero et al. [2010] survey was published in July 2010, late in the writing of this thesis. Consequently, the results are not discussed further in this work.

Orion KL remains one of the richest molecular-line sources and has become the prototype upon which much of our understanding of high-mass star formation, and astrochemistry in general, has been built. An excellent overview of the source's geometry and prior studies can be found in Comito et al. [2005], and Genzel and Stutzki [1989] offers a very detailed, albeit slightly older, review.

The strong molecular lines in protostellar surveys have largely been identified, so recent laboratory spectroscopy has been focused on measuring the lines of complex organic molecules [e.g., Widicus Weaver, 2005], including some with biological significance. Parallel efforts have been made via submillimeter observations to find these lines in protostellar cores. From a theoretical perspective, constraining abundances of these molecules is important to further development of astrochemistry models. In addition, finding evidence of prebiotic molecules in the primeval components of star formation could offer important clues to determining how the seeds of life were planted on Earth during our solar system's development.

Because these molecules are expected to occur in relatively lower abundance, the richer, brighter spectra of high-mass protostars represent an obvious target. Even so, deep observations are required to find evidence of the molecules' existence. A positive identification requires a simultaneous detection of multiple lines, which requires the observations to encompass a moderate bandwidth and can be quite difficult in the presence of a crowded spectrum and heavily blended spectral peaks. Rather than continuing to search for these molecules individually as new lines are discovered, we have decided to assemble very broad, unbiased line surveys of several massive star-forming regions. This allows the existence of previously undetected molecules to be determined quite rapidly once the spectra are known from laboratory studies. The large bandwidth of our line survey permits many more lines to be detected, and also provides the context for developing robust deconvolution and line-identification tools.

The sensitivity of the prior 230-GHz Orion KL survey was estimated to be ~ 300 mK [Sutton et al., 1985, Blake et al., 1986]. We aim to improve this sensitivity by at least an order of magnitude, to ~ 15 mK. This provides a particular challenge to our deconvolution algorithms, as they must be capable of preserving the high sensitivity achieved in the observations. Much of the effort in developing and testing the deconvolution software has been focused on positively demonstrating that the software can recover spectra down to this level.

6.2.1 Observations

Observations of Orion KL took place during the same observing runs (and often on the same nights) as the L1157 observations described in Section 6.1.1, incorporating scans from six different nights. The observations, including pointing checks, required approximately 25 hours of observatory (clock) time. Four of the nights offered $\tau \sim 0.07$ - 0.1, resulting in system noise temperatures of $T_{sys} \sim 350$ - 400 K. One night had $\tau \sim 0.14$ and $T_{sys} \sim 450$ - 500 K while the worst night produced $\tau \sim 0.2$ - 0.25 and $T_{sys} \sim 550$ - 700 K. Pointing was checked every 2 - 3 hours and corrected as needed. Observations were taken at coordinates of $5^{h}32^{m}47\overset{s}{.}19$ and $-5^\circ 24' 21''$ (J1950), matching those of Sutton et al. [1985] and Blake et al. [1986]. Off-position subtraction was performed using the chopping secondary with a throw of 70″ at a frequency of 1.121 Hz, producing good integration efficiency. The receiver configuration was identical to that used for L1157. LO frequencies were chosen using similar principles to those described for L1157 so that each frequency would be observed eight times. A slightly different spacing strategy was chosen so that LO steps were ~ 1 GHz apart, but a small, random offset was still included in each setting. Integration times were chosen so that each DSB scan would have $T_{RMS} \sim 45$ mK according to Equation 2.1, equivalent to $T_{RMS} \sim 15$ mK (before deconvolution) for this LO scheme.

6.2.2 Deconvolution

The DSB data was deconvolved to produce a single-sideband spectrum using the techniques described in Chapter 4. Deconvolution was done both with and without baseline fitting, and found to produce better results when second-order baselines were included. A multi-step continuation approach was used with constraint values of $\lambda_S = 0.018$, $\lambda_G = 2160$, and $\lambda_B = 0.018$.[4] Major spurs were removed, but the spectra were not reviewed as thoroughly as the L1157 data. Until that review can be completed, these should be considered preliminary results.

An overview of the deconvolved spectrum is shown in Figure 6.6, and several expanded spectra are displayed in the discussion that follows. The deconvolution required

[4]These correspond to `lambda_s_fraction`, `lambda_g_fraction`, and `lambda_b_fraction` each being set to 0.01. (See Footnote 3 in Chapter 5.)

just under 30 minutes to run and produced the following goodness-of-fit values (for 60,551 degrees of freedom):

$$\begin{aligned}\chi_o^2 &= 425,318\\ \chi_S^2 &= 2,834\\ \chi_\gamma^2 &= 256\\ \chi_\beta^2 &= 3,406\\ \chi_{mod}^2 &= 431,815.\end{aligned}$$

The χ_o^2 value corresponds to a reduced chi-square of $\chi_{o,\nu}^2 \approx 7$, which is exceedingly high. This is indicative of the fact that the predicted and observed DSB spectra showed differences with RMS error $\sim$ 80 mK. The results for the sideband-gain factors were quite reasonable, with $-0.1 \lesssim \gamma^a \lesssim +0.1$. As shown in Figure 6.7, the baseline fits also appear to be quite reasonable.

This is a preliminary deconvolution using raw spectra that have had very little processing performed on them. Doing the same type of manual review of calibration scans and spectra that was applied to the L1157 data should considerably improve these results. Additionally, adjusting some of the constraints or the solution method might allow the optimizer to find a better solution more easily, and it is reassuring to see that the gains and baselines are qualitatively reasonable, indicating that the optimizer did not locate an undesirable portion of parameter space. These results are also encouraging in that they demonstrate the capability of the software to be used during an observing run to guide the remaining observations. A deconvolution launched at the end of a night of observing would likely be completed by the beginning of the next night. Since the software can produce results like these on minimally processed data, this represents a realistic scenario for harried observers.

Finally, these results are exciting because they show the benefits of including the baseline fitting as part of the deconvolution. For comparison, the same data set was deconvolved using identical methodology except that baseline fitting was turned off. Although

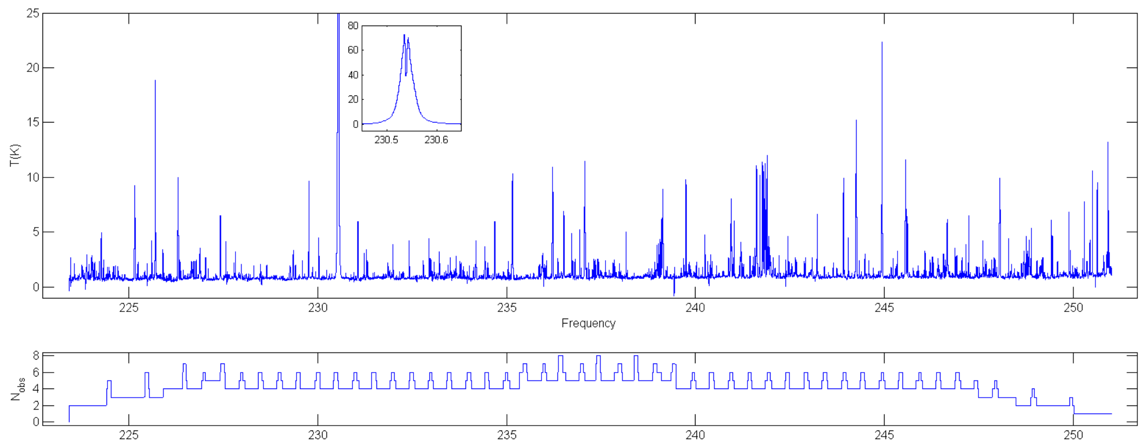


Figure 6.6: Deconvolved spectrum from part of the Orion KL line survey. The full survey offers similar sensitivity up from $\sim$ 220 - 270 GHz. The bottom plot shows the number of times each channel in the deconvolved spectrum was observed while the inset shows the full extent of the CO line.

the resulting deconvolved spectrum looks very similar to the eye, the deconvolution resulted in $\chi_o^2 = 573,382$ for $60,797$ degrees of freedom, corresponding to reduced chi-square of $\chi_{o,\nu}^2 \approx 9.5$, an increase of more than 35%. Given that many of our spectra reached the line-confusion limit, at which nearly every channel appears to be part of a line, determining what portion of the flux should be attributed to a baseline offset is nontrivial. Performing the baseline fit as part of the deconvolution allows the algorithm to take advantage of all the information it has about the spectrum in making this determination. We hope that this not only makes it more convenient for observers, but also improves the reliability of baseline fits.

6.2.3 Comparison to Prior Surveys

Figure 6.8 shows a comparison between the previous 230-GHz line survey of Orion [Sutton et al., 1985] and the current results. The overall similarity of peak shapes and relative strengths is striking, and gives us further confidence in our deconvolution results. It remains unclear why the current survey measured stronger line intensities, but the SSB results are consistent with the underlying DSB spectra. The negative peak near 239.45 GHz is presumably a ghost caused by over aggressive subtraction of a strong line in the other sideband.

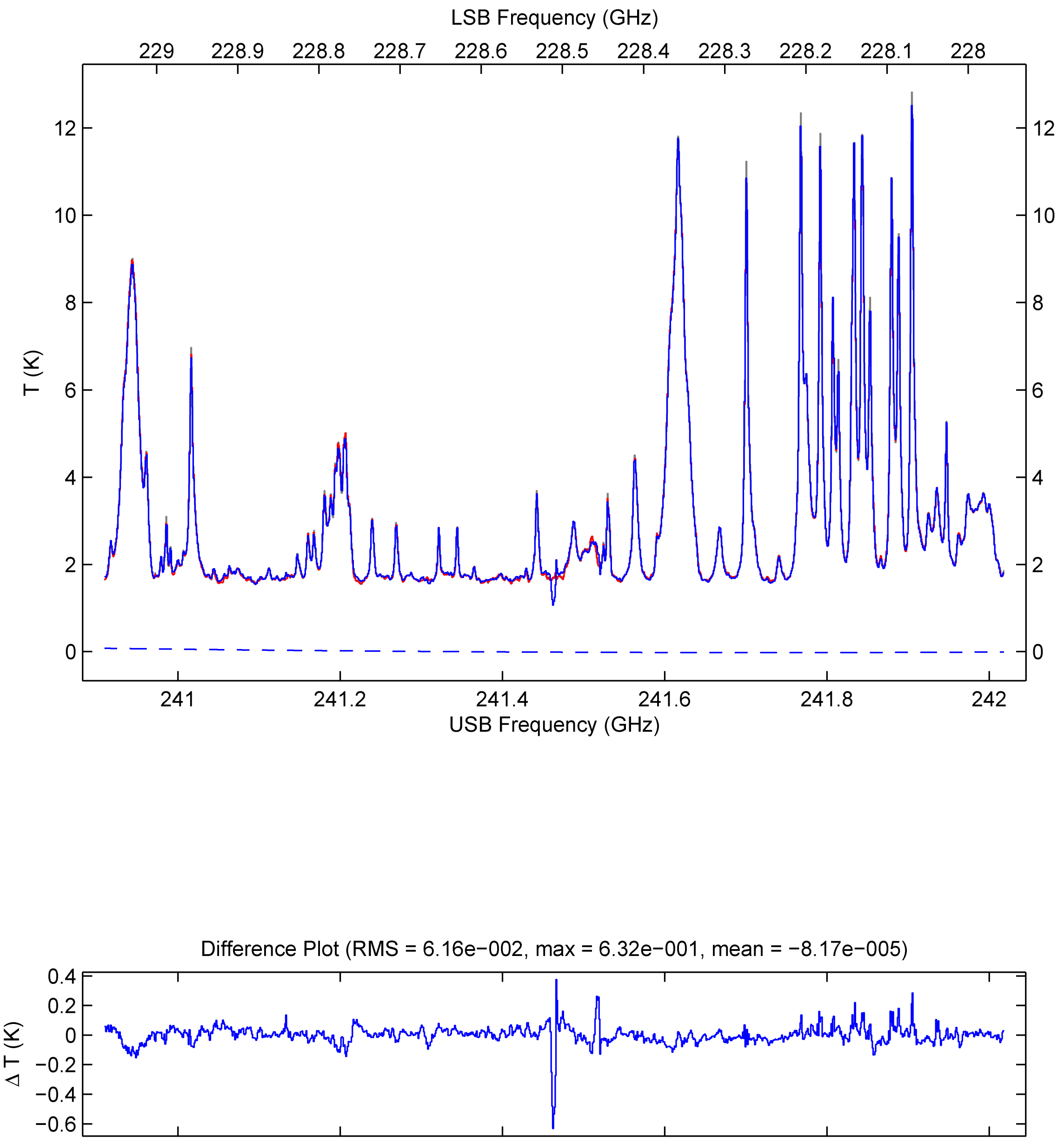


Figure 6.7: Sample DSB-comparison plot from Orion KL deconvolution. The blue line represents the predicted DSB spectrum while the red line shows the resampled DSB data. In general, agreement is good, although glitches can be seen (e.g., near the center of the plot). The gray line in the background represents the raw (non-resampled) DSB data, indicating that the resampling process preserved the data accurately. The dotted blue line shows the baseline fit by the deconvolution algorithm; although quite small, it is non-zero. The bottom panel shows the difference between the resampled DSB data and the predicted DSB spectrum; this particular scan has RMS error of 61.6 mK and a mean offset of 81.7 μK. The worst error across the band is 632 mK.

The smoother spectrum in the current work is indicative of the improved sensitivity, which greatly clarifies spectral analysis. The small bumps that persist are believed to be spectral peaks, not statistical fluctuations. For instance, the lowest-frequency peak of CH_3CCH shown in the top figure is barely visible above the noise in the prior survey, but shows up clearly in the current one. Conversely, the possible peak just to the right of 239.3 GHz disappears in the current survey. The left-hand side of the SO_2 peak shows three small bumps of approximately equal intensity in the prior survey, but the current results imply that two of these are noise artifacts while the third (lowest-frequency) bump could represent a blended peak. There are several peaks in the current survey, such as the set of three blended peaks near 239.33 GHz and the lone peak near 241.74 GHz, that hint at additional lines to be identified, although further analysis is needed to ensure they are not deconvolution ghosts or noise artifacts. It is intriguing that each of these corresponds to a similar structure in the prior survey that is marginally lost in the noise.

The fact that barely visible peaks in the previous results can now be seen clearly is a testament both to the benefits offered by improved sensitivity and to the effort that was put into the previous work to retrieve small lines from the noise. The science to be done with the new spectrum will require equally careful analysis, as the most interesting results will be found near its noise threshold. As shown in Figure 6.9, once we expand the scale on the new survey, the discrimination between peaks and noise is no longer so simple.

6.2.4 Comparison to XCLASS

Peter Schilke and Claudia Comito, both of the Max-Planck-Institut für Radioastronomie in Bonn, Germany, have also put significant work into developing reliable deconvolution techniques for double-sideband spectra [Comito and Schilke, 2002]. They have developed a tool called XCLASS that adds deconvolution capabilities onto the popular GILDAS CLASS spectral-analysis package and have used it successfully for deconvolving large data sets, such as the 850-GHz Orion survey described in Comito et al. [2005].

To compare the different deconvolution techniques, Darek Lis, who is familiar with using XCLASS, ran its deconvolution facility on the Orion data set used here. He cut off noise at the band edges, removed major spurs, and fit zeroth-order baselines to the individual

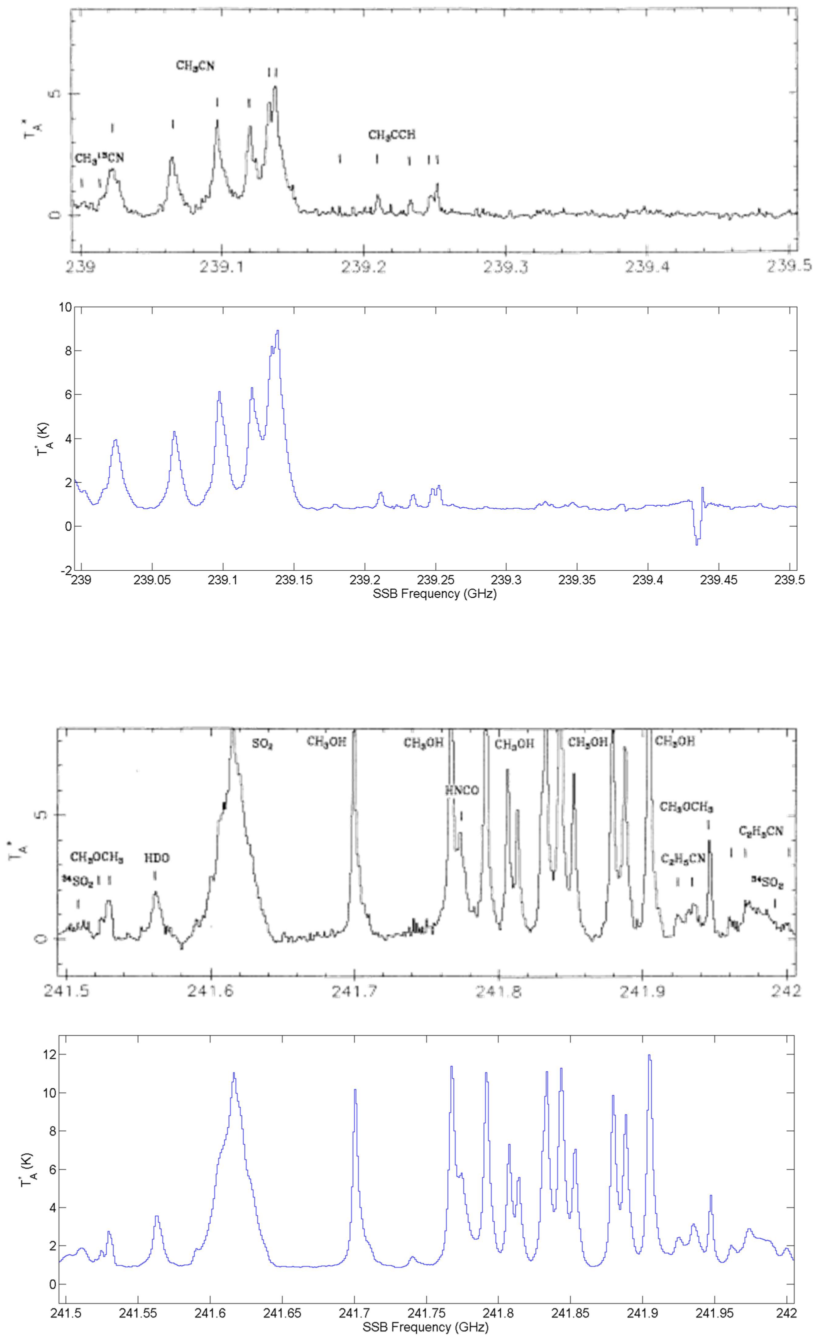


Figure 6.8: Comparison between previous 230-GHz survey of Orion (top panels, Sutton et al., 1985) and preliminary deconvolution of current results (bottom panels). The prior survey has a sensitivity of $\sim$ 300 mK while we expect the current survey to have a sensitivity of $\sim$ 15 mK.

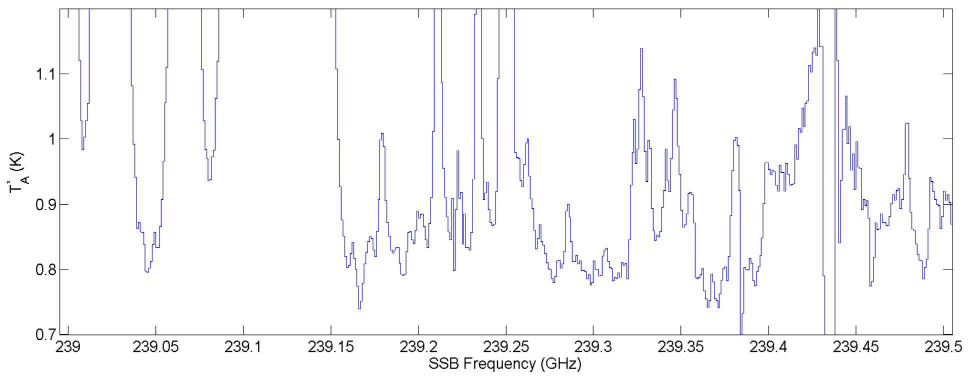


Figure 6.9: An expanded view of the deconvolved Orion spectrum, showing both the peril and the promise that awaits as we attempt to identify the small peaks revealed by our higher-sensitivity results.

spectra before running the deconvolution. The XCLASS deconvolution shown here used a chi-square minimization, rather than the maximum-entropy method described in Comito and Schilke [2002]. It is important to emphasize that these represent preliminary deconvolution attempts and that neither deconvolution algorithm was necessarily achieving its optimal performance. The MATLAB implementation discussed in this thesis is sufficiently new that we are still exploring the parameter space and learning how to improve its performance. Likewise, more experience with XCLASS would likely allow us to improve the results it produces.

Comparisons between the XCLASS deconvolution and the MATLAB deconvolution are shown in in Figures 6.10 and 6.11. The results demonstrate both the advantage of having multiple deconvolution methods available, as well as the challenges that arise as we push line surveys to higher sensitivities. It is quite reassuring that both deconvolution methods achieve very similar results for the major spectral features, giving us confidence in the results. Interestingly, some of the defects that might have been attributed to a glitch in the convolution algorithm appear in both results, indicating that they are probably real and related to some feature of the data.

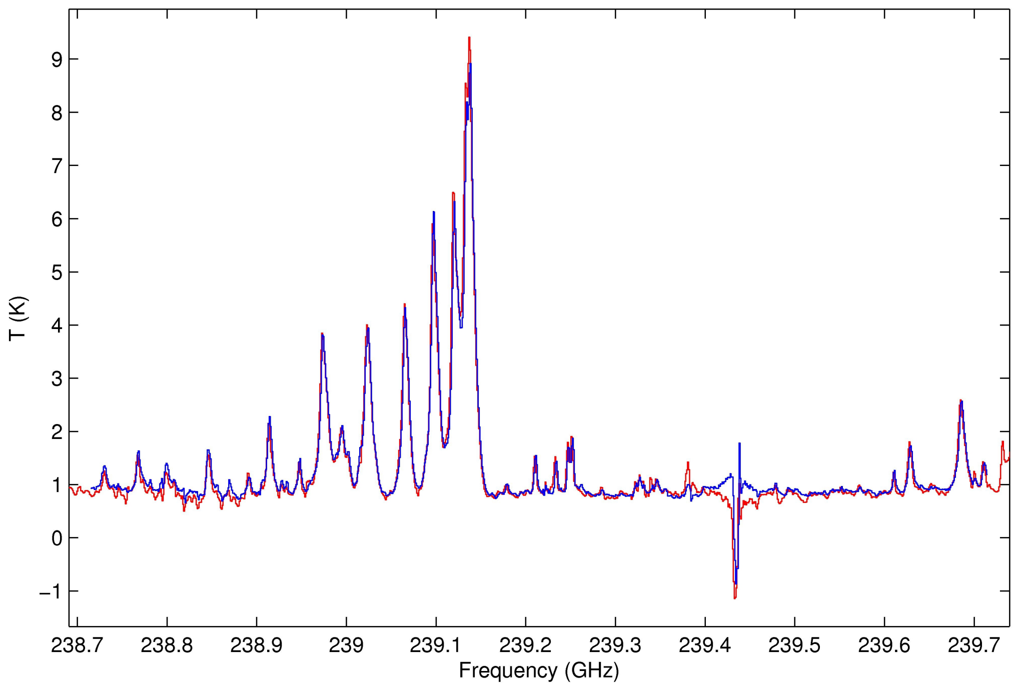
The top plot in Figure 6.10 shows a portion of the single-sideband spectrum in which the two methods agree quite well. Overall line shapes and peak intensities are nearly identical. It is also interesting to note that both algorithms reproduce the negative artifact near 239.45 GHz. This type of close agreement is typical across the SSB spectrum. The bottom panel shows a case in which the two algorithms produce noticeably different results. Of particular interest is the qualitative difference near 233.7 GHz; one algorithm identifies a moderate-sized peak while the other finds nothing. The degree of difference observed across this entire example is unusually large; however, it is not uncommon to identify points in the spectrum where one algorithm found a peak that is significantly larger than the peak found by the second algorithm. The significance of these differences cannot be determined until both SSB spectra are examined for "ghost" lines from the deconvolution.

At the same time, the small-scale peaks demonstrate the effort that remains. While the differences between the two deconvolved spectra seem negligible on the scale of the plot, some are significant at our level of sensitivity. Before we can confidently declare that we have produced deconvolved spectra with $\sim$ 15-mK sensitivity, we will need to reconcile, or at least understand, why the two algorithms produce different results. Rather than insisting that all channels of the two deconvolved spectra match to within their statistical errors, it might be more productive to focus on whether the derived peak parameters (e.g., amplitude and width) are consistent, since those are the values that drive the scientific analysis.

This comparison should be repeated after properly cleaning the Orion data and identifying likely ghosts. We will need to try different configurations of both algorithms to see how we can optimize the recovery of small spectral features, and it would be prudent for us to seek feedback from the authors of XCLASS to guide this work. Allowing both algorithms to work on the same set of simulated data would also be quite interesting.

6.3 Spectral Analysis

Conceptually, characterizing the spectral lines in a single-sideband spectrum is separate from the process of deconvolution. Practically, however, there is a feedback loop in which the spectral interpretation informs the deconvolution. Tuning the deconvolution algorithm requires subjective decisions, and the spectral analysis provides an opportunity to determine whether the resulting spectrum is self consistent. Developing software to perform automated line analysis is key in this process, as it allows us to understand whether the small features are actual spectral lines, residual noise, or artifacts of the deconvolution.



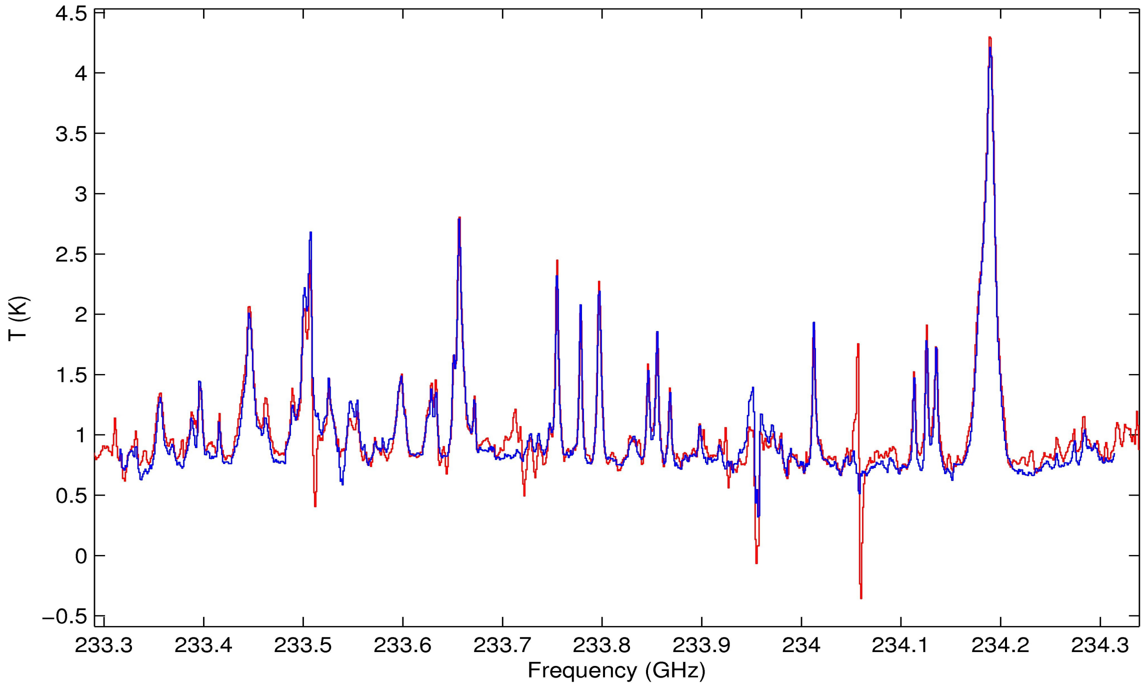


Figure 6.10: Comparison between deconvolved results using XCLASS (red) and the MATLAB-based deconvolution algorithm (blue). The XCLASS spectrum has been vertically offset to align baselines. The extra channels of red spectrum at the edges of the spectrum are caused by the plotting method and do not represent real differences between spectra.

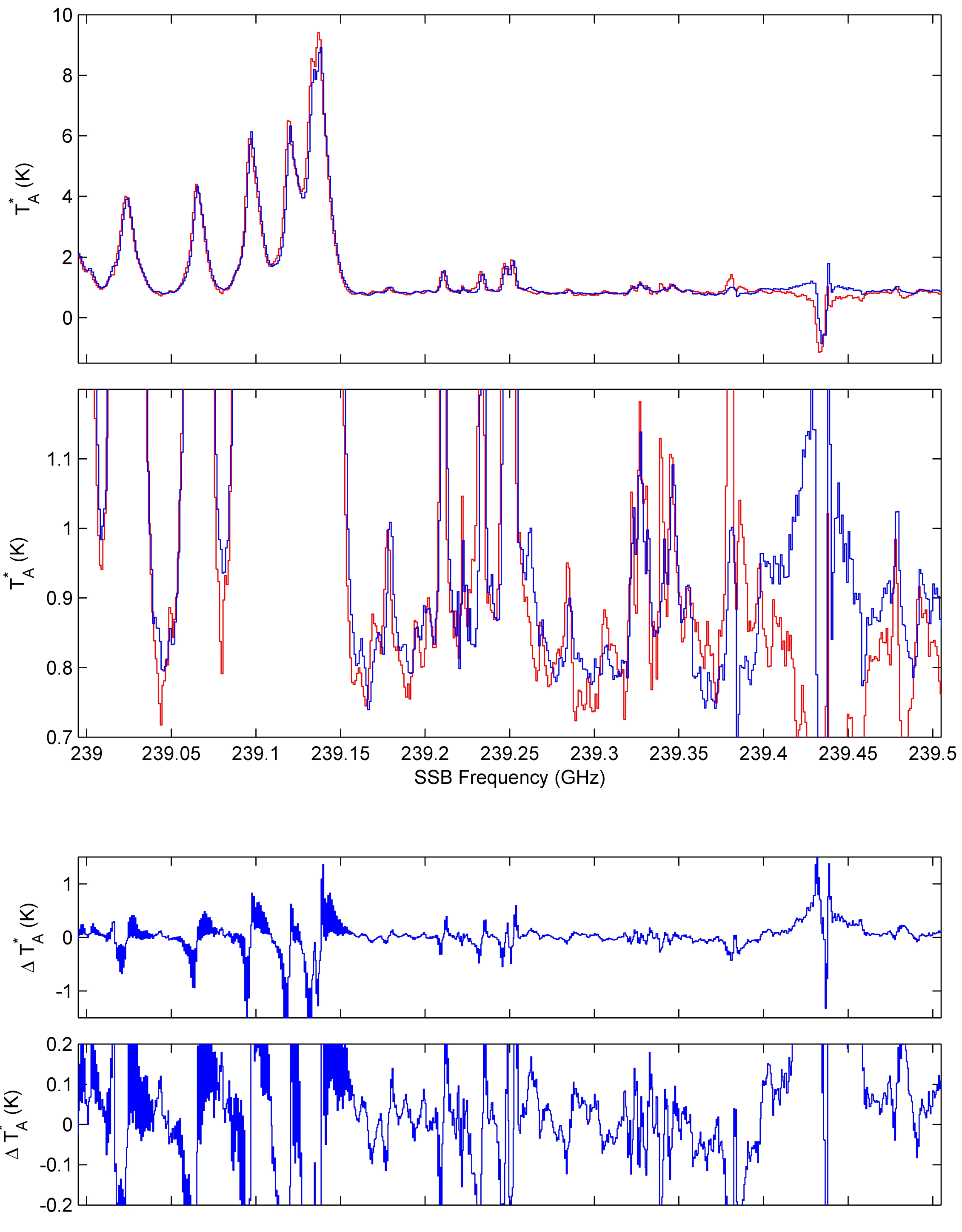


Figure 6.11: Comparison between deconvolved results using XCLASS (red) and the MATLAB-based deconvolution method (blue). In the second and fourth plots, the vertical scale has been expanded to show how the two algorithms differ for the peaks that are likely to represent lines relevant to current science. Compare to Figure 6.9. (The XCLASS spectrum has been vertically offset to align baselines.)

6.3.1 Line-Analysis Software

As discussed in Section 1.7, the increased data rate of the new receivers requires similar advances in software to deconvolve and analyze the spectra. This must occur in a largely automated pipeline; otherwise, manually analyzing the spectra becomes a bottleneck. The deconvolution software we have developed requires little user intervention; the most time-consuming task is examining the raw spectra, and that could easily be accomplished in real time while observing.

Traditionally, spectral analysis for large line surveys has involved manually fitting each peak individually, or at best, trying to fit all of the lines from a single molecule simultaneously. Ideally, this process should be automated, and Susanna Widicus Weaver's group is currently developing such software. It accepts a set of initial estimates for a molecule's velocity, linewidth, and temperature; using a least-squares analysis, it then optimizes these parameters based on the observed spectrum. The software first analyzes a single molecule, then iteratively adds additional molecules to the mix, reoptimizing all of the molecules' parameters each time. This provides more robust way of dealing with strongly blended lines and provides hope for identifying lines that are nearly lost in the confusion. User intervention is required to develop the initial values of the parameters, which are estimated by performing a "by-eye" optimization of the parameter values. The software has not yet been completed, but it is under active development.

6.3.2 Baselines and Spectral Confusion

Modeling all of the lines in a spectrum simultaneously is critical to our understanding of how to fit a baseline beneath a spectral peak. Typically, the peaks are assumed to be built upon continuum emission that creates an underlying offset (as can be seen in the Orion spectrum in Figure 6.7). Presumably, the "continuum" actually consists of a combination of truly broadband emission from sources such as dust and a set of hopelessly blended small spectral lines. Treating all of this as continuum emission and using the top of that as the baseline for fitting large peaks is probably acceptable. The relative error in the integrated intensity is likely to be small compared to other errors, such as calibration, and it makes sense to remove the contribution of smaller lines to the larger line's flux. For small peaks, however, this could represent a significant source of systematic error. Fitting the visible peak to the top of the "continuum" might be the equivalent of fitting the tip of the iceberg; the majority of the line's flux could be hidden below the presumed baseline.

By generating simulated spectra that include all of the emission lines from known molecules, we should be able to quantitatively estimate the offset caused by confused spectral lines. Moreover, even if we cannot recover some of the lines buried beneath the confusion limit, we can at least test whether the observed spectrum is consistent with their presence.

This knowledge feeds back through the entire system. If it is important to include the offset due to confused spectral lines in the single-sideband spectrum, this has important implications for observing and deconvolution techniques. In particular, it implies that fitting baselines to individual spectra is not reliable in the presence of line confusion.

6.4 Analysis of L1157 Spectrum

Although spectral-identification software is not yet complete (Section 6.3), we have analyzed the single-sideband spectrum of L1157 by hand. Consistent with the results of Avery and Chiao [1996], we find that the peaks cannot be fit by a simple Gaussian function [e.g., Weisstein, 2010]; however, an "asymmetric Gaussian" of the form

$$T(f) = \begin{cases} T_o e^{-\left(\frac{f-f_o}{\sqrt{2}\,\sigma_L}\right)^2} & \text{for } f < f_o \\ T_o e^{-\left(\frac{f-f_o}{\sqrt{2}\,\sigma_R}\right)^2} & \text{for } f \geq f_o \end{cases} \tag{6.1}$$

gives good fits for all peaks except CO, which often has a complex peak shape. The asymmetric Gaussian function is normalized to a maximum amplitude of 1 so that T_o gives the peak intensity of the line. The peak-fitting results are shown in Figures D.1 through D.9 of Appendix D and summarized in Table 6.1. All peaks were fit using the asymmetric Gaussian function, and when necessary, multiple peaks were fit simultaneously.

As indicated by Equation 6.1, f_o is the frequency at which the peak reaches its maximum amplitude, T_o is the maximum amplitude of the peak (on the T_A^* scale), and σ_L and

σ_R represent the low-frequency ("left") and high-frequency ("right") widths of the Gaussian, respectively. The column labeled "Baseline RMS" gives the root-mean-square of the deviations for channels outside the peak-fitting region while "Peak RMS" lists the similar quantity for the channels included in the peak fit. Each letter in the "Blend" column indicates a set of peaks that were fit simultaneously due to line blending. Spectral lines were matched to likely molecular species by comparing the center frequencies of the peak fits to the line identifications in other works, particularly the spectral-line tables of Lovas [2004]. Peak frequencies were also compared to previous surveys of point B1 in L1157 [Avery and Chiao, 1996, Bachiller and Gutierrez, 1997], the surveys of IRAS 16293-2422, another Class-0 source, by Blake et al. [1994] and van Dishoeck et al. [1995], and the surveys of Orion KL [Blake et al., 1986, Sutton et al., 1985], a high-mass hot core. Dashes (–) in the line-identification column indicate that no likely match was found. These assignments should be considered tentative until futher analysis can be performed using the spectral-analysis software described above.

The columns labeled "B&G," "A&C," "16293," and "Orion" show an "X" if the line is detected in the Bachiller and Gutierrez [1997] survey of point B1 in L1157, the Avery and Chiao [1996] survey of that same location in L1157, the Blake et al. [1994] and van Dishoeck et al. [1995] survey of IRAS 16293-2422, and the Sutton et al. [1985] and Blake et al. [1986] surveys of Orion KL, respectively. Because the Orion KL study includes an unbiased survey similar to the one presented here, additional comparisons could be made between the results for L1157 and those of Orion KL. An "O" in the "Orion" column indicates the frequency was included in the Orion survey, but no line was observed while dashes indicate frequencies that were not covered in that survey. Table 6.2 summarizes the number of lines observed for each molecular species, based on the tentative line identifications in Table 6.1.

Table 6.1: Peak-fitting results for the deconvolved (SSB) spectrum of L1157. (See text for full description.)

f_o (GHz)	T_o (K)	σ_L (MHz)	σ_R (MHz)	Baseline RMS (mK)	Peak RMS (mK)	Blend	Tentative ID	B&G	A&C	16293	Orion
217.1068	0.185	1.56	3.58	40.3	22.6		SiO	X	X	X	X
218.2231	0.792	0.93	2.46	20.6	12.0		H2CO			X	X
218.4417	0.210	1.21	2.15	18.5	11.9	a	CH3OH			X	X
218.4770	0.218	1.15	2.13	–	–	a	H2CO			X	X
218.7616	0.244	1.39	1.95	18.8	13.5		H2CO				X
219.5595	0.078	0.01	2.54	14.4	18.8		C18O	X		X	X
219.9503	0.368	0.80	3.01	15.6	13.1		SO	X	X	X	X
220.3989	0.596	0.65	1.38	13.7	45.4		13CO	X			X
221.9666	0.048	1.49	4.18	16.3	9.3		SO2	X		X	X
225.6989	1.043	1.15	2.61	15.1	13.1		H2CO	X		X	X
226.1749	0.042	2.04	2.87	11.4	9.9	b	–				O
226.2134	0.043	1.22	1.03	–	–	b	–				O
226.6337	0.024	11.88	7.97	14.0	11.9	c	CN				X
226.6611	0.111	1.20	3.91	–	–	c	CN	X			X
226.6823	0.046	5.97	1.48	–	–	c	CN				X
226.8759	0.276	1.25	2.85	15.1	11.4		CN	X		X	X
229.7603	0.365	1.23	2.35	13.6	11.8		CH3OH		X	X	X
230.5390	10.514	0.71	2.82	46.0	705.8		CO	X	X	X	X
239.7480	0.197	1.30	2.68	13.4	12.1		CH3OH		X	X	X
241.0190	0.068	2.06	1.69	12.1	6.1		C34S	X		X	X
241.6155	0.082	0.02	2.42	13.6	40.5		SO2		X	X	X
241.7019	0.525	1.30	2.42	11.5	9.0		CH3OH		X	X	X
241.7687	1.794	1.28	2.74	14.3	19.1	d	CH3OH		X	X	X
241.7928	2.227	1.26	2.71	–	–	d	CH3OH	X	X	X	X
241.8810	0.183	1.38	2.54	13.2	9.2	e	CH3OH		X	X	X
241.9063	0.303	1.43	2.58	–	–	e	CH3OH		X	X	X

(table continued on next page)

Table 6.1 – Continued

f_o (GHz)	T_o (K)	σ_L (MHz)	σ_R (MHz)	Baseline RMS (mK)	Peak RMS (mK)	Blend	Tentative ID	B&G	A&C	16293	Orion
243.9177	0.336	1.25	2.44	13.2	14.2		CH3OH			X	X
244.9377	1.038	1.62	2.72	14.2	29.2		CS	X		X	X
250.5087	0.094	1.64	2.30	16.0	12.2		CH3OH				X
251.8279	0.185	1.71	2.95	17.6	15.9		SO		X		–
254.0170	0.269	1.33	2.85	15.8	13.2		CH3OH		X		–
258.2582	0.158	1.81	2.74	17.0	15.8		SO			X	X
259.0146	0.079	1.92	2.30	19.2	8.8		H13CN			X	X
260.5208	0.378	1.94	3.18	16.5	10.4	f	SiO		X	X	X
260.5289	0.109	1.94	4.23	–	–	f	SiO wing				
261.8073	0.206	1.46	2.48	18.0	14.0	g	CH3OH		X	X	X
261.8453	0.399	1.25	3.26	–	–	g	SO		X	X	X
261.9144	0.032	1.39	3.32	–	–	h	–				O
262.0075	0.130	2.12	3.27	19.8	12.2	h	CCH				X
262.0693	0.100	2.21	1.12	–	–	h	CCH				X
265.8887	1.440	2.37	4.31	17.8	20.5	i	HCN				–
265.8992	0.078	5.59	4.35	–	–	i	–				–
266.8393	0.414	0.89	3.12	21.8	16.1		CH3OH				–
267.5585	0.689	1.39	2.22	22.4	18.6		HCO+				–
271.9819	0.186	1.24	2.28	22.2	14.1		HNC				–
274.0590	-0.062	0.79	2.50	22.9	11.9	j	–				–
274.0779	-0.102	4.57	2.25	–	–	j	–				–
274.2269	-0.176	4.37	1.95	23.4	21.0		–				–

Table 6.3 shows the differences between line centers found in this work and those from other works, based on the tentative identifications shown in 6.1. The value of f_o represents the center frequency used in each study while Δf represents the difference between the center frequency used in that study with respect to the center frequency found in this work. For the Lovas [2004] catalog values, the frequency error is also used to calculate a corresponding line-of-sight velocity for the region of the source emitting the line (Δv).

Species	Number of Lines
CH3OH	13
CN	4
H2CO	4
SO	4
SiO and SiO wing	3
CCH	2
SO2	2
CO	1
13CO	1
C18O	1
CS	1
C34S	1
HCN	1
H13CN	1
HCO+	1
HNC	1
–	7
TOTAL	48

Table 6.2: Number of lines found for each chemical species, based on tentative identifications from Table 6.1.

The columns labelled "Lovas" correspond to the Lovas [2004] catalog while the other caption headings use the abbreviations defined for Table 6.1. As before, dashes (–) in the line-identification column indicate that no likely match was found. Parentheses around a value of f_o in the "Lovas" column indicate that the center frequency shown is the closest frequency match that could be identified in Lovas [2004]. These frequencies differed significantly from the results of the line fits, producing a large value of Δf. No line assignments were made for such cases.

Table 6.1 shows that several lines observed in this work were not reported in either of the previous surveys of point B1 in L1157, and a few lines were not observed in either of those papers or in the survey of 16293. However, we have reasonable confidence that the lines with tentative identifications represent real spectral lines, and not artifacts from the receiver or the data analysis, as all such lines were also observed in the Orion survey. This fact suggests it is reasonable to expect that the corresponding species could be created in a prestellar core and that the observed lines could be excited in such an environment.

However, it should be noted that the physical environment, and hence the chemistry, of large, hot cores, such as Orion KL, are usually believed to be qualitatively different than those of smaller cores, such as L1157, so caution must be taken when applying the line identifications from Orion KL to the lines observed in L1157.

Table 6.3: Differences between line centers found in this work and those from other works, based on the tentative identifications shown in Table 6.1. (See text for full description.)

Tentative ID	This work	Lovas			B&G		A&C		16293		Orion	
	f_o (GHz)	f_o (GHz)	Δf (MHz)	Δv (km/s)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)
SiO	217.1068	217.1050	-1.8	-2.5	217.1049	-1.902	217.104	-2.802	217.1049	-1.902	217.105	-1.802
H2CO	218.2231	218.2222	-0.9	-1.2					218.2222	-0.861	218.222	-1.081
CH3OH	218.4417	218.4401	-1.6	-2.2					218.4400	-1.672	218.440	-1.672
H2CO	218.4770	218.4756	-1.4	-1.9					218.4756	-1.35	218.476	-0.95
H2CO	218.7616	218.7601	-1.5	-2.0							218.760	-1.582
C18O	219.5595	219.5604	0.9	1.2	219.5603	0.791			219.5603	0.791	219.560	0.491
SO	219.9503	219.9494	-0.9	-1.2	219.9494	-0.856	219.951	0.744	219.9494	-0.856	219.949	-1.256
13CO	220.3989	220.3987	-0.2	-0.3							220.399	0.056
SO2	221.9666	221.9652	-1.4	-1.9	221.9652	-1.439			221.9652	-1.439	221.965	-1.639
H2CO	225.6989	225.6978	-1.1	-1.4	225.6978	-1.061			225.6978	6.09	225.698	-0.861
–	226.1749											
–	226.2134	(226.217)	3.6	4.8								
CN	226.6337	226.6322	-1.5	-2.0							226.634	0.297
CN	226.6611	226.6595	-1.6	-2.1	226.6595	-1.615					226.660	-1.115

(table continued on next page)

Tentative ID	This work	Lovas			B&G		A&C		16293		Orion	
	f_o (GHz)	f_o (GHz)	Δf (MHz)	Δv (km/s)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)
CN	226.6823	226.6793	-2.9	-3.9							226.679	-3.284
CN	226.8759	226.8748	-1.1	-1.5	226.8748	-1.104			226.8742	-1.704	226.875	-0.904
CH3OH	229.7603	229.7588	-1.5	-1.9			229.763	2.73	229.7587	-1.57	229.759	-1.27
CO	230.5390	230.5380	-1.0	-1.3	230.5380	-0.984			230.5380	-0.984	230.538	-0.984
CH3OH	239.7480	239.7462	-1.8	-2.2			239.746	-2.01	239.7463	-1.71	239.746	-2.01
C34S	241.0190	241.0161	-2.9	-3.6					241.0162	-2.839	241.016	-3.039
SO2	241.6155	241.6158	0.2	0.3			241.612	-3.526	241.6158	0.274	241.616	0.474
CH3OH	241.7019	241.7002	-1.8	-2.2			241.700	-1.935	241.7002	-1.735	241.700	-1.935
CH3OH	241.7687	241.7672	-1.4	-1.8			241.768	-0.663	241.7672	-1.463	241.767	-1.663
CH3OH	241.7928	241.7914	-1.4	-1.8	241.7914	-1.377	241.791	-1.777	241.7914	-1.377	241.791	-1.777
CH3OH	241.8810	241.8790	-2.0	-2.5			241.879	-2.039	241.8791	-1.939	241.879	-2.039
CH3OH	241.9063	241.9046	-1.7	-2.1			241.905	-1.32	241.9040	-2.32	241.904	-2.32
CH3OH	243.9177	243.9158	-1.9	-2.3					243.9158	-1.875	243.916	-1.675
CS	244.9377	244.9356	-2.1	-2.6	244.9356	-2.1			244.9356	-2.1	244.936	-1.7
CH3OH	250.5087	250.5070	-1.7	-2.1							250.507	-1.723
SO	251.8279	251.8258	-2.1	-2.5			251.830	2.128				
CH3OH	254.0170	254.0154	-1.6	-1.9			254.017	0.035				
SO	258.2582	258.2559	-2.3	-2.7					258.2558	-2.402	258.256	-2.202

(table continued on next page)

	This work	Lovas			B&G		A&C		16293		Orion	
Tentative ID	f_o (GHz)	f_o (GHz)	Δf (MHz)	Δv (km/s)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)	f_o (GHz)	Δf (MHz)
H13CN	259.0146	259.0118	-2.8	-3.2					259.0110	-3.562	259.011	-3.562
SiO	260.5208	260.5181	-2.7	-3.1			260.518	-2.796	260.5180	-2.796	260.518	-2.796
SiO wing	260.5289	(260.53567)	6.7	7.7								
CH3OH	261.8073	261.8057	-1.5	-1.7			261.809	1.749	261.8057	-1.551	261.806	-1.251
SO	261.8453	261.8438	-1.5	-1.7			261.840	-5.274	261.8437	-1.574	261.844	-1.274
–	261.9144	(261.899788)	-14.6	-16.7								
CCH	262.0075	262.0043	-3.2	-3.7					262.0067	-0.759	262.005	-2.459
CCH	262.0693	262.0675	-1.8	-2.1					262.0676	-1.69	262.067	-2.29
HCN	265.8887	265.8864	-2.3	-2.6								
–	265.8992											
CH3OH	266.8393	266.8381	-1.2	-1.3								
HCO+	267.5585	267.5576	-0.9	-1.0								
HNC	271.9819	271.9811	-0.8	-0.9								
–	274.0590											
–	274.0779											
–	274.2269											

Chapter 7

Summary and Conclusions

Our understanding of the early stages of star formation has advanced greatly over the past few decades, largely through improvements in submillimeter astronomy combined with new theoretical models. Submillimeter observations provide an exciting window into regions of the universe inaccessible at other wavelengths and are particularly well suited to peering through the cold, dusty gas of molecular clouds. As with other wavelength regimes, submillimeter astronomy relies on two types of complementary observing, imaging and spectroscopy. The resolving power of a single-dish telescope like the CSO is not sufficient for imaging a protostar, but it offers excellent capabilities for spectroscopic studies. Spectral lines at these frequencies can be used to study source dynamics and determine physical conditions, in addition to identifying the molecular source.

An unbiased line survey, covering a broad range of frequencies, provides a particularly powerful way of applying spectroscopy to a given source. While each of the individual lines is informative, the line survey can tell a story greater than the sum of its parts. Understanding that story requires an interdisciplinary approach, combining physics, astronomy, and chemistry. The results are impressive and include information needed to understand the physics, chemistry, and geometry of the source.

Traditionally, line surveys have required a significant investment of observing time that often spans several observing seasons, followed by an equally intense data-analysis effort. Consequently, few sources have been fully surveyed, and much of our understanding of star formation has been built upon a few prototypical sources that are particularly amenable to observing. This thesis describes our efforts to enable rapid, sensitive line surveys by taking advantage of recent advances in submillimeter heterodyne receivers and by developing new software tools to minimize the amount of manual effort required during the data analysis.

7.1 Receivers

The graduate work leading to this thesis includes the installation of two instruments at the CSO, progenitors of a new generation of high-bandwidth receivers. In both cases, the SIS mixer at the heart of the receiver has been an unqualified success. New modeling and fabrication tools have permitted the design of SIS mixers with unprecedented performance goals, and the resulting devices have not disappointed. The new receivers offer vastly greater instantaneous bandwidths, high sensitivity, and easy, stable tuning. While the mixers themselves have been a joy to use, there has been a learning curve associated with moving to a broader IF bandwidth, going up as high as 20 GHz from the previous 2 GHz. The extra bandwidth requires additional caution in designing IF systems and exposes sources of interference that never caused problems before.

Synthesizer-driven LO chains seem to offer substantial promise, but the results have been mixed. The improved frequency agility simplifies the task of observing and substantially reduces the overhead associated with tuning the receiver to a new frequency. However, it is difficult to generate a sufficiently clean signal from the LO chain. That situation has been largely remedied through the use of a YIG filter to clean the synthesizer signal, but the added components increase the complexity of the device, with a corresponding decrease in the reliability of the LO system. Further, this approach does not remove all of the spurious signals from the output of the LO chain, leading us to choose the traditional Gunn-based LO for our observing efforts. Nonetheless, work with the prototype demonstrates that the synthesized LO can offer similar performance to the Gunn under the right conditions and provides several insights that have gone into designing the next series of facility receivers.

7.2 Data Analysis

The new receivers fulfilled their promise to substantially increase the speed at which line surveys could be completed, thereby revealing the need for equally fast data analysis. To this end, we have created new sideband-deconvolution software and are currently developing spectral-analysis software. The data pipeline has been designed with an eye toward minimizing the amount of manual intervention required from the user, although some features, such as integrated baseline fitting, need further work.

A thorough analytic analysis of the sideband-deconvolution problem has been completed, including the derivation of Jacobian matrices needed for numerical algorithms. The deconvolution technique is built upon the minimization off a χ^2-like error function. Several methods of finding the optimal solution have been considered, including direct optimization of the model parameters, nonlinear root finding, and a hybrid pseudo-linear approach. The third concept is the most intriguing, as it isolates the nonlinearities inherent in the problem, leaving the majority of parameters accessible to linear solution. However, it also looks to be the most complicated to implement. In order to confirm the results of the analytic derivations and to facilitate future expansions to technique, a utility has been created to allow the calculation of arbitrary gradients within *Mathematica*.

7.3 Line Surveys

To demonstrate the capabilities of the new wideband systems, several unbiased line surveys were completed, and more are being undertaken by our collaborators. The first of these surveys focused on a position in the outflow of L1157, a low-mass protostar with geometry that makes it ideal for studying shock chemistry in the outflow. This particular position has been heavily studied by other researchers, both through targeted line searches and interferometric observations, providing a significant foundation of background information. The DSB spectra resulting from these observations had excellent baselines, and the deconvolution software performed well. The entire single-sideband spectrum, covering more than 50 GHz of spectral range, is included in this thesis.

The Orion KL hot core, arguably one of the most studied submillimeter sources in the galaxy, was first surveyed in the 230-GHz band about 25 years ago. With the new advances in receiver technology, we have repeated that survey, driving the noise levels down by an additional order of magnitude. The new survey should provide sufficient sensitivity to identify complex organic molecules in the spectrum, if they are present in the source. These molecules are important for understanding the chemistry that occurs around hot cores, and some have implications for origin-of-life theories. A partial deconvolution of the Orion data is presented here to demonstrate that the data-analysis routines can preserve the quality of such high-sensitivity observations.

7.4 Prospects for Future Work

The L1157 and Orion KL surveys show the potential of the new systems. The broadband receivers greatly simplify and speed the observing process, and the data-analysis software has demonstrated that it can rapidly deconvolve large data sets while preserving the quality of high-sensitivity observations. These initial successes are rewarding, but there remains much work to be done.

A suite of new facility receivers has been designed for the CSO and is currently under construction. This represents a substantial undertaking that has not only necessitated new mixer and receiver designs, but has also required new bias-control electronics, new IF systems, and new methods of computer control. In addition to offering broad instantaneous bandwidths at a variety of observing frequencies, these receivers are also configured in a balanced configuration, making them ideal for use with synthesizer-driven LO chains. However, while much has been learned about active LO chains from their use on Z-Rex, open questions remain. As these receivers are tested and installed on the telescope, it will be particularly important to look for signs of spurious signals or excess noise in the resulting spectra. As more heterodyne instruments move toward a 4 - 8 GHz IF output, it will be necessary to identify and eliminate sources of RF interference that fall in the new IF band, ranging from the microwave in the CSO galley to the lock signal of the LO's phase-locked loop. Ensuring that these receivers present the correct power levels at their interface to the IF electronics will also be critical.

The analytic derivations underlying the deconvolution software appear to be quite reliable, having converged on the same solutions through calculations by hand and in *Mathematica*, and the deconvolution software has proven the value of this methodology. However, the MATLAB implementation of the deconvolution algorithm does have some unexplained behavior, particularly when fitting data with baselines. New display and debugging tools are needed to determine whether there are errors in the current code and to support the creation of additional solution modes, such as non-linear root finding or the hybrid pseudo-linear approach. In the near term, medium-sized simulations are needed to validate the deconvolution techniques for $\lambda_B \neq 0$. On longer timescales, reformulating χ^2_{mod} to use memory more efficiently should allow larger simulations and line surveys to be deconvolved with $\lambda_B \neq 0$, if desired.

Finally, significant work remains on the spectral-analysis tools being developed by collaborators at Emory University. This software will greatly expanded our ability to model molecular-line spectra, particularly in line-confused spectra. This analysis is critical for reaching sensitivities needed to further constrain astrochemical models and to search for prebiotic molecules in star-forming regions.

Once completed, the hardware and software systems described here should make it possible for submillimeter line surveys to become routine observational tools. As demonstrated with L1157, sources can be surveyed to moderate sensitivity in only a few nights of observing time. Even sensitive studies, such as the Orion hot-core survey, can be completed in a single run, improving the quality of the data calibration and minimizing the likelihood of ending up with a half-completed survey.

Appendix A

Calculating Gradients and Jacobians

A.1 General Derivations

The results in Chapter 4 rely heavily on taking derivatives of equations containing matrices and vectors. The basic approach for such calculations is to expand a matrix operation into the underlying components, which can be treated as ordinary numbers. To avoid recalculating similar quantities for each derivation, we start by considering some general cases in which $\boldsymbol{x}$ and $\boldsymbol{y}$ are arbitrary vectors and $\boldsymbol{A}$ is an arbitrary matrix. Throughout this appendix, sums are shown explicitly; there are no implied summations. The results of this section are summarized in Table A.1 for easy reference.

We begin with the simplest calculation:

$$\begin{aligned}\frac{\partial x_i}{\partial x_j} &= \delta_{ij} \\ &= \mathbb{1}_{ij}.\end{aligned} \tag{A.1}$$

We can then consider cases in which $\boldsymbol{x}$ is multiplied by a matrix from the left,

$$\begin{aligned}\frac{\partial}{\partial x_j}\left[\boldsymbol{A}\cdot\boldsymbol{x}\right]_i &= \frac{\partial}{\partial x_j}\sum_k A_{ik}x_k \\ &= \sum_k A_{ik}\frac{\partial x_k}{\partial x_j} \\ &= \sum_k A_{ik}\delta_{kj} \\ &= A_{ij},\end{aligned} \tag{A.2}$$

and from the right:

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left[\boldsymbol{x}^T \cdot \boldsymbol{A}\right]_i &= \frac{\partial}{\partial x_j}\sum_k x_k A_{ki} \\
&= \sum_k \frac{\partial x_k}{\partial x_j} A_{ki} \\
&= \sum_k \delta_{kj} A_{ki} \\
&= A_{ji}. \qquad \text{(A.3)}
\end{aligned}$$

Now we add another vector into the mix:

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left(\boldsymbol{y}^T \cdot \boldsymbol{A} \cdot \boldsymbol{x}\right) &= \frac{\partial}{\partial x_j}\sum_{k,m} y_k A_{km} x_m \\
&= \sum_{k,m} y_k A_{km} \frac{\partial x_m}{\partial x_j} \\
&= \sum_{k,m} y_k A_{km} \delta_{mj} \\
&= \sum_k y_k A_{kj} \\
&= \left[\boldsymbol{y}^T \cdot \boldsymbol{A}\right]_j. \qquad \text{(A.4)}
\end{aligned}$$

This result could be expressed equivalently as $\left[\boldsymbol{A}^T \cdot \boldsymbol{y}\right]_j$, since the component representations of both forms are identical. If $\boldsymbol{x}$ multiplies from the left, we find the following result:

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T \cdot \boldsymbol{A} \cdot \boldsymbol{y}\right) &= \frac{\partial}{\partial x_j}\sum_{k,m} x_k A_{km} y_m \\
&= \sum_{k,m} \frac{\partial x_k}{\partial x_j} A_{km} y_m \\
&= \sum_{k,m} \delta_{kj} A_{km} y_m \\
&= \sum_m A_{jm} y_m \\
&= \left[\boldsymbol{A} \cdot \boldsymbol{y}\right]_j. \qquad \text{(A.5)}
\end{aligned}$$

As a special case, we can consider

$$\frac{\partial}{\partial x_j}\left(\boldsymbol{y}^T\cdot\boldsymbol{x}\right)=y_j, \tag{A.6}$$

which can trivially be derived from Equation A.4 by setting $\boldsymbol{A}\rightarrow\mathbb{1}$.[1] Similarly, we find

$$\frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T\cdot\boldsymbol{y}\right)=y_j \tag{A.7}$$

from Equation A.5.

Next, we calculate $\frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T\cdot\boldsymbol{A}\cdot\boldsymbol{x}\right)$:

$$\begin{aligned}\frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T\cdot\boldsymbol{A}\cdot\boldsymbol{x}\right)&=\frac{\partial}{\partial x_j}\sum_{k,m}x_kA_{km}x_m\\&=\sum_{k,m}\frac{\partial x_k}{\partial x_j}A_{km}x_m+\sum_{k,m}x_kA_{km}\frac{\partial x_m}{\partial x_j}\\&=\sum_{k,m}\delta_{kj}A_{km}x_m+\sum_{k,m}x_kA_{km}\delta_{mj}\\&=\sum_{m}A_{jm}x_m+\sum_{k}x_kA_{kj}\\&=\left[\boldsymbol{A}\cdot\boldsymbol{x}\right]_j+\left[\boldsymbol{x}^T\cdot\boldsymbol{A}\right]_j.\end{aligned} \tag{A.8}$$

If preferred, this result can also be written as $\left[\left(\boldsymbol{A}+\boldsymbol{A}^T\right)\cdot\boldsymbol{x}\right]_j$. This result can be used to calculate two other general cases that appear frequently when taking derivatives of χ^2_{mod}. By setting $\boldsymbol{A}\rightarrow\mathbb{1}$, we find

$$\frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T\cdot\boldsymbol{x}\right)=2x_j. \tag{A.9}$$

Another, case that appears commonly is $\frac{\partial}{\partial x_j}\left(\boldsymbol{z}^T\cdot\boldsymbol{z}\right)$, in which $\boldsymbol{z}=\boldsymbol{A}\cdot\boldsymbol{x}$. As shown below, this can be addressed easily using Equation A.8 with $\boldsymbol{A}\rightarrow\boldsymbol{A}^T\cdot\boldsymbol{A}$:

[1]It would be equally valid to consider Equation A.4 as a special case of Equation A.6 since $\boldsymbol{y}^T\cdot\boldsymbol{A}$ simply represents another vector that could be used in the latter equation to generate the former.

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left(\boldsymbol{z}^T \cdot \boldsymbol{z}\right) &= \frac{\partial}{\partial x_j}\left(\boldsymbol{x}^T \cdot \boldsymbol{A}^T \cdot \boldsymbol{A} \cdot \boldsymbol{x}\right) \\
&= \left[\left(\boldsymbol{A}^T \cdot \boldsymbol{A} + \left(\boldsymbol{A}^T \cdot \boldsymbol{A}\right)^T\right) \cdot \boldsymbol{x}\right]_j \quad \text{(by Eq. A.8)} \\
&= 2\left[\boldsymbol{A}^T \cdot \boldsymbol{A} \cdot \boldsymbol{x}\right]_j \\
&= 2\left[\boldsymbol{A}^T \cdot \boldsymbol{z}\right]_j . \qquad \text{(A.10)}
\end{aligned}$$

The specific implementation of the deconvolution algorithm in MATLAB also requires that we calculate derivatives of the form $\frac{\partial}{\partial x_j}\left(x_i\right)^2$ and $\frac{\partial}{\partial x_j}\left(z_i\right)^2$, where $\boldsymbol{z}$ is again defined as $\boldsymbol{z} = \boldsymbol{A} \cdot \boldsymbol{x}$. It is important to emphasize that these quantities are not implicit sums; rather, they represent single terms of $\boldsymbol{x} \cdot \boldsymbol{x}$ and $\boldsymbol{z} \cdot \boldsymbol{z}$, respectively. The first derivative can be calculated quite simply:

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left(x_i\right)^2 &= 2x_i \frac{\partial x_i}{\partial x_j} \\
&= 2x_i \delta_{ij}. \qquad \text{(A.11)}
\end{aligned}$$

Despite the repeated index, there is no summation over i, so this result represents a diagonal matrix with the elements of $2x_i$ along the diagonal. Similarly, we find

$$\begin{aligned}
\frac{\partial}{\partial x_j}\left(z_i\right)^2 &= 2z_i \frac{\partial}{\partial x_j}\left(z_i\right) \\
&= 2z_i \frac{\partial}{\partial x_j}\left(\sum_k A_{ik} x_k\right) \\
&= 2z_i \left(\sum_k A_{ik} \delta_{jk}\right) \\
&= 2z_i A_{ij}. \qquad \text{(A.12)}
\end{aligned}$$

Again, there is no summation over i, so this result represents the matrix $\boldsymbol{A}$, in which the i^{th} row has been multiplied by $2z_i$.

Derivative	Result	Equation
$\frac{\partial x_i}{\partial x_j}$	δ_{ij} or $\mathbb{1}_{ij}$	A.1
$\frac{\partial}{\partial x_j} [\boldsymbol{A} \cdot \boldsymbol{x}]_i$	A_{ij}	A.2
$\frac{\partial}{\partial x_j} \left[\boldsymbol{x}^T \cdot \boldsymbol{A}\right]_i$	A_{ji}	A.3
$\frac{\partial}{\partial x_j} \left(\boldsymbol{y}^T \cdot \boldsymbol{A} \cdot \boldsymbol{x}\right)$	$\left[\boldsymbol{y}^T \cdot \boldsymbol{A}\right]_j$ or $\left[\boldsymbol{A}^T \cdot \boldsymbol{y}\right]_j$	A.4
$\frac{\partial}{\partial x_j} \left(\boldsymbol{x}^T \cdot \boldsymbol{A} \cdot \boldsymbol{y}\right)$	$[\boldsymbol{A} \cdot \boldsymbol{y}]_j$	A.5
$\frac{\partial}{\partial x_j} \left(\boldsymbol{y}^T \cdot \boldsymbol{x}\right)$	y_j	A.6
$\frac{\partial}{\partial x_j} \left(\boldsymbol{x}^T \cdot \boldsymbol{y}\right)$	y_j	A.7
$\frac{\partial}{\partial x_j} \left(\boldsymbol{x}^T \cdot \boldsymbol{A} \cdot \boldsymbol{x}\right)$	$\left[\boldsymbol{A} \cdot \boldsymbol{x}\right]_j + \left[\boldsymbol{x}^T \cdot \boldsymbol{A}\right]_j$ or $\left[\left(\boldsymbol{A} + \boldsymbol{A}^T\right) \cdot \boldsymbol{x}\right]_j$	A.8
$\frac{\partial}{\partial x_j} \left(\boldsymbol{x}^T \cdot \boldsymbol{x}\right)$	$2x_j$	A.9
$\frac{\partial}{\partial x_j} \left(\boldsymbol{z}^T \cdot \boldsymbol{z}\right)$	$2\left[\boldsymbol{A}^T \cdot \boldsymbol{z}\right]_j$	A.10
$\frac{\partial}{\partial x_j} (x_i)^2$	$2x_i\delta_{ij}$	A.11
$\frac{\partial}{\partial x_j} (z_i)^2$	$2z_iA_{ij}$	A.12

Table A.1: General results for differentiation of matrix and vector quantities. $\boldsymbol{A}$ represents an arbitrary matrix while $\boldsymbol{x}$ and $\boldsymbol{y}$ are arbitrary vectors, all of appropriate dimension. It is assumed that $\boldsymbol{z} = \boldsymbol{A} \cdot \boldsymbol{x}$. Equivalent forms are listed for some results. Note that there is no sum over the index i in Equations A.11 and A.12.

A.2 Common Derivatives

Armed with these general results for arbitrary $\boldsymbol{A}$, $\boldsymbol{x}$, and $\boldsymbol{y}$, we can quickly calculate some important derivatives. In particular, it is useful to differentiate many of the quantities defined in Chapter 4 with respect to their underlying independent parameters. We start by taking derivatives of the modeled DSB data, $\boldsymbol{d}^{\boldsymbol{NR}}$, as expressed in Equation 4.61, which gives

$$\begin{aligned}
\frac{\partial d_i^{NR}}{\partial s_j} &= \frac{\partial}{\partial s_j}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_i \\
&= \frac{\partial}{\partial s_j}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \\
&= M_{ij}^{\Gamma,NR} \quad \text{(by Eq. A.2)} \qquad \text{(A.13)}
\end{aligned}$$

and

$$\begin{aligned}
\frac{\partial d_i^{NR}}{\partial b_j} &= \frac{\partial}{\partial s_j}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \frac{\partial}{\partial s_n} \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_i \\
&= \eta_B \frac{\partial}{\partial b_j}\left[\boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_i \\
&= \eta_B C_{ij}^{NR} \quad \text{(by Eq. A.2).} \qquad \text{(A.14)}
\end{aligned}$$

The derivative with respect to γ^j is discussed in Section A.2.1,where we consider some general properties of such calculations.

Similarly, we calculate derivatives of $\boldsymbol{\beta}^{\boldsymbol{NR}}$ with respect each of the independent parameters. From Equation 4.60, it is evident that $\frac{\partial \beta_i^{NR}}{\partial s_j} = 0$ and $\frac{\partial \beta_i^{NR}}{\partial \gamma^j} = 0$, so we only need to calculate one derivative:

$$\begin{aligned}
\frac{\partial \beta_i^{NR}}{\partial b_j} &= \frac{\partial}{\partial b_j}\left[\boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_i \\
&= C_{ij}^{NR} \quad \text{(by Eq. A.2).} \qquad \text{(A.15)}
\end{aligned}$$

Finally, the following identity is useful when simplifying some of the elements of the Jacobian matrices and relies on the fact that $\mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}} = \delta_{aj} \mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}$ (Equation 4.44):

$$\begin{aligned}
\mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Gamma,NR} &= \mathbb{1}^{j,R} \cdot \left[\boldsymbol{M}^{\Sigma,NR} + \left(\sum_a \gamma^a \mathbb{1}^{a,R}\right) \cdot \boldsymbol{M}^{\Delta,NR}\right] \\
&= \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Sigma,NR} + \left(\sum_a \gamma^a \mathbb{1}^{j,R} \cdot \mathbb{1}^{a,R}\right) \cdot \boldsymbol{M}^{\Delta,NR} \\
&= \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Sigma,NR} + \left(\sum_a \gamma^a \delta_{aj} \mathbb{1}^{j,R}\right) \cdot \boldsymbol{M}^{\Delta,NR} \\
&= \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Sigma,NR} + \gamma^j \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR}. \qquad \text{(A.16)}
\end{aligned}$$

A.2.1 Derivatives with respect to γ

Derivatives with respect to the sideband-gain factors deserve special consideration. Although γ can be considered a vector, it exists in a different space than the other matrices and vectors discussed in previous sections of this appendix. In particular, the index a on γ^a pairs with the "third" index on $\mathbb{1}_{ij}^{a,R}$. To avoid confusion, we have consistently described operations involving γ in terms of explicit sums over its components.[2] Therefore, we can apply the techniques of the previous section to the components of γ without needing to expand the dot products into matrix and vector components first. For instance, consider taking the derivative of $\boldsymbol{\Psi_1} \cdot \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{\Psi_2}$, where $\boldsymbol{\Psi_1}^T$ and $\boldsymbol{\Psi_2}$ represent arbitrary matrices, vectors, or one of each.[3] Since the vector and matrix multiplication commute with multiplication by a scalar, taking the derivative turns out to be quite easy:

$$\begin{aligned}
\frac{\partial}{\partial \gamma^j}\left(\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{\Psi_2}\right) &= \frac{\partial}{\partial \gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \left(\boldsymbol{M}^{\Sigma,NR} + \boldsymbol{\Gamma}^R \cdot \boldsymbol{M}^{\Delta,NR}\right) \cdot \boldsymbol{\Psi_2}\right] \text{ by Equation 4.59} \\
&= \frac{\partial}{\partial \gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \left(\sum_a \gamma^a \mathbb{1}^{a,R}\right) \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2}\right] \text{ by Equation 4.45} \\
&= \frac{\partial}{\partial \gamma^j} \sum_a \gamma^a \boldsymbol{\Psi_1}^T \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2} \\
&= \sum_a \frac{\partial \gamma^a}{\partial \gamma^j} \boldsymbol{\Psi_1}^T \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2} \\
&= \sum_a \delta_{aj} \boldsymbol{\Psi_1}^T \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2} \\
&= \boldsymbol{\Psi_1}^T \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2}. \qquad \text{(A.17)}
\end{aligned}$$

[2]For instance, we use $\sum_a \gamma^a \mathbb{1}^{a,R}$, rather than attempting a calculation akin to a "third-index dot product."

[3]By letting $\boldsymbol{\Psi_1}$ and/or $\boldsymbol{\Psi_2}$ be a multiple of the identity matrix ($\mathbb{1}$), we can also incorporate multiplication by scalars.

We have been able to arrive at this result without specifying the nature of $\boldsymbol{\Psi_1}$; the only requirement is that the multiplication operator allows us to pull out the γ^a from between between $\boldsymbol{\Psi_1}$ and $\mathbb{1}^{a,R}$ (i.e., the multiplication must be distributive over addition and commutative with respect to scalar multiplication). Since multiplication by matrices, multiplication by vectors, and multiplication by scalars meet these constraints, we can treat Equation A.17 as a general result.

Taking the derivative of the transpose is equally easy:

$$\begin{aligned}
\frac{\partial}{\partial\gamma^j}\left(\boldsymbol{\Psi_2}^T \cdot \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{\Psi_1}\right) &= \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_2}^T \cdot \left(\boldsymbol{M}^{\Sigma,NR^T} + \boldsymbol{M}^{\Delta,NR^T} \cdot \boldsymbol{\Gamma}^{R^T}\right) \cdot \boldsymbol{\Psi_1}\right] \\
&= \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_2}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \left(\sum_a \gamma^a \mathbb{1}^{a,R}\right) \cdot \boldsymbol{\Psi_1}\right] \\
&= \sum_a \frac{\partial\gamma^a}{\partial\gamma^j} \boldsymbol{\Psi_2}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{\Psi_1} \\
&= \boldsymbol{\Psi_2}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{\Psi_1}^T. \qquad \text{(A.18)}
\end{aligned}$$

Another useful derivative is

$$\begin{aligned}
\frac{\partial}{\partial\gamma^j}\left(\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{\Psi_2}\right) &= \frac{\partial}{\partial\gamma^j}\left(\boldsymbol{\Psi_1}^T \cdot \left[\boldsymbol{M}^{\Sigma,NR^T} + \boldsymbol{M}^{\Delta,NR^T} \cdot \boldsymbol{\Gamma}^{R^T}\right] \cdot \right. \\
&\qquad \left. \left[\boldsymbol{M}^{\Sigma,NR} + \boldsymbol{\Gamma}^{R} \cdot \boldsymbol{M}^{\Delta,NR}\right] \cdot \boldsymbol{\Psi_2}\right) \\
&= \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Sigma,NR^T} \cdot \boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{\Psi_2}\right] \\
&\quad + \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Sigma,NR^T} \cdot \boldsymbol{\Gamma}^{R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2}\right] \\
&\quad + \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \boldsymbol{\Gamma}^{R^T} \cdot \boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{\Psi_2}\right] \\
&\quad + \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \boldsymbol{\Gamma}^{R^T} \cdot \boldsymbol{\Gamma}^{R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2}\right].
\end{aligned}$$
$$\text{(A.19)}$$

The first term of this equation is clearly zero since it has no dependence on the sideband-gain factors. Comparing the next term to some of the intermediate steps in Equation A.17 shows that the second term is equal to $\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Sigma,NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{\Psi_2}$. Similarly, by comparing to Equation A.18, it can be seen that the third term is equal to $\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{j,R} \cdot$

$\boldsymbol{M}^{\boldsymbol{\Sigma},NR} \cdot \boldsymbol{\Psi_2}$. The fourth term can be simplified as follows:

$$\begin{aligned}
(\text{Term 4}) &= \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \boldsymbol{\Gamma}^{R^T} \cdot \boldsymbol{\Gamma}^{R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}\right] \\
&= \frac{\partial}{\partial\gamma^j}\left[\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \left(\sum_a \gamma^a \mathbb{1}^{a,R}\right) \cdot \left(\sum_b \gamma^b \mathbb{1}^{b,R}\right) \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}\right] \\
&= \frac{\partial}{\partial\gamma^j}\left[\sum_{a,b} \gamma^a\gamma^b \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{a,R} \cdot \mathbb{1}^{b,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}\right] \\
&= \frac{\partial}{\partial\gamma^j}\left[\sum_{a,b} \gamma^a\gamma^b \delta_{ab} \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}\right] \text{ by Equation 4.44} \\
&= \frac{\partial}{\partial\gamma^j}\left[\sum_{a} (\gamma^a)^2 \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}\right] \\
&= \sum_a \frac{\partial\,(\gamma^a)^2}{\partial\gamma^j} \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2} \\
&= \sum_a 2\gamma^a \delta_{aj} \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{a,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2} \\
&= 2\gamma^j \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2}.
\end{aligned} \tag{A.20}$$

Combining these terms generates the desired formula:

$$\begin{aligned}
\frac{\partial}{\partial\gamma^j}\left(\boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},NR^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},NR} \cdot \boldsymbol{\Psi_2}\right) &= \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Sigma},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2} \\
&\quad + \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Sigma},NR} \cdot \boldsymbol{\Psi_2} \\
&\quad + 2\gamma^j \boldsymbol{\Psi_1}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \cdot \boldsymbol{\Psi_2} \\
&= \boldsymbol{\Psi_1}^T \cdot \Big(\boldsymbol{M}^{\boldsymbol{\Sigma},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR} \\
&\qquad + \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Sigma},NR} \\
&\qquad + 2\gamma^j \boldsymbol{M}^{\boldsymbol{\Delta},NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},NR}\Big) \cdot \boldsymbol{\Psi_2}.
\end{aligned} \tag{A.21}$$

We can easily calculate the derivative of $\boldsymbol{d}^{NR}$ using these results:[4]

[4]The fact that we are differentiating the i^{th} component of $\boldsymbol{d}^{NR}$ does not prevent us from using the general result, a fact that can be demonstrated as follows. Let $\hat{\boldsymbol{\imath}}$ represent a basis vector with all components equal to 0, except the i^{th}, which is equal to 1. Then we can write

$$\frac{\partial d_i^{NR}}{\partial\gamma^j} = \frac{\partial}{\partial\gamma^j}\hat{\boldsymbol{\imath}} \cdot \boldsymbol{d}^{NR},$$

which allows us to apply Equation A.17 by including $\hat{\boldsymbol{\imath}}$ in $\boldsymbol{\Psi_1}$.

Quantity	Value	Equation
$\dfrac{\partial d_i^{NR}}{\partial s_j}$	$M_{ij}^{\Gamma,NR}$	A.13
$\dfrac{\partial d_i^{NR}}{\partial b_j}$	$\eta_B C_{ij}^{NR}$	A.14
$\dfrac{\partial \beta_i^{NR}}{\partial b_j}$	C_{ij}^{NR}	A.15
$\dfrac{\partial d_i^{NR}}{\partial \gamma^j}$	$\left[\mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}} \cdot \boldsymbol{s}\right]_i$	A.22
$\mathbb{1}^{\boldsymbol{j,R}} \cdot \mathbb{1}^{\boldsymbol{a,R}}$	$\delta_{aj}\mathbb{1}^{\boldsymbol{a,R}}$	4.44
$\mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma,NR}}$	$\mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M}^{\boldsymbol{\Sigma,NR}} + \gamma^j \mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}}$	A.16

Table A.2: Results often needed during the discussion of different deconvolution routines.

$$\begin{aligned}\frac{\partial d_i^{NR}}{\partial \gamma^j} &= \frac{\partial}{\partial \gamma^j}\left[\boldsymbol{M}^{\boldsymbol{\Gamma,NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_i \\ &= \left[\mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}} \cdot \boldsymbol{s}\right]_i . \end{aligned} \tag{A.22}$$

A.3 Derivation of Nonlinear Optimization Equations

As discussed in Section 4.12, the minimum of the χ^2_{mod} surface can be found by solving the set of coupled, nonlinear equations that result from calculating $\frac{\partial \chi^2_{mod}}{\partial x_i} = 0$, where x_i is used to symbolically represent each of the independent parameters ($\boldsymbol{s}, \boldsymbol{b},$ and γ). Before diving into these calculations, we explicitly note how the pieces of χ^2_{mod} depend on each of the parameters (Equation 4.65):

$$\begin{aligned} \chi^2_{o,i} &= \chi^2_{o,i}\left(\boldsymbol{s}, \boldsymbol{b}, \gamma\right), \\ \chi^2_{S,j} &= \chi^2_{S,j}\left(\boldsymbol{s}\right), \\ \chi^2_{\beta,k} &= \chi^2_{\beta,k}\left(\boldsymbol{b}\right), \text{ and} \\ \chi^2_{\gamma,m} &= \chi^2_{\gamma,m}\left(\gamma\right). \end{aligned} \tag{4.65}$$

Taking the partial derivatives of χ^2_{mod} with respect to s_n and setting the results equal to zero generates a set of equations that can be used to determine the values of the deconvolved spectrum:

$$\frac{\partial \chi^2_{mod}}{\partial s_n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial s_n} + \sum_j \frac{\partial \chi^2_{S,j}}{\partial s_n} = 0. \tag{4.78}$$

Calculating each of these terms gives

$$\begin{aligned}
0 &= \sum_i \frac{\partial \chi^2_{o,i}}{\partial s_n} + \sum_j \frac{\partial \chi^2_{S,j}}{\partial s_n} \\
&= \lambda_\chi \sum_i \frac{\partial}{\partial s_n}\left[\left(d_i^{NR} - d_i^{o,NR}\right)^2\right] + \lambda_S \sum_j \frac{\partial}{\partial s_n}\left(s_j\right)^2 \\
&= 2\lambda_\chi \sum_i \left(d_i^{NR} - d_i^{o,NR}\right)\frac{\partial d_i^{NR}}{\partial s_n} + 2\lambda_S \sum_j s_j \delta_{jn} \quad \text{by Equation A.11} \\
&= 2\lambda_\chi \sum_i \left(d_i^{NR} - d_i^{o,NR}\right) M_{in}^{\Gamma,NR} + 2\lambda_S \sum_j s_j \delta_{jn} \quad \text{by Equation A.13} \\
&= \lambda_\chi \sum_i M_{ni}^{\Gamma,NR^T}\left(d_i^{NR} - d_i^{o,NR}\right) + \lambda_S s_n \\
&= \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_n + \lambda_S s_n.
\end{aligned}$$

This equation represents the n^{th} element of an N_{SSB}-component vector; if each component is equal to 0, then the entire vector must be equal to 0, allowing all N_{SSB} equations to be written as a single vector equation:

$$\lambda_\chi \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) + \lambda_S \boldsymbol{s} = 0. \tag{A.23}$$

Similarly, the coefficients of the baseline terms can be found by taking derivatives with respect to the b_n and setting the result equal to 0:

$$\frac{\partial \chi^2_{mod}}{\partial b_n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial b_n} + \sum_k \frac{\partial \chi^2_{\beta,k}}{\partial b_n} = 0. \tag{4.79}$$

We can expand this equation to find

$$
\begin{aligned}
0 &= \sum_i \frac{\partial \chi^2_{o,i}}{\partial b_n} + \sum_k \frac{\partial \chi^2_{\beta,k}}{\partial b_n} \\
&= \lambda_\chi \sum_i \frac{\partial}{\partial b_n} \left[\left(d_i^{NR} - d_i^{o,NR} \right)^2 \right] + \eta_B \lambda_B \sum_k \frac{\partial}{\partial b_n} \left(\beta_k^{NR} \right)^2 \\
&= 2\lambda_\chi \sum_i \left(d_i^{NR} - d_i^{o,NR} \right) \frac{\partial d_i^{NR}}{\partial b_n} + 2\eta_B \lambda_B \sum_k \beta_k^{NR} \frac{\partial}{\partial b_n} \left(\beta_k^{NR} \right) \\
&= 2\lambda_\chi \eta_B \sum_i C_{ni}^{NR^T} \left(d_i^{NR} - d_i^{o,NR} \right) + 2\eta_B \lambda_B \sum_k C_{nk}^{NR^T} \beta_k^{NR} \quad \text{by Equations A.14 and A.15} \\
&= \eta_B \left[\lambda_\chi \left[\boldsymbol{C^{NR}}^T \cdot \left(\boldsymbol{d^{NR}} - \boldsymbol{d^{o,NR}} \right) \right]_n + \lambda_B \left[\boldsymbol{C^{NR}}^T \cdot \boldsymbol{\beta^{NR}} \right]_n \right].
\end{aligned}
$$

As before, this represents the n^{th} component of a vector, and we can write the full set of equations as

$$
\eta_B \boldsymbol{C^{NR}}^T \cdot \left[\lambda_\chi \left(\boldsymbol{d^{NR}} - \boldsymbol{d^{o,NR}} \right) + \lambda_B \boldsymbol{\beta^{NR}} \right] = 0. \tag{A.24}
$$

Finally, the set of sideband gains can be found by taking the derivative of χ^2_{mod} with respect to γ^n and setting the result equal to zero:

$$
\frac{\partial \chi^2_{mod}}{\partial \gamma^n} = \sum_i \frac{\partial \chi^2_{o,i}}{\partial \gamma^n} + \sum_m \frac{\partial \chi^2_{\gamma,m}}{\partial \gamma^n} = 0. \tag{4.80}
$$

Calculating the derivatives yields

$$
\begin{aligned}
0 &= \sum_i \frac{\partial \chi^2_{o,i}}{\partial \gamma^n} + \sum_m \frac{\partial \chi^2_{\gamma,m}}{\partial \gamma^n} \\
&= \sum_i \frac{\partial}{\partial \gamma^n} \left[\left(d_i^{NR} - d_i^{o,NR} \right)^2 \right] + \lambda_G \sum_m \frac{\partial}{\partial \gamma^n} \left(\gamma^m \right)^2 \\
&= 2\lambda_\chi \sum_i \left(d_i^{NR} - d_i^{o,NR} \right) \frac{\partial d_i^{NR}}{\partial \gamma^n} + 2\lambda_G \sum_m \gamma^m \delta_{mn} \quad \text{by Equation A.11} \\
&= 2\lambda_\chi \sum_i \left(d_i^{NR} - d_i^{o,NR} \right) \left[\mathbb{1}^{\boldsymbol{n,R}} \cdot \boldsymbol{M^{\Delta,NR}} \cdot \boldsymbol{s} \right]_i + 2\lambda_G \gamma^n \quad \text{by EquationA.22} \\
&= \lambda_\chi \left(\boldsymbol{d^{NR}} - \boldsymbol{d^{o,NR}} \right)^T \cdot \mathbb{1}^{\boldsymbol{n,R}} \cdot \boldsymbol{M^{\Delta,NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^n.
\end{aligned} \tag{A.25}
$$

Setting each of the three sets of derivatives from Equations 4.78, 4.79, and 4.80 equal to zero produces a set of nonlinear equations that can be used to solve for the individual parameters:

$$\frac{\partial \chi^2_{mod}}{\partial s_n} = 0 \Longrightarrow \qquad \lambda_\chi \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) + \lambda_S \boldsymbol{s} = 0,$$

$$\frac{\partial \chi^2_{mod}}{\partial b_n} = 0 \Longrightarrow \qquad \eta_B \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \left[\lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) + \lambda_B \boldsymbol{\beta}^{\boldsymbol{NR}}\right] = 0, \text{ and}$$

$$\frac{\partial \chi^2_{mod}}{\partial \gamma^n} = 0 \Longrightarrow \qquad \lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)^T \cdot \mathbb{1}^{\boldsymbol{n},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^n = 0.$$

A.4 Jacobian for Nonlinear Root Finding

A.4.1 Calculating Individual Components

Section 4.12.2 uses the Jacobian, defined as (Equation 4.82)

$$\boldsymbol{J}^{\boldsymbol{nonlin}} = \left(\begin{array}{c:c:c} \dfrac{\partial L_1}{\partial s_n} & \dfrac{\partial L_1}{\partial b_p} & \dfrac{\partial L_1}{\partial \gamma^q} \\ \hdashline \dfrac{\partial L_2}{\partial s_n} & \dfrac{\partial L_2}{\partial b_p} & \dfrac{\partial L_2}{\partial \gamma^q} \\ \hdashline \dfrac{\partial L_3}{\partial s_n} & \dfrac{\partial L_3}{\partial b_p} & \dfrac{\partial L_3}{\partial \gamma^q} \end{array}\right), \tag{A.26}$$

where $L_{1,i}$, $L_{2,i}$, and $L_{3,i}$ represent the i^{th} components of the left-hand sides of the first, second, and third lines in Equations 4.81, respectively, expanded using Equations 4.60 and 4.61:

$$L_{1,i} = \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i + \lambda_S s_i, \tag{A.27a}$$

$$L_{2,i} = \eta_B \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \left(\lambda_\chi \boldsymbol{d}^{\boldsymbol{NR}} + \lambda_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \lambda_\chi \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i \text{ and} \tag{A.27b}$$

$$L_{3,i} = \lambda_\chi \left(\boldsymbol{d}^{\boldsymbol{NR}} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}^T}\right) \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^i. \tag{A.27c}$$

The first component of the Jacobian is equal to

$$\begin{aligned} \frac{\partial L_{1,i}}{\partial s_n} &= \lambda_\chi \frac{\partial}{\partial s_n} \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i + \lambda_S \frac{\partial s_i}{\partial s_n} \\ &= \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}}\right]_{in} + \lambda_S \mathbb{1}_{in} \quad \text{by Equations A.2 and A.1.} \end{aligned}$$

The second component can be similarly calculated:

$$\begin{aligned}\frac{\partial L_{1,i}}{\partial b_p} &= \lambda_\chi \eta_B \frac{\partial}{\partial b_p}\left(\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right]_i\right)\\ &= \lambda_\chi \eta_B \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\right]_{ip} \quad \text{by Equation A.2.}\end{aligned}$$

The third element can be calculated using Equations A.18 and A.21:

$$\begin{aligned}\frac{\partial L_{1,i}}{\partial \gamma^q} = {} & \lambda_\chi \frac{\partial}{\partial \gamma^q}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}}\cdot \boldsymbol{s}\right]_i + \lambda_\chi \eta_B \frac{\partial}{\partial \gamma^q}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right]_i\\ & - \lambda_\chi \frac{\partial}{\partial \gamma^q}\left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T}\cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right]_i\\ = {} & \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\cdot \boldsymbol{s}\right]_i + \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}}\cdot \boldsymbol{s}\right]_i\\ & + 2\lambda_\chi \gamma^q \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\cdot \boldsymbol{s}\right]_i\\ & + \lambda_\chi \eta_B \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right]_i - \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right]_i\\ = {} & \lambda_\chi \left[\left(\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}^T} + 2\gamma^q \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\right)\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\cdot \boldsymbol{s}\right]_i +\\ & \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T}\cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}}\cdot \left(\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}}\cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i\end{aligned}$$

We can now begin working on the matrix elements from the second row:

$$\begin{aligned}\frac{\partial L_{2,i}}{\partial s_n} &= \frac{\partial}{\partial s_n}\left(\eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \boldsymbol{d}^{\boldsymbol{NR}}\right]_i\right)\\ &= \eta_B \lambda_\chi \frac{\partial}{\partial s_n}\left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}}\cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right)\right]_i\\ &= \eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}}\right]_{in} \quad \text{by Equation A.2.}\end{aligned}$$

The next element contains two terms:

$$\begin{aligned}\frac{\partial L_{2,i}}{\partial b_p} &= \eta_B \frac{\partial}{\partial b_p}\left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \left(\lambda_\chi \boldsymbol{d}^{\boldsymbol{NR}} + \lambda_B \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right)\right]_i\\ &= \eta_B \frac{\partial}{\partial b_p}\left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \left(\lambda_\chi \eta_B \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b} + \lambda_B \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right)\right]_i\\ &= \eta_B \left(\lambda_\chi \eta_B + \lambda_B\right) \frac{\partial}{\partial b_p}\left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\cdot \boldsymbol{b}\right]_i\\ &= \eta_B \left(\eta_B \lambda_\chi + \lambda_B\right)\left[\boldsymbol{C}^{\boldsymbol{NR}^T}\cdot \boldsymbol{C}^{\boldsymbol{NR}}\right]_{ip} \quad \text{by Equation A.2.}\end{aligned}$$

Equation A.17 allows us to quickly calculate the final element of the second row:

$$\begin{aligned}
\frac{\partial L_{2,i}}{\partial \gamma^q} &= \eta_B \lambda_\chi \frac{\partial}{\partial \gamma^q} \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{d}^{\boldsymbol{NR}} \right]_i \\
&= \eta_B \lambda_\chi \frac{\partial}{\partial \gamma^q} \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} \right) \right]_i \\
&= \eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right]_i .
\end{aligned}$$

Moving on to the first element of the third row gives

$$\begin{aligned}
\frac{\partial L_{3,i}}{\partial s_n} &= \lambda_\chi \frac{\partial}{\partial s_n} \left(\boldsymbol{d}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right) \\
&= \lambda_\chi \frac{\partial}{\partial s_n} \left[\left(\boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} + \eta_B \boldsymbol{b}^T \cdot \boldsymbol{C}^{\boldsymbol{NR}^T} \right) \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right] \\
&= \lambda_\chi \frac{\partial}{\partial s_n} \left[\boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right] \\
&= \lambda_\chi \left[\left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} + \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \right) \cdot \boldsymbol{s} \right]_n .
\end{aligned}$$

The next element can be calculated easily using Equation A.7:

$$\begin{aligned}
\frac{\partial L_{3,i}}{\partial b_p} &= \lambda_\chi \frac{\partial}{\partial b_p} \left(\boldsymbol{d}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right) \\
&= \eta_B \lambda_\chi \frac{\partial}{\partial b_p} \left(\boldsymbol{b}^T \cdot \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right) \\
&= \eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right]_p .
\end{aligned}$$

The final element of the Jacobian can be calculated using Equations A.1 and A.18:

$$\begin{aligned}
\frac{\partial L_{3,i}}{\partial \gamma^q} &= \lambda_\chi \frac{\partial}{\partial \gamma^q} \left(\boldsymbol{d}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^i \right) \\
&= \lambda_\chi \frac{\partial}{\partial \gamma^q} \left(\boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \gamma^i \right) \\
&= \lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{q},\boldsymbol{R}} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \delta_{iq} \\
&= \left(\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \right) \delta_{iq} \quad \text{by Equation 4.44.}
\end{aligned}$$

A.4.2 Full Jacobian

Putting all of these together generates the full Jacobian:

$$\boldsymbol{J}^{nlin} = \left(\begin{array}{c:c:c} \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} + \lambda_S \mathbb{1} & \lambda_\chi \eta_B \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} & \boldsymbol{J}^{nlin(L_1,\gamma)} \\ \hdashline \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} & \eta_B \left(\eta_B \lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} & \boldsymbol{J}^{nlin(L_2,\gamma)} \\ \hdashline \boldsymbol{J}^{nlin(L_3,s)} & \boldsymbol{J}^{nlin(L_3,b)} & \boldsymbol{J}^{nlin(L_3,\gamma)} \end{array}\right), \quad \text{(A.27)}$$

where:

$$J_{ij}^{nlin(L_1,\gamma)} = \lambda_\chi \left[\left(\boldsymbol{M}^{\Sigma,NR^T} + 2\gamma^q \boldsymbol{M}^{\Delta,NR^T}\right) \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_i + \lambda_\chi \left[\boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{j,R} \cdot \left(\boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right)\right]_i, \quad \text{(A.28a)}$$

$$J_{ij}^{nlin(L_2,\gamma)} = \eta_B \lambda_\chi \left[\boldsymbol{C}^{NR^T} \cdot \mathbb{1}^{j,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_i, \quad \text{(A.28b)}$$

$$J_{ij}^{nlin(L_3,s)} = \lambda_\chi \left[\left(\boldsymbol{M}^{\Gamma,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} + \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Gamma,NR}\right) \cdot \boldsymbol{s}\right]_j, \quad \text{(A.28c)}$$

$$J_{ij}^{nlin(L_3,b)} = \eta_B \lambda_\chi \left[\boldsymbol{C}^{NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right]_j, \text{ and} \quad \text{(A.28d)}$$

$$J_{ij}^{nlin(L_3,\gamma)} = \left(\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s} + \lambda_G\right) \delta_{ij}. \quad \text{(A.28e)}$$

A.5 Perturbation Method for Pseudo-Linear Optimization

Section 4.13 uses a first-order perturbation analysis to estimate the changes that would occur in $\boldsymbol{s}$ and $\boldsymbol{b}$ as a result of a given change in γ. The results are summarized in that section while the detailed calculations are shown below.

Let γ_o, $\boldsymbol{s}_o$, and $\boldsymbol{b}_o$ represent a solution to Equations 4.85, so that

$$\left(\lambda_\chi \boldsymbol{M}_o^{\Gamma,NR^T} \cdot \boldsymbol{M}_o^{\Gamma,NR} + \lambda_S \mathbb{1}\right) \cdot \boldsymbol{s}_o + \eta_B \lambda_\chi \boldsymbol{M}_o^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} \cdot \boldsymbol{b}_o = \lambda_\chi \boldsymbol{M}_o^{\Gamma,NR^T} \cdot \boldsymbol{d}^{o,NR} \quad \text{(A.29a)}$$

$$\eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}_o^{\Gamma,NR} \cdot \boldsymbol{s}_o + \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \cdot \boldsymbol{b}_o = \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{d}^{o,NR}, \quad \text{(A.29b)}$$

where $\boldsymbol{M}_o^{\Gamma,NR}$ represents the convolution matrix formed using the components of γ_o according to Equation 4.41.

We can approximate the derivative by calculating the changes in $\boldsymbol{s}$ and $\boldsymbol{b}$ that would be generated by a small deviation from γ_o. Mathematically, we represent this by rewriting Equations 4.85 with the following substitutions:

$$\begin{aligned} \gamma &\rightarrow \boldsymbol{\gamma_o} + \delta\gamma \\ \boldsymbol{s} &\rightarrow \boldsymbol{s_o} + \delta\boldsymbol{s} \\ \boldsymbol{b} &\rightarrow \boldsymbol{b_o} + \delta\boldsymbol{b}, \end{aligned} \tag{A.30}$$

where $\delta\gamma$ is specified and we wish to find the resulting $\delta\boldsymbol{s}$ and $\delta\boldsymbol{b}$. It is worth noting that the variation in γ can be expressed as a similar variation in $\boldsymbol{\Gamma^R}$. Equation 4.45 demonstrates that $\boldsymbol{\Gamma^R}$ is linear with respect to the elements of γ, so when the variation is applied,

$$\begin{aligned} \boldsymbol{\Gamma^R} &\rightarrow \sum_a \left(\gamma_o^a + \delta\gamma^a\right) \mathbb{1}^{\boldsymbol{a,R}} \\ &= \sum_a \gamma_o^a \mathbb{1}^{\boldsymbol{a,R}} + \sum_a \delta\gamma^a \mathbb{1}^{\boldsymbol{a,R}} \\ &= \boldsymbol{\Gamma_o^R} + \delta\boldsymbol{\Gamma^R}, \end{aligned}$$

and we can identify $\delta\boldsymbol{\Gamma^R} = \sum_a \delta\gamma^a \mathbb{1}^{\boldsymbol{a,R}}$ as the corresponding change in $\boldsymbol{\Gamma}$. This allows us to write the effects of the variation on the convolution matrix as:

$$\begin{aligned} \boldsymbol{M}^{\boldsymbol{\Gamma,N}} &\rightarrow \boldsymbol{M}^{\boldsymbol{\Sigma,NR}} + \left(\boldsymbol{\Gamma_o^R} + \delta\boldsymbol{\Gamma^R}\right) \boldsymbol{M}^{\boldsymbol{\Delta,NR}} \\ &= \boldsymbol{M_o^{\Gamma,NR}} + \delta\boldsymbol{\Gamma^R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}}. \end{aligned}$$

We start by applying the variation to Equation A.29a:

$$\begin{aligned} &\left[\lambda_\chi \left(\boldsymbol{M_o^{\Gamma,NR}}^T + \boldsymbol{M}^{\boldsymbol{\Delta,NR}^T} \cdot \delta\boldsymbol{\Gamma^R}^T\right) \cdot \left(\boldsymbol{M_o^{\Gamma,NR}} + \delta\boldsymbol{\Gamma^R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}}\right) + \lambda_S \mathbb{1}\right] \cdot \left(\boldsymbol{s_o} + \delta\boldsymbol{s}\right) \\ &\qquad + \eta_B \lambda_\chi \left(\boldsymbol{M_o^{\Gamma,NR}}^T + \boldsymbol{M}^{\boldsymbol{\Delta,NR}^T} \cdot \delta\boldsymbol{\Gamma^R}^T\right) \cdot \boldsymbol{C^{NR}} \cdot \left(\boldsymbol{b_o} + \delta\boldsymbol{b}\right) \\ &\qquad\qquad = \lambda_\chi \left(\boldsymbol{M_o^{\Gamma,NR}}^T + \boldsymbol{M}^{\boldsymbol{\Delta,NR}^T} \cdot \delta\boldsymbol{\Gamma^R}^T\right) \cdot \boldsymbol{d^{o,NR}}. \end{aligned} \tag{A.31}$$

Distributing the multiplication and noting that $\delta\boldsymbol{\Gamma^R}^T = \delta\boldsymbol{\Gamma^R}$ gives

$$\begin{aligned} \Bigg[\lambda_\chi \Bigg(&\boldsymbol{M_o^{\Gamma,NR}}^T \cdot \boldsymbol{M_o^{\Gamma,NR}} + \boldsymbol{M_o^{\Gamma,NR}}^T \cdot \delta\boldsymbol{\Gamma^R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}} + \boldsymbol{M}^{\boldsymbol{\Delta,NR}^T} \cdot \delta\boldsymbol{\Gamma^R} \cdot \boldsymbol{M_o^{\Gamma,NR}} \\ &+ \underbrace{\boldsymbol{M}^{\boldsymbol{\Delta,NR}^T} \cdot \delta\boldsymbol{\Gamma^R} \cdot \delta\boldsymbol{\Gamma^R} \cdot \boldsymbol{M}^{\boldsymbol{\Delta,NR}}}_{\delta\boldsymbol{\Gamma^R} \cdot \delta\boldsymbol{\Gamma^R} \rightarrow 0}\Bigg) + \lambda_S \mathbb{1}\Bigg] \cdot \boldsymbol{s_o} \end{aligned}$$

$$+\left[\lambda_\chi\left(\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{M}_o^{\Gamma,NR}+\underbrace{\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}^{\Delta,NR}}_{\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{s}\to 0}+\underbrace{\boldsymbol{M}^{\Delta,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}_o^{\Gamma,NR}}_{\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{s}\to 0}\right.\right.$$

$$\left.\left.+\underbrace{\boldsymbol{M}^{\Delta,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}^{\Delta,NR}}_{\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{s}\to 0}\right)+\lambda_S\mathbb{1}\right]\cdot\delta\boldsymbol{s}$$

$$+\eta_B\lambda_\chi\left(\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{C}^{NR}\cdot\boldsymbol{b}_o+\boldsymbol{M}^{\Delta,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{C}^{NR}\cdot\boldsymbol{b}_o\right.$$

$$\left.+\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{C}^{NR}\cdot\delta\boldsymbol{b}+\underbrace{\boldsymbol{M}^{\Delta,NR^T}\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{C}^{NR}\cdot\delta\boldsymbol{b}}_{\delta\boldsymbol{\Gamma}^R\cdot\delta\boldsymbol{b}\to 0}\right)$$

$$=\lambda_\chi\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{d}^{o,NR}+\lambda_\chi\boldsymbol{M}^{\Delta,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{d}^{o,NR}. \quad \text{(A.32)}$$

Since we want to approximate the Jacobian, which represents a first derivative, we can ignore terms that are second-order and higher in the variation, as indicated by the braces in this equation. If we then group the coefficients of the variational quantities, we find

$$\underbrace{\lambda_\chi\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\left(\boldsymbol{M}_o^{\Gamma,NR}\cdot\boldsymbol{s}_o+\eta_B\boldsymbol{C}^{NR}\cdot\boldsymbol{b}_o-\boldsymbol{d}^{o,NR}\right)+\lambda_S\cdot\boldsymbol{s}_o}_{\text{zeroth-order terms}\to 0}$$

$$+\left(\lambda_\chi\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{M}_o^{\Gamma,NR}+\lambda_S\mathbb{1}\right)\cdot\delta\boldsymbol{s}+\eta_B\lambda_\chi\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\boldsymbol{C}^{NR}\cdot\delta\boldsymbol{b}$$

$$=\lambda_\chi\boldsymbol{M}^{\Delta,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\left(\boldsymbol{d}^{o,NR}-\boldsymbol{M}_o^{\Gamma,NR}\cdot\boldsymbol{s}_o-\eta_B\boldsymbol{C}^{NR}\cdot\boldsymbol{b}_o\right)$$

$$-\lambda_\chi\boldsymbol{M}_o^{\Gamma,NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}^{\Delta,NR}\cdot\boldsymbol{s}_o. \quad \text{(A.33)}$$

The terms in the first line sum to zero based on Equation A.47a, leaving relatively simple coefficients for $\delta\boldsymbol{s}$ and $\delta\boldsymbol{b}$.

We repeat the process by applying the variation to Equation A.29b:

$$\eta_B\lambda_\chi\boldsymbol{C}^{NR^T}\cdot\left(\boldsymbol{M}_o^{\Gamma,NR}+\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}^{\Delta,NR}\right)\cdot\left(\boldsymbol{s}_o+\delta\boldsymbol{s}\right)+$$

$$\eta_B\left(\lambda_\chi+\lambda_B\right)\boldsymbol{C}^{NR^T}\cdot\boldsymbol{C}^{NR}\cdot\left(\boldsymbol{b}_o+\delta\boldsymbol{b}\right)=\eta_B\lambda_\chi\boldsymbol{C}^{NR^T}\cdot\boldsymbol{d}^{o,NR}. \quad \text{(A.34)}$$

The only higher-order term in this equation is $\eta_B\lambda_\chi\boldsymbol{C}^{NR^T}\cdot\delta\boldsymbol{\Gamma}^R\cdot\boldsymbol{M}^{\Delta,NR}\cdot\delta\boldsymbol{s}$, which we again set to zero, leaving

$$\underbrace{\eta_B \boldsymbol{C}^{NR^T} \cdot \left[\lambda_\chi \left(\boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR} \cdot \lambda_\chi \boldsymbol{s_o} - \boldsymbol{d}^{o,NR} + \boldsymbol{C}^{NR} \cdot \boldsymbol{b_o}\right) + \lambda_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b_o}\right]}_{\text{zeroth-order terms}\to 0}$$
$$+ \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR} \cdot \delta\boldsymbol{s} + \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \cdot \delta\boldsymbol{b}$$
$$= -\eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \delta\boldsymbol{\Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s_o}, \quad \text{(A.35)}$$

where the terms on the first line can be set to zero by Equation A.29b.

Combining these results gives a set of equations describing the $\delta\boldsymbol{s}$ and $\delta\boldsymbol{b}$ that result from a small deviation about the solution point, denoted by $\delta\gamma$:

$$\left(\lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR^T} \cdot \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR} + \lambda_S \mathbb{1}\right) \cdot \delta\boldsymbol{s} + \eta_B \lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} \cdot \delta\boldsymbol{b}$$
$$= \lambda_\chi \boldsymbol{M}^{\Delta,NR^T} \cdot \delta\boldsymbol{\Gamma}^{\boldsymbol{R}} \cdot \left(\boldsymbol{d}^{o,NR} - \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR} \cdot \boldsymbol{s_o} - \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b_o}\right) -$$
$$\lambda_\chi \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR^T} \cdot \delta\boldsymbol{\Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s_o} \quad \text{(A.36a)}$$

$$\eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}_{\boldsymbol{o}}^{\Gamma,NR} \cdot \delta\boldsymbol{s} + \eta_B \left(\lambda_\chi + \lambda_B\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \cdot \delta\boldsymbol{b}$$
$$= -\eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \delta\boldsymbol{\Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s_o}. \quad \text{(A.36b)}$$

A.6 *Mathematica* Utility

To improve accuracy and decrease the barrier to testing new strategies, we have developed software capable of analytically calculating these derivatives.[5] By considering the previous calculations, we can abstract a general algorithm for taking the needed derivatives:

1. Rewrite quantities in terms of the underlying parameters ($\boldsymbol{s}$, $\boldsymbol{b}$, and γ).

2. Expand matrix and vector multiplication in terms of the individual components, using either explicit or implicit summation.

[5]We found it necessary to develop custom routines, as the vector calculus package in *Mathematica* appears to be focused on three-dimensional analysis, rather than the many-dimensional methodology discussed here. Extensions to higher-dimensionality geometries exist (such as those for general relativity), but they also assumed a fixed, finite dimensionality. Moreover, the use of a third matrix index (the a index on $\mathbb{1}_{ij}^{a}$ and γ^a) requires a custom approach.

3. Differentiate the component equations. Since the individual components are simply numbers, the usual rules of differentiation apply, thus avoiding any problems with the non-commutative multiplication of matrices.

4. Arrange the remaining components so that the right index of one component matches the left index of the next, with the free indices at the leftmost and rightmost positions. If two components have matched left or right indices $\left(\text{e.g., } A_{jk}B_{ji}\right)$, taking the transpose of one results in the desired ordering $\left(B_{ji} \to \left(B^T\right)_{ij}\right)$.

5. Once all of the components have been paired off in this fashion, the component equation can be converted back into a matrix multiplication.

6. If desired, regroup fundamental parameters in terms of more compact quantities to undo the substitutions from Step 1.

The powerful symbol-manipulation abilities of *Mathematica* allow us to automate this process. We have developed a *Mathematica* utility that converts formulas written out in a matrix form into the underlying component-based formulas, takes the derivative, and then converts the results back into a matrix form. Additional routines address the issue of having a third index, a, for use on the sideband gains, γ^a, and the truncated identity functions, $\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}$. The utility can also apply the variational calculus needed for Section A.5.[6]

The above discussion indicates how individual steps of matrix multiplication and differentiation can be translated into the corresponding *Mathematica* functions. For convenience, we have also designed several functions that can display results and convert them into publication-friendly formats. Throughout the remainder of this section, there are several examples showing how to combine these functions to perform complete calculations using the *Mathematica* utility.

As the hand calculations in previous sections show, the calculations are relatively straightforward, but the sheer number of terms generates opportunities for error. The *Mathematica* gradient utility has been very useful in helping to generate reliable results for this appendix. Since this utility appears to add previously unavailable functionality to *Mathematica*, it is included in Appendix B.

[6]The routines handle the third index on $\mathbb{1}^a_{ij}$, as well as the index of γ^a, making them sufficient for this work. However, these exceptions are hard-coded into the routines so that *only* a few specially defined variables receive special handling. Expanding these capabilities to handle such an index on arbitrarily named variables would require significant modification of the code, although it is relatively easy to add individual exceptions (such as $\delta\gamma$ and $\delta\boldsymbol{\Gamma}$ for Section A.7.3).

A.7 Validation of the *Mathematica* Utility

To ensure the validity of the software, the hand-derived results of Sections 4.11, A.3 and A.4.1 were recalculated in *Mathematica*.

A.7.1 Verification of the Direct Optimization Results

All of the results of Section 4.11 can be derived quickly using the *Mathematica* routines.[7] The terms of χ^2_{mod} are defined with the following commands

```
fChi[i_]:  = √λChi[] convertToComponents[d[i,•] -
   do[i,•]//.expandTerms];
fS[i_]:  = √λS[] convertToComponents[s[i,•]//.expandTerms];
fC[i_]:  = √λG[] convertToComponents[gamma[i]//.expandTerms];
fB[i_]:  =
   √λB[] √ηB[] convertToComponents[β[i,•]//.expandTerms];
```

and the calculation can be run and displayed using these commands

```
{resultA, resultAcomp} = calculateJacobian[{fChi[i], fS[i], fB[i],
   fC[i]}, {s[j], b[j], gamma[j]}, "jacobian"];
printElements[resultA]
```

The results are returned both as individual components and as a block matrix, the latter of which is shown below:

$$\begin{pmatrix} \texttt{m[i,j]}\sqrt{\lambda\texttt{Chi[]}} & \texttt{c[i,j]}\eta\texttt{B[]}\sqrt{\lambda\texttt{Chi[]}} & \texttt{ones[j][i,}\bullet\texttt{]}\cdot\texttt{mDelta[}\bullet\texttt{,}\bullet\texttt{]}\cdot\texttt{s[}\bullet\texttt{]}\sqrt{\lambda\texttt{Chi[]}} \\ \texttt{delta[i,j]}\sqrt{\lambda\texttt{S[]}} & 0 & 0 \\ 0 & \texttt{c[i,j]}\sqrt{\eta\texttt{B[]}}\sqrt{\lambda\texttt{B[]}} & 0 \\ 0 & 0 & \texttt{delta[i,j]}\sqrt{\lambda\texttt{G[]}} \end{pmatrix}. \tag{A.37}$$

[7]The representation of vectors in the *Mathematica* utility varies for input and output. On input, a vector must be given two indices; $x[i,\bullet]$ is used to represent a column vector while $x[\bullet,i]$ is used to represent the row vector. On output, these are represented as $x[i]$ and $x[i]^\dagger$, respectively. When converting *Mathematica* results into the equations shown here, the distinction between $\boldsymbol{x}$ and $\boldsymbol{x}^T$ was generally ignored.

The *Mathematica* results use a simplified form of implied summation. Indices represented by a circle, •, are part of the matrix multiplication indicated by the center-dot operator, so the formula $F_1[i,\bullet] \cdot F_2[\bullet,\bullet] \cdot x[\bullet]$ should be read as the i^{th}component of the matrix multiplication $\boldsymbol{F_1} \cdot \boldsymbol{F_2} \cdot \boldsymbol{x}$, which is denoted $[F_1 \cdot F_2 \cdot x]_i$ in this thesis. A quick comparison to Equation 4.76 shows that further translation is necessary to make the *Mathematica* results match those presented here. In particular, the conversions shown in Table A.3 need to be incorporated, yielding a result, that exactly reproduces Equation 4.76:[8]

$$\boldsymbol{J^{opt}} = \left(\begin{array}{c:c:c} \sqrt{\lambda_\chi} M_{ij}^{\Gamma,NR} & \eta_B \sqrt{\lambda_\chi} C_{ij}^{NR} & \sqrt{\lambda_\chi} \left[\mathbb{1}^{\boldsymbol{j,R}} \cdot \boldsymbol{M^{\Delta,NR}} \cdot \boldsymbol{s}\right]_i \\ \hdashline \sqrt{\lambda_S} \mathbb{1}_{ij} & 0 & 0 \\ \hdashline 0 & \sqrt{\eta_B \lambda_B} C_{ij}^{NR} & 0 \\ \hdashline 0 & 0 & \sqrt{\lambda_G} \mathbb{1}_{ij} \end{array}\right). \tag{A.38}$$

A.7.2 Verification of Nonlinear Root-Finding Results

The equations for optimization via nonlinear root finding presented earlier in this Appendix, as well as the corresponding Jacobian, are easy to calculate in *Mathematica*. We start by defining χ^2_{mod} :

```
chisq = convertToComponents[
  λChi[](d[•,k]-do[•,k])·(d[k,•]-do[k,•]) + λS[]s[•,k]·s[k,•] +
  λB[]ηB[]β[•,k]·β[k,• ] + λG[]gainsSquared[] //.expandTerms];
```

The nonlinear equations can be calculated and displayed

[8]When comparing this $\boldsymbol{J^{opt}}$ to Equation 4.76, it is important to remember the shorthand used in both cases. Equation 4.76 represents a block matrix in which each block is filled with the specified sub-matrix. The 0 elements are assumed to be matrices of the appropriate dimension filled with zeros. The matrix shown above is also assumed to be a block matrix in which i and j are measured within each block. Thus, the first element could equivalently be replaced with $\sqrt{\lambda_\chi}\boldsymbol{M^{\Gamma,NR}}$.

Description	This Work	*Mathematica*
Model DSB spectrum (with baselines)	d_i^{NR}	`d[i,●]`
Model DSB spectrum (without baselines)	$d_i^{NR} - \eta_B \beta_i^{NR}$	`d[i,●]` $-\ \eta$`B[ ]`β`[i,●]`
SSB spectrum	s_i	`s[i,●]`
Observed DSB spectrum	$d_i^{o,NR}$	`do[i,●]`
Baselines	β_i^{NR}	β`[i,●]`
Sideband gain factors	γ^a	`gamma[a]`
Sideband gain matrix	Γ_{ij}^R	`deltaG[i,j]`
Truncated identity matrix	$\mathbb{1}_{ij}^{a,R}$	`ones[a][i,j]`
Norm of gain-factors vector	$\sum_a \gamma^a \gamma^a$	`gainsSquared[ ]`
Full convolution matrix	$M_{ij}^{\Gamma,NR}$	`m[i,j]`
Sum convolution matrix	$M_{ij}^{\Sigma,NR}$	`mSigma[i,j]`
Difference convolution matrix	$M_{ij}^{\Delta,NR}$	`mDelta[i,j]`
Baseline coefficients	b_i	`b[i,●]`
Baseline functions matrix	C_{ij}^{NR}	`c[i,j]`
Baseline "switch"	η_B	η`B[ ]`
Variation in $\boldsymbol{s}$	δs_i	δ`s[i]`
Variation in $\boldsymbol{b}$	δb_i	δ`b[i]`
Variation in $\boldsymbol{\gamma}$	$\delta\gamma^a$	δ`gamma[ia]`
Variation in $\boldsymbol{\Gamma^R}$	$\delta\Gamma_{ij}^R$	δ`deltaG[i,j]`
Lagrange multiplier for χ_o^2	λ_χ	λ`Chi[ ]`
Lagrange multiplier for χ_S^2	λ_S	λ`S[ ]`
Lagrange multiplier for χ_β^2	λ_B	λ`B[ ]`
Lagrange multiplier for χ_γ^2	λ_G	λ`G[ ]`
Kronecker delta	δ_{ij}	δ`[i,j]`
Matrix transpose	^{T}	†
Matrix multiplication	$\cdot$	$.$

Table A.3: Conversion between notation used in this work and that of the *Mathematica* gradient-calculation package. Most quantities either translate directly or correspond to their normalized equivalents (as described in Section 4.8).

```
{eqns, eqnsComp} = calculateJacobian[{chisq/2}, {s[i], b[i],
   gamma[i]}];
eqnsOut = Transpose[printElements[eqns]][[2]];
```

to give the following results:[9]

$$\lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_S \boldsymbol{s} = 0 \tag{A.39a}$$

$$\eta_B \boldsymbol{C}^{NR^T} \cdot \left(\lambda_\chi \left(\boldsymbol{d}^{NR} - \boldsymbol{d}^{o,NR}\right) + \lambda_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) = 0 \tag{A.39b}$$

$$\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_G \gamma^i = 0 \tag{A.39c}$$

The Jacobian can be calculated as

```
{jacobian,jComp} = calculateJacobian[{eqnsComp[[1,1]],
   eqnsComp[[1,2]], eqnsComp[[1,3]]}, {s[j],b[j],gamma[j]},
   "jacobian"];
jacobianOut = printElements[jacobian];
```

[9]Actually, the *Mathematica* output needs to undergo some simplification by hand to match the previous results. The output generated for the second equation is

$$\eta_B \boldsymbol{C}^{NR^T} \cdot \left(\lambda_\chi \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} - \boldsymbol{d}^{o,NR}\right) + (\lambda_B + \eta_B \lambda_\chi) \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) = 0.$$

The form shown above can be derived by inserting the $\eta_B \lambda_\chi \boldsymbol{C}^{NR} \cdot \boldsymbol{b}$ term into the first set of parentheses and using Equation 4.61 to simplify the result.

Likewise, *Mathematica* returns the third equation in the following form:

$$\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Delta,N^T} \cdot \mathbb{1}^{i,R} \cdot \left(-\boldsymbol{d}^{o,NR} + \tfrac{1}{2}\boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{s} + \gamma^i \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) \\ + \tfrac{1}{2}\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Sigma,NR^T} \cdot \mathbb{1}^{i,R} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s} + \lambda_G \gamma^i = 0.$$

Since $\left(\tfrac{1}{2}\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Sigma,NR^T} \cdot \mathbb{1}^{i} \cdot \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s}\right)$ is just a scalar quantity, it can be transposed and combined with the $\tfrac{1}{2}\boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{s}$ term in the first set of parentheses to give

$$\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\Delta,NR^T} \cdot \mathbb{1}^{i,R} \cdot \left(-\boldsymbol{d}^{o,NR} + \boldsymbol{M}^{\Sigma,NR} \cdot \boldsymbol{s} + \gamma^i \boldsymbol{M}^{\Delta,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) + \lambda_G \gamma^i = 0,$$

which can be simplified using Equation A.16 to yield the form shown above.

to give[10]

$$\boldsymbol{J}^{\boldsymbol{nlin}} = \left(\begin{array}{c:c:c} \boldsymbol{J}^{\boldsymbol{nlin}(L_1,s)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_1,b)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_1,\gamma)} \\ \hdashline \boldsymbol{J}^{\boldsymbol{nlin}(L_2,s)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_2,b)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_2,\gamma)} \\ \hdashline \boldsymbol{J}^{\boldsymbol{nlin}(L_3,s)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_3,b)} & \boldsymbol{J}^{\boldsymbol{nlin}(L_3,\gamma)} \end{array}\right), \tag{A.40}$$

where

$$\begin{aligned}
\boldsymbol{J}^{\boldsymbol{nlin}(L_1,s)} &= \lambda_\chi \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} + \lambda_S \mathbb{1}, \\
\boldsymbol{J}^{\boldsymbol{nlin}(L_1,b)} &= \eta_B \lambda_\chi \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}}, \\
J_{ij}^{nlin(L_1,\gamma)} &= \left[\lambda_\chi \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j}} \cdot \left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i \\
&\qquad + \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{N}^T} \cdot \mathbb{1}^{\boldsymbol{j}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{N}} \cdot \boldsymbol{s}\right]_i, \\
J_{ij}^{nlin(L_2,s)} &= \eta_B \lambda_\chi \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}}, \\
J_{ij}^{nlin(L_2,b)} &= \eta_B \left(\eta_B \lambda_\chi + \lambda_B\right) \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}}, \\
J_{ij}^{nlin(L_2,\gamma)} &= \eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i, \\
J_{ij}^{nlin(L_3,s)} &= \lambda_\chi \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \left(\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right) \\
&\qquad + \lambda_\chi \left[\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_j, \\
&= J_{ji}^{nlin(L_1,\gamma)},
\end{aligned}$$

[10] As with Equations A.39, the raw results from *Mathematica* need to be simplified somewhat. $J_{ij}^{nlin(L_1,\gamma)}$ starts out as

$$\begin{aligned}
J_{ij}^{nlin(L_1,\gamma)} = \lambda_\chi \Big(&\left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \left(\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i + \left[\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \\
&+ 2\gamma^j \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \Big),
\end{aligned}$$

which can be rearranged to give

$$\begin{aligned}
= \lambda_\chi \Big(&\left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \left(\left(\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}} + \gamma^j \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}}\right) \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} - \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right)\right]_i + \\
&\left[\left(\gamma^j \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} + \boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}^T}\right) \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_i \Big).
\end{aligned}$$

Equation A.16 can then be used to produce the results shown above. Likewise, the raw result for $\boldsymbol{J}^{\boldsymbol{nlin}(L_3,s)}$ is

$$\begin{aligned}
\boldsymbol{J}^{\boldsymbol{nlin}(L_3,s)} = \lambda_\chi \Big(&- \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}}\right]_j + \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_j \\
&+ \left[\boldsymbol{M}^{\boldsymbol{\Sigma},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_j + 2\left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right]_j \gamma^i \\
&+ \left[\boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\right]_j \eta_B \Big),
\end{aligned}$$

which is identical to the result for $J_{ij}^{nlin(L_1,\gamma)}$ with *i* and *j* switched.

$$\begin{aligned} J_{ij}^{nlin(L_3,b)} &= \eta_B \lambda_\chi \left[\boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right]_j, \\ &= J_{ji}^{nlin(L_2,\gamma)}, \text{ and} \\ J_{ij}^{nlin(L_3,\gamma)} &= \left(\lambda_\chi \boldsymbol{s}^T \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{i},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \lambda_G \right) \delta_{ij}. \end{aligned}$$

In practice, calculating these quantities can be simplified by noting that $\boldsymbol{J}^{\boldsymbol{nlin}(\boldsymbol{L_3},\boldsymbol{s})} = \boldsymbol{J}^{\boldsymbol{nlin}(\boldsymbol{L_1},\gamma)^T}$ and $\boldsymbol{J}^{\boldsymbol{nlin}(\boldsymbol{L_3},\boldsymbol{b})} = \boldsymbol{J}^{\boldsymbol{nlin}(\boldsymbol{L_2},\gamma)^T}$, as shown.[11]

A.7.3 Verification of Variational Results for Pseudo-Linear Jacobian

In Section 4.13, variational calculus is used to calculate the Jacobian for the outer loop of the pseudo-linear optimization routine. Since this calculation was completed both by hand and by computer, it provides another opportunity to verify the accuracy of the *Mathematica* utility.

The *Mathematica* utility is capable of handling arbitrary matrices and vectors, so adding the variational quantities $\delta \boldsymbol{s}$ and $\delta \boldsymbol{b}$ was trivial; however, it was necessary to modify the code to ensure that $\delta\gamma$ would be handled like γ instead of a normal vector. (See Footnote 6.)

The parts of χ^2_{mod} (as in Equation 4.65) can be defined using the following commands:

```
chisqData := convertToComponents[(d[•,i] - do[•,i])·(d[i,•] -
   do[i,•]) //.expandTerms];
chisqS := convertToComponents[s[•,j]·s[j,•] //.expandTerms];
chisqB := convertToComponents[β[•,i]·β[i,•] //.expandTerms];
chisqMod := λChi[]chisqData + λS[]chisqS + ηB[]² λB[]chisqB;
```

We start by taking the derivative $\frac{\partial \chi^2_{mod}}{\partial s_j}$ and setting the result equal to zero, generating an equation for the jth component of $\boldsymbol{s}$,

[11] When implementing the deconvolution algorithm (e.g., in MATLAB), computational speed will likely be improved by calculating entire row or column vectors of each sub-matrix instead of calculating individual components one by one. For instance, $\left(\eta_B \lambda_\chi \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \mathbb{1}^{\boldsymbol{j},\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s}\right)$ represents the jth column of $\boldsymbol{J}^{\boldsymbol{nlin}(\boldsymbol{L_2},\gamma)}$. Unfortunately, it is necessary to treat each column separately, as the column number corresponds to the index on the truncated identity matrix $\left(\mathbb{1}^{\boldsymbol{a},\boldsymbol{R}}\right)$.

```
mD[chisqMod, s[j]];

addIndex[%, left,j];

convertToMatrix[%];

dChidS = removeIndices[%];

collapseTerms[%]
```

which can then be generalized to the vector equation:

$$\lambda_\chi \left(-\boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}} + \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} + \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b}\eta_B \right) + \lambda_S \boldsymbol{s} \;=\; 0. \quad \text{(A.41)}$$

Generating the equations for $\boldsymbol{b}$ follows an identical process, except that the first line of Mathematica commands needs to be changed to take a derivative with respect to b_j, and the results in the fourth line should be saved to a different variable:

```
mD[chisqMod, b[j]];
   ⋮
dChidB = removeIndices[%];
   ⋮
```

The *Mathematica* output represents an equation for the j^{th} component of $\boldsymbol{b}$, which can be trivially converted to the corresponding vector equation:

$$\eta_B \left[\lambda_\chi \left(-C_j^{NR^T} \cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}} + \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} \right) + \eta_B(\lambda_B + \lambda_\chi) \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} \right] = 0. \quad \text{(A.42)}$$

As described in Section A.5, variational calculus can be used to estimate the derivative around the solution point. We start by defining one replacement rule to implement the variations shown in Equation A.30:[12]

```
variations = {gamma[a_]→ gamma[a] + δ[gamma[a]], s[i_]→ s[i] +
   δ[s[i]], b[j_]→ b[j] + δ[b[j]]};
```

[12]In contrast to Footnote 7, the variations ($\delta\boldsymbol{s}$, $\delta\boldsymbol{b}$, and $\delta\gamma$) do not require a • in unused index positions. The variations are introduced at a later step in the algorithm, at which time the • is no longer used. Therefore, the variations can be defined as δs[i], etc.

and another to eliminate any terms that are second-order or higher in the variations:

```
firstOrder = {δ[a_] δ[b_]→ 0};
```

A third rule converts the functional `δ[_]` notation used in the prior rules into the `δ_` notation used in Section A.5:

```
fixDeltas = {δ[gamma[a_]]→ δgamma[a], δ[s[i_]]→ δs[i], δ[b[j_]]→
   δb[j]};
```

Finally, we prepare the inputs for the `zeroTest` function, which identifies combinations of terms that are equal zero according to Equations A.47a and A.47b:

```
dChidS = FactorTermsList[Expand[dChidS]][[2]];
dChidB = FactorTermsList[Expand[dChidB]][[2]];
dChidSterms = Sort[removeCoefficients[termList[dChidS]]];
dChidBterms = Sort[removeCoefficients[termList[dChidB]]];
allEqns = {dChidS, dChidB};
allTerms = Union[dChidSterms, dChidBterms];
```

We can generate the first set of equations by taking the $\frac{\partial \chi^2_{mod}}{\partial s_j} = 0$ results, applying the variations, removing higher-order terms, and eliminating terms that sum to zero:

```
mD[chisqMod, s[j]];
addIndex[%, left,j];
%/.variations;
%//.firstOrder;
%//.fixDeltas;
convertToMatrix[%];
removeIndices[%];
zeroTest[Expand[%], allEqns, allTerms];
collapseTerms[%]
```

This results in the equation

$$
\begin{aligned}
\lambda_\chi \Big(& \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{\delta s} - \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \boldsymbol{\delta \Gamma}^{\boldsymbol{R}^T} \cdot \boldsymbol{d}^{\boldsymbol{o},\boldsymbol{NR}} \\
& + \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{\delta \Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} + \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \boldsymbol{\delta \Gamma}^{\boldsymbol{R}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{s} \\
& + \eta_B \left(+ \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{\delta b} + \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}^T} \cdot \boldsymbol{\delta \Gamma}^{\boldsymbol{R}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{b} \right) \Big) + \lambda_S \boldsymbol{\delta s} = 0
\end{aligned}
\tag{A.43}
$$

We can then repeat the process for $\frac{\partial \chi^2_{mod}}{\partial b_j} = 0$, which follows identical steps to those shown above, except that the first line is replaced with

```
mD[chisqMod, b[j]];
```

This result represents a second set of equations,

$$
\begin{aligned}
2\eta_B \Big[& \lambda_\chi \left(+ \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{M}^{\boldsymbol{\Gamma},\boldsymbol{NR}} \cdot \boldsymbol{\delta s} + \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{\delta \Gamma}^{\boldsymbol{R}} \cdot \boldsymbol{M}^{\boldsymbol{\Delta},\boldsymbol{NR}} \cdot \boldsymbol{s} \right) \\
& + \eta_B \left(\lambda_B + \lambda_\chi \right) \boldsymbol{C}^{\boldsymbol{NR}^T} \cdot \boldsymbol{C}^{\boldsymbol{NR}} \cdot \boldsymbol{\delta b} \Big] = 0.
\end{aligned}
\tag{A.44}
$$

These results are equivalent to the hand-derived results shown in Equations A.36.

A.8 Other Cases

A few simple models are also worth calculating, particularly since some are relevant for the linear case in which the sideband-gain parameters are known *a priori*.

A.8.1 Root Finding

A.8.1.1 Simple χ^2

As the simplest case, we start by defining the basic equation for χ^2. We exclude the constraint terms, and we subtract off the baseline terms from $\boldsymbol{d}^{\boldsymbol{NR}}$:

```
chisq = convertToComponents[(d[●,k] - ηB[] β[●,k] -
   do[●,k])·(d[k,●] - (ηB[] β[k,●] - do[k,●])) //.expandTerms];
```

If we assume that the values in γ are fixed, the only variables are the s_i. The linear optimization equations can be calculated by taking the derivatives with respect to s_i and setting the results equal to zero:

```
{eqns, eqnsComp} = calculateJacobian[{chisq/2}, {s[i]}]; eqnsOut =
   Transpose[printElements[eqns]][[2]];
```

Mathematica calculates the derivative, which can be translated into the variables used in this thesis to give the set of linear equations:

$$\boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} = 0 \tag{A.45}$$

The Jacobian can be calculated by taking derivatives of the equations with respect to s_j :

```
{jacobian, jComp} = calculateJacobian[{eqnsComp[[1,1]]}, {s[j]},
   "jacobian"]; jacobianOut = printElements[jacobian];
```

Simplifying the *Mathematica* output gives the Jacobian, although it is unlikely that the linear solver would need it:

$$\boldsymbol{J} = \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} \tag{A.46}$$

A.8.1.2 Linear

We now consider another linear case that includes all of the terms affecting $\boldsymbol{s}$ and $\boldsymbol{b}$ (including the constraint terms), but assume the sideband gains are fixed. We define a simplified version of χ^2_{mod}:

```
chisq = convertToComponents[λChi[](d[•,k] - do[•,k])·(d[k,•] -
   do[k,•]) + λS[]s[•,k]·s[k,•] + λB[]ηB[] β[•,k]·β[k,•]
   //.expandTerms];
```

Taking a derivative with respect to $\boldsymbol{s}$ and $\boldsymbol{b}$

```
{eqns, eqnsComp} = calculateJacobian[{chisq/2}, {s[i], b[i]}];
   eqnsOut = Transpose[printElements[eqns]][[2]];
```

and setting the results equal to zero generates a set of linear, coupled equations:

$$\lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_S \boldsymbol{s} = 0 \tag{A.47a}$$

$$\eta_B \boldsymbol{C}^{NR^T} \cdot \left(\lambda_\chi \left(\boldsymbol{M}^{\Gamma,NR} \cdot \boldsymbol{s} + \eta_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b} - \boldsymbol{d}^{o,NR}\right) + \lambda_B \boldsymbol{C}^{NR} \cdot \boldsymbol{b}\right) = 0 \tag{A.47b}$$

The Jacobian, if needed, can be calculated by taking the derivative of each equation with respect to $\boldsymbol{s}$ and $\boldsymbol{b}$

```
{jacobian, jComp} = calculateJacobian[{eqnsComp[[1,1]],
   eqnsComp[[1,2]]}, {s[j], b[j]}, "jacobian"]; jacobianOut =
   printElements[jacobian];
```

to generate a 2×2 block matrix of the form

$$\boldsymbol{J} = \begin{pmatrix} \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} + \lambda_S \mathbb{1} & \eta_B \lambda_\chi \boldsymbol{M}^{\Gamma,NR^T} \cdot \boldsymbol{C}^{NR} \\ \eta_B \lambda_\chi \boldsymbol{C}^{NR^T} \cdot \boldsymbol{M}^{\Gamma,NR} & \eta_B \left(\lambda_B + \eta_B \lambda_\chi\right) \boldsymbol{C}^{NR^T} \cdot \boldsymbol{C}^{NR} \end{pmatrix}. \tag{A.48a}$$

Appendix B

Listing of *Mathematica* Utility

The following pages contain a listing of a *Mathematica* utility for calculating gradients and Jacobian matrices, as described in Section A.6.

Setup for Gradient Calculator

Must be run before other "Gradient calculator..." notebooks.

Translation to thesis variables

Setup

```
ClearAll[gradientCalculatorSetupComplete];
```

- **Run all self tests**

```
doTests = False;
```

■ Helper functions

■ Description

uniqueFromRoot[x]
Creates a unique variable, using the "root" from *x* by using Unique[ ] to add a unique $nnn suffix. If *x* already contains the $nnn suffix, it will be stripped off first.
Examples:

i1 → i1$14
i1$142 → i1$447

independentListQ[list1, list2]
If all elements of list1 and list2 are distinct, this function will return True.

replaceWithUnique[x]
Creates rule to replace *x* with unique variables of the same root. *x* can be either a list or a single value.
Example, replaceWithUnique[{i1, i2}] will return {i1 → i1$1017, i2 → i2$1018}

convertToRoots[x]
Takes a list of indices and produces the corresponding roots by removing the $nnn index (if present)

allAtomQ[x]
True if x a list containing only atomic elements. x should be provided as a list, even if just a single value

onesAtomQ[x]
True if x a list containing only the following elements

atomic expressions
ones[_] matrices

x should be sent as a list, even if just a single value

nullQ[x]
True if x is Null

validMatrixQ[matrix]
True if matrix appears to be a valid matrix for the purposes of this workbook. Function tests for the following criteria:

onesAtomQ[{matrix}] = True (matrix is either an atomic expression or ones[_])
matrix is not mSum, Plus, Times, Power, or CenterDot

singleEntryQ[list, x]
True if *x*is atomic and appears once, and only once, in list

allSingleEntryQ[list]
True if each element in the list is atomic and appears once, and only once, in the list.

allFreeQ[x, list]
True if x is free of each element in list (applies FreeQ[x, ...] to each list element)

extractSign[x]
Returns any explicit sign attached to *x*, based on *Mathematica* documentation, which indicates that $-x$ = Minus[*x*] is saved as Times[−1, *x*] internally.

$x \to +1$
$0 \to 0$
$-x \to -1$

termList[x]
Returns a list of all the terms in *x* with leading negative signs removed. Expand[*x*] is run on the input to expand all factors. If needed, those signs could be returned in a separate array (note yet implemented).

removeCoefficients[list]
Removes numeric coefficients (inlcuding leading negative signs) from each term in list. Pure numbers are removed from list.

removeEntry[list, x]
Removes x from list. Only works on the highest level of the list, so variables in array notation (e.g. *x*[*i*])are treated as distinct from the same variable without the added brackets.
Examples:

removeEntry[{*x*, *x*[*i*], *i*}, *i*] = {*x*, *x*[*i*]}
removeEntry[{*x*, *x*[*i*], *i*}, *x*] = {*x*[*i*], *i*}
removeEntry[{*x*, *x*[*i*], *i*}, *x*[*i*]] = {*x*, *i*}

nullComponent
Defines a symbol to represent an empty index position. Vectors should be represented as either x[i, <nullComponent>] or x[<nullComponent>, i], which allows row and column vectors to be distinguished when they are being placed into mSum[ ] functions.

The variable can be referenced by saying "null component" and is currently set ●, which can be entered using {Esc}gci{Esc} or "gray circle". *This should not be changed unless absolutely necessary, as ● is probably hard-coded into lots of equations and comments.*

indexToSum[i1a, i1b, i2a, i2b]
Accepts two pairs of indices from two matrices. The first pair represents the left and right indices from matrix 1, while the second pair represents the indices of matrix 2. The function checks to see if one of the indices from matrix 1 matches one from matrix 2. If so, it returns a value of {iSum, {match1, match2}}, where iSum lists the index that matches and match1, match2 give the position of iSum in each pair. (Note that match1 and match2 will have values of either 1 or 2.) Indices equal to nullComponent are ignored. The function returns unevaluated if there are no matching indices or more than one matching index.
Examples:

indexToSum[a, b, b, c] = {b, {2, 1}} b matches, occupying position 2 in matrix 1 and position 1 in matrix 2
indexToSum[●, b, a, b] = {b, {2, 2}} b matches, occupying position 2 in matrix 1 and position 2 in matrix 2
indexToSum[●, b, a, ●] = indexToSum[●, b, a, ●] Although the ●'s match, they are ignored since they match the value of nullComponent
indexToSum[a, b, c, d] = indexToSum[a, b, c, d] No matches
indexToSum[a, b, b, a] = indexToSum[a, b, b, a] More than one match
indexToSum[a, b, a, a] = indexToSum[a, b, a, a] Multiple matches
indexToSum[a, a, c, d] = indexToSum[a, a, c, d] Matching indices are on the same matrix

indexToSumTestQ[iSum, iA, iB]
Helper function for indexToSum[ ]. iSum is the matching index on two matrices, with iA and iB representing the other two indices. This function checks to make sure that the following conditions are met

iSum is not null component
iA and iB are either distinct or equal to nullComponent
iA and iB are distinct from iSum

outputIndices[matrix, i1, i2]
Returns matrix with the appropriate indices:

i1 = nullComponent or Null ⇒ matrix[i2]
i2 = nullComponent or Null ⇒ matrix[i1]
i1 and i2 ≠ nullComponent or Null ⇒ matrix[i1, i2]

matchingIndexQ[i1, i2, i3, i4]
Returns True if indexToSum[ ] finds that there is a valid set of indices that can be summed on the two matrices; False otherwise.

error[message]
Prints message and generates an Abort[ ] command

releaseHoldAll[x]
Repeatedly applies ReleaseHold[ ] until all Hold[ ]'s have been cleared.

validDerivativeQ[f, x, summedIndices]
Determines whether differentiating *f* wrt *x* meets certain validity tests (see discussion in notes for mD[ ] function)

▪ Function Definitions

```
ClearAll[uniqueFromRoot, independentListQ, replaceWithUnique, convertToRoots, allAtomQ,
  onesAtomQ, nullQ, validMatrixQ, singleEntryQ, allSingleEntryQ, allFreeQ, extractSign,
  termList, removeCoefficients, removeEntry, nullComponent, indexToSum, indexToSumTestQ,
  matchingIndexQ, outputIndices, error, releaseHoldAll, validDerivativeQ];
uniqueFromRoot[x_] := Unique[Symbol[Part[StringSplit[ToString[x], "$"], 1]]];
independentListQ[list1_, list2_] := Apply[And, Map[Function[FreeQ[list1, #1]], list2]] ;
replaceWithUnique[x_] := Map[Function[#1 → uniqueFromRoot[#1]], Flatten[{x}]];
convertToRoots[x_] :=
  Map[Function[Symbol[Part[StringSplit[ToString[#1], "$"], 1]]], x] /; ListQ[x];
allAtomQ[x_] := Apply[And, Map[Function[AtomQ[#1]], x]] /; ListQ[x];
onesAtomQ[x_] := Apply[And, Map[Function[ MatchQ[#1, ones[_]] || AtomQ[#1]], x]] /; ListQ[x];
nullQ[x_] := SameQ[x, Null];
validMatrixQ[matrix_] :=
  onesAtomQ[{matrix}] && allFreeQ[matrix, {Plus, Times, Power, mSum, CenterDot}];
singleEntryQ[list_, x_] := TrueQ[AtomQ[x] && Length[Flatten[Position[list, x]]] == 1];
allFreeQ[x_, list_] := Apply[And, Map[Function[FreeQ[x, #1]], list]] /; ListQ[list];
allSingleEntryQ[list_] := Apply[And, Map[Function[singleEntryQ[list, #1]], list]] ;
removeEntry[list_, x_] := Drop[list, Flatten[Position[list, x, 1]]];
nullComponent = ●;

extractSign[Times[y_, ___]] := Sign[y] /; NumberQ[y];
extractSign[y_] := Sign[y] /; NumberQ[y];
```

```
indexToSum[iSum_, iA_, iSum_, iB_] := {iSum, {1, 1}} /; indexToSumTestQ[iSum, iA, iB];
indexToSum[iSum_, iA_, iB_, iSum_] := {iSum, {1, 2}} /; indexToSumTestQ[iSum, iA, iB];
indexToSum[iA_, iSum_, iSum_, iB_] := {iSum, {2, 1}} /; indexToSumTestQ[iSum, iA, iB];
indexToSum[iA_, iSum_, iB_, iSum_] := {iSum, {2, 2}} /; indexToSumTestQ[iSum, iA, iB];
indexToSumTestQ[iSum_, iA_, iB_] := Not[iSum === Null || iSum === nullComponent] &&
   (Not[SameQ[iA, iB]] || iA === Null || iA === nullComponent) &&
   Not[iSum === iA] && Not[iSum === iB];
matchingIndexQ[i1_, i2_, i3_, i4_] := MatchQ[indexToSum[i1, i2, i3, i4], {_, {_, _}}];

termList[x_] := Module[
   {i, terms = {}, xLength, xFull},
   xFull = Expand[x];
   If[Head[xFull] === Plus,
    { (* Has multiple terms *)
     xLength = Length[xFull];
     For[i = 1, i ≤ xLength, i++, AppendTo[terms, xFull⟦i⟧]];
    },
    { (* Just one term *)
     terms = {x};
    }];
   Return[terms];
  ];

removeCoefficients[x_] := Module[
   {i, output = {}, xLength = Length[x]},
   If[! ListQ[x], error["removeCoefficients: Input must be list."]];
   For[i = 1, i ≤ xLength, i++, AppendTo[output, Part[FactorTermsList[x⟦i⟧], 2]]];
   Return[output];
  ];


outputIndices[matrix_, i1_, i2_] := Module[
   {output},
   If[(i1 === nullComponent || i1 === Null) && (i2 === nullComponent || i2 === Null),
    error["Matrix cannot have two null indices."]];
   If[(i1 === nullComponent || i1 === Null),
    {output = matrix[i2]},
    {
     If[(i2 === nullComponent || i2 === Null),
       {output = matrix[i1]},
       {output = matrix[i1, i2]}
      ];
    }];
   Return[output];
  ];

error[message___] := Module[
   {},
   Print[message];
   Abort[];
  ];

releaseHoldAll[x_] := x /; FreeQ[x, Hold];
```

```
releaseHoldAll[x_] := releaseHoldAll[ReleaseHold[x]] /; ! FreeQ[x, Hold];

validDerivativeQ[f_, x_, indices_] := Module[
   {result, summedIndices, xRoot, xTest, currentIndex, j, k, y},

   summedIndices = indices[[1]];

   result = True;
   result = result && FreeQ[indices, x];
   If[MatchQ[x, _[_]],
    {
     result = result && ! MatchQ[x, _[_][_]];
     xRoot = Part[Cases[{x}, xRoot_[_] :> xRoot], 1];
     currentIndex = Part[Cases[{x}, _[k_] :> k], 1];
     result = result && FreeQ[summedIndices, currentIndex];
     result = result && FreeQ[f /. xRoot[_] :> -1, xRoot];
    }
   ];
   result = result && FreeQ[x, SuperDagger];
   result = result && FreeQ[f, SuperDagger];
   result = result && FreeQ[f, x[_]];
   Return[result];
  ];
```

■ Define · (matrix multiplication) and convertToComponents[ ]

■ Description

Matrix multiplication should be written using a center dot:

$A[i, j] \cdot B[j, k] \cdot x[k, ●]$

where A and x can be either matrices (two indices) or vectors (one index). Within a given line (function call), the indices must line up appropriately so that the right index of one matrix matches the left index of the next, and each index is only used once (free) or twice (summed).

To differentiate between column and row vectors, a vector should have a single index in the first or second place, respectively, with the blank spot filled with ● ({Esc}gci{Esc} or \ [GrayCircle]). Symbol can be changed in all function definitions using the nullComponent variable. The ● will be used to determine whether the mSum should have free indices on the left or right, and will then be discarded. Vector components inside the mSum will NOT contain the ●.

Enter the ones[ ] matrix as ones[a][i, j], where a indicates which matrix is being used (matching the terminology in the lab notebook), and i, j represent the row and column (also see notes below).

These can then be converted to a sum over multiplied components using the convertToComponents[ ] function:

convertToComponents[$A[i, j] \cdot B[j, k] \cdot x[k, ●]$] ⇒

$$\sum_{j,k} A_{i\,j} B_{j\,k} x_k$$

which is represented as mSum[$A[i, j]\, B[j, k]\, x[k]$, {{j, k}, {i}, { }, {}}]

convertToComponents[$A[i, j] \cdot$ ones[a][j, k] $\cdot$ gamma[b] $\cdot$ ones[b][k, m]] ⇒

$$\sum_{b,j,k} A_{i\,j}\ \mathbb{1}^{a}{}_{j\,k}\,\gamma^{b}\,\mathbb{1}^{b}{}_{k\,m}$$

which is represented as mSum[A[i, j] ones[a][j, k] gamma[b] ones[b][k, m], {{b, j, k}, {i}, {m}, {a}}]
The function works right-to-left, so an error to the right will prevent all leftward matrices from being evaluated.

The results are expressed using the matrix sum function (mSum[ ]) as follows:
mSum[components, { {summedIndices}, {leftFreeIndices}, {rightFreeIndices}, {onesFreeIndices} }]
Free indices will match the values used in the initial function call, while summed indices will be replaced with unique variable names. For clarity, the unique names are based on the value specified by the user (e.g. if the user label an index i1, the unique replacement will be i1$nnn). The function automatically renames some indices as needed, so one equation can be substituted into another without problem. When substituting one equation using mSum[ ] into another, the free indices must be carefully considered, as free indices will be automatically summed whenever a right and left free index match

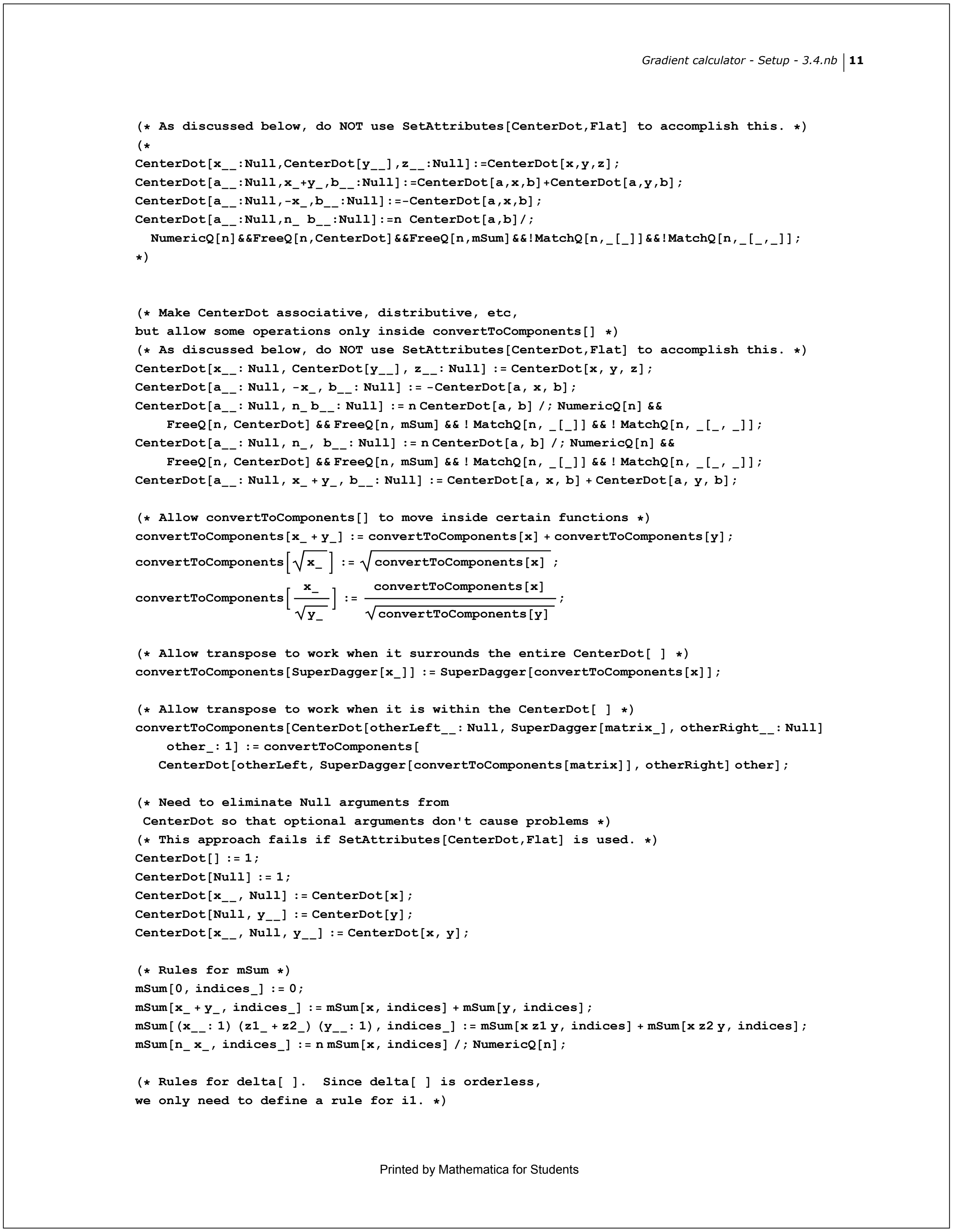
Notes:
- The only sum allowed over the a index of ones[a][i, j] is gamma[a] ones[a][i, j]. If other combinations are needed, additional functions will need to be added.
- If two different mSum functions are combined, they must have a single matching right and left free index.
- The function is not perfect, so the following rules must be observed:
  - When defining a variable (e.g. m1 = A[i, j] B[j, k]), it should be defined as
    m1[i_, k_] := Module[{j}, convertToComponents[A[i, j] B[j, k]]]
    The Module[ ] command is necessary to make sure that the j variable won't conflict with the values of i, k. (Without it, m[i, j] would generate an error.)
- If indices match properly, matrices can be multiplied, even if the multiplication is not ordered correctly. For instance, both $x^{\dagger}\cdot x$ and $x\cdot x^{\dagger}$ will produce the same (scalar) value.

Entering functions:
Mathematica represents this function as CenterDot[ ], and the symbol can be entered with the voice command "matrix multiply" or using {Esc}.{Esc}. The mSum[ ] function can be enetered using voice command "matrix sum" or "M. sum".

■ Function Definition

```
ClearAll[CenterDot, convertToComponents, mSum, delta];

(* Seed - place each matrix in an mSum[ ] function and let
  combining rules for the mSum[ ]'s complete the multiplication. *)
(* Need to pull mSum out of CenterDot to avoid infinite recursion *)
(* For unknown reasons,
this pattern only works if other_:1 is used rather than other___ *)
(* Errors in this definition have identifier convertToComponents(1) *)
convertToComponents[CenterDot[otherLeft___ : Null, matrix1_[i1_, i2_]] other_: 1] := Module[
    {leftFreeIndices, rightFreeIndices, onesFreeIndices, newIndices},
    If[! allAtomQ[{i1, i2}],
     error["Nonatomic index in convertToComponents(1): \n\tmatrix1= ",
      matrix1, "\n\ti1 = ", i1, "\n\ti2 = ", i2]];
    If[! onesAtomQ[{matrix1}], error["Invalid matrix in convertToComponents(1): ",
      matrix1]];
    newIndices = {};
    If[i1 === nullComponent,
     leftFreeIndices = {},
     leftFreeIndices = {i1}
    ];
    If[i2 === nullComponent,
     rightFreeIndices = {},
     rightFreeIndices = {i2}
    ];
    onesFreeIndices = Cases[{matrix1}, ones[a_] :> a];
    If[! allSingleEntryQ[Flatten[{leftFreeIndices, rightFreeIndices, onesFreeIndices}]],
     error["Unexpected repeated entry in index components in convertToComponents(1): ",
      {leftFreeIndices, rightFreeIndices, onesFreeIndices}]];
    convertToComponents[CenterDot[otherLeft] mSum[outputIndices[matrix1, i1, i2],
       { {}, leftFreeIndices, rightFreeIndices, onesFreeIndices }] other]
   ] /; validMatrixQ[matrix1];

(* Also need a rule for isolated matrices (not part of a CenterDot[ ]) *)
(* Errors in this definition have identifier convertToComponents(1a) *)
convertToComponents[matrix1_[i1_, i2_]] := Module[
    {leftFreeIndices, rightFreeIndices, onesFreeIndices},
    If[! allAtomQ[{i1, i2}],
     error["Nonatomic index in convertToComponents(1a): \n\tmatrix1 = ",
      matrix1, "\n\ti1 = ", i1, "\n\ti2 = ", i2]];
    If[! onesAtomQ[{matrix1}], error["Invalid matrix in convertToComponents(1a): ",
      matrix1]];
    If[i1 === nullComponent,
     leftFreeIndices = {},
     leftFreeIndices = {i1}
    ];
    If[i2 === nullComponent,
     rightFreeIndices = {},
     rightFreeIndices = {i2}
    ];
    onesFreeIndices = Cases[{matrix1}, ones[a_] :> a];
    If[! allSingleEntryQ[Flatten[{leftFreeIndices, rightFreeIndices, onesFreeIndices}]],
     error["Unexpected repeated entry in index components in convertToComponents(1a): ",
      {leftFreeIndices, rightFreeIndices, onesFreeIndices}]];
    convertToComponents[mSum[outputIndices[matrix1, i1, i2],
      { {}, leftFreeIndices, rightFreeIndices, onesFreeIndices }]]
   ] /; validMatrixQ[matrix1] && FreeQ[{matrix1, i1, i2}, mSum];

(* Convert isolated gamma's (not part of a CenterDot[ ]) *)
(* Errors in this definition have identifier convertToComponents(1b) *)
convertToComponents[gamma[a_]] := Module[
    {},
    If[! allAtomQ[{a}],
     error["Nonatomic index in convertToComponents(1b): \n\tgamma\n\ta = ", a]];
    convertToComponents[mSum[gamma[a], { {}, {}, {}, {a} }]]
   ] /; FreeQ[a, mSum];

(* Need to specifically address gamma[a]
 since it does not have matrix or vector indices *)
(* Errors in this definition have identifier convertToComponents(2) *)
convertToComponents[CenterDot[otherLeft___ : Null, gamma[a_], otherRight___ : Null]
```

```
     mSum[ones[a_][i1_, i2_] mSumOther_: 1, indices_] other_: 1] := Module[
    {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices, aSum},
    {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices} = indices;
    If[! AtomQ[a], error["Nonatomic index in convertToComponents(2): ", a]];
    If[! MemberQ[onesFreeIndices, a],
     error["Index not included in onesFreeIndices in convertToComponents(2): ", a]];
    If[MemberQ[summedIndices, a] || MemberQ[leftFreeIndices, a] || MemberQ[rightFreeIndices, a],
     error["Repeated index in convertToComponents(2): ", a]];
    If[! FreeQ[{otherLeft, otherRight, mSumOther, other}, _[a] | _[a, _] | _[_, a]],
     error["Unexpected repetition of index (", a,
      ") in other variables in convertToComponents(2): ",
      "\n otherLeft: ", otherLeft, "\notherRight: ", otherRight,
      "\nmSumOther: ", mSumOther, "\nother: ", other]];
    onesFreeIndices = removeEntry[onesFreeIndices, a];
    aSum = uniqueFromRoot[a];
    AppendTo[summedIndices, aSum];
    convertToComponents[
     CenterDot[otherLeft, otherRight] mSum[gamma[aSum] ones[aSum][i1, i2] mSumOther,
       {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices} ] other]
   ];

(* Merge mSum[]'s when they have appropriate matching
 free indices.  In order to have a valid multiplication, *)
(* the matching index MUST appear in rightFreeIndices for the first mSum and
 leftFreeIndices for the second mSum.  *)(* The individual matrix components
 can have any combination of placements (due to transpose operations), *)
(* but the free indices in the mSum's must align as described. Moroever,
there should never be a situation in which *)
(* there is more than one leftFreeIndices or rightFreeIndices *)
(* Errors in this definition have identifier convertToComponents(3) *)
convertToComponents[
   mSum[matrix1_[i1_, i2_: Null] other1_: 1, {summedIndices1_, leftFreeIndices1_,
      {iSum_}, onesFreeIndices1_}] mSum[matrix2_[i3_, i4_: Null] other2_: 1 ,
     {summedIndices2_, {iSum_}, rightFreeIndices2_, onesFreeIndices2_}] other__: 1] := Module[
    {iOut1, iOut2, iOut3, iOut4, summedIndicesOut, leftFreeIndicesOut, rightFreeIndicesOut,
     onesFreeIndicesOut, matchResults, newIndices, iSumRules, iSumPosition1, iSumPosition2},

    matchResults = indexToSum[i1, i2, i3, i4];
    If[! (iSum == Part[matchResults, 1]),
     error["Invalid iSum in convertToComponents(3): ", iSum]];
    iSumPosition1 = Part[matchResults, 2, 1];
    iSumPosition2 = Part[matchResults, 2, 2];
    iSumRules = replaceWithUnique[iSum];

    If[Part[{{i1, i2}, {i3, i4}}, 1, iSumPosition1] ≠ iSum ||
      Part[{{i1, i2}, {i3, i4}}, 2, iSumPosition2] ≠ iSum,
     error["iSum does not appear correct matrix indices in convertToComponents(3): ", iSum]
    ];
    If[Part[{{i1, i2}, {i3, i4}}, 1, 3 - iSumPosition1] == iSum ||
      Part[{{i1, i2}, {i3, i4}}, 2, 3 - iSumPosition2] == iSum,
     error["iSum appears in additional matrix indices in convertToComponents(3): ", iSum]
    ];
    (* iSum should not appear in any of the other index lists or in any other variables *)
    If[! FreeQ[{summedIndices1, leftFreeIndices1, onesFreeIndices1,
        summedIndices2, rightFreeIndices2, onesFreeIndices2}, iSum],
     error["iSum appears in unexpected places in index lists in convertToComponents(3): ",
      iSum]
    ];
    If[
     ! FreeQ[{other, other1, other2, matrix1, matrix2}, _[iSum] | _[iSum, _] | _[_, iSum]], error[
      "iSum appears in unexpected places in other variables in convertToComponents(3): ",
      iSum]
    ];

    (* Make sure there are no other free indices that match *)
    If[! singleEntryQ[
       Join[leftFreeIndices1, onesFreeIndices1, rightFreeIndices2, onesFreeIndices2]],
     error["Free index lists are not independent in convertToComponents(3)."]
    ];

    (* If any of the summedIndices match, need to replace them *)
    If[! independentListQ[summedIndices1, summedIndices2],
     summedIndicesOut = Join[summedIndices1, ReplaceAll[summedIndices2,
        replaceWithUnique[Intersection[summedIndices1, summedIndices2]]]],
     summedIndicesOut = Join[summedIndices1, summedIndices2]
    ];
    AppendTo[summedIndicesOut, iSum /. iSumRules];
    (*{iOut1,iOut2,iOut3,iOut4}={i1,i2,i3,i4}/.iSumRules;*)
    leftFreeIndicesOut = leftFreeIndices1;
    rightFreeIndicesOut = rightFreeIndices2;
    onesFreeIndicesOut = Sort[Join[onesFreeIndices1, onesFreeIndices2]];
    convertToComponents[other mSum[ReplaceAll[
        other1 outputIndices[matrix1, i1, i2] outputIndices[matrix2, i3, i4] other2, iSumRules],
       {summedIndicesOut, leftFreeIndicesOut, rightFreeIndicesOut, onesFreeIndicesOut}]]
   ](* end module *) /; matchingIndexQ[i1, i2, i3, i4] && iSum ==
     Part[indexToSum[i1, i2, i3, i4], 1] && validMatrixQ[matrix1] && validMatrixQ[matrix2];

(* Remove mSum from convertToComponents once
 all factors have been combined into a single sum. *)
convertToComponents[
   mSum[x_, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]
    other_: 1] := other mSum[x, {Sort[summedIndices], Sort[leftFreeIndices],
      Sort[rightFreeIndices], Sort[onesFreeIndices]}] /; NumericQ[other];

(* Remove mSum's from CenterDot *)
convertToComponents[CenterDot[otherLeft__: Null, mSum[x__], otherRight__: Null] other_: 1] :=
  convertToComponents[CenterDot[otherLeft, otherRight] mSum[x] other];

(* Pull numeric quantities out of convertToComponents *)
convertToComponents[other_ matrices_: 1] := other convertToComponents[matrices] /;
   NumericQ[other] && FreeQ[other, CenterDot] && FreeQ[other, mSum];

(* Modified to only work inside convertToComponents[]
 so that similar terms could be gathered together. *)
(* Make CenterDot associative, distributive, etc. *)
```



```
(* As discussed below, do NOT use SetAttributes[CenterDot,Flat] to accomplish this. *)
(*
CenterDot[x__:Null,CenterDot[y__],z__:Null]:=CenterDot[x,y,z];
CenterDot[a__:Null,x_+y_,b__:Null]:=CenterDot[a,x,b]+CenterDot[a,y,b];
CenterDot[a__:Null,-x_,b__:Null]:=-CenterDot[a,x,b];
CenterDot[a__:Null,n_ b__:Null]:=n CenterDot[a,b]/;
  NumericQ[n]&&FreeQ[n,CenterDot]&&FreeQ[n,mSum]&&!MatchQ[n,_[_]]&&!MatchQ[n,_[_,_]];
*)


(* Make CenterDot associative, distributive, etc,
but allow some operations only inside convertToComponents[] *)
(* As discussed below, do NOT use SetAttributes[CenterDot,Flat] to accomplish this. *)
CenterDot[x__: Null, CenterDot[y__], z__: Null] := CenterDot[x, y, z];
CenterDot[a__: Null, -x_, b__: Null] := -CenterDot[a, x, b];
CenterDot[a__: Null, n_ b__: Null] := n CenterDot[a, b] /; NumericQ[n] &&
    FreeQ[n, CenterDot] && FreeQ[n, mSum] && ! MatchQ[n, _[_]] && ! MatchQ[n, _[_, _]];
CenterDot[a__: Null, n_, b__: Null] := n CenterDot[a, b] /; NumericQ[n] &&
    FreeQ[n, CenterDot] && FreeQ[n, mSum] && ! MatchQ[n, _[_]] && ! MatchQ[n, _[_, _]];
CenterDot[a__: Null, x_ + y_, b__: Null] := CenterDot[a, x, b] + CenterDot[a, y, b];

(* Allow convertToComponents[] to move inside certain functions *)
convertToComponents[x_ + y_] := convertToComponents[x] + convertToComponents[y];
convertToComponents[√x_] := √convertToComponents[x] ;
                      x_          convertToComponents[x]
convertToComponents[ ---- ] := ---------------------------;
                     √y_        √convertToComponents[y]

(* Allow transpose to work when it surrounds the entire CenterDot[ ] *)
convertToComponents[SuperDagger[x_]] := SuperDagger[convertToComponents[x]];

(* Allow transpose to work when it is within the CenterDot[ ] *)
convertToComponents[CenterDot[otherLeft__: Null, SuperDagger[matrix_], otherRight__: Null]
    other_: 1] := convertToComponents[
   CenterDot[otherLeft, SuperDagger[convertToComponents[matrix]], otherRight] other];

(* Need to eliminate Null arguments from
 CenterDot so that optional arguments don't cause problems *)
(* This approach fails if SetAttributes[CenterDot,Flat] is used. *)
CenterDot[] := 1;
CenterDot[Null] := 1;
CenterDot[x__, Null] := CenterDot[x];
CenterDot[Null, y__] := CenterDot[y];
CenterDot[x__, Null, y__] := CenterDot[x, y];

(* Rules for mSum *)
mSum[0, indices_] := 0;
mSum[x_ + y_, indices_] := mSum[x, indices] + mSum[y, indices];
mSum[(x__: 1) (z1_ + z2_) (y__: 1), indices_] := mSum[x z1 y, indices] + mSum[x z2 y, indices];
mSum[n_ x_, indices_] := n mSum[x, indices] /; NumericQ[n];

(* Rules for delta[ ].  Since delta[ ] is orderless,
we only need to define a rule for i1. *)
```

```
(* This function DOES NOT fix the free indices after
 substituting for a summed index. See notes with mD[ ]. *)
SetAttributes[delta, Orderless];
delta[i_, i_] := 1;
mSum[ delta[i1_, i2_] other_: 1,
   {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}] :=
  mSum[other /. i1 :→ i2, {removeEntry[summedIndices, i1], leftFreeIndices,
     rightFreeIndices, onesFreeIndices}] /; MemberQ[summedIndices, i1] &&
    FreeQ[{leftFreeIndices, rightFreeIndices, onesFreeIndices}, i2] &&
    FreeQ[{leftFreeIndices, rightFreeIndices, onesFreeIndices}, i1];

(* Rules for ones matrices *)
(* Bad idea to include them here; ruins ability to put deltaG matrices back together *)
(* mSum[(ones[a_][i1_,iSum_]ones[b_][iSum_,i2_]other_:1)|
    (ones[a_][i1_,iSum_]ones[b_][i2_,iSum_] other_:1)|
    (ones[a_][iSum_,i1_]ones[b_][iSum_,i2_] other_:1)  ,
   {summedIndices_,leftFreeIndices_,rightFreeIndices_,onesFreeIndices_}]:=
  mSum[delta[a,b]ones[a][i1,i2] other,{removeEntry[summedIndices,iSum],leftFreeIndices,
     rightFreeIndices,onesFreeIndices}]/;MemberQ[summedIndices,iSum]; *)
```

▪ Tests

▪ Define † (Transpose)

▪ Description

SuperDagger[ ] = Ctrl+6 {Esc}dg{Esc} or Ctrl+6 \ [Dagger]
Must be put AFTER index specification: $A[i, j]^\dagger$, which **represents the (*i*, *j*) component of $A^\dagger$**. Thus, the correct way to write a matrix multiplication including a transpose is

$A[j, k] \cdot B[k, m]^\dagger \cdot C[m, p]$ (i.e. the left-hand index on the transposed matrix should line up with the right-hand index on the matrix to the left, etc.)
Equivalently, $B[k, m]^\dagger$ has leftFreeIndices $= \{k\}$ and rightFreeIndices = $\{m\}$

convertToComponents[ ] deduces left and rght free indices based on the position of the index in the matrix. Thus, for $A[i, j]$, i is assumed to be the left index and j the right one, and they can only be summed against matching indices on the right and left, respectively. This approach will fail unless the transpose operator waits to act until the matrices have been assigned indices within the mSum[ ] function.

Consider how the transpose operator works on a matrix:

Let $M = A \cdot B \cdot C$ so that $M_{i\,j} = A_{i\,k}\, B_{k\,l}\, C_{l\,j}$

We know that $\left(M^\dagger\right)_{i\,j} = M_{j\,i}$so we can just use the formula shown above to determine $\left(M^\dagger\right)_{i\,j} = A_{j\,k}\, B_{k\,l}\, C_{l\,i}$. (Just swap the two free indices on components.)

However, we also know that $M^\dagger = (A \cdot B \cdot C)^\dagger = C^\dagger \cdot B^\dagger \cdot A^\dagger$, giving us $\left(M^\dagger\right)_{i\,j} = C^\dagger{}_{i\,k}\, B^\dagger{}_{k\,l}\, A^\dagger{}_{l\,j} = A_{j\,l}\, B_{l\,k}\, C_{k\,i}$.

Since the summed indices are just dummy variables, these two methods are consistent.

Note, however, that the following approach does NOT work: $\left(M_{i\,j}\right)^\dagger = \left(A_{i\,k}\, B_{k\,l}\, C_{l\,j}\right)^\dagger \neq C^\dagger{}_{l\,j}\, B^\dagger{}_{k\,l}\, A^\dagger{}_{i\,k} = A_{k\,i}\, B_{l\,k}\, C_{j\,l}$

Also note that this is the matrix component for $\left(M^\dagger\right)_{i\,j}$, so it should be multiplied from the left by i, even though i is a right-hand index on the individual components.

Thus, to form the transpose of a matrix element, the two free indices should be swapped on the components, but leftFreeIndices and rightFreeIndices stay the same.
Now consider operation on a vector:

Let $y = A \cdot B \cdot x$ so that $y_i = y_i = A_{i\,k}\, B_{k\,l}\, x_l$

The transpose operator just changes y from a column to a row vector, but the value of each component is the same: $\left(y^\dagger\right)_i = y_i$.

This can be confirmed using the alternate identity $y^\dagger = (A \cdot B \cdot x)^\dagger = x^\dagger \cdot B^\dagger \cdot A^\dagger$

We can expand this to find $\left(y^\dagger\right)_i = x_j \cdot B^\dagger{}_{j\,k} \cdot A^\dagger{}_{k\,i} = A_{i\,k}\, B_{k\,j}\, x_j$

As one would expect, the values of y_i and $\left(y^\dagger\right)_i$ are exactly the same (no change in matrix components), but they multiply to the right and left respectively.

Thus, taking the transpose of a vector is merely a bookkeeping operation that swaps leftFreeIndices ↔ rightFreeIndices without changing the components.

■ Function Definition

```
ClearAll[SuperDagger];

SuperDagger[mSum[components_, {summedIndices_, {left_}, {right_}, onesFreeIndices_}]] :=
  mSum[components /. {left :→ right, right :→ left},
    {summedIndices, {left}, {right}, onesFreeIndices}] /; AtomQ[left] && AtomQ[right] &&
    Count[components, left, Infinity] == 1 && Count[components, right, Infinity] == 1;
SuperDagger[mSum[components_ , {summedIndices_, {left_}, {}, onesFreeIndices_}]] :=
  mSum[components, {summedIndices, {}, {left}, onesFreeIndices}] /;
   AtomQ[left] && Count[components, left, Infinity] == 1;
SuperDagger[mSum[components_, {summedIndices_, {}, {right_}, onesFreeIndices_}]] :=
  mSum[components, {summedIndices, {right}, {}, onesFreeIndices}] /;
   AtomQ[right] && Count[components, right, Infinity] == 1;
SuperDagger[x_ y__: 1] := x SuperDagger[y] /; NumericQ[x];
SuperDagger[x_] := x /; NumericQ[x];
SuperDagger[x_ + y_] := SuperDagger[x] + SuperDagger[y];
```

■ Tests

■ Define convertToMatrix[ ]

■ Description

Comverts components in mSum[ ] notation back to matrices in CenterDot[ ] notation.

Cases:

two matrices match indices -> add dot and remove from summedIndices
gamma's - handle them last; treat δgamma's equivalently

Internally uses superscripted T† (T dagger) to represent a transpose.

▪ Function Definition

```
ClearAll[convertToMatrix];

convertToMatrix[n_] := n /; NumericQ[n];
convertToMatrix[n_ x_] := n convertToMatrix[x] /; NumericQ[n];
convertToMatrix[x_ + y_] := convertToMatrix[x] + convertToMatrix[y];
convertToMatrix[x_/√y_] := convertToMatrix[x]/√convertToMatrix[y];
convertToMatrix[√x_] := √convertToMatrix[x] ;

(* *** INITIALIZATION *** *)
(* Put one of the end matrices into CenterDot[ ]
 structure by matching against left and right free indices. *)
(* If there are no free indices, than expression must include 2 vectors at either end,
so start with one of those. *)
(* Express in CenterDot notation such that
 leftmost and rightmost indices are the free indices *)

(* Seed statements;
the FreeQ[_, CenterDot]'s ensure that only one condition will be used. *)
(* Find a matrix with one index matching the corresponding
 left or right free index and put it in CenterDot[ ]. *)
convertToMatrix[mSum[matrix_[i1_, i2_] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[matrix[i1, i2]] If[! nullQ[other], other, 1],
     {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix] && FreeQ[other, CenterDot] &&
    (MemberQ[leftFreeIndices, i1] || MemberQ[rightFreeIndices, i2]);

(* Find a matrix such that the transpose has one index
 matching the left or right free index and put it in CenterDot[ ]. *)
convertToMatrix[mSum[matrix_[i1_, i2_] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[matrix[i2, i1]^T†] If[! nullQ[other], other, 1],
     {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix] && FreeQ[other, CenterDot] &&
    (MemberQ[leftFreeIndices, i2] || MemberQ[rightFreeIndices, i1]);

(* Find a vector with an index matching the left free index and put it in CenterDot[ ]. *)
convertToMatrix[mSum[vector_[i1_] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[vector[i1]] If[! nullQ[other], other, 1],
     {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[vector] && FreeQ[other, CenterDot] && MemberQ[leftFreeIndices, i1] &&
    ! (vector === gamma) && ! (vector === δgamma);

(* Find a vector with an index matching the
 right free index and put its transpose in CenterDot[ ]. *)
convertToMatrix[mSum[vector_[i1_] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[vector[i1]^T†] If[! nullQ[other], other, 1],
     {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[vector] && FreeQ[other, CenterDot] && MemberQ[rightFreeIndices, i1] &&
    ! (vector === gamma) && ! (vector === δgamma);

(* Find a matrix with an index matching one
 of the ones free indices and put it in CenterDot[ ]. *)
(* This should only happen with an isolated delta[] matrix. In principle,
there could be multiple such *)
(*  delta[] matrices,
but that is unlikely for current calculations.  Therefore remove other_:Null *)
(* argument so that previous methods are favored. *)
convertToMatrix[mSum[matrix_[i1_, i2_],
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[matrix[i1, i2]],
     {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   matrix === delta && (MemberQ[onesFreeIndices, i1] || MemberQ[onesFreeIndices, i2]);

(* If none of the previous conditions holds,
then the mSum must be a dot product between two vectors. *)
(* Find one of those vectors and put it into
  a CenterDot. This will become the right-hand vector. *)
convertToMatrix[mSum[vector_[i1_] other_: Null,
    {summedIndices_, {}, {}, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[vector[i1]] If[! nullQ[other], other, 1],
     {summedIndices, {}, {}, onesFreeIndices}]] /; validMatrixQ[vector] &&
    FreeQ[other, CenterDot] && ! (vector === gamma) && ! (vector === δgamma);
(* *** END INITIALIZATION *** *)


(* *** MERGE TERMS *** *)
(* Merge matrix components into CenterDot,
transposing is necessary to keep free indices on left and right. *)
(* Combine CenterDot[..., A[i,j]] with B[j,k] or x[j] *)
convertToMatrix[
   mSum[CenterDot[otherLeft__: Null, matrix1_[i1_, iSum_]^p1_:1] matrix2_[iSum_, i2_: Null]
     other_: Null, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[otherLeft, matrix1[i1, iSum]^p1,
       outputIndices[matrix2, iSum, i2]] If[! nullQ[other], other, 1], {removeEntry[
       summedIndices, iSum], leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix1] && validMatrixQ[matrix2] && MemberQ[summedIndices, iSum];

(* Combine CenterDot[..., A[i,j]] with B[k,j] *)
convertToMatrix[mSum[
    CenterDot[otherLeft__: Null, matrix1_[i1_, iSum_]^p1_:1] matrix2_[i2_, iSum_] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[otherLeft, matrix1[i1, iSum]^p1, matrix2[iSum, i2]^T†]
      If[! nullQ[other], other, 1], {removeEntry[summedIndices, iSum],
      leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix1] && validMatrixQ[matrix2] && MemberQ[summedIndices, iSum];
```

```
(* Combine B[k,i] with CenterDot[A[i,j], ...] or CenterDot[x[i], ...] *)
convertToMatrix[
   mSum[matrix2_[i2_, iSum_] CenterDot[matrix1_[iSum_, i1_: Null]^p1_:1, otherRight__: Null]
     other_: Null, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[matrix2[i2, iSum], outputIndices[matrix1, iSum, i1]^p1 ,
       otherRight] If[! nullQ[other], other, 1], {removeEntry[summedIndices, iSum],
      leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix1] && validMatrixQ[matrix2] && MemberQ[summedIndices, iSum];

(* Combine B[i,k] with CenterDot[A[i,j], ...] or CenterDot[x[i], ...] *)
convertToMatrix[mSum[matrix2_[iSum_, i2_: Null]
     CenterDot[matrix1_[iSum_, i1_: Null]^p1_:1, otherRight__: Null] other_: Null,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  convertToMatrix[mSum[CenterDot[outputIndices[matrix2, i2, iSum]^T†,
       outputIndices[matrix1, iSum, i1]^p1 , otherRight] If[! nullQ[other], other, 1],
     {removeEntry[summedIndices, iSum], leftFreeIndices,
      rightFreeIndices, onesFreeIndices}]] /;
   validMatrixQ[matrix1] && validMatrixQ[matrix2] && MemberQ[summedIndices, iSum];

(* Convert a matching gamma[a] and ones[a][i,j] into a deltaG[i,j] *)
(* Likewise for δdeltaG *)
convertToMatrix[
   mSum[gamma[a_] CenterDot[otherLeft__: Null, ones[a_][i1_, i2_]^p1_:1, otherRight__: Null]
     other_: Null, {summedIndices_, leftFreeIndices_,
     rightFreeIndices_, onesFreeIndices_}]] := convertToMatrix[
    mSum[CenterDot[otherLeft, deltaG[i1, i2]^p1, otherRight] If[! nullQ[other], other, 1],
     {removeEntry[summedIndices, a], leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   MemberQ[summedIndices, a] && FreeQ[leftFreeIndices, a] &&
    FreeQ[rightFreeIndices, a ] && FreeQ[onesFreeIndices, a];
convertToMatrix[mSum[δgamma[a_] CenterDot[otherLeft__: Null,
      ones[a_][i1_, i2_]^p1_:1, otherRight__: Null] other_: Null, {summedIndices_,
     leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] := convertToMatrix[
    mSum[CenterDot[otherLeft, δdeltaG[i1, i2]^p1, otherRight] If[! nullQ[other], other, 1],
     {removeEntry[summedIndices, a], leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   MemberQ[summedIndices, a] && FreeQ[leftFreeIndices, a] &&
    FreeQ[rightFreeIndices, a ] && FreeQ[onesFreeIndices, a];
(* *** END MERGE TERMS *** *)


(* *** TERMINATION CONDITIONS *** *)
(* Pull out an isolated gamma[a].  If the a is not summed,
it is NOT included in CenterDot[] *)
convertToMatrix[mSum[gamma[a_] other_,
    {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}]] :=
  gamma[a] convertToMatrix[mSum[other, {summedIndices, leftFreeIndices, rightFreeIndices,
       onesFreeIndices}]] /; FreeQ[summedIndices, a] && FreeQ[other, ones[a]];
convertToMatrix[mSum[δgamma[a_] other_, {summedIndices_, leftFreeIndices_,
     rightFreeIndices_, onesFreeIndices_}]] := δgamma[a] convertToMatrix[
     mSum[other, {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}]] /;
   FreeQ[summedIndices, a] && FreeQ[other, ones[a]];

(* If mSum[] only contains a gamma,
terminate as long as there are no summed or left/right free indices. *)
(* Likewise for δgamma *)
convertToMatrix[mSum[gamma[a_], {{}, {}, {}, onesFreeIndices_}]] :=
  gamma[a] /; onesFreeIndices === {a};
convertToMatrix[mSum[δgamma[a_], {{}, {}, {}, onesFreeIndices_}]] :=
  δgamma[a] /; onesFreeIndices === {a};

(* Terminate if all summed indices have been
 exhausted and CenterDot[ ] is outermost function. *)
(* Double-check that indices on components match the
  ordering of leftFreeIndices and rightFreeIndices. *)
(* Check that onesFreeIndices only appear once per term and on ones[] matrices *)
(* Errors called from this section will be labeled convertToMatrix(1) *)
convertToMatrix[mSum[CenterDot[matrices__],
    {{}, {left_: Null}, {right_: Null}, onesFreeIndices_}]] := Module[
   {x, i, currentIndex, numIndices, output},

   If[! left === Null,
    {
     (* Counts the number of times that left free index
      appears as a matrix index in either the left or right position *)
     If[Count[{matrices}, _[left, _ : Null] | _[_, left], Infinity] ≠ 1,
      error["convertToMatrix(1): Left free index (", left,
       ") appears too many or too few times in matrix components: ", matrices]];

     (* Makes sure left free index is in the correct position *)
     If[Count[{CenterDot[matrices]},
        CenterDot[_[left, _ : Null]^_:1, ___] | CenterDot[gamma[_] ones[left, _]^_:1, ___]] ≠ 1,
      error["convertToMatrix(1): Left free index (", left,
       ") appears in wrong position: ", matrices]];
     If[Count[{CenterDot[matrices]}, CenterDot[_[left, _ : Null]^_:1, ___] |
         CenterDot[δgamma[_] ones[left, _]^_:1, ___]] ≠ 1,
      error["convertToMatrix(1): Left free index (", left,
       ") appears in wrong position: ", matrices]];
    }];

   If[! right === Null,
    {
     (* Counts the number of times that right free index
      appears as a matrix index in either the left or right position *)
     If[Count[{matrices}, _[right, _ : Null] | _[_, right], Infinity] ≠ 1,
      error["convertToMatrix(1): Right free index (", right,
       ") appears too many or too few times in matrix components: ", matrices]];

     (* Makes sure right free index is in the correct position *)
     If[Count[{CenterDot[matrices]}, CenterDot[___, _[_ : Null, right]^_:1]] ≠ 1,
      error["convertToMatrix(1): Right free index (",
       right, ") appears in wrong position: ", matrices]];
    }];

   If[! onesFreeIndices === Null,
    {
     numIndices = Length[onesFreeIndices];
     For[i = 1, i ≤ numIndices, i++,
      currentIndex = onesFreeIndices⟦i⟧;

      (* Counts the number of times that each onesFreeIndices
        appears as a matrix index in either the left or right position on a non-
       ones matrix.  Subtracts number of occurrences in delta[]
        functions, since those are OK.  *)

      If[(Count[{matrices}, _[currentIndex, _ : Null] | _[_, currentIndex], Infinity] - Count[
           {matrices}, delta[currentIndex, _ : Null] | delta[_, currentIndex], Infinity]) ≠ 0 ,
       error["convertToMatrix(1): Ones free index (", currentIndex,
        ") appears in left/right matrix index: ", matrices]];

      (* Makes sure each onesFreeIndices only appears once, in either ones[] or delta *)

      If[(Count[{matrices}, ones[currentIndex][_, _], Infinity] + Count[{matrices},
           delta[currentIndex, _ : Null] | delta[_, currentIndex], Infinity]) ≠ 1 ,
       error["convertToMatrix(1): Ones free index (", currentIndex,
        ") appears too many times: ", matrices]];
     ]
    }];

   If[MatchQ[{matrices}, {x_[__]} /; validMatrixQ[x]] ||
     MatchQ[{matrices}, {Power[x_[__], T†]} /; validMatrixQ[x]],
    output = matrices,
    output = CenterDot[matrices]
   ];
   Return[output /. {Power[x_, T†] ⧴ SuperDagger[x]} /. {SuperDagger[ones[a_][i1_, i2_]] ⧴
       ones[a][i1, i2], SuperDagger[deltaG[j1_, j2_]] ⧴ deltaG[j1, j2]}];
  ];
```

■ Tests

▪ Define mD[ ] = matrix derivative

▪ Description

mD[f ,x]
Takes the partial derivative of f with respect to x. Unlike *Mathematica*'s `D[]`command, this function is capable of taking the derivative with respect to a vector component so that where mD[x[i], x[j]] = delta[i, j] = $\delta_{i\,j}$. The mSum[ ] function will properly drop summed indices if the delta[i, j] demands it.

Important notes:
- x can be a scalar or a vector component x[i] , but it cannot be a matrix component A[i, j]
- This function relies on D[ ] to do the actual differentiation.
- Differentiation with respect to a vector adds a dimension to f, so that mD[f, x[i]] is a vector if f is a scalar and a matrix if f is a vector. However, this function does not fix the the values of leftFreeIndices and rightFreeIndices to reflect that fact. If the result will be used in additional matrix multiplications, use the functions addIndex[ ] and dropIndex[ ] to manually manage indices. (See comments for definition of delta[ ] above.)

Needs to deal with the following cases:

```
    1. f wrt x, where x is simple (not indexed)
        validDerivativeQ[ ] ensures that there are no indexed forms of x in f
        ⇒ pass to D[f, x]
    2.f wrt x where x is indexed (x[i])
        validDerivativeQ[ ] ensures that there are no non-indexed forms of x in f
        If there are no instances of x[j] in f, then the derivatives is just 0
        Otherwise,
            make a list of all the x[j]
            call D[f, x[j] ] for each one
            insert a delta[i, j] into each result and sum
```

Sum where components of x in f do not have summation index - just swap sum and d[]
Sum where components of x in f do have summation index - forms delta(i,j) - will remain unevaluated if other terms of sum also have x[j]

Should never allow differentiation of f wrt x in the following cases:
- x is an index (e.g. i or i$nnn) ⇒ `FreeQ[summedIndices,x]`*
- x = x[i$nnn], where i$nnn is a summation index
⇒ `FreeQ[summedIndices, Part[Cases[{x}, _[k_] :→ k], 1]]`*
- x is an unindexed version of a variable that is indexed in f since this probably represents a typo (e.g. x = gamma but f contains gamma[a])
⇒ `FreeQ[f,x[_]]` *
- x is an indexed version of a variable that is not indexed in f since this probably represents a typo (e.g. x = gamma[a] but f contains gamma)
⇒ `FreeQ[f, Head[x]]` *
- x has more than one set of indices (x[a][b])
⇒ `Cases[{x}, xRoot_[_] :→ xRoot)]== Head[x]` *
- f or x includes a transpose operation ⇒ `FreeQ[x,SuperDagger]`* and `FreeQ[f, SuperDagger]`*

* indicates these rules are contained in the validDerivativeQ[ ] function.

Useful rules:

`FreeQ[f /.x[_]→1, x]` (removes indexed forms of *x* and just checks for *x*)

FreeQ[*f*, Except[*x*[*j*], *x*[_]]]

Notes about D[ ] and FreeQ[ ]

```
 FreeQ[x[j],x] = False
FreeQ[x, x[_]] = True
D[x[j], x] = D[x, x[j]] = 0
D[x[i], x[j]] = 0
```

Voice commands: "matrix derivative" or "M. D."

Function Definition

```
ClearAll[mD];

mD[f1_ + f2_, x_] := mD[f1, x] + mD[f2, x];
mD[Sqrt[f1_], x_] := 1/2 1/Sqrt[f1] mD[f1, x];
mD[n1_ f1_, x_] := n1 mD[f1, x] /; NumericQ[n1];

(* Case 1 *)
mD[mSum[f_, indices_], x_] := mSum[D[f, x], indices] /;
   FreeQ[x, _[_]] && validDerivativeQ[f, x, indices];

 (* Case 2 *)
mD[mSum[f_, indices_], x_] := Module[
    {xList, xRoot, inputRoot, inputIndex, answer, k},

    If[MatchQ[x, _[_][_]], error["Invalid indexing on x in mD(2): ", x]];

    inputRoot = Part[Cases[{x}, xRoot_[_] :→ xRoot], 1];
    inputIndex = Part[Cases[{x}, inputRoot[k_] :→ k], 1];
    xList = Cases[{f}, inputRoot[_], Infinity];
    If[Length[xList] > 0,
     answer = Apply[Plus,
       Map[Function[D[f, #1] delta[inputIndex, Part[Cases[{#1}, _[k_] :→ k], 1]]], xList]],
     answer = 0;
    ];
    mSum[answer, indices]
   ] /; MatchQ[x, _[_]] && validDerivativeQ[f, x, indices];
```

Tests

Define index manipulation tools

whichIndex = "left"," right", ones, or summed

addIndex[mSum[...], whichIndex, newIndex]

```
ClearAll[addIndex];

addIndex[mSum[x_, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}],
   whichIndex_, newIndex_] := Module[
   {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew},

   If[! FreeQ[{summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices}, newIndex],
    error["addIndex: Index (", newIndex, ") already in use.",
     "\n\tsummedIndices: ", summedIndices, "\n\tleftFreeIndices: ", leftFreeIndices,
     "\n\trightFreeIndices: ", rightFreeIndices, "\n\tonesFreeIndices: ", onesFreeIndices]
   ];
   {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew} =
    {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices};

   Switch[whichIndex,
    "summed", {
     AppendTo[summedIndicesNew, newIndex];
    },
    "left", {
     If[Length[leftFreeIndices] > 0,
      error["addIndex: Left index already exists: ", leftFreeIndices]];
     AppendTo[leftFreeIndicesNew, newIndex];
    },
    "right", {
     If[Length[rightFreeIndices] > 0,
      error["addIndex: Right index already exists: ", rightFreeIndices]];
     AppendTo[rightFreeIndicesNew, newIndex];
    },
    "ones", {
     AppendTo[onesFreeIndicesNew, newIndex];
    },
    _, error["Invalid whichIndex provided to addIndex: ", whichIndex]
   ];
   Return[mSum[x,
     {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew}]];
  ];
addIndex[f1_ + f2_, whichIndex_, newIndex_] :=
  addIndex[f1, whichIndex, newIndex] + addIndex[f2, whichIndex, newIndex];
             f1_                                  addIndex[f1, whichIndex, newIndex]
addIndex[ -------, whichIndex_, newIndex_] := ------------------------------------------ ;
           √f2_                                 √addIndex[f2, whichIndex, newIndex]
addIndex[√f1_ , whichIndex_, newIndex_] := √addIndex[f1, whichIndex, newIndex] ;
addIndex[n1_ f1_, whichIndex_, newIndex_] :=
  n1 addIndex[f1, whichIndex, newIndex] /; NumericQ[n1];
addIndex[n1_, whichIndex_, newIndex_] := n1 /; NumericQ[n1];
```

▪ dropIndex[mSum[...], whichIndex, dropIndex]

```
ClearAll[dropIndex];

dropIndex[mSum[x_, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}],
   whichIndex_, dropIndex_] := Module[
   {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew},

   {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew} =
    {summedIndices, leftFreeIndices, rightFreeIndices, onesFreeIndices};

   Switch[whichIndex,
    "summed", {
     If[! MemberQ[summedIndices, dropIndex], error["dropIndex: index (",
       dropIndex, ") not present in summedIndices: ", summedIndices]];
     summedIndicesNew = removeEntry[summedIndicesNew, dropIndex];
    },
    "left", {
     If[! MemberQ[leftFreeIndices, dropIndex], error["dropIndex: index (",
       dropIndex, ") not present in leftFreeIndices: ", leftFreeIndices]];
     leftFreeIndicesNew = removeEntry[leftFreeIndicesNew, dropIndex];
    },
    "right", {
     If[! MemberQ[rightFreeIndices, dropIndex], error["dropIndex: index (",
       dropIndex, ") not present in rightFreeIndices: ", rightFreeIndices]];
     rightFreeIndicesNew = removeEntry[rightFreeIndicesNew, dropIndex];
    },
    "ones", {
     If[! MemberQ[onesFreeIndices, dropIndex], error["dropIndex: index (",
       dropIndex, ") not present in onesFreeIndices: ", onesFreeIndices]];
     onesFreeIndicesNew = removeEntry[onesFreeIndicesNew, dropIndex];
    },
    _, error["Invalid whichIndex provided to dropIndex: ", whichIndex]
   ];
   Return[mSum[x,
     {summedIndicesNew, leftFreeIndicesNew, rightFreeIndicesNew, onesFreeIndicesNew}]];
  ];
dropIndex[f1_ + f2_, whichIndex_, newIndex_] :=
  dropIndex[f1, whichIndex, newIndex] + dropIndex[f2, whichIndex, newIndex];
```

$$\texttt{dropIndex}\left[\frac{\texttt{f1\_}}{\sqrt{\texttt{f2\_}}}, \texttt{whichIndex\_}, \texttt{newIndex\_}\right] := \frac{\texttt{dropIndex[f1, whichIndex, newIndex]}}{\sqrt{\texttt{dropIndex[f2, whichIndex, newIndex]}}};$$

$$\texttt{dropIndex}\left[\sqrt{\texttt{f1\_}}, \texttt{whichIndex\_}, \texttt{newIndex\_}\right] := \sqrt{\texttt{dropIndex[f1, whichIndex, newIndex]}};$$

```
dropIndex[n1_ f1_, whichIndex_, newIndex_] :=
  n1 dropIndex[f1, whichIndex, newIndex] /; NumericQ[n1];
dropIndex[n1_, whichIndex_, newIndex_] := n1 /; NumericQ[n1];
```

■ Define getIndexList[ ]

■ Description

Returns the free indices from the matrix or vector x, which must be represented by one or more mSum[...] functions. (If indices are not consistent among terms, will return error.) Returned in the form
{ {leftFreeIndices}, {rightFreeIndices}, {onesFreeIndices} }

■ Function Definition

```
ClearAll[getIndexList];
getIndexList[x_] := Module[
   {i, terms, currentIndexList, indexList, onesFreeIndices,
    leftFreeIndices, rightFreeIndices, summedIndices, numTerms,},
   indexList = {};
   terms = termList[x];
   numTerms = Length[terms];
   For[i = 1, i ≤ numTerms, i++,
    currentIndexList = Cases[terms〚i〛,
      mSum[_, {summedIndices_, leftFreeIndices_, rightFreeIndices_, onesFreeIndices_}] other_:
         1 /; NumericQ[other] → {leftFreeIndices, rightFreeIndices, onesFreeIndices}];
    If[currentIndexList == {} || Length[currentIndexList] > 1,
     Print["getIndexList Unexpected format: ", currentIndexList];
     Abort[];
    ];
    currentIndexList = currentIndexList〚1〛;
    If[i == 1,
     indexList = currentIndexList,
     If[! (indexList === currentIndexList),
       Print["getIndexList Unmatched indices: ", indexList, " vs. ", currentIndexList];
       Abort[];
      ];
    ];
   ];
   Return[currentIndexList];
  ];
```

■ Define simplifying rules (run in order shown)

■ Expand to fundamental expressions using //.expandTerms BEFORE running convertToComponents[ ]

MUST use delayed rule (:>) for these definitions; otherwise, all definitions of deltaG get same a1, etc.

```
ClearAll[expandTerms];

expandTerms = {
   deltaG[i1_, i2_] :→ Module[{a1}, gamma[a1] · ones[a1][i1, i2]],
   gainsSquared[] :→ Module[{a2}, gamma[●, a2] · gamma[a2, ●]],
   d[i1_, ●] :→ Module[{j1}, m[i1, j1] · s[j1, ●] + ηB[] β[i1, ●]],
   d[●, i1_] :→ Module[{j2}, s[●, j2] · m[j2, i1]† + ηB[] β[●, i1]],
   β[i1_, ●] :→ Module[{j1}, c[i1, j1] · b[j1, ●]],
   β[●, i1_] :→ Module[{j2}, b[●, j2] · c[j2, i1]†],
   m[i1_, i2_] :→ Module[{j3}, mSigma[i1, i2] + deltaG[i1, j3] · mDelta[j3, i2]]
  };
```

■ **Simplify by removing dummy (summation) indices with removeIndices[ ]**

Free indices will be left on the first and last matrix, but all other indices will be replaced with ● as long as they match properly.

```
ClearAll[removeIndices];

removeIndices[results_] := Module[
   {rules, resultsModified},

   (* Convert SuperDagger[x] notation to Power[x,T†] *)
   resultsModified = results //. SuperDagger[x_] :→ x^T†;

   (* Replacement rules for removing summed indices *)
   (* Since this is a replacement rule,
   outputIndices seems to be called BEFORE i2 is set. *)
   (* Thus, outputIndices cannot be used to weed out cases
    in which i2_:Null has been set to Null. *)
   (* Instead, make a separate case for matrices and vectors. *)
   rules = {
     CenterDot[otherLeft__: Null,
        matrix1_[i1_, iSum_]^p1_:1, matrix2_[iSum_, i2_]^p2_:1, otherRight__: Null]
       /; (! iSum === nullComponent) && validMatrixQ[matrix1] && validMatrixQ[matrix2]
      :→ CenterDot[otherLeft,
       matrix1[i1, nullComponent]^p1, matrix2[nullComponent, i2]^p2, otherRight],

     CenterDot[otherLeft__: Null,
        matrix1_[i1_, iSum_]^p1_:1, matrix2_[iSum_]^p2_:1, otherRight__: Null]
       /; (! iSum === nullComponent) && validMatrixQ[matrix1] && validMatrixQ[matrix2]
      :→ CenterDot[otherLeft,
       matrix1[i1, nullComponent]^p1, matrix2[nullComponent]^p2, otherRight],

     CenterDot[otherLeft__: Null,
        matrix1_[iSum_]^p1_:1, matrix2_[iSum_, i2_]^p2_:1, otherRight__: Null]
       /; (! iSum === nullComponent) && validMatrixQ[matrix1] && validMatrixQ[matrix2]
      :→ CenterDot[otherLeft,
       matrix1[nullComponent]^p1, matrix2[nullComponent, i2]^p2, otherRight],

     CenterDot[otherLeft__: Null,
        matrix1_[iSum_]^p1_:1, matrix2_[iSum_]^p2_:1, otherRight__: Null]
       /; (! iSum === nullComponent) && validMatrixQ[matrix1] && validMatrixQ[matrix2]
      :→ CenterDot[otherLeft, matrix1[nullComponent]^p1, matrix2[nullComponent]^p2, otherRight]
    };
   resultsModified = resultsModified //. rules;
   Return[resultsModified /. {Power[x_, T†] :→ SuperDagger[x]}];

  ];
```

■ Collapse to higher-level expressions using collapseTerms[ ]

collapseTerms must be used AFTER calling convertToMatrix[ ] and removeIndices[ ]

```
ClearAll[collapseTerms, collapse1, collapse2, collapse3];
collapse1 := {
   CenterDot[x_[i__]] :→ x[i] /; validMatrixQ[x],
   CenterDot[otherLeft__ : Null, deltaG[i1_, iSum_], ones[a_][iSum_, i2_],
     otherRight__ : Null] :→ gamma[a] CenterDot[otherLeft, ones[a][i1, i2], otherRight],
   CenterDot[otherLeft__ : Null, ones[a_][i1_, iSum_], deltaG[iSum_, i2_], otherRight__ : Null] :→
    gamma[a] CenterDot[otherLeft, ones[a][i1, i2], otherRight],
   CenterDot[otherLeft__ : Null, ones[a_][i1_, iSum_], ones[b_][iSum_, i2_], otherRight__ :
      Null] :→ delta[a, b] CenterDot[otherLeft, ones[Part[Sort[{a, b}], 1]][i1, i2], otherRight]
   (* , convertToMatrix[mSum[delta[i_,j_],{{},{},{},{}}]]:→delta[i,j]*)
  };
collapse2 := {
   CenterDot[otherLeft__ : Null, mSigma[i1_, i2_], otherRight__ : Null] other_ : 1 +
     CenterDot[otherLeft__ : Null,
       deltaG[i1_, iSum_], mDelta[iSum_, i2_] , otherRight__ : Null] other_ : 1
    :→ other CenterDot[otherLeft, m[i1, i2], otherRight],
   mSigma[i1_, i2_] other_ : 1 +
     CenterDot[deltaG[i1_, iSum_], mDelta[iSum_, i2_]] other_ : 1 :→ other m[i1, i2],
   CenterDot[otherLeft__ : Null, mSigma[i1_, i2_]†, otherRight__ : Null] other_ : 1 +
     CenterDot[otherLeft__ : Null,
       mDelta[i1_, iSum_] †, deltaG[iSum_, i2_] , otherRight__ : Null] other_ : 1
    :→ other CenterDot[otherLeft, m[i1, i2]†, otherRight],
   mSigma[i1_, i2_]† other_ : 1 + CenterDot[mDelta[i1_, iSum_] †, deltaG[iSum_, i2_]] other_ : 1 :→
    other CenterDot[m[i1, i2]†] (*,
   (* Take out \beta replacements so that equations are
    kept in terms of unknown parameters *)
   CenterDot[otherLeft__:Null,c[i1_,iSum_],b[iSum_], otherRight__:Null]→
    CenterDot[otherLeft,β[i1],otherRight],
   CenterDot[otherLeft__:Null,b[iSum_]†,c[iSum_,i1_]†, otherRight__:Null]→
    CenterDot[otherLeft,β[i1]†,otherRight] *)
  };
collapse3 = {
   CenterDot[Longest[a__], x__] coeff1__ : 1 + CenterDot[a__, y__] coeff2__ : 1 + other2__ : 0 :→
    CenterDot[a, (CenterDot[x] coeff1 + CenterDot[y] coeff2)] + other2,
   CenterDot[x__, Longest[a__]] coeff1__ : 1 + CenterDot[y__, a__] coeff2__ : 1 + other2__ : 0 :→
    CenterDot[(CenterDot[x] coeff1 + CenterDot[y] coeff2), a] + other2,
   CenterDot[x_[i__]] :→ x[i]
  };

collapseTerms[x_] := ReplaceRepeated[
   FullSimplify[ReplaceRepeated[ReplaceRepeated[x, collapse1], collapse2]], collapse3];
```

▪ Break into smaller matrices using matrixSimplify[ ]

Starts with the expressions in the deepest level of *x* (the leaves), then works its way upward looking for "CenterDot" (or "Plus") headers. Whenever it finds a CenterDot[] it takes the contents and assigns them to a new matrix, and substitutes that matrix into *x* wherever possible. After making a substitution, it re-examines all of the other expressions in the current level to make sure none of them have changed. It returns a simplified form for *x*, plus a list of replacement rules that can be used to convert xSimple back into *x*. Before exiting, it doublechecks to make sure those substitution rules exactly reproduce *x*.
The second (optional) argument specifies whether CenterDot or Plus headers should be used
    CenterDot - makes each set of multiplied matrices into a new matrix (smaller pieces, but more of them)
    Plus - makes each set of added matrices into a new matrix (fewer pieces, but each is larger)
Can print results nicely like this:
    r = matrixSimplify[m];
    MatrixForm[r⟦1⟧]
    Column[r⟦2⟧]

```
ClearAll[matrixSimplify];
matrixSimplify[x_, searchMethod_: Plus] := Module[
  {xSimple, currentLevel, currentValue, uniqueMatrix,
    uniqueCounter, replacementList = {}, i, j, startOver, isUnique},

   If[! (searchMethod === CenterDot) && ! (searchMethod === Plus),
    Print["WARNING: Invalid setting for searchMethod (",
     searchMethod, "). Changing to Plus."];
    searchMethod = Plus;
   ];
   xSimple = x;
   replacementList = {};
   uniqueCounter = 1;
   startOver = False;
   For[i = Depth[xSimple], i ≥ 1, If[startOver, startOver = False, i--],
    currentLevel = Level[xSimple, {i}];
    For[j = 1, j ≤ Length[currentLevel] && Not[startOver],
     j++, (* Print["[", i, ", ", j, "]"];*)
     currentValue = currentLevel⟦j⟧;
     (* Could also use Plus instead of CenterDot - works about the same *)
     If[Head[currentValue] == searchMethod,
      uniqueMatrix = Unique["m"];
      AppendTo[replacementList, uniqueMatrix → currentValue];
      xSimple = xSimple //. {currentValue → uniqueMatrix};
      startOver = True;
     ];
    ];
   ];
   If[Not[(xSimple //. replacementList) === x],
    error["Replacement rules do not reproduce input."];
   ];
   Return[{xSimple, replacementList}];
  ];
```

▪ Define functions to calculate Jacobian and display results

▪ calculateJacobian[functions, variables]

Calculates all of the elements of the Jacobian matrix:

$$\frac{\partial F_{(i)}}{\partial x_{(j)}}$$

where $F_{(i)}$ is the ith function provided in the < functions > list and $x_{(j)}$ is the jth variable provided in the < variables > list.

Returns two formats:

first output = human-readable form

second output = components form (for further calculations)

mode (optional) = "gradient" for gradient (default), "jacobian" for Jacobian

Index position:

When calculating a gradient...

...with respect to s or b, it adds a left index (to make a vector).

...with respect to gamma, it adds an entry to the ones index.

When calculating a Jacobian...

...of a normal vector (free left index) with respect to s or b, it adds a right index.

...of a "ones vector" (free ones index) with respect to s or b, it adds a left index.

...with respect to gamma, it adds an entry to the ones index.

The left index on a "ones vector" is necessary to avoid creating the transpose of the desired quantity.

Variables MUST be standard vectors of the form name[index].

```
ClearAll[calculateJacobian];
calculateJacobian[functions_, variables_, mode_: "gradient"] := Module[
   {r, c, derivativeIndex, resultsMatrix, numRows, numColumns, currentFunction,
    currentVariable, currentValue, componentsMatrix, i, indexPosn, currentIndexList},
   If[UnsameQ[mode, "gradient"] && UnsameQ[mode, "jacobian"],
    Print["Invalid mode: ", mode]; Abort[]];
   numRows = Length[functions];
   numColumns = Length[variables];
   resultsMatrix = ConstantArray[Null, {numRows, numColumns}];
   componentsMatrix = ConstantArray[Null, {numRows, numColumns}];
   For[r = 1, r ≤ numRows, r++,
    For[c = 1, c ≤ numColumns, c++,
      currentFunction = functions⟦r⟧;
      currentVariable = variables⟦c⟧;
      derivativeIndex = currentVariable /. _[i_] → i;
      currentValue = mD[currentFunction, currentVariable];
      If[MatchQ[currentVariable, gamma[_]],
       indexPosn = "ones",
       If[mode == "gradient",
        {
         indexPosn = "left"
        },
        { (* Jacobian *)
         If[mode ≠ "jacobian", Print["Unexpected mode: ", mode]];
         currentIndexList = getIndexList[currentFunction];
         If[currentIndexList⟦3⟧ == {},
          { (* Expect a free left index and no right index *)
           If[Length[currentIndexList⟦1⟧] ≠ 1 || currentIndexList⟦2⟧ ≠ {},
            Print["calculateJacobian: unexpected index list: ", currentIndexList];
            Abort[];
           ];
           indexPosn = "right";
          },
          {(* Expect no left or right indices *)
           If[currentIndexList⟦1⟧ ≠ {} || currentIndexList⟦2⟧ ≠ {},
            Print["calculateJacobian: unexpected index list: ", currentIndexList];
            Abort[];
           ];
           indexPosn = "left";
          }
         ]
        }
       ]
      ];
      (*Print["Adding index: ",derivativeIndex," (",indexPosn,")"];*)
      currentValue = addIndex[Numerator[currentValue],
         indexPosn, derivativeIndex] / Denominator[currentValue];
      componentsMatrix⟦r, c⟧ = currentValue;
      currentValue = collapseTerms[removeIndices[convertToMatrix[currentValue]]];
      resultsMatrix⟦r, c⟧ = currentValue;
     ];
   ];
   Return[{resultsMatrix, componentsMatrix}];
  ];
```

▪ matrixCompare[m1, m2, show]

Compares two Jacobian matrices and prints a results matrix showing True in positions that match and False in positions that disagree. It then prints each matrix's contents at the positions that disagree (optional).

m1, m2 = name of functions containing matrices to be compared

show = if True, prints the values of the differing elements

Useful for determining which elements need to be recalculated when using different methods.

```
ClearAll[matrixCompare];
matrixCompare[m1_, m2_, show_] := Module[
   {resultsMatrix, r, c, numRows, numColumns},
   If[Dimensions[m1] ≠ Dimensions[m2],
    Print["Inconsistent array sizes in matrixCheck."];
    Abort[];
   ];
   {numRows, numColumns} = Dimensions[m1];

   resultsMatrix = ConstantArray[Null, {numRows, numColumns}];
   For[r = 1, r ≤ numRows, r++,
    For[c = 1, c ≤ numColumns, c++,
      resultsMatrix⟦r, c⟧ = (m1⟦r, c⟧ === m2⟦r, c⟧);
     ];
   ];
   Print[MatrixForm[resultsMatrix]];
   If[show,
    For[r = 1, r ≤ numRows, r++,
      For[c = 1, c ≤ numColumns, c++,
        If[! resultsMatrix⟦r, c⟧,
          Print["Row ", r, ", Column ", c, ":"];
          Print["\tMatrix 1: ", m1⟦r, c⟧];
          Print["\tMatrix 2: ", m2⟦r, c⟧];
         ];
       ];
     ];
   ];
  ];
```

- **printElements[m]**

Returns a table consisting of {"(r,c):", value} entries that can be displayed with Table[...] command (will not word-wrap properly)
If doIndex = False, supresses index values when printing (default = True)
If doPrint = True (default), also prints table using word-wrapping format.

```
ClearAll[printElements];
printElements[m_, doIndex_: True, doPrint_: True] := Module[
   {r, c, numRows,  numColumns, tableOut, position},

   tableOut = {};
   {numRows, numColumns} = Dimensions[m];
   For[r = 1, r ≤ numRows, r++,
    For[c = 1, c ≤ numColumns, c++,
      position = "(" <> ToString[r] <> "," <> ToString[c] <> "):";
      AppendTo[tableOut, {position, m⟦r, c⟧}];
      If[doPrint,
       If[doIndex,
        Print[position, " ", m⟦r, c⟧],
        Print[m⟦r, c⟧]
       ]
      ];
     ];
   ];
   Return[tableOut];
  ];
```

Variational calculus

zeroTest[ ]

zeroTest[x, zeroEqns, zeroTerms]
Uses "zeroth-order" equations to set matching terms to 0.
 x = polynomial to be simplified
 zeroEqns = polynomial(s) that should be replaced with 0
 zeroTerms = list of all terms in zeroEqns
Not entierly reliable, but if it does produce answer, appears to be correct. Has problems in following cases:
 x = dChidS + 7.2 dChidS (problem with non-integer coefficients?)
 x = dChidS + 7.2 dChidB (unable to sort terms well enough to handle both eqns together?)

```
ClearAll[zeroTest];
(*zeroTest[x__,zeroEqns_,zeroTerms_ ]:=0/;x==zeroEqns⟦1⟧;
zeroTest[x__,zeroEqns_,zeroTerms_ ]:=0/;x==zeroEqns⟦2⟧;*)
zeroTest[x__, zeroEqns_, zeroTerms_] := 0 /; Apply[Or, Map[Function[x == #1], zeroEqns]];
zeroTest[a_, zeroEqns_, zeroTerms_] := a /; allFreeQ[a, zeroTerms];
zeroTest[a_ + x__, zeroEqns_, zeroTerms_] :=
  a + zeroTest[Plus[x], zeroEqns, zeroTerms] /; allFreeQ[a, zeroTerms];
zeroTest[a_ x__, zeroEqns_, zeroTerms_] :=
  a zeroTest[Times[x], zeroEqns, zeroTerms] /; NumericQ[a];
zeroTest[x__, zeroEqns_, zeroTerms_] := Module[{y},
    y = FactorTermsList[x];
    Return[y⟦1⟧ zeroTest[y⟦2⟧, zeroEqns, zeroTerms]];
   ] /; Part[FactorTermsList[x], 1] ≠ 1;
```

Define Lagrange multipliers

Use SetAttributes to tell *Mathematica* that the λ's are numeric. However, since SetAttributes only controls functions, need to use λChi[], etc. That will satisfy the NumericQ test (but not the NumberQ test).

```
ClearAll[λG, λS, λChi, λB, ηB];

SetAttributes[λG, NumericFunction];
SetAttributes[λS, NumericFunction];
SetAttributes[λChi, NumericFunction];
SetAttributes[λB, NumericFunction];
SetAttributes[ηB, NumericFunction];
```

Track setup status

Define a function to see whether the specified variable has been set to True; if not abort calculation.
x = name of setup confirmation variable AS STRING

```
ClearAll[confirmSetup];
confirmSetup[x_] := Module[{},
   If[SymbolName[Symbol[x]] === x || (! Symbol[x] === True),
    {
     MessageDialog["Must run setup notebook first for " <> x];
     Print[Style["Must run setup notebook for " <> ToString[x] <> " first!", Red]];
     EmitSound[Sound[SoundNote[{"B3", "C3", "G3"}, 0.5]]];
     Abort[];
    },
    {
     Print[x, " OK."];
    }
   ];
   Return[True];
  ];
```

Setup complete (gradientCalculatorSetupComplete)

```
gradientCalculatorSetupComplete = True;
```

Appendix C

IF Processor Parts List

This appendix contains the parts list and vendor information for the wideband IF processor described in Section 3.4.1.

Table C.1: As-designed parts list for wideband IF processor.

Description	**Manufacturer**	**Model #**	**Vendor**
Amplifiers			
4 - 8 GHz amplifiers	AML	AML 48L4002	AML
2 - 20 GHz amplifier	AML	AML 220L3401	AML
DROs			
8.75 GHz (+13 dBm)	Nexyn	NXOS-1375-01044	Nexyn
12.25 GHz (+13 dBm)	Nexyn	NXOS-1725-01044	Nexyn
13.75 GHz (+13 dBm)	Nexyn	NXOS-0875-01044	Nexyn
17.25 GHz (+13 dBm)	Nexyn	NXOS-1225-01044	Nexyn
Filters			
5.75 - 9.75 GHz bandpass filter	Reactel	4CX11-7750-X4000S11	Reactel
9.25 - 13.25 GHz bandpass filter	Reactel	4C11-11250-X4000S11	Reactel

(table continued on next page)

Table C.1: As-designed parts list for wideband IF processor.

Description	**Manufacturer**	**Model #**	**Vendor**
12.75 - 16.75 GHz bandpass filter	Reactel	5C11-14750-X4000S11	Reactel
16.25 - 20.25 GHz bandpass filter	Reactel	5C11-18250-X4000S11	Reactel
Low-pass filter 12.5 - 16.5 GHz passband	Reactel	10L0-X16.5GS11	Reactel
Low-pass filter 9.25 - 13.5 GHz passband	Reactel	10L0-X13.5GS11	Reactel
Low-pass filter DC - 10 GHz passband	Reactel	9LO-X10GS11	Reactel
4 - 8 GHz bandpass filter	Reactel	9CX11-6G-X4GS11	Reactel
Mixers			
Triple-balanced mixer	Marki Microwave	M2H-0220LA	Marki Microwave
Triple-balanced mixer	Advanced Microwave	M3006L	Hytech
DC Power Supplies			
12 V, 0.7 A	Acopian	12EB70	Acopian
15 V, 2.0 A	Acopian	A15NT200	Acopian
5 V, 1.0 A	Acopian	5EB100	Acopian
Attenuators			
1-dB attenuator (SMA, 23 GHz)	Inmet	23AH-01	Hytech
3-dB attenuator (SMA, 23 GHz)	Inmet	23AH-03	Hytech
6-dB attenuator (SMA, 23 GHz)	Inmet	23AH-06	Hytech
10-dB attenuator (SMA, 23 GHz)	Inmet	23AH-10	Hytech
20-dB attenuator (SMA, 23 GHz)	Inmet	23AH-20	Hytech
3-dB attenuator (2.9 mm, 26 GHz)	Inmet	26AH-3	Hytech

(table continued on next page)

Table C.1: As-designed parts list for wideband IF processor.

Description	**Manufacturer**	**Model #**	**Vendor**
Cables and Interconnects			
Super SMA F - F adapter (27 GHz)	Southwest Microwave	232-02SF	Hytech
Super SMA M - M adapter (27 GHz)	Southwest Microwave	231-02SF	Hytech
Right-angle SMA M - M (DC - 27 GHz)	Hasco	SMAP-SMAP-RA	Hytech
Flange-mount SMA F - F (for 4 - 8 GHz outputs)	Inmet	5313	Hytech
Flange-mount 2.9 mm F - F (for input and test port)	S.G. McGeary	111-35-35-000	C.W. Swift
Hand-formable cable (SMA connectors)	United Microwave	Microform 139	United Microwave
Flexible coaxial cable (SMA connectors)	United Microwave	Microflex 165	United Microwave
Low-loss coaxial cable (SMA connectors)	United Microwave	Micropore 190	United Microwave
Misc			
Four-way power splitter	MAC Technology	P8248-4	MAC Technology
10-dB coupler	MAC Technology	C4258-10	MAC Technology
50-ohm terminator (SMA, DC - 26.5 GHz)	Inmet	TS260MC	Hytech
Chassis	Proline Metal Fabricators (PMF)	10-007S, 67-019, 65-003	PMF

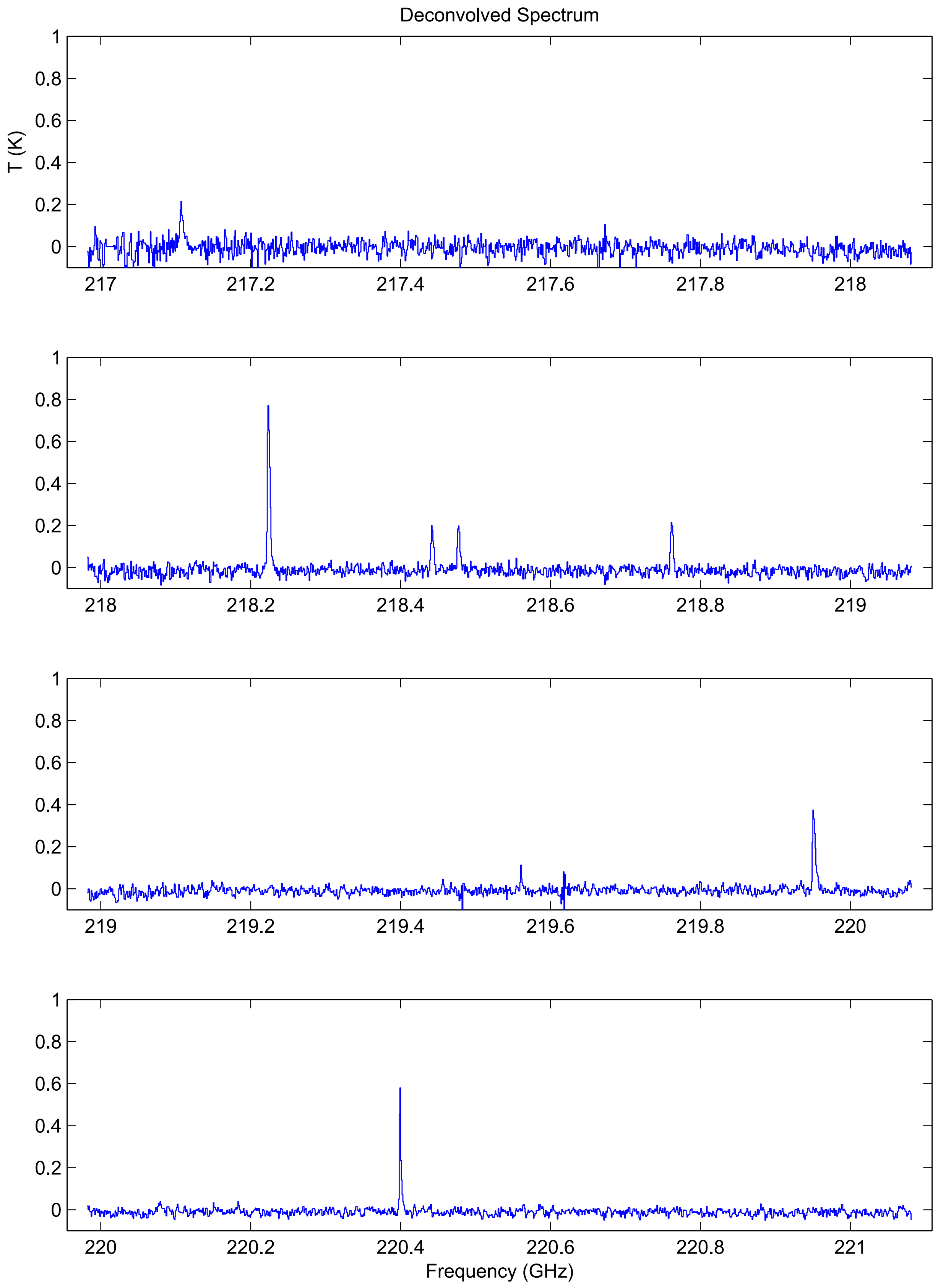
Vendor	Location	Website
Acopian Technical Company	Easton, PA	www.acopian.com
AML Communications	Camarillo, CA	www.amlj.com
C.W. Swift and Associates	Van Nuys, CA	www.cwswift.com
Hytech Associates	Westlake Village, CA	www.hytechassociatesinc.com
MAC Technology	Klamath Falls, OR	www.mactechnology.com
Marki Microwave	Morgan Hill, CA	www.markimicrowave.com
Nexyn Corporation	Sunnyvale, CA	www.nexyn.com
Proline Metal Fabricators	Fremont, CA	www.gotopmf.com
Reactel	Gaithersburg, MD	www.reactel.com
United Microwave Products	Torrance, CA	www.unitedmicrowave.com

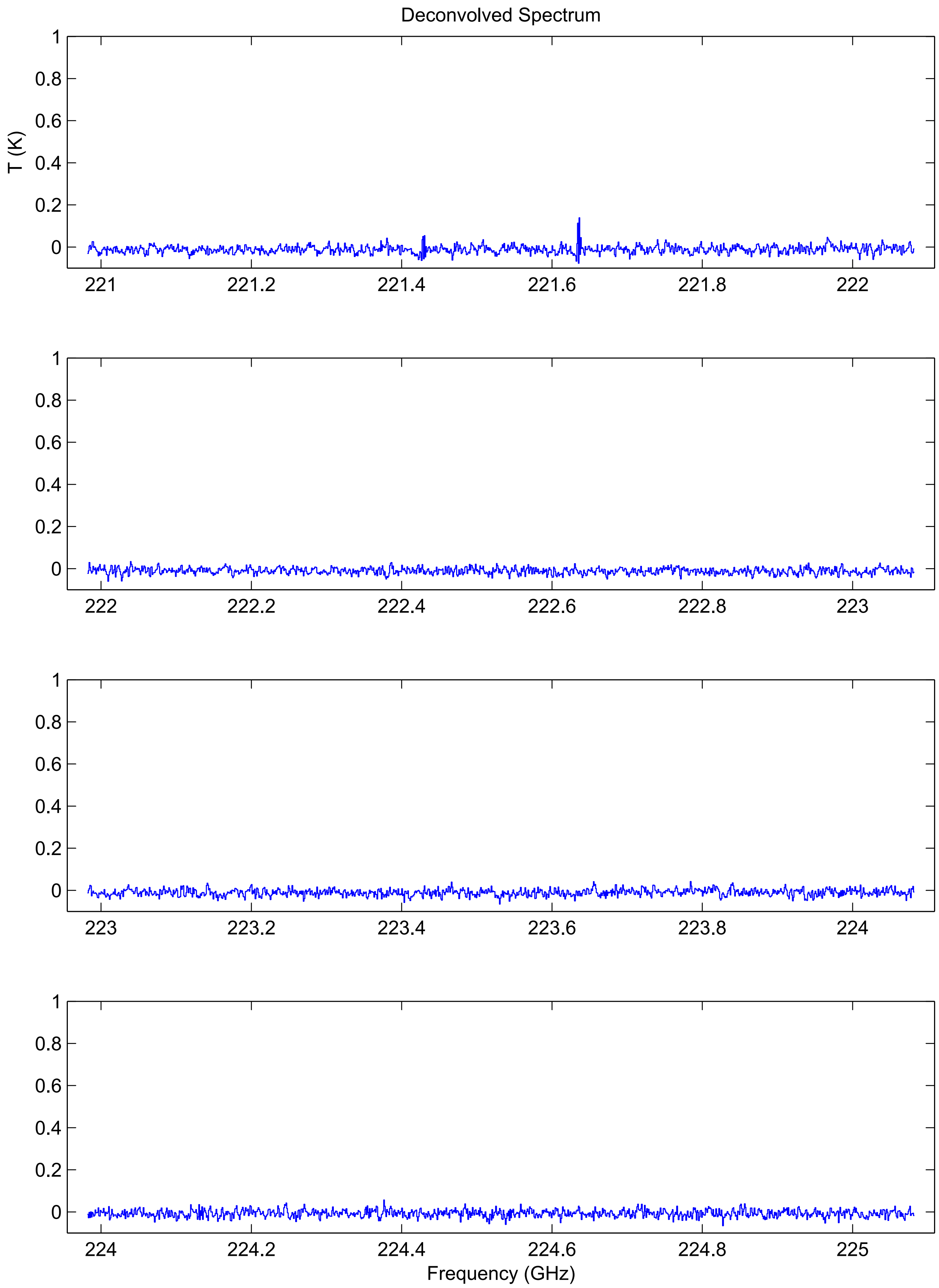
Table C.2: Contact information for vendors listed in Table C.1.

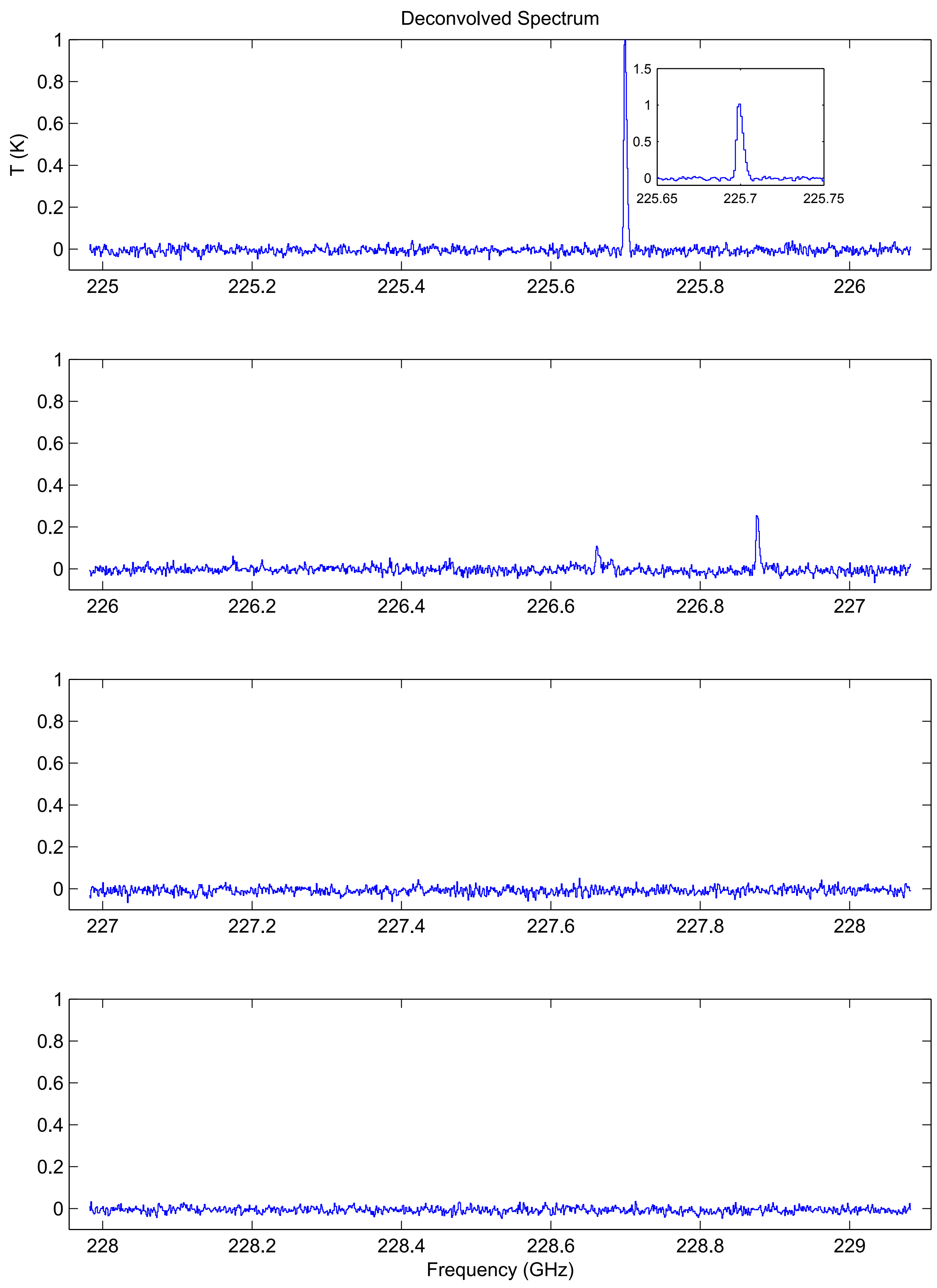
Appendix D

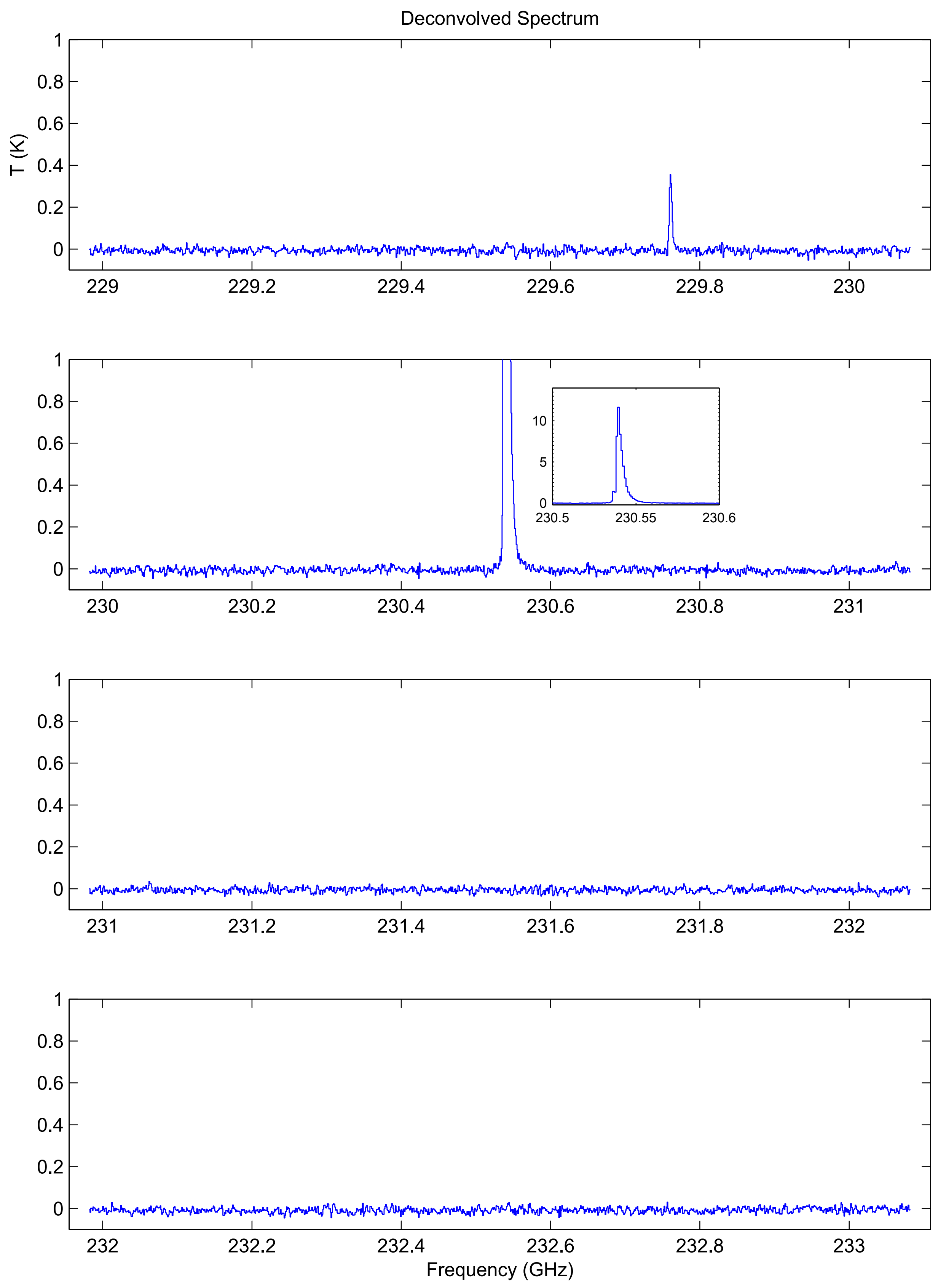
L1157 Deconvolution Results

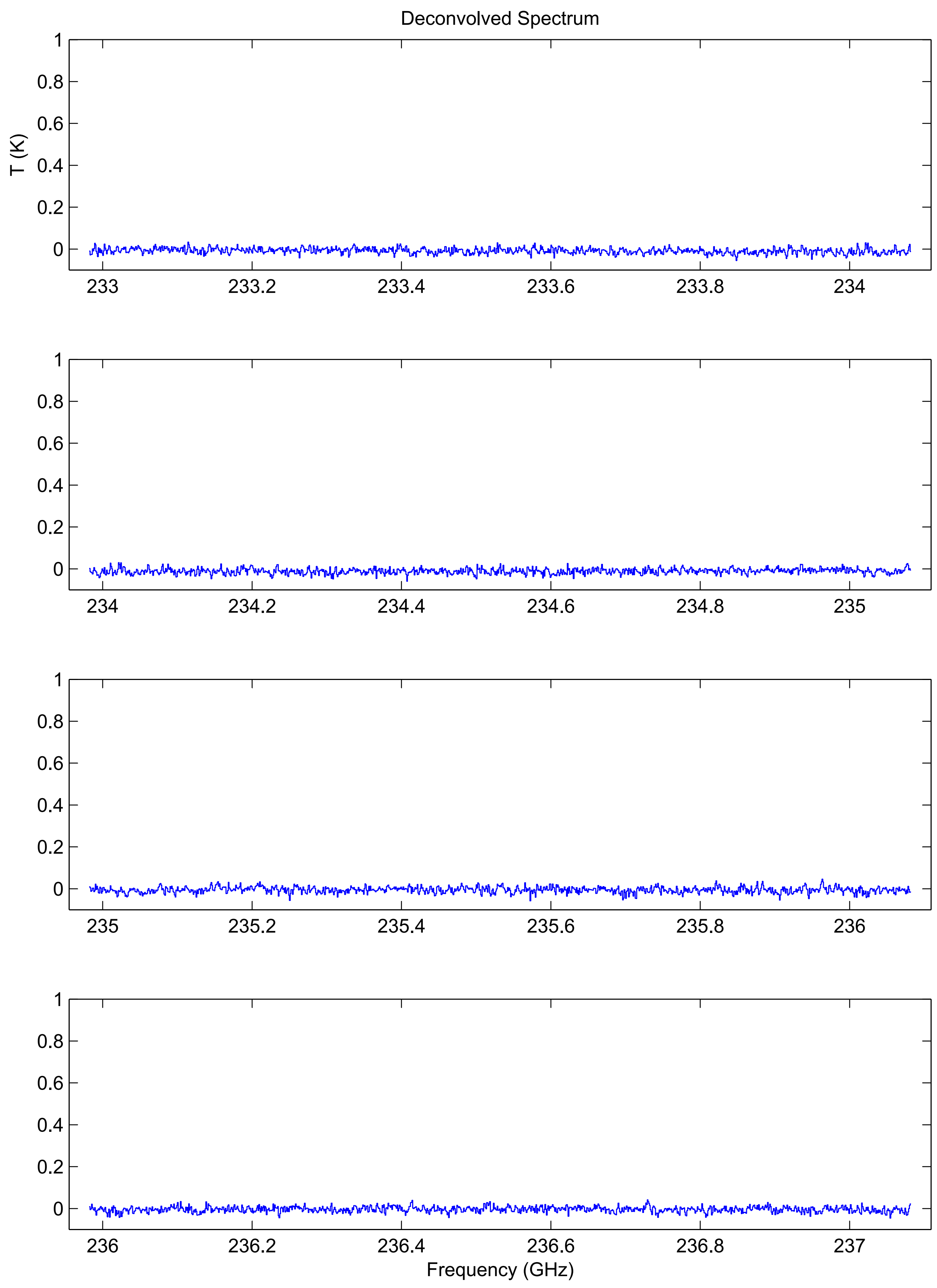
This appendix contains the deconvolved, single-sideband spectrum of position B1 in the outflow of L1157 (Section 6.1). Brightness temperatures are calibrated to the T_A^* scale, as described in Equation 2.10. The peak fits used for the analysis in Section 6.4 are shown in Figures D.1 through D.9.

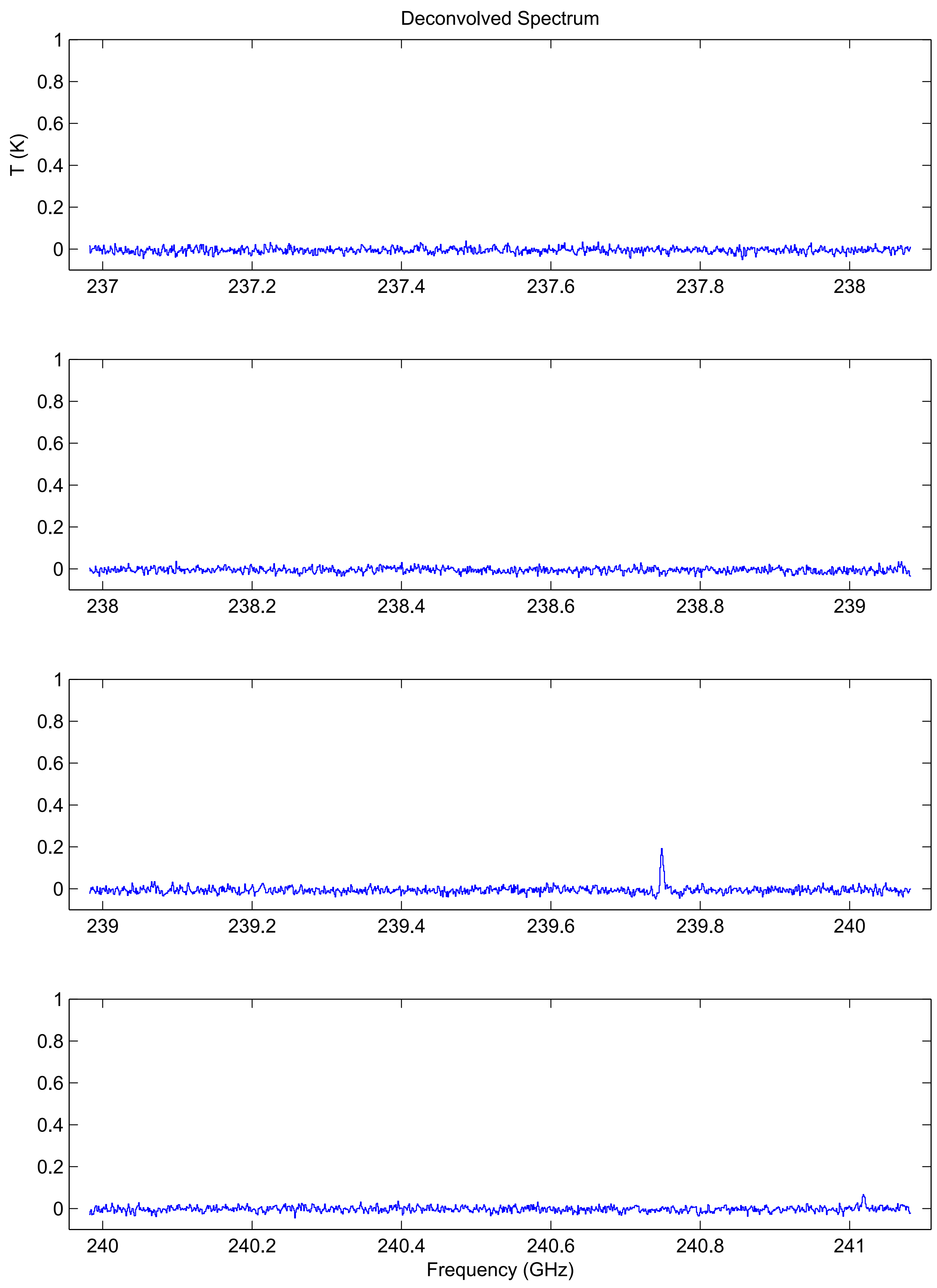

Deconvolved Spectrum
T (K)
Frequency (GHz)
217
217.2
217.4
217.6
217.8
218
218.2
218.4
218.6
218.8
219
219.2
219.4
219.6
219.8
220
220.2
220.4
220.6
220.8
221
0
0.2
0.4
0.6
0.8
1

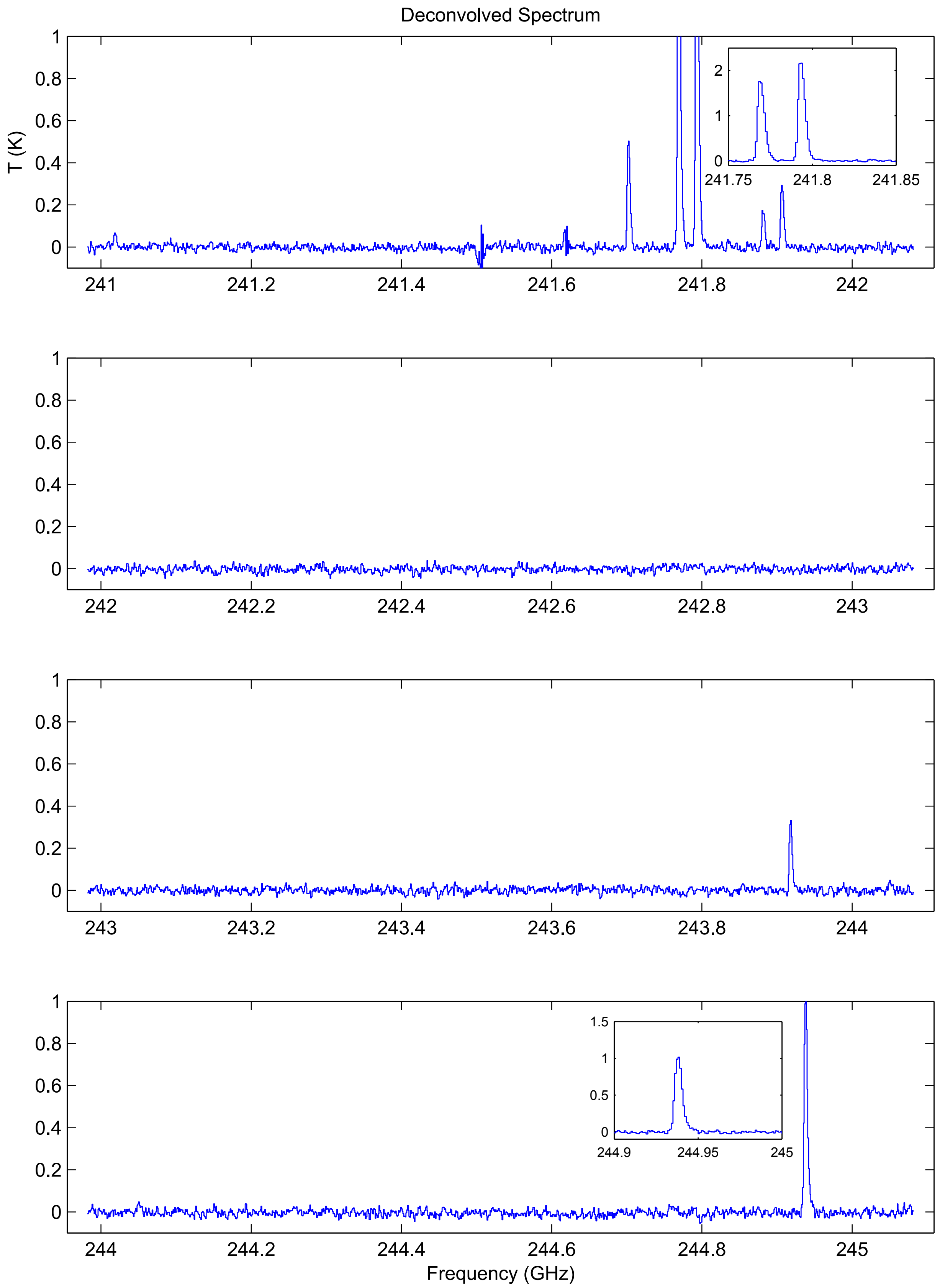

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
221
221.2
221.4
221.6
221.8
222
222.2
222.4
222.6
222.8
223
223.2
223.4
223.6
223.8
224
224.2
224.4
224.6
224.8
225
Frequency (GHz)

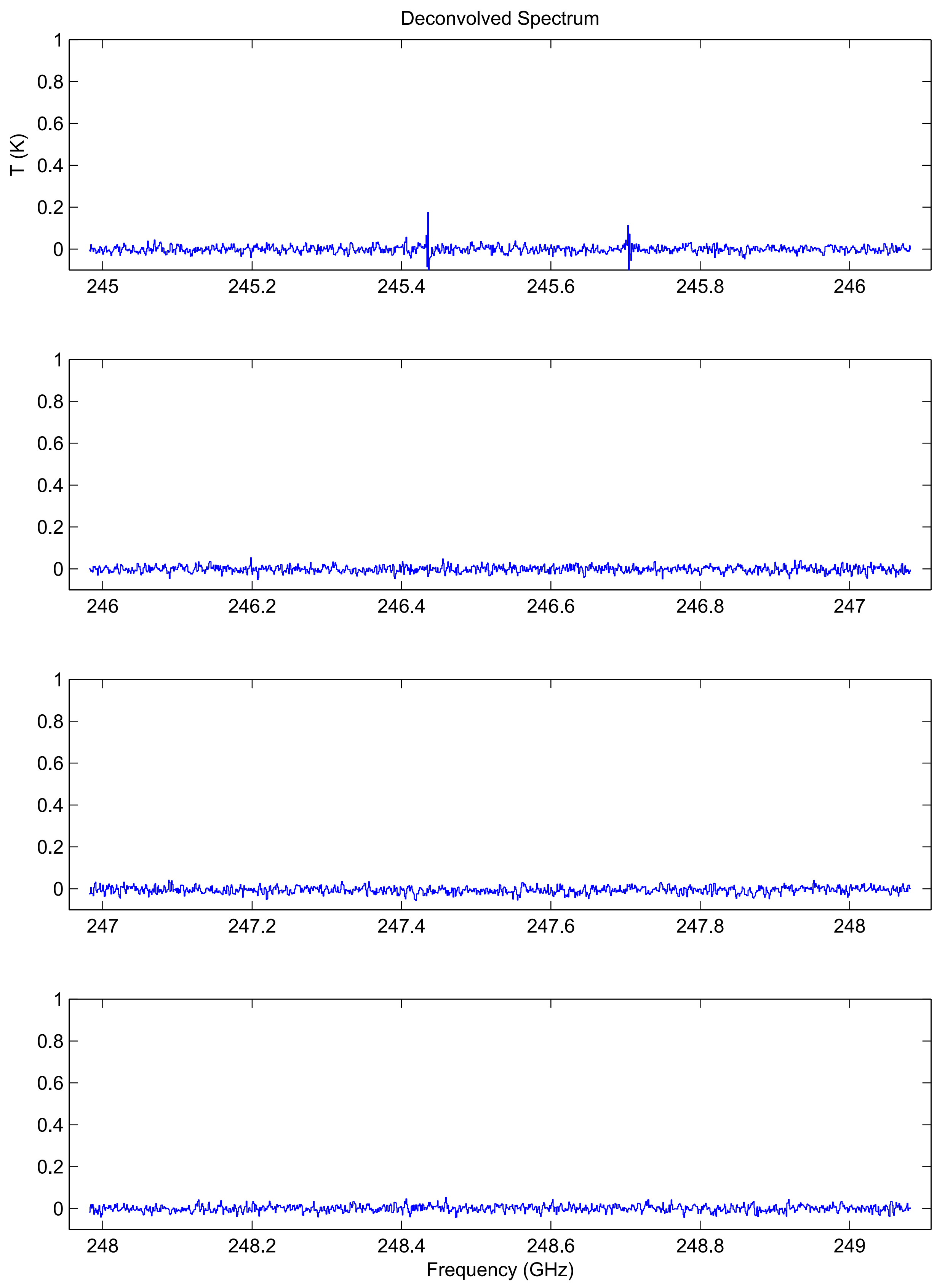

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
225
225.2
225.4
225.6
225.8
226
1.5
1
0.5
0
225.65
225.7
225.75
226.2
226.4
226.6
226.8
227
227.2
227.4
227.6
227.8
228
228.2
228.4
228.6
228.8
229
Frequency (GHz)

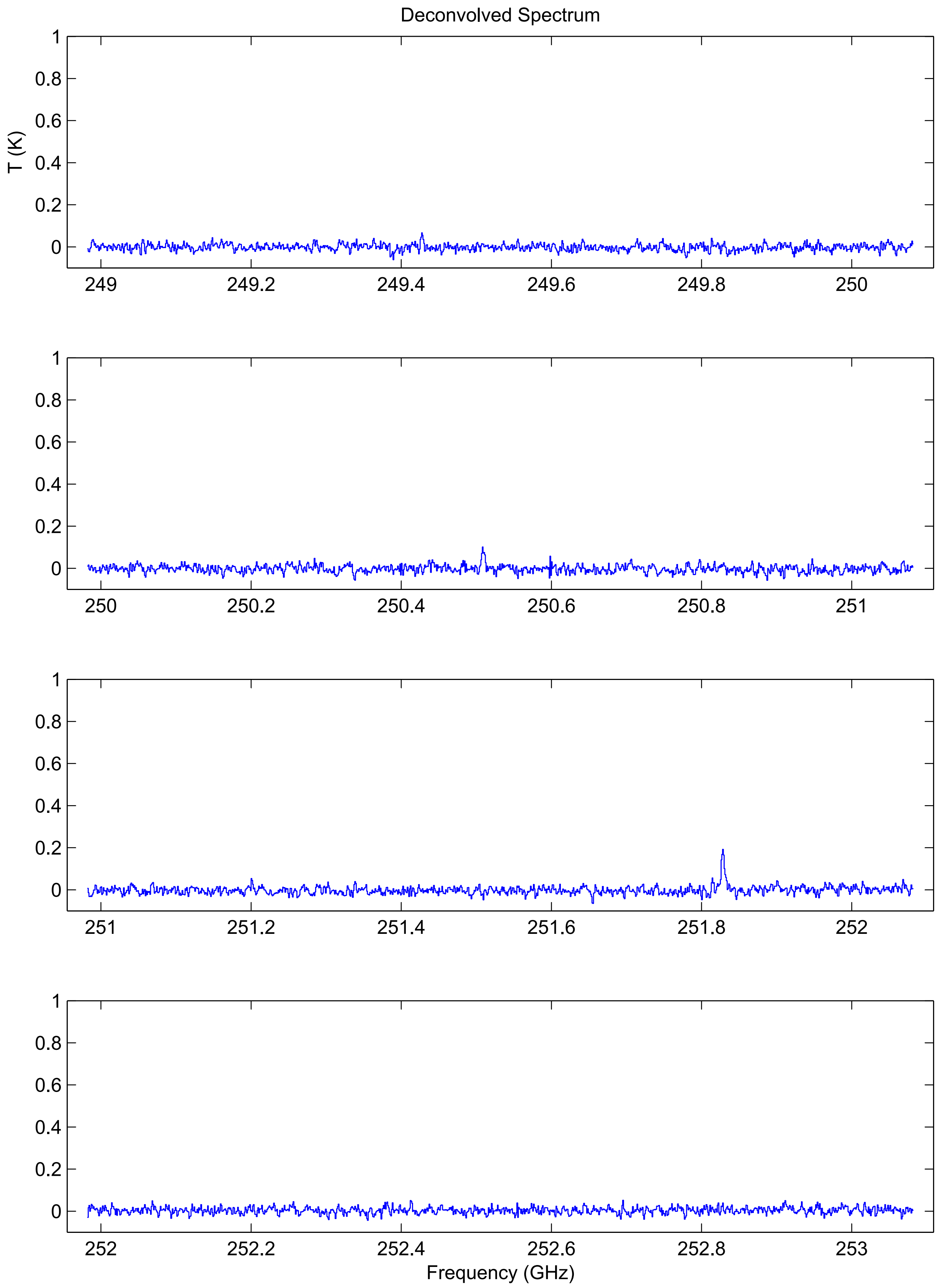

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
229
229.2
229.4
229.6
229.8
230
230.2
230.4
230.6
230.8
231
10
5
230.5
230.55
230.6
231.2
231.4
231.6
231.8
232
232.2
232.4
232.6
232.8
233
Frequency (GHz)

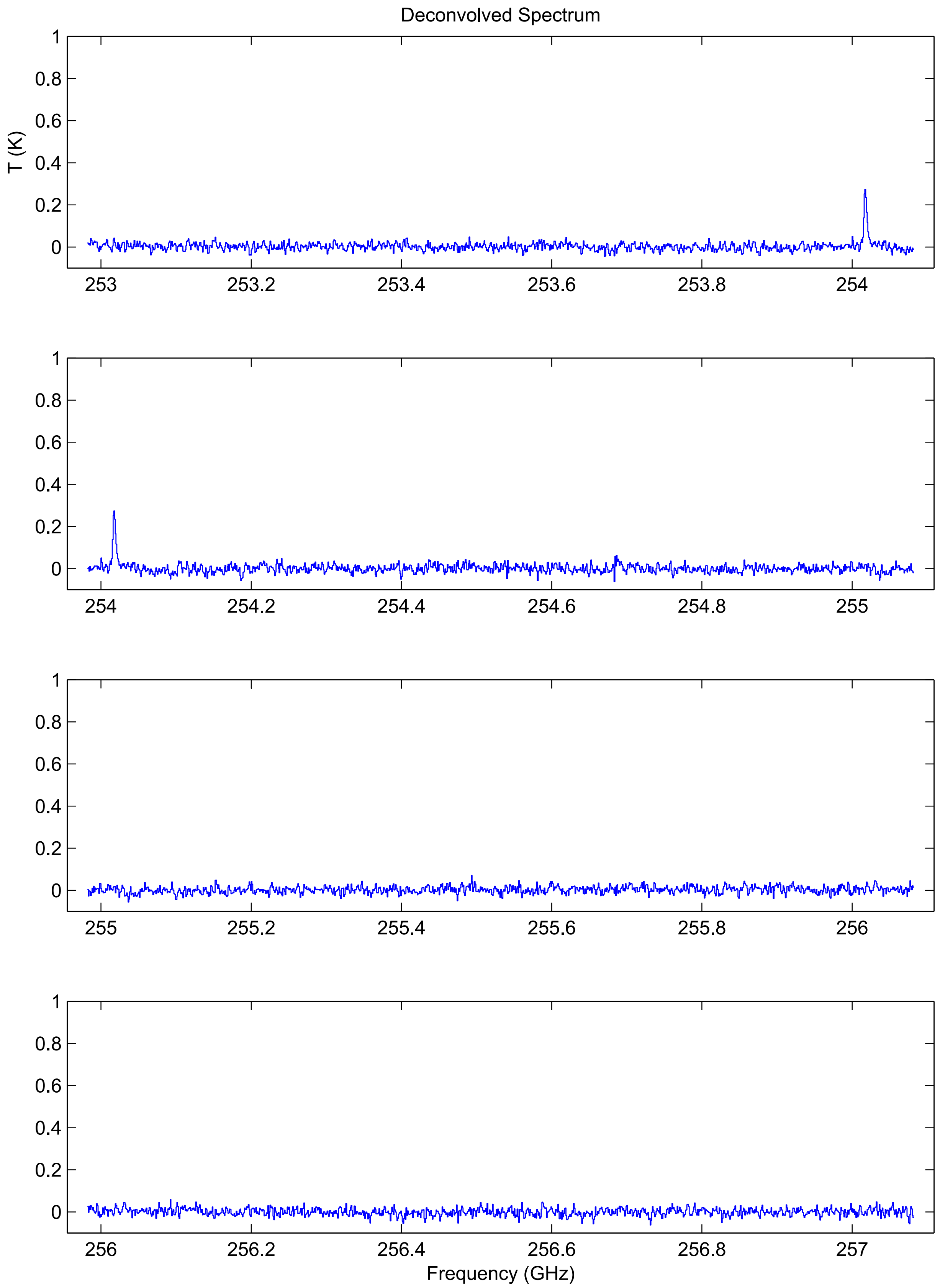

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
233
233.2
233.4
233.6
233.8
234
234.2
234.4
234.6
234.8
235
235.2
235.4
235.6
235.8
236
236.2
236.4
236.6
236.8
237
Frequency (GHz)

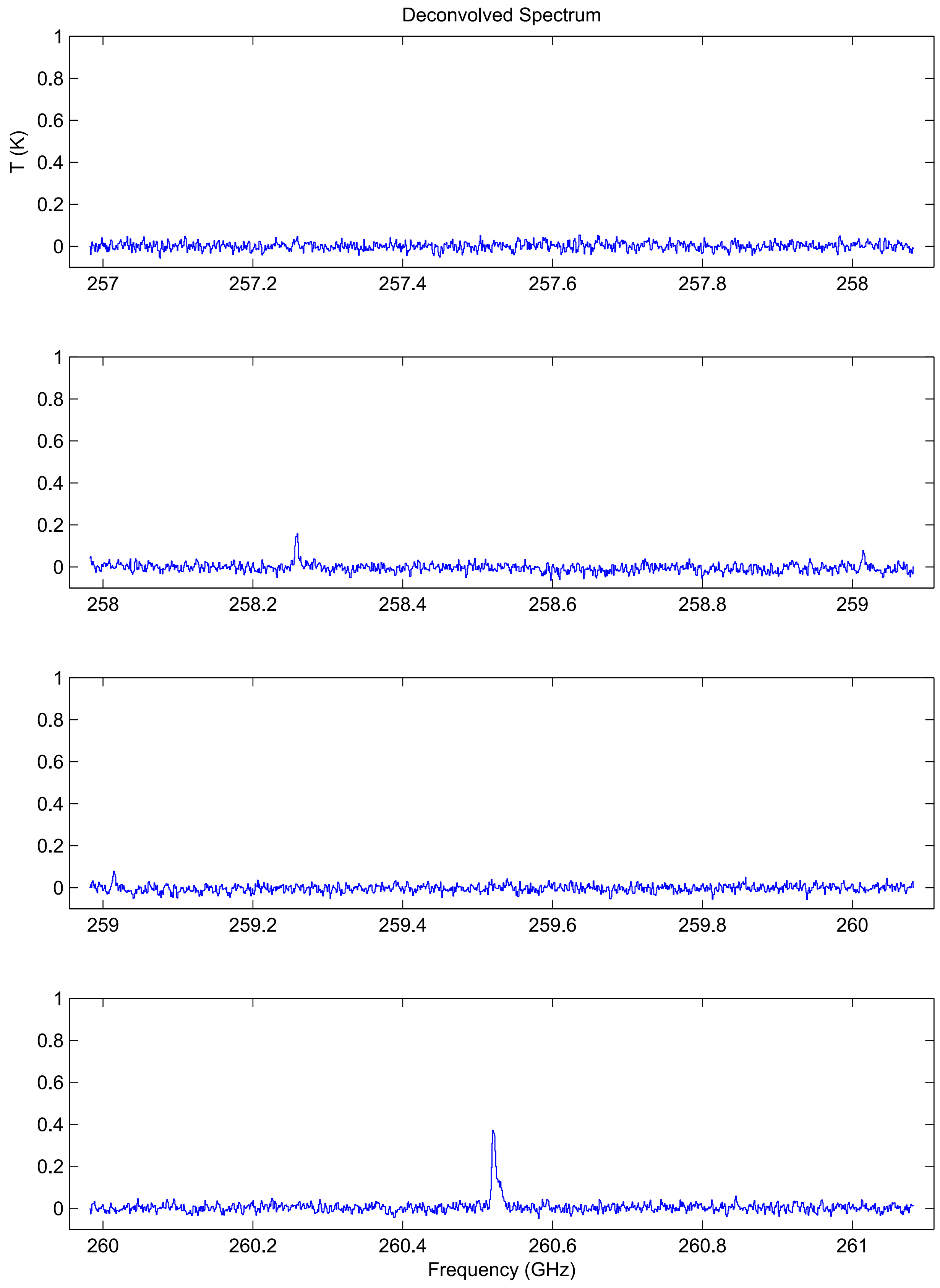

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
237
237.2
237.4
237.6
237.8
238
238.2
238.4
238.6
238.8
239
239.2
239.4
239.6
239.8
240
240.2
240.4
240.6
240.8
241
Frequency (GHz)

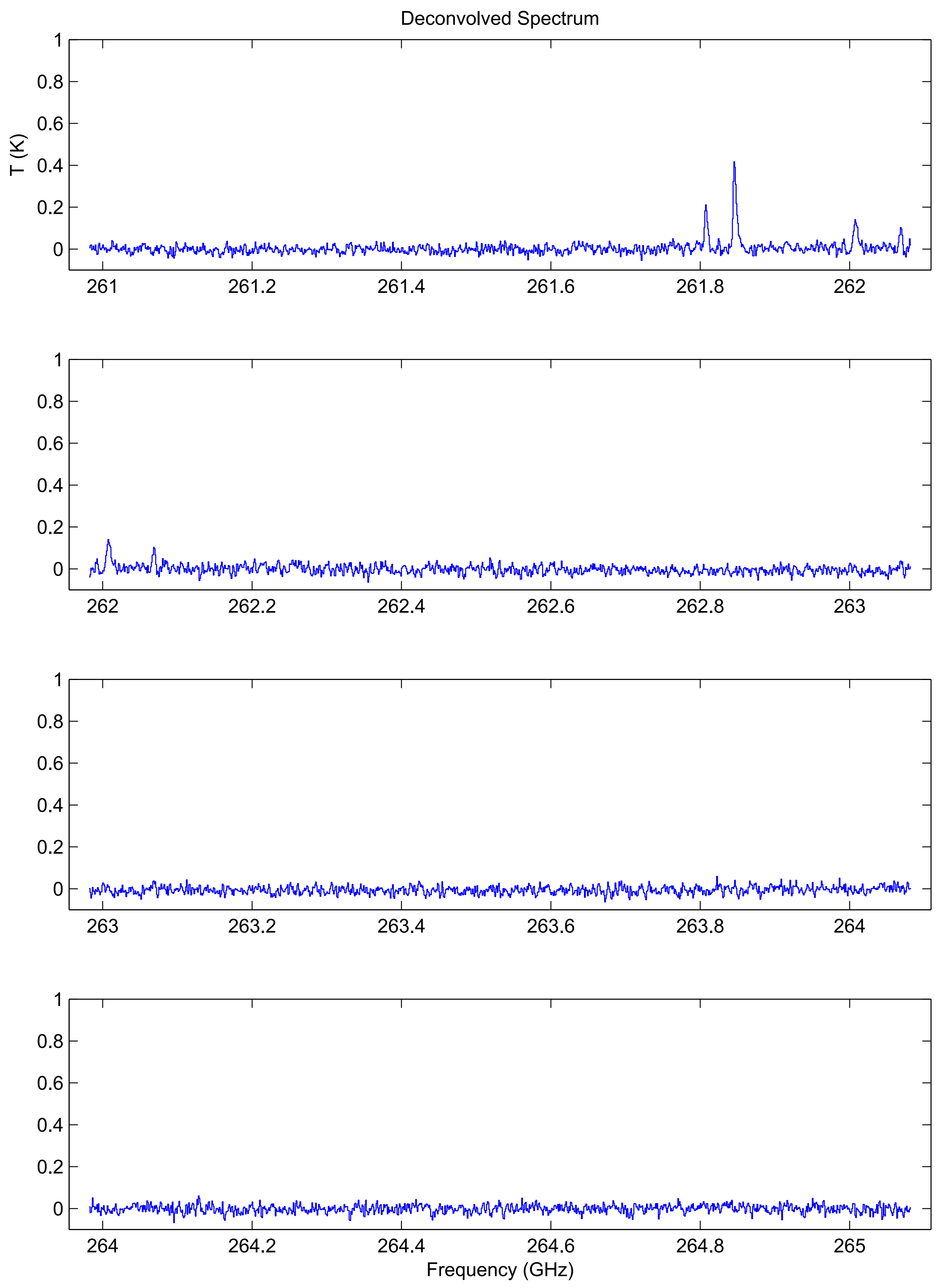

Deconvolved Spectrum
T (K)
Frequency (GHz)
241
241.2
241.4
241.6
241.8
242
241.75
241.8
241.85
242.2
242.4
242.6
242.8
243
243.2
243.4
243.6
243.8
244
244.2
244.4
244.6
244.8
245
244.9
244.95


Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
245
245.2
245.4
245.6
245.8
246
246.2
246.4
246.6
246.8
247
247.2
247.4
247.6
247.8
248
248.2
248.4
248.6
248.8
249
Frequency (GHz)

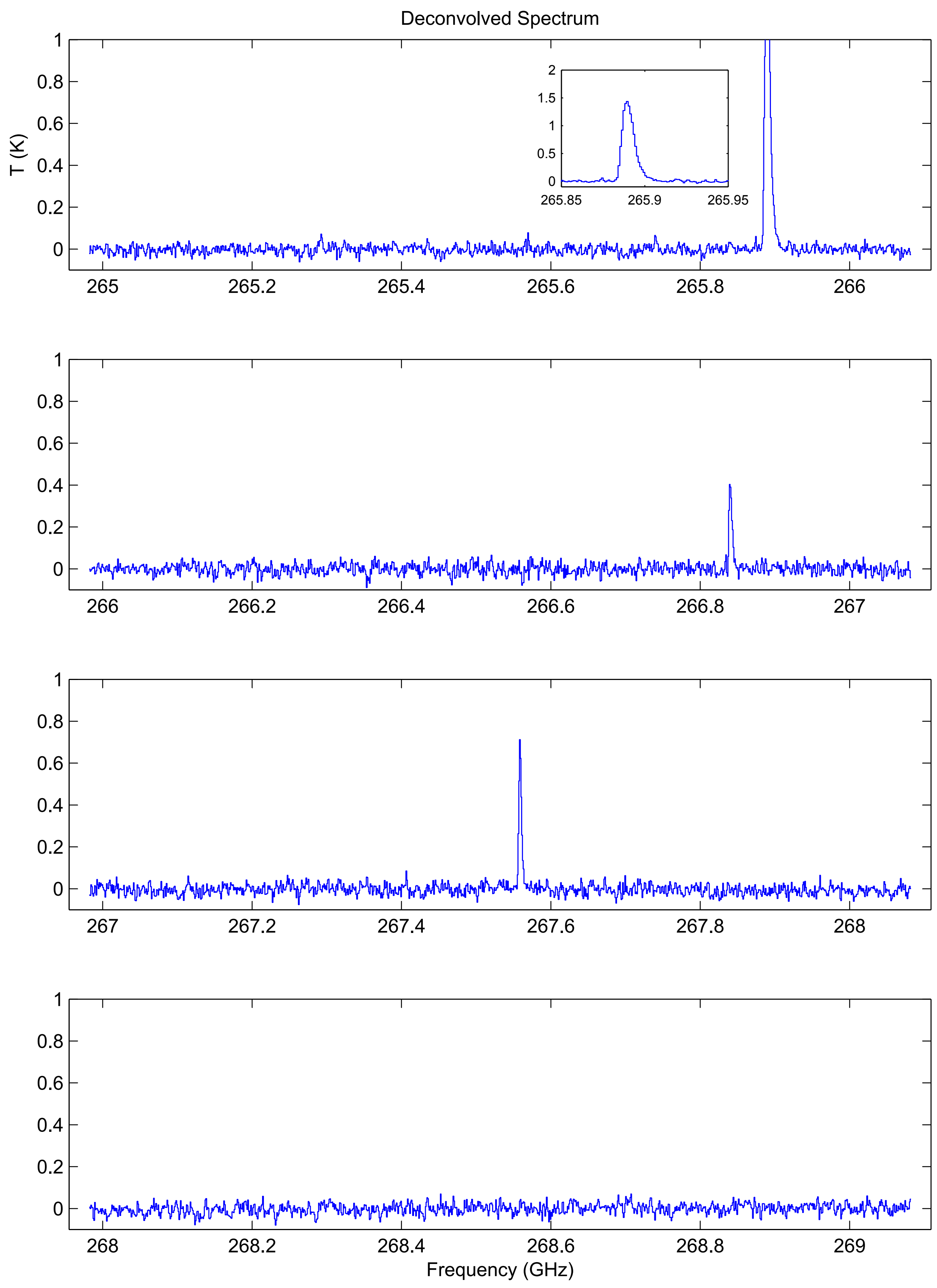

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
249
249.2
249.4
249.6
249.8
250
250.2
250.4
250.6
250.8
251
251.2
251.4
251.6
251.8
252
252.2
252.4
252.6
252.8
253
Frequency (GHz)

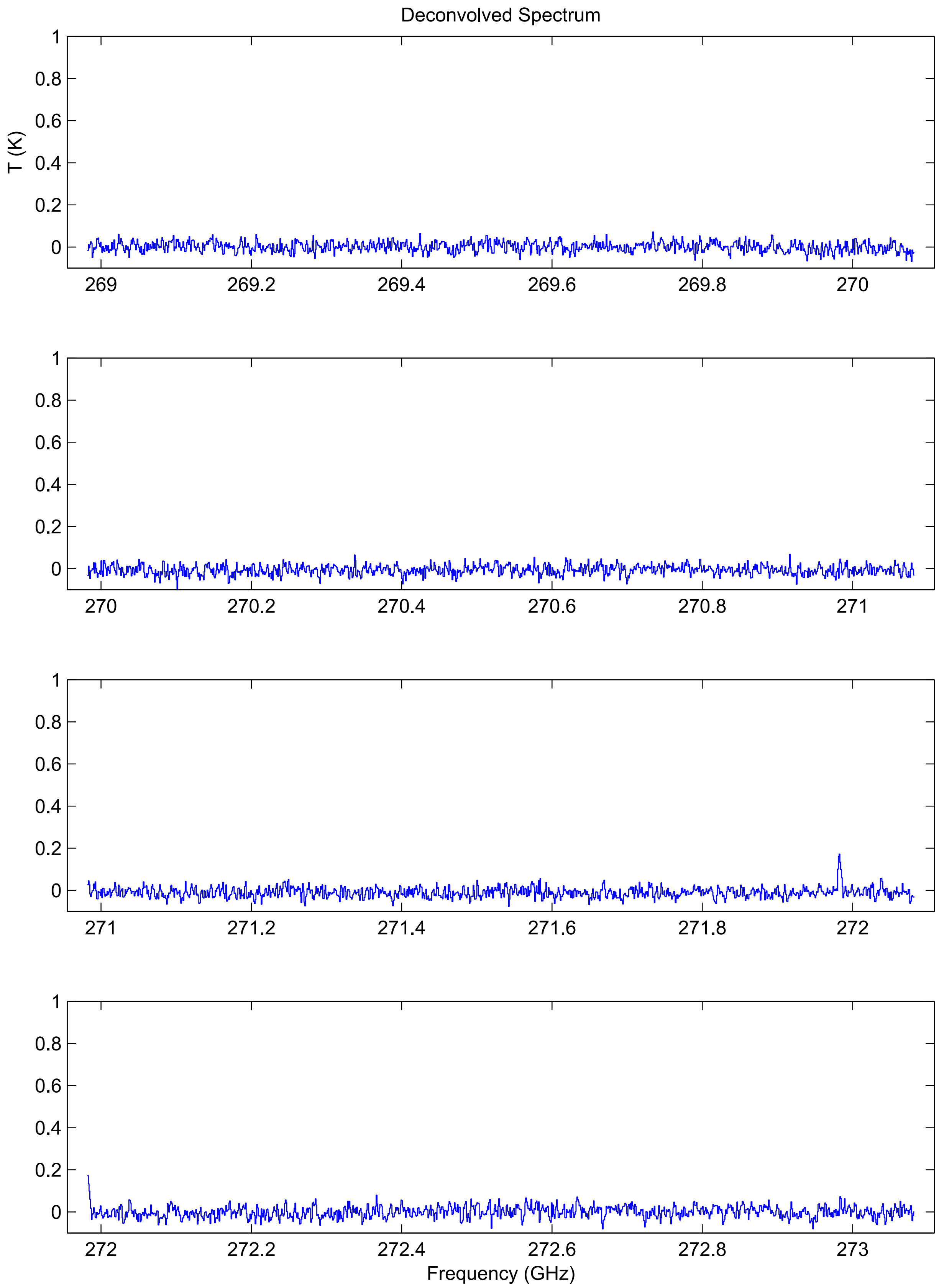

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
253
253.2
253.4
253.6
253.8
254
254.2
254.4
254.6
254.8
255
255.2
255.4
255.6
255.8
256
256.2
256.4
256.6
256.8
257
Frequency (GHz)

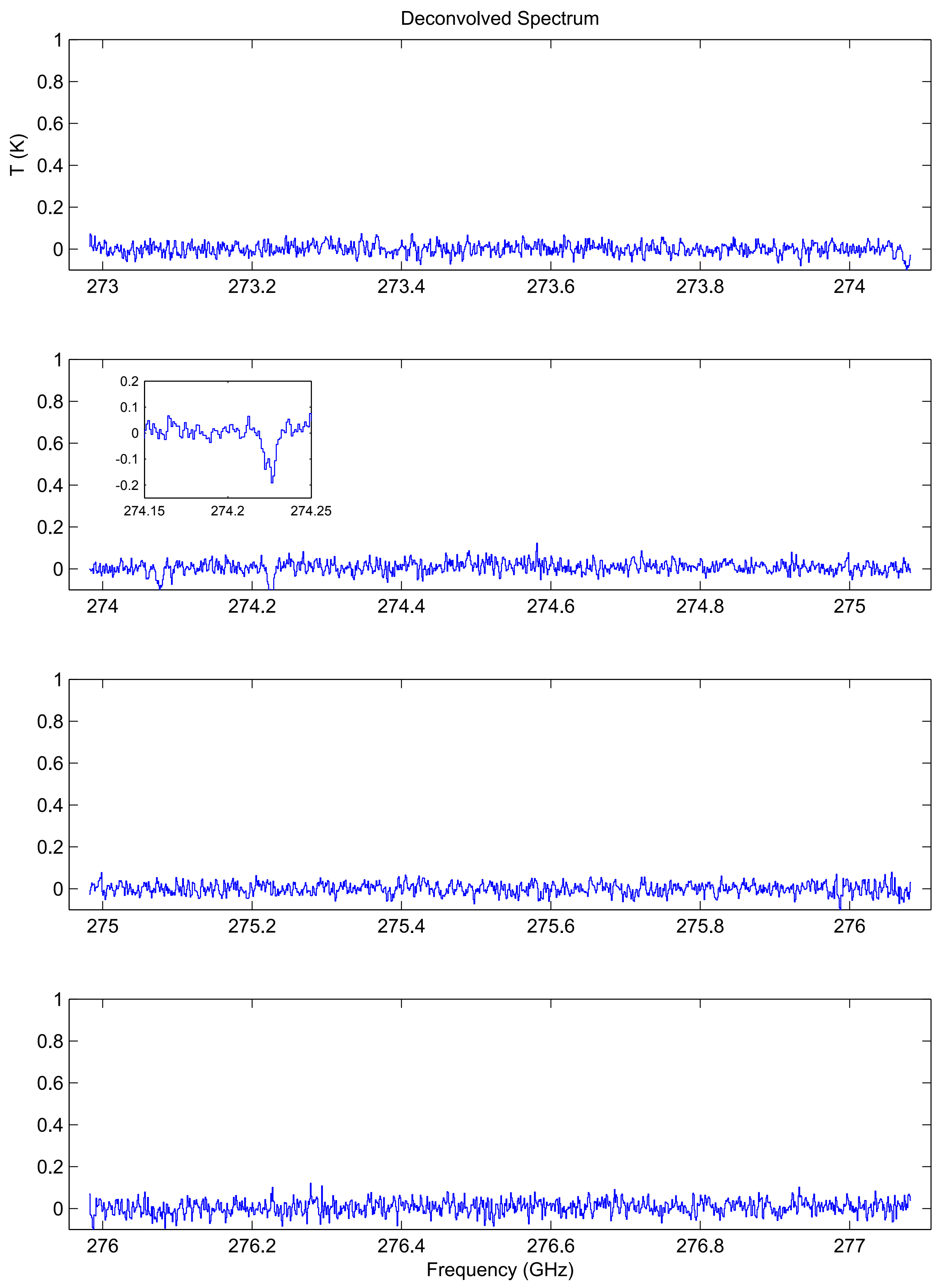

Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
257
257.2
257.4
257.6
257.8
258
258.2
258.4
258.6
258.8
259
259.2
259.4
259.6
259.8
260
260.2
260.4
260.6
260.8
261
Frequency (GHz)

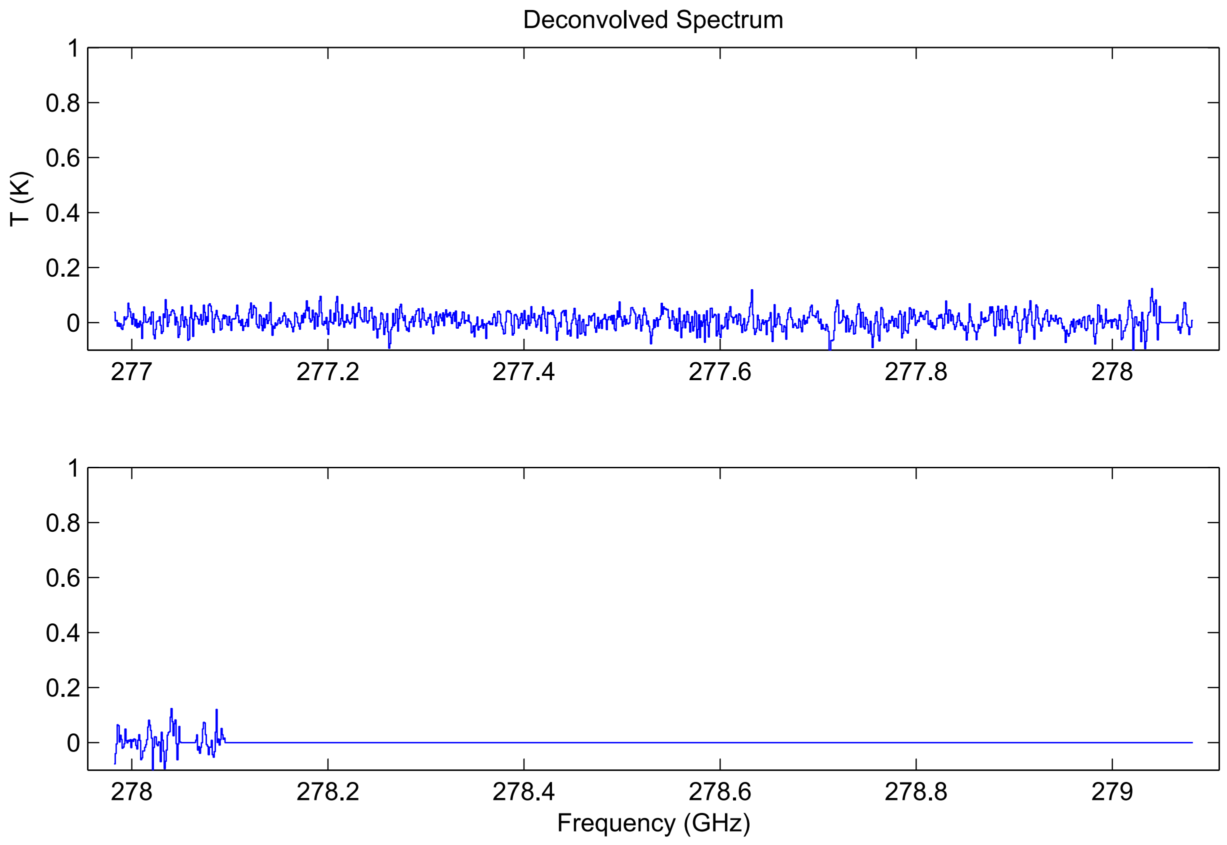


Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
261
261.2
261.4
261.6
261.8
262
262.2
262.4
262.6
262.8
263
263.2
263.4
263.6
263.8
264
264.2
264.4
264.6
264.8
265
Frequency (GHz)


Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
265
265.2
265.4
265.6
265.8
266
2
1.5
1
0.5
0
265.85
265.9
265.95
266.2
266.4
266.6
266.8
267
267.2
267.4
267.6
267.8
268
268.2
268.4
268.6
268.8
269
Frequency (GHz)


Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
269
269.2
269.4
269.6
269.8
270
270.2
270.4
270.6
270.8
271
271.2
271.4
271.6
271.8
272
272.2
272.4
272.6
272.8
273
Frequency (GHz)


Deconvolved Spectrum
T (K)
Frequency (GHz)
273
273.2
273.4
273.6
273.8
274
274.15
274.2
274.25
274.4
274.6
274.8
275
275.2
275.4
275.6
275.8
276
276.2
276.4
276.6
276.8
277


Deconvolved Spectrum
T (K)
1
0.8
0.6
0.4
0.2
0
277
277.2
277.4
277.6
277.8
278
278
278.2
278.4
278.6
278.8
279
Frequency (GHz)

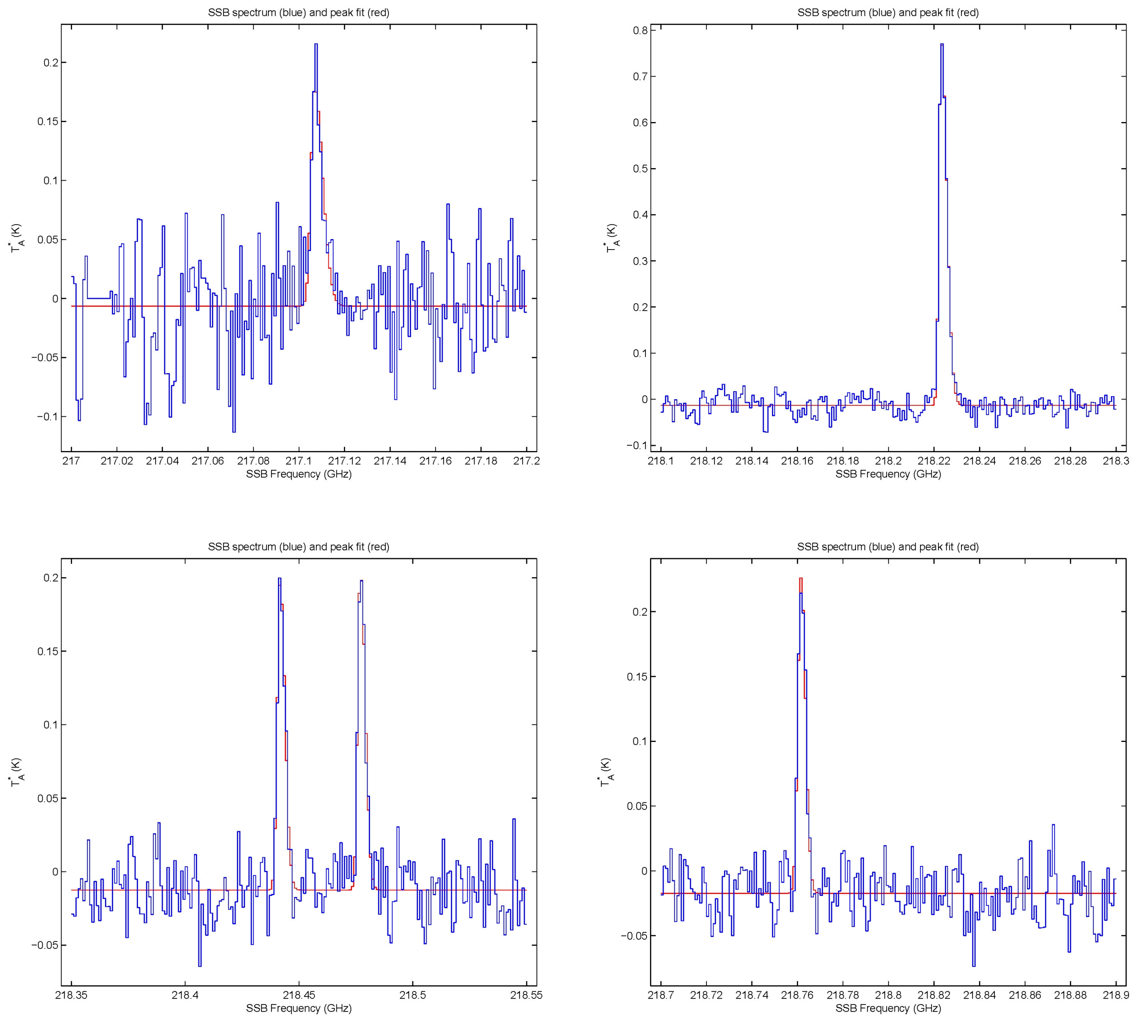


Figure D.1: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

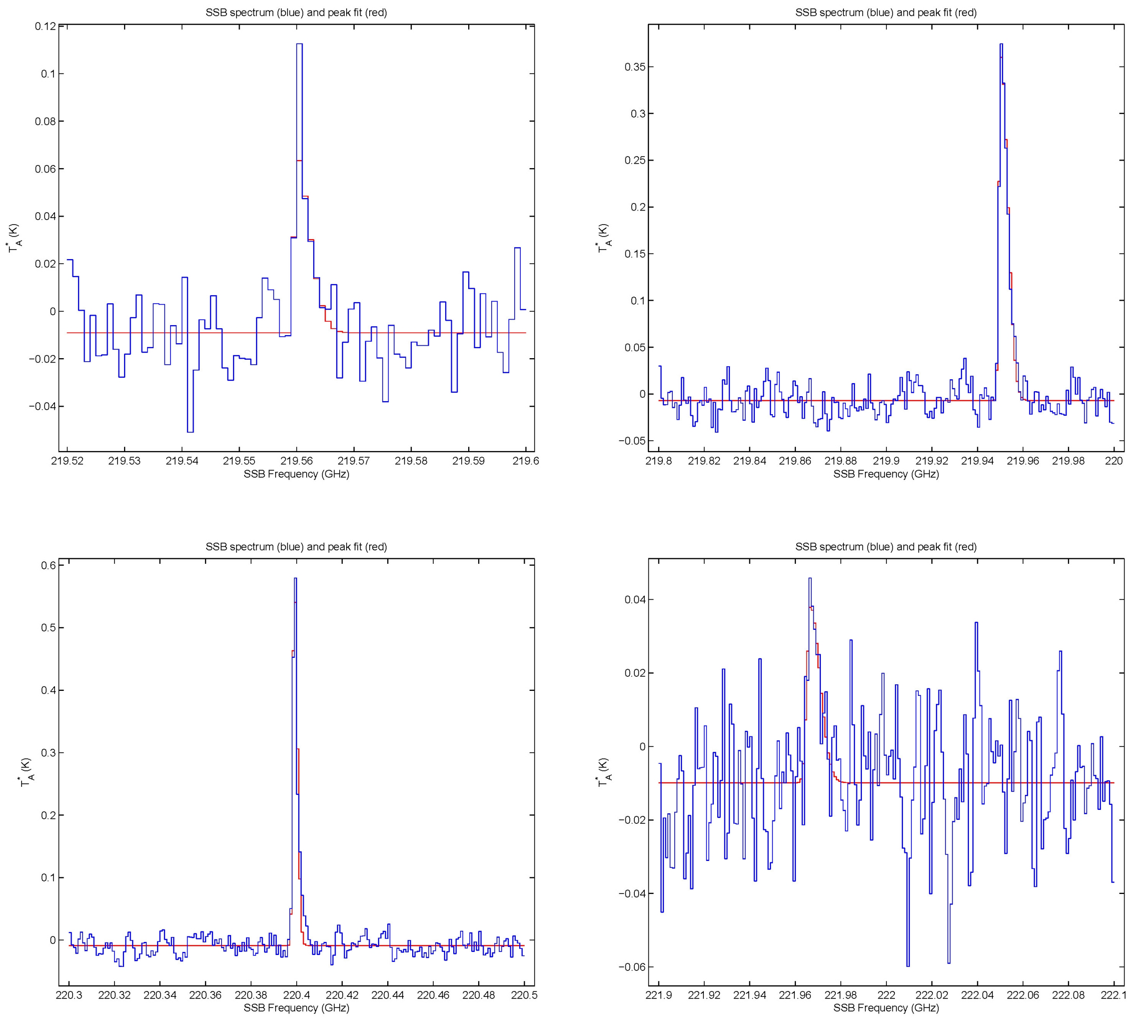


Figure D.2: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

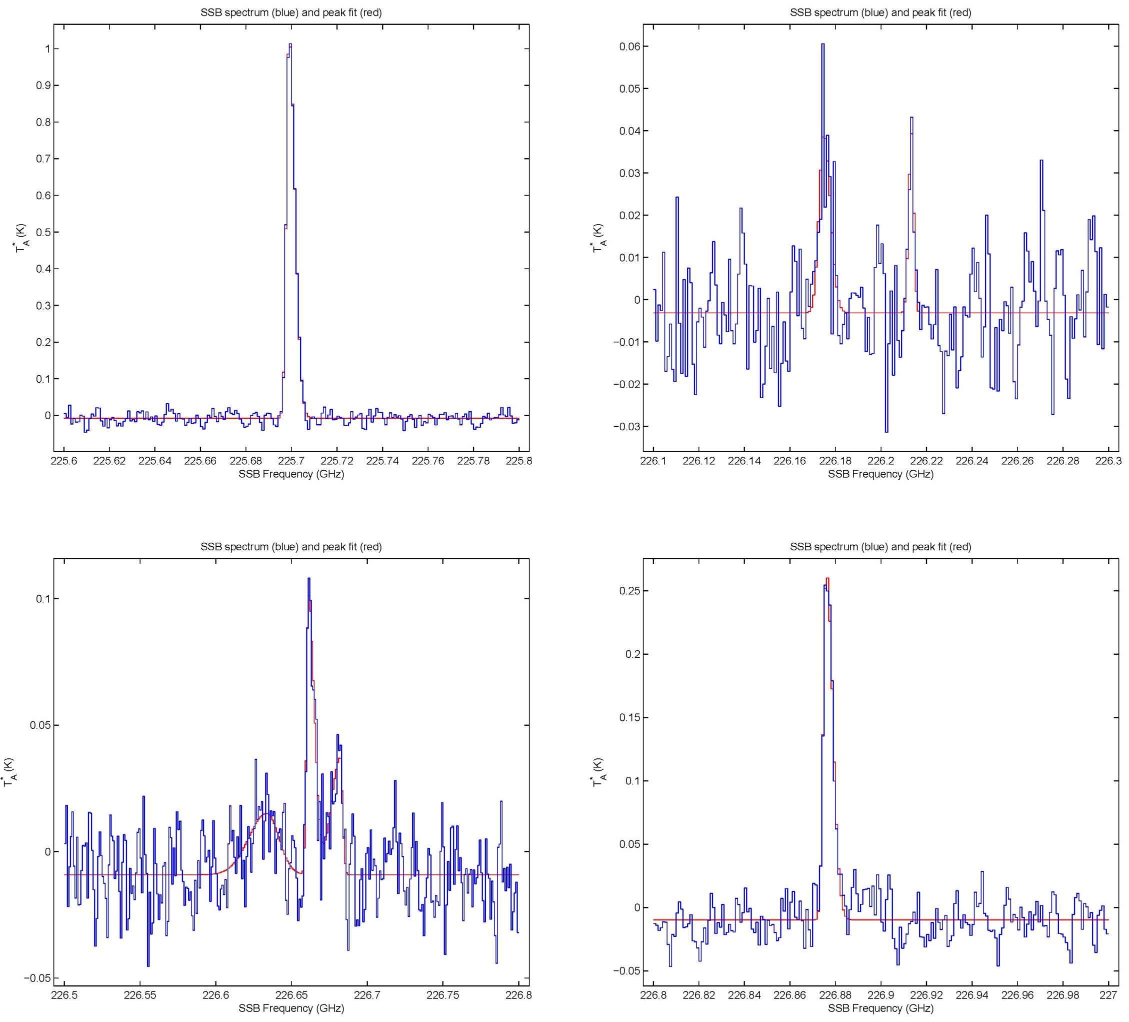


Figure D.3: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

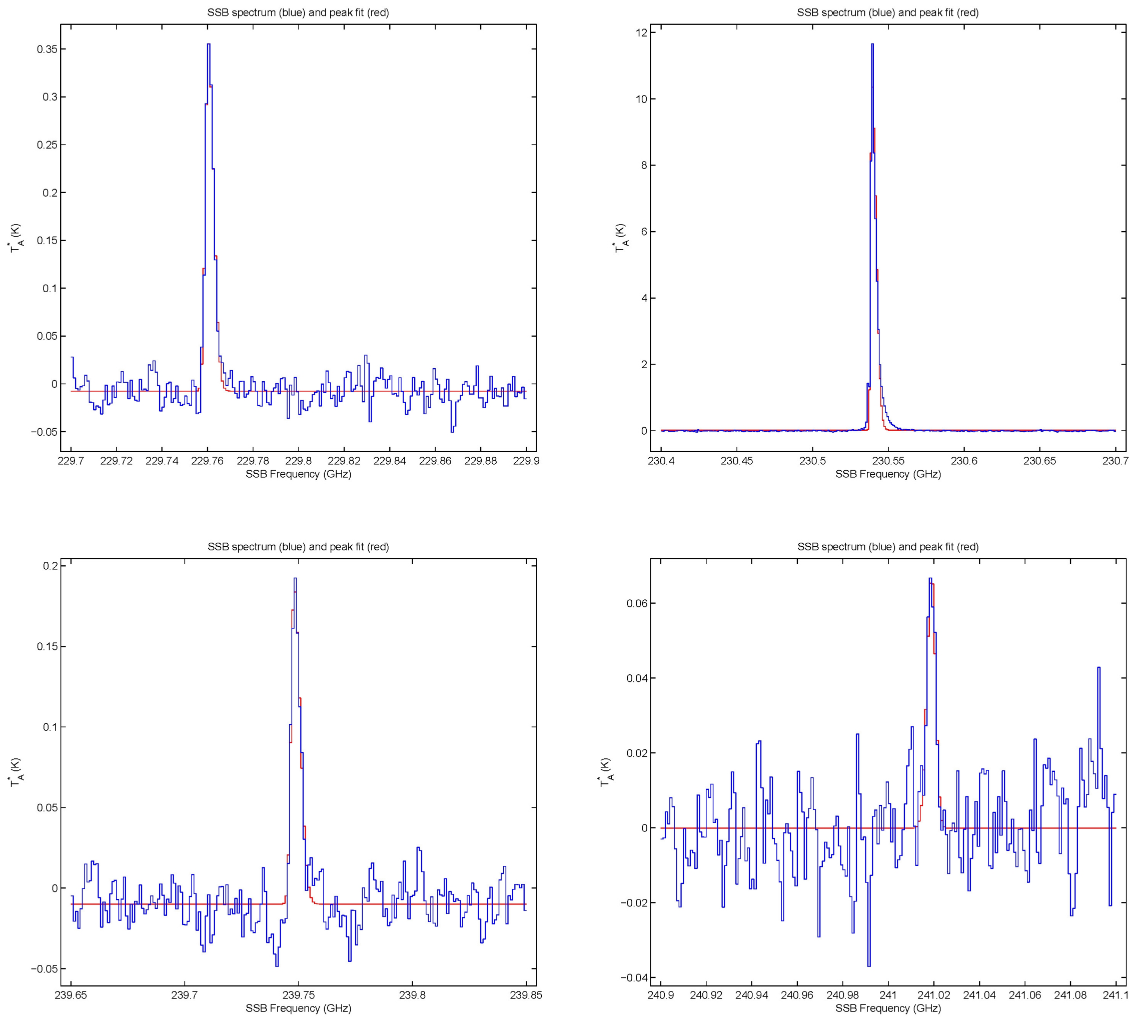


Figure D.4: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

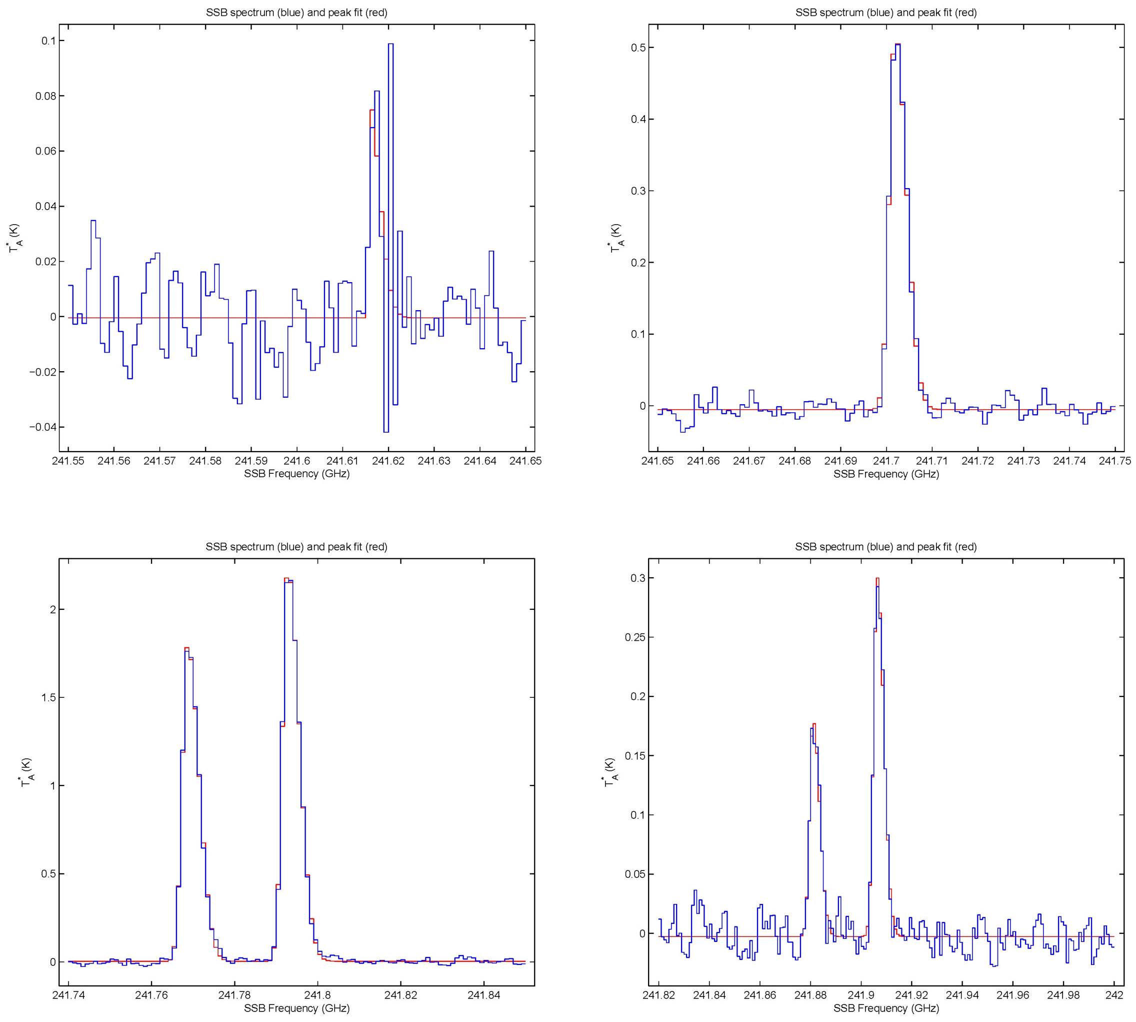


Figure D.5: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

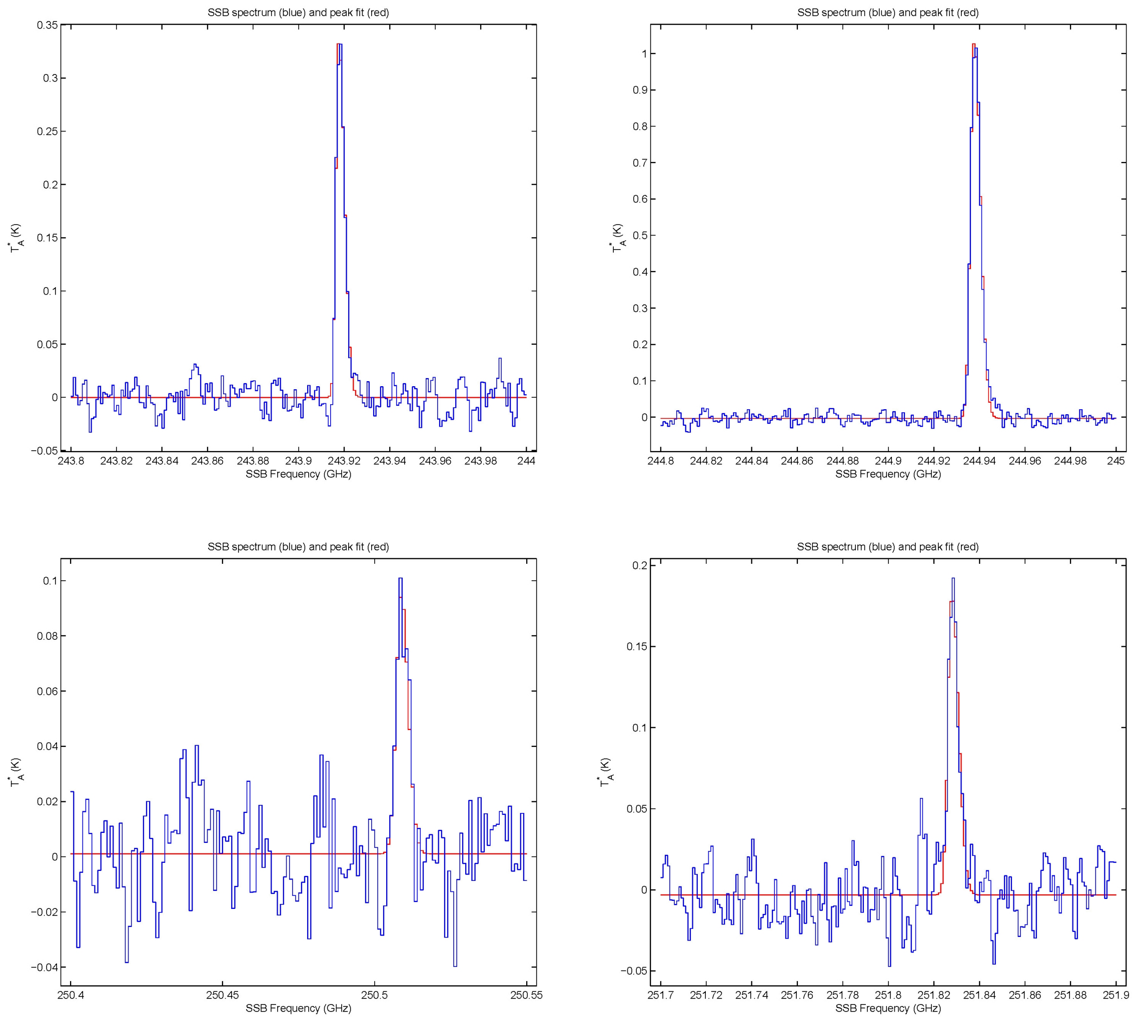
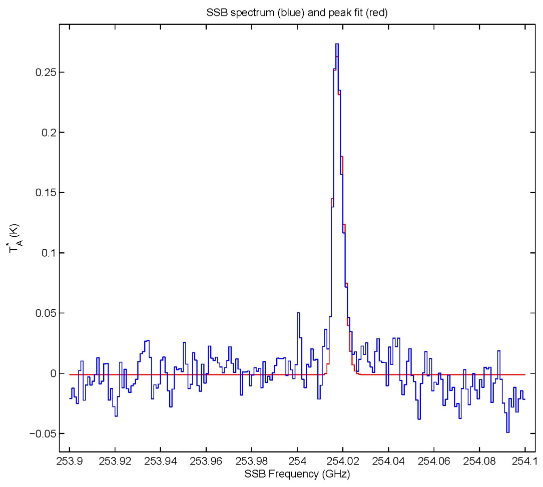
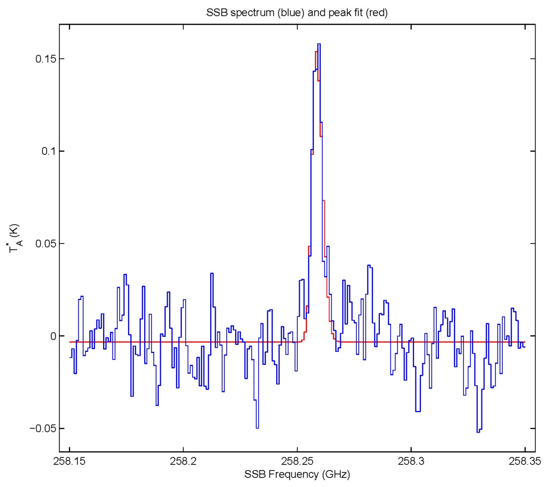
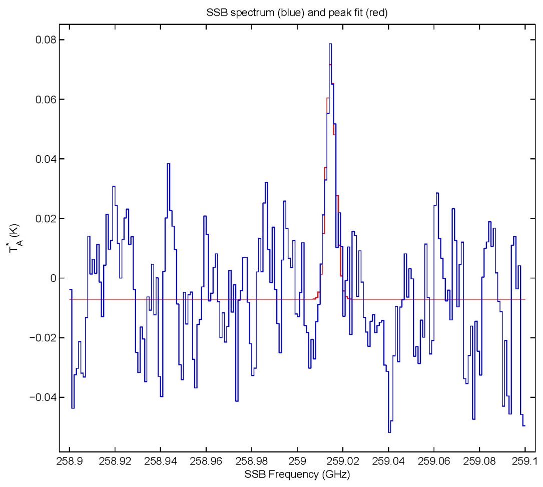


Figure D.6: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)







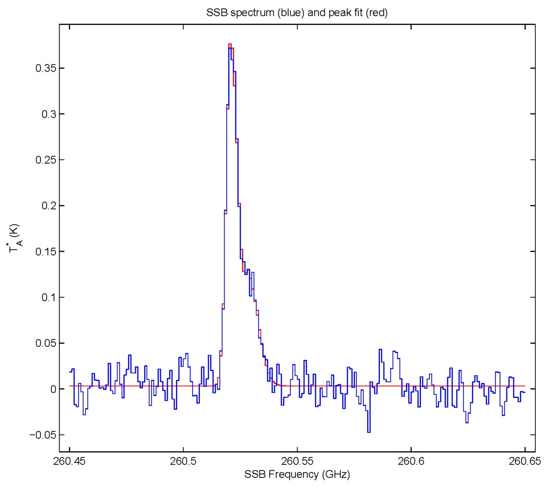


Figure D.7: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

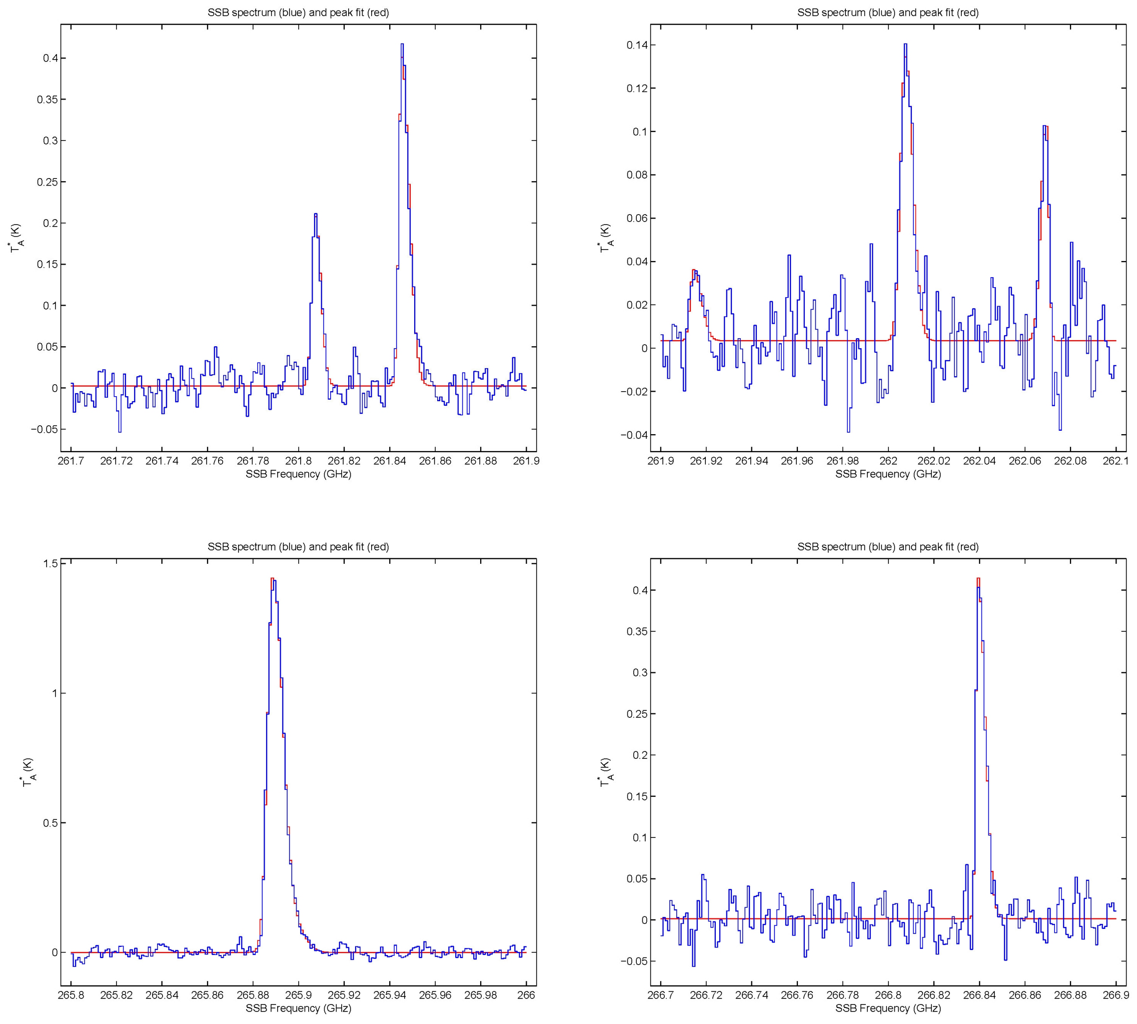


Figure D.8: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

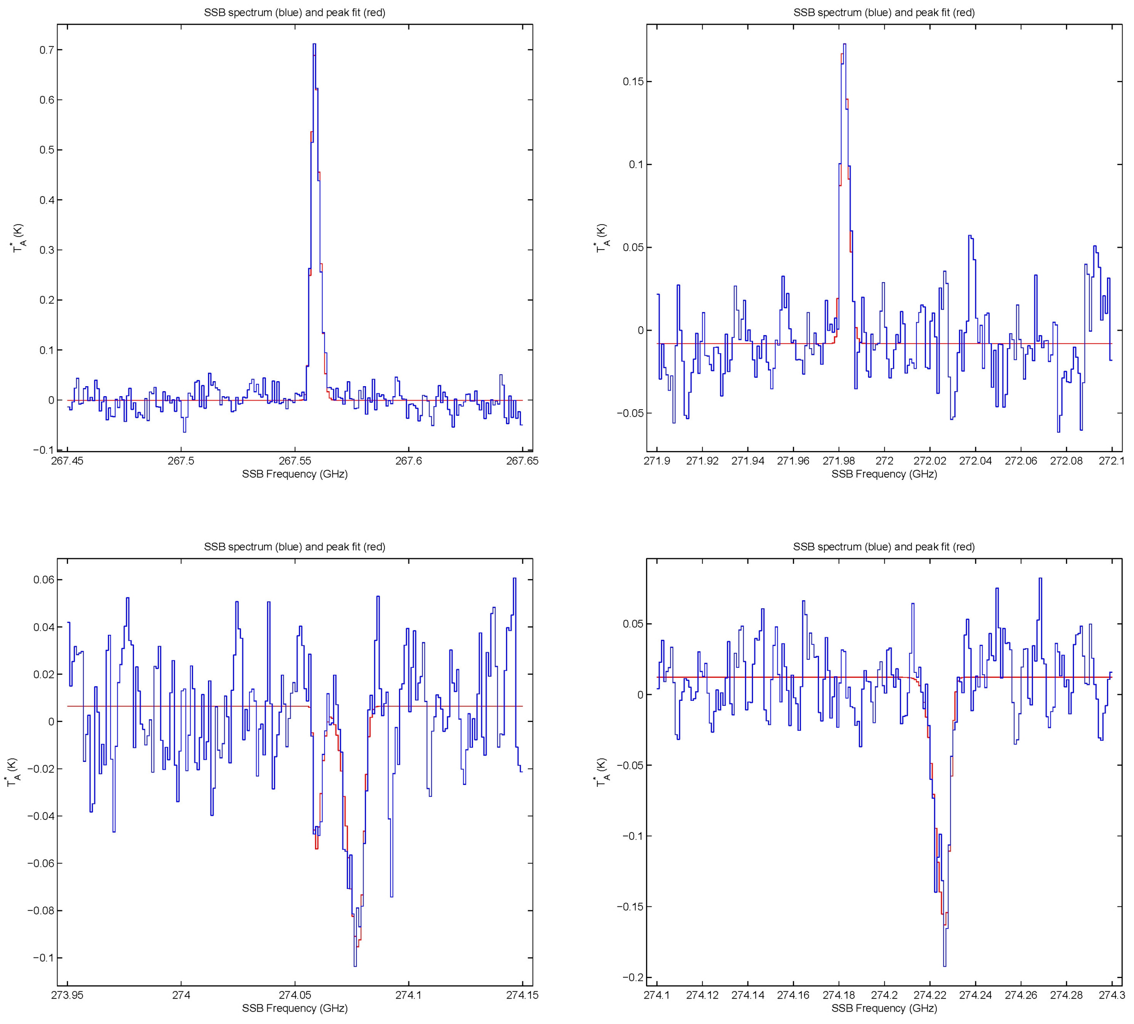


Figure D.9: Peak fits used for L1157 analysis in Section 6.4. (See Figures D.1 through D.9.)

Bibliography

P. Andre, D. Ward-Thompson, and M. Barsony. Submillimeter Continuum Observations of Rho Ophiuchi A - The Candidate Protostar VLA 1623 and Prestellar Clumps. *The Astrophysical Journal*, 406:122–141, Mar. 1993.

P. Andre, D. Ward-Thompson, and M. Barsony. From Prestellar Cores to Protostars: The Initial Conditions of Star Formation. In *Protostars and Planets IV*, page 59, May 2000.

L. W. Avery and M. Chiao. Enhanced Chemical Abundances in the L1157 Outflow: SiO and CH3OH Observations. *The Astrophysical Journal*, 463:642, June 1996.

R. Bachiller and M. P. Gutierrez. Shock Chemistry in the Young Bipolar Outflow L1157. *The Astrophysical Journal*, 487:L93, Sept. 1997.

R. Bachiller, J. Martin-Pintado, and A. Fuente. High-Velocity Hot Ammonia in Bipolar Outflows. *The Astrophysical Journal*, 417:L45, Nov. 1993.

R. Bachiller, M. P. Gutierrez, M. S. N. Kumar, and M. Tafalla. Chemically Active Outflow L 1157. *Astronomy and Astrophysics*, 372:899–912, June 2001.

A. W. Blain, I. Smail, R. J. Ivison, J. P. Kneib, and D. T. Frayer. Submillimeter Galaxies. *Physics Reports*, 369(2):111–176, Oct. 2002. ISSN 0370-1573. doi: 10.1016/S0370-1573(02)00134-5.

G. A. Blake, C. R. Masson, T. G. Phillips, and E. C. Sutton. The Rotational Emission-Line Spectrum of Orion A Between 247 and 263 GHz. *The Astrophysical Journal Supplement Series*, 60:357–374, Jan. 1986.

G. A. Blake, E. C. Sutton, C. R. Masson, and T. G. Phillips. Molecular Abundances in OMC-1 - The Chemical Composition of Interstellar Molecular Clouds and the Influence of Massive Star Formation. *The Astrophysical Journal*, 315:621–645, Apr. 1987.

G. A. Blake, E. F. van Dishoek, D. J. Jansen, T. D. Groesbeck, and L. G. Mundy. Molecular Abundances and Low-Mass Star Formation. 1: Si- and S-Bearing Species Toward IRAS 16293-2422. *The Astrophysical Journal*, 428:680, 1994. ISSN 0004-637X. doi: 10.1086/174278.

A. C. A. Boogert. *The Interplay between Dust, Gas, Ice, and Protostars*. PhD thesis, University of Groningen, Mar. 1999. available at `http://irs.ub.rug.nl/ppn/180933256`.

C. M. Caves. Quantum Limits on Noise in Linear Amplifiers. *Physical Review D*, 26(8):1817, Oct. 1982. doi: 10.1103/PhysRevD.26.1817.

S. C. Chapman, A. W. Blain, R. J. Ivison, and I. R. Smail. A Median Redshift of 2.4 for Galaxies Bright at Submillimetre Wavelengths. *Nature*, 422(6933):695–698, Apr. 2003. ISSN 0028-0836. doi: 10.1038/nature01540.

A. A. Clerk, M. H. Devoret, S. M. Girvin, F. Marquardt, and R. J. Schoelkopf. Introduction to Quantum Noise, Measurement, and Amplification. *Reviews of Modern Physics*, 82(2): 1155, Apr. 2010. doi: 10.1103/RevModPhys.82.1155.

C. Comito and P. Schilke. Reconstructing Reality: Strategies for Sideband Deconvolution. *Astronomy and Astrophysics*, 395(1):357–371, 2002. ISSN 0004-6361. doi: 10.1051/0004-6361:20021277.

C. Comito, P. Schilke, T. G. Phillips, D. C. Lis, F. Motte, and D. Mehringer. A Molecular Line Survey of Orion KL in the 350 Micron Band. *The Astrophysical Journal Supplement Series*, 156:127–167, Feb. 2005.

R. Genzel and J. Stutzki. The Orion Molecular Cloud and Star-Forming Region. *Annual Review of Astronomy and Astrophysics*, 27(1):41–85, 1989. ISSN 0066-4146. doi: 10.1146/annurev.aa.27.090189.000353.

T. D. Groesbeck, T. G. Phillips, and G. A. Blake. The Molecular Emission-Line Spectrum of IRC +10216 between 330 and 358 GHz. *The Astrophysical Journal Supplement Series*, 94: 147–162, Aug. 1994.

F. Gueth, S. Guilloteau, A. Dutrey, and R. Bachiller. Structure and Kinematics of a Protostar: MM-Interferometry of L 1157. *Astronomy and Astrophysics*, 323:943–952, July 1997.

A. I. Harris and J. Zmuidzinas. A Wideband Lag Correlator for Heterodyne Spectroscopy of Broad Astronomical and Atmospheric Spectral Lines. *Review of Scientific Instruments*, 72(2):1531, 2001. ISSN 00346748. doi: 10.1063/1.1334629.

J. Horn, O. Siebertz, F. Schmulling, C. Kunz, R. Schieder, and G. Winnewisser. A 4×1 GHz Array Acousto-Optical Spectrometer. *Experimental Astronomy*, 9:17–38, 1999.

J. W. Kooi, A. Kovacs, M. C. Sumner, G. Chattopadhyay, R. Ceria, D. Miller, B. Bumble, H. G. Leduc, J. A. Stern, and T. G. Phillips. A 275–425 GHz Tunerless Waveguide Receiver Based on AlN-Barrier SIS Technology. *IEEE Transactions on Microwave Theory Techniques*, 55:2086–2096, Oct. 2007.

J. D. Kraus. *Radio Astronomy*. Cygnus-Quasar Books, 2nd edition, June 1986. ISBN 1882484002.

M. L. Kutner and B. L. Ulich. Recommendations for Calibration of Millimeter-Wavelength Spectral Line Data. *Astrophysical Journal*, 250:341–348, Nov. 1981.

C. J. Lada. Star Formation - From OB Associations to Protostars. In M. Peimbert and J. Jugaku, editors, *Star Forming Regions*, pages 1–18, 1987.

L. W. Looney, J. J. Tobin, and W. Kwon. A Flattened Protostellar Envelope in Absorption around L1157. *The Astrophysical Journal*, 670(2):L131–L134, 2007. ISSN 0004-637X. doi: 10.1086/524361.

F. J. Lovas. NIST Recommended Rest Frequencies for Observed Interstellar Molecular Microwave Transitions - 2002 Revision. *Journal of Physical and Chemical Reference Data*, 33 (1):177, 2004. ISSN 00472689. doi: 10.1063/1.1633275. Available at `http://www.nist.gov/data/PDFfiles/jpcrd644.pdf`.

H. Mikami, T. Umemoto, S. Yamamoto, and S. Saito. Detection of SiO Emission in the L1157 Dark Cloud. *The Astrophysical Journal*, 392:L87–L90, June 1992.

B. J. Naylor. *Broadband Millimeter-Wave Spectroscopy with Z-Spec: An Unbiased Molecular-Line Survey of the Starburst Galaxy M82*. PhD thesis, California Institute of Technology, 2008. Available at `http://thesis.library.caltech.edu/2208/`.

R. Peng. *CSO Observing Manual*. 2002. Available at `http://www.cso.caltech.edu/Obs_manual/manual.pdf`.

T. G. Phillips and D. P. Woody. Millimeter- and Submillimeter-Wave Receivers. *Annual Review of Astronomy and Astrophysics*, 20:285–321, 1982.

W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in FORTRAN 77: The Art of Scientific Computing*. Cambridge University Press, 2nd edition, Sept. 1992. ISBN 052143064X.

F. Rice, M. Sumner, J. Zmuidzinas, R. Hu, H. G. LeDuc, A. I. Harris, and D. Miller. SIS Mixer Design for a Broadband Millimeter Spectrometer Suitable for Rapid Line Surveys and Redshift Determinations. In *Proceedings of SPIE*, pages 301–311, Waikoloa, HI, USA, 2003. doi: 10.1117/12.459710.

K. Rohlfs and T. L. Wilson. *Tools of Radio Astronomy*. Springer, 4th rev. and enlarged edition, Oct. 2003. ISBN 3540403876.

P. Schilke, T. G. Phillips, and D. M. Mehringer. FIRST Wide Band Submillimeter Line Surveys. In A. Wilson, editor, *The Far Infrared and Submillimetre Universe*, volume 401, page 73, Aug. 1997.

P. Schilke, D. J. Benford, T. R. Hunter, D. C. Lis, and T. G. Phillips. A Line Survey of Orion KL from 607 to 725 GHz. *The Astrophysical Journal Supplement Series*, 132(2):281–364, 2001. ISSN 0067-0049. doi: 10.1086/318951.

F. L. Schoier, J. K. Jorgensen, E. F. van Dishoeck, and G. A. Blake. Does IRAS 16293-2422 Have a Hot Core? Chemical Inventory and Abundance Changes in its Protostellar Environment. *Astronomy and Astrophysics*, 390:1001–1021, Aug. 2002.

F. Shu, J. Najita, D. Galli, E. Ostriker, and S. Lizano. The Collapse of Clouds and the Formation and Evolution of Stars and Disks. In *Protostars and Planets III*, pages 3–45, 1993.

F. H. Shu, J. R. Najita, H. Shang, and Z. Li. X-Winds Theory and Observations. In *Protostars and Planets IV*, page 789, May 2000.

G. L. Squires. *Practical Physics*. Cambridge University Press, 3rd edition, Oct. 1985. ISBN 0521270952.

E. C. Sutton, G. A. Blake, C. R. Masson, and T. G. Phillips. Molecular Line Survey of Orion A from 215 to 247 GHz. *The Astrophysical Journal Supplement Series*, 58:341–378, July 1985.

E. C. Sutton, R. Peng, W. C. Danchi, P. A. Jaminet, G. Sandell, and A. P. G. Russell. The Distribution of Molecules in the Core of OMC-1. *The Astrophysical Journal Supplement Series*, 97:455, 1995. ISSN 0067-0049. doi: 10.1086/192147.

B. Tercero, J. Cernicharo, J. R. Pardo, and J. R. Goicoechea. A Line Confusion Limited Millimeter Survey of Orion KL (I): Sulfur Carbon Chains. *Astronomy and Astrophysics*, 517:A96, 2010. doi: 10.1051/0004-6361/200913501.

B. Tercero, L. Vincent, J. Cernicharo, S. Viti, and N. Marcelino. A Line-Confusion Limited Millimeter Survey of Orion KL - II. Silicon-Bearing Species. *Astronomy & Astrophysics*, 528:22, 2011. doi: 10.1051/0004-6361/201015837.

B. L. Ulich and R. W. Haas. Absolute Calibration of Millimeter-Wavelength Spectral Lines. *Astrophysical Journal Supplement Series*, 30:247–258, Mar. 1976.

T. Umemoto, T. Iwata, Y. Fukui, H. Mikami, S. Yamamoto, O. Kameya, and N. Hirano. The Outflow in the L1157 Dark Cloud - Evidence for Shock Heating of the Interacting Gas. *The Astrophysical Journal*, 392:L83–L86, June 1992.

E. F. van Dishoeck, G. A. Blake, D. J. Jansen, and T. D. Groesbeck. Molecular Abundances and Low-Mass Star Formation. II. Organic and Deuterated Species toward IRAS 16293-2422. *The Astrophysical Journal*, 447:760, 1995. ISSN 0004-637X. doi: 10.1086/175915.

J. Ward, F. Rice, G. Chattopadhyay, and J. Zmuidzinas. SuperMix: A Flexible Software Library for High-Frequency Circuit Simulation, Including SIS Mixers and Superconducting Elements. In *Proceedings of the Tenth International Symposium on Space Terahertz Technology*, Mar. 1999.

E. W. Weisstein. Mathworld, 2010. A Wolfram Web Resource, available at `http://mathworld.wolfram.com/GaussianFunction.html` (last accessed Sept. 29, 2010).

M. J. Wengler. Submillimeter-Wave Detection with Superconducting Tunnel Diodes. *Proceedings of the IEEE*, 80(11):1810–1826, Nov. 1992.

S. L. Widicus Weaver. *Rotational Spectroscopy and Observational Astronomy of Prebiotic Molecules*. PhD thesis, California Institute of Technology, 2005. Available at `http://thesis.library.caltech.edu/1835/`.