UNIVERSITY OF CALIFORNIA

Los Angeles

High-Precision Calibration and Reduction Techniques for Molecular Line Emission in Radio Astronomy & Their Application to the Galactic Chemical Evolution of Silicon

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Geochemistry

by

Nathaniel Nels Monson

2019

ABSTRACT OF THE DISSERTATION

High-Precision Calibration and Reduction Techniques for Molecular Line Emission in Radio Astronomy & Their Application to the Galactic Chemical Evolution of Silicon

by

Nathaniel Nels Monson

Doctor of Philosophy in Geochemistry

University of California, Los Angeles, 2019

Professor Edward Donald Young, Co-Chair

Professor Mark R. Morris, Co-Chair

A report on the relative abundances of the three stable isotopes of silicon, ^{28}Si, ^{29}Si and ^{30}Si, across the Galaxy using the $v = 0, J = 1 \rightarrow 0$ transition of silicon monoxide. The chosen sources represent a range in Galactocentric radii (R_{GC}) from 0 to 9.8 kpc. The high spectral resolution and sensitivity afforded by the Robert C. Byrd Green Bank Telescope (GBT) permit isotope ratios to be corrected for optical depth, using a novel method developed as part of this study. The optical-depth-corrected data indicate that the secondary-to-primary silicon isotope ratios [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] vary much less than predicted on the basis of other stable isotope ratios in the Galaxy. Indeed, there is *no* detectable variation in Si isotope ratios with R_{GC}. This lack of an isotope ratio gradient stands in stark contrast to the monotonically decreasing trend with R_{GC} in published secondary-to-primary oxygen isotope ratios. These results, when considered in the context of the expectations for galactic chemical evolution, suggest that the reported isotope ratio trends in oxygen, and likely carbon as well, may be in error and require further investigation. The methods developed in this study for SiO isotopologue ratio measurements are equally applicable to Galactic oxygen, carbon and nitrogen isotope ratio measurements, and should prove useful for future observations of these isotope systems.

The dissertation of Nathaniel Nels Monson is approved.

University of California, Los Angeles

2019

To my friend Juno…

For her spirit, which never faltered; even when mine did.

TABLE OF CONTENTS

1 Introduction **1**

2 Previous Work **4**

2.1 Stellar Metallicity 4

2.2 Galactic Chemical Evolution of Light Stable Isotopes 5

2.2.1 Primary Nuclides 6

2.2.2 Secondary Nuclides 8

2.3 Previous Observations 12

3 Testing GCE using Silicon **16**

4 Observations **20**

5 Calibration and Data Reduction **23**

5.1 Flux Calibration 23

5.2 Baselines 25

5.2.1 Interfering Lines 25

5.3 Optical Depth Effects 26

5.3.1 The Shape Parameter 27

5.4 Extracting Column Densities from Line Intensities 35

5.4.1 Correcting for Finite Optical Depth 36

5.4.2 Excitation Effects 37

5.5 Evaluation of Uncertainties 40

6 Results **42**

7 Discussion **47**

7.1 Secondary/Primary Si Isotope Ratios 47

7.2 Secondary/Secondary Si Isotope Ratios 49

8 Future Work **52**

9 Conclusions **54**

Appendix A **55**

A.1 Preface 55

A.2 Calculating Abundance Ratios 57

A.2.1 Accounting for Beam-Weighting 57

A.2.2 Correcting for Optical Depth 60

A.3 The Principle of Detailed Balance 65

A.3.1 The Excitation Temperature 65

A.3.2 The Critical Density 68

10 HYDRA User's Guide & Cookbook **70**

10.1 User Agreement 72

11 The HYDRA Program Suite **73**

11.1 Preface 73

12 HYDRA Cookbook **75**

12.1 Installing HYDRA 75

12.2 Setting Up HYDRA 77

12.2.1 Setdir.pro 77

12.2.2 Getfits.pro 78

12.3 Managing Data With HYDRA 80
12.3.1 Gather.pro 80
12.4 Calibrating Data With HYDRA 83
12.4.1 Dreamcatcher.pro 83
12.5 Reducing Data With HYDRA 86
12.5.1 Outerlimits.pro 86
12.5.2 Polybase.pro 87
12.5.3 Dreamweaver.pro 89

13 How HYDRA Calibrates Data 92
13.1 Calibration Sources 92
13.2 The Noise Diodes 94
13.3 Calibrating Science Data 95

14 HYDRA Source Code 98
14.1 Hydra.pro 98
14.2 Getfits.pro 106
14.3 Setdir.pro 116
14.4 Gather.pro 120
14.5 Extract.pro 139
14.6 Dreamcatcher.pro 151
14.7 Outerlimits.pro 183
14.8 Polybase.pro 198
14.9 Dreamweaver.pro 214
14.10 Looper.pro 235
14.11 Accessory Programs And Definitions 245

LIST OF FIGURES

2.1 Stellar Metallicity vs. Galactocentric Radius 5

2.2 δ'^{18}O vs. R_{GC} 13

2.3 δ'^{17}O vs. δ'^{18}O in the ISM 15

3.1 Predicted Dependence of O and Si Isotopic Ratios on Stellar Metallicity. 17

3.2 Silicon Isotope Ratios in Mainstream Presolar SiC Grains 19

5.1 Interfering H(83)δ Emission in W51e2 25

5.2 Full vs. Optically-Thin Approximation of the Opacity Term 28

5.3 Fractional Error in the Optically-Thin Approximation of the Opacity Term . . . 28

5.4 Scaled ^{29,30}SiO Emission Lines in L1157 and GCM-0.13-0.08 29

5.5 Effects of Optical Depth on Emission Line Profiles 30

5.6 Quasilinear Relation Between Γ'' and τ_0 34

5.7 Polynomial Fit to $d\tau_0/d\Gamma''$ 34

5.8 Contours of SiO Isotopologue Ratio Errors from RADEX 39

6.1 Calibrated and Baseline-Corrected ^{28,29,30}SiO Emission Lines 43

6.2 Uncorrected SiO Silicon Isotope Abundance Ratios 45

6.3 Optical-Depth-Corrected SiO Silicon Isotope Abundance Ratios 45

6.4 δ'^{29}Si vs. R_{gc} 46

LIST OF TABLES

4.1 List of Sources and Observed SiO v=0, J=1 $\rightarrow$ 0 Emission Lines 21

6.1 Corrected and Uncorrected SiO Isotopologue Abundance Ratios 44

6.2 Corrected and Uncorrected SiO δ'^{29}Si and δ'^{30}Si Values 44

ACKNOWLEDGMENTS

This work was supported by a grant from the NASA Origins program (NNX10AH35G) and from the NASA Emerging Worlds program (NNX17AE78G). Chapters one through nine of this work constitute a revised and expanded version of Monson et al. (2017).

My thanks and appreciation are extended to (in no particular order): Mark Morris, Edward Young, Benjamin Zuckerman, David Jewitt, Kevin McKeegan, Lauri Holbrook, Issaku Kohl, Coleen Young, Françoise Quéval, Ron Maddalena, Alexandra Schneider, Bonnie Zucker, Lucas Monson, Margaret Maldonado, Steven Monson, Siyi Xu, the staff at the Green Bank Observatory, the Arizona Radio Observatory, the W. M. Keck Observatory and the European Southern Observatory.

VITA

2011	B.A. (Biochemistry), University of Colorado Boulder. Boulder, Colorado
2013–2015	Teaching Assistant: Earth, Planetary & Space Sciences, UCLA. Los Angeles, California.
2015–2016	Teaching Assistant: Physics and Astronomy, UCLA.
2016	M.A. (Geochemistry),University of California, Los Angeles.
2016	Visiting Research Assistant: The European Southern Observatory. Garching bei München, Germany.
2016–present	Research Assistant: Earth, Planetary & Space Sciences, UCLA. Los Angeles, California

PUBLICATIONS

Monson, N. N., Morris, M. R., and Young, E. D. Uniform Silicon Isotope Ratios Across the Milky Way Galaxy. *The Astrophysical Journal*, 839:123, April 2017. doi:10.3847/1538-4357/aa67e6.

CHAPTER 1

Introduction

The utility of interstellar isotope abundance ratios as diagnostic tools for probing metallicity variations across the Galaxy was realized well over thirty years ago (e.g. Frerking et al., 1980; Linke et al., 1977; Penzias, 1981a,b; Wilson et al., 1981; Wolff, 1980). In conjunction with models for galactic chemical evolution (GCE), the distribution of stable isotopes with distance from the Galactic center provides a quantitative probe into stellar nucleosynthesis (Henkel et al., 2014; Prantzos et al., 1996; Timmes et al., 1995), galaxy formation and evolution (Kobayashi et al., 2006; Martín et al., 2009, 2010; Prantzos et al., 1996; Spite et al., 2006) and levels of heterogeneity in the interstellar medium (ISM) (Lugaro et al., 2003; Nittler, 2005; Young et al., 2011). For these purposes, Galactocentric radius (R_{GC})[1] serves as a proxy for time because stellar processing of material increases with both time and decreasing R_{GC}.

Galactic chemical evolution of light stable isotopes leads to shifts in isotope ratios over time in what should be broadly predictable ways. The shifts are especially pronounced for ratios of secondary nuclides to primary nuclides, and the details of the process are clearer where two or more such ratios are available. For example, Riquelme et al. (2010a,b) used $[^{12}\mathrm{C}]/[^{13}\mathrm{C}]$ ratios to trace the infall of more chemically primitive gas in the halo and the outer disk into the Galactic center region. Their study illustrates the utility of GCE of stable isotopes as tracers of gas motions over time. When studied as functions of R_{GC}, isotopic abundance ratios delineate the extent of stellar processing within the Galaxy, and serve as signposts for chemical variations with time (Clayton, 1984; Clayton and Pantelaki, 1986; Kobayashi et al., 2011; Prantzos, 2008; Prantzos et al., 1996; Timmes et al., 1995).

[1] i.e. distance from the Galactic center

The ratios of secondary to primary silicon isotopes in the Solar System are surprisingly low compared with older presolar SiC grains found in meteorites. This aberration has been used as possible evidence for extraordinary enrichment of the primary isotope ^{28}Si by supernovae in the region in which the Sun formed (Alexander and Nittler, 1999; Young et al., 2011). In order to verify or contravene the idea that the birth environment of the Solar System was atypical of the Galaxy 4.6 Gyr before present, one needs an understanding of how the relevant stable isotope ratios have evolved with time and place in the Galaxy (i.e. over the last 4.6 Gyrs). We need to understand whether the Solar System formed from typical material and by typical processes, or whether it formed in some atypical environment and/or by unusual processes. In other words, *are we normal* in the context of the isotopic evolution of our local Galactic environs?

The Solar System is expected to be representative of the interstellar medium at $R_{\rm GC} \approx 8$ kpc, 4.6 Gyr before present, in the absence of some extraordinary local enrichment processes during its formation. GCE over this time interval must be accounted for before drawing comparisons between the Solar System and the present-day ISM in a meaningful way. Studies of isotope ratios vs. $R_{\rm GC}$ therefore provide the context for interpreting the significance of Solar System stable isotope ratios. If the Solar System fits with the general picture of secular variations in stable isotope ratios in the Galaxy, then it would suggest that the Solar System formed under conditions that were unremarkable. Conversely, if the Solar System exhibits significant departures from the averages expected from an analysis of the distribution and evolution of isotopes in the Galaxy, then one would be impelled to search for extraordinary circumstances to explain these departures in isotopic abundances (enrichment by nearby supernovae is the most obvious example). Isotopes of silicon are thought to be an example of the latter case but a firm Galactic reference frame for interpretation of the Solar data is not in place.

Studies of isotope ratios vs. Galactocentric radius therefore help place the Solar System in a Galactic perspective, and provide the context for interpreting the significance of stable isotope ratios in the Solar System. The first step is to establish the baseline isotopic characteristics of the Galaxy, which in turn involves defining the mean distributions of isotope ratios as functions of R_{GC}, and establishing the magnitude of dispersion about this trend. This is the objective of the present study.

Secondary isotopes are typically rare, often comprising a couple percent or less of the total abundance of the element of interest. As a result, signal-to-noise ratio (SNR) for emission lines from rare isotopologues are typically poor and contribute significantly to the error budgets. Measurements of the abundance ratios of the three stable isotopes of silicon by Wolff (1980) and soon after by Penzias (1981a), using the $v = 0, J = 2 \rightarrow 1$ and $J = 3 \rightarrow 2$ lines of SiO, were hampered by low signal-to-noise. However, modern cryogenic high-electron-mobility transistor (HEMT) amplifiers and superconducting tunnel junction (STJ) mixers provide such exceptionally low noise that sensitivities have been increased in excess of an order of magnitude since those early studies. The data reported by Penzias (1981a) and Wolff (1980) have statistical errors as high as 40%. The measurements of $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ ratios based on the $v = 0, J = 1 \rightarrow 0$ transitions of SiO reported herein have 1σ statistical errors one tenth of that value. In part for this reason, stable isotope abundance ratios as tracers for variations in the degree of astration across the Galaxy should see a resurgence (e.g., Adande and Ziurys, 2012; Henkel et al., 2014).

CHAPTER 2

Previous Work

2.1 Stellar Metallicity

To first order, metallicity is known to increase towards the Galactic center. Recent studies of H II regions (Balser et al., 2011) and classical Cepheids (Pedicelli et al., 2009) define a clear gradient in metallicity in the Galactic disk (Figure 2.1). This gradient is traced by iron and the α-elements O, Ca, Si, Mg and Ti, all relative to H. However these gradients are slight, and measurements indicate that [α/H] and [Fe/H] [1]deviate from the Solar value by little more than 0.5 dex as far out as 16 kpc from the Galactic center.

For the outer disk ($R_{\rm GC} > 8$ kpc), [Fe/H] ratios in Cepheids increase with decreasing $R_{\rm GC}$ with a gradient of ≈ -0.05 dex kpc^{-1}. Between 8 and 4 kpc from the Galactic Center the [Fe/H] gradient is observed to be ≈ -0.02 dex kpc^{-1} with a maximum of 0.3 ± 0.1 dex at $R_{\rm GC} \approx 4$ kpc (Figure 2.1). Inside of $R_{\rm GC} = 4$ kpc the [Fe/H] trend seems to "roll over," and decrease as $R_{\rm GC}$ decreases, as evidenced by luminous blue variables (LBVs) and red supergiants (RSGs) in the Galactic center having observed values of [Fe/H] within error of the Solar value (Cunha et al., 2007). Measurements of oxygen and the other α-elements exhibit slightly more variability, with estimated maxima in [O/H] and [α/H] at the Galactic center of ≈ 0.5 dex, and values near 0.2 dex being more typical (Davies et al., 2009; Najarro et al., 2009). These results imply that the outer disk evolves somewhat differently than the inner disk and Galactic center.

[1]Brackets are used to distinguish atomic abundances from mass abundances. In this context, [x/y] represents the atomic abundance ratio of element x to element y in dex units

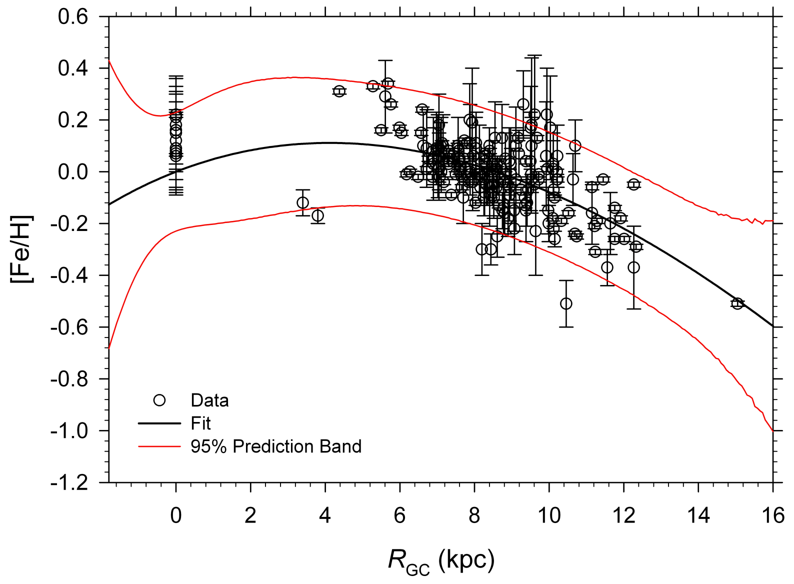


Figure 2.1: Stellar metallicity (with respect to the Solar value) vs. Galactocentric radius with a fit ($\pm$ 95% confidence) for illustrative purposes. Data represent Cepheids, Quintuplet cluster LBVs and the Scutum Red Supergiant clusters (Andrievsky et al., 2002a,b,c; Luck et al., 2006; Pedicelli et al., 2009).

2.2 Galactic Chemical Evolution of Light Stable Isotopes

Ratios of the stable isotopes of oxygen, carbon and nitrogen have been used as tracers of GCE. Galactic chemical evolution leads to time dependent shifts in the isotopic makeup of the Galaxy, and this variability that results from the varying rates of astration and production should also be evident in variations with R_{GC}. Isotope ratios have the advantage of normalizing some of the vagaries associated with production terms for the elements. Tinsley (1975) provided a basis for a mathematical formalism to describe the GCE of nuclides. In this treatment and those that followed, the rate of nuclide growth in the Galaxy is expressed as a function of both the star formation rate (SFR) within the Galaxy, $\Psi(t)$, and the initial mass function (IMF), $\phi(m)$, for the stellar sources.

2.2.1 Primary Nuclides

Nucleosynthetic processes requiring only primordial matter are termed primary processes, and produce *primary* nuclides. Assuming that $M_{\text{gas}}(0) = M_{\text{tot}}$ and the mass of nuclide i at time zero $M_i(0) = 0$, the equation for the evolution of the mass of a primary nuclide p takes the form

$$\frac{d}{dt}(M_{\text{gas}}X_p(t)) = -\psi(t)X_p(t) + E_p(t) + [f_{\text{in}}X'_p - f_{\text{out}}X_p(t)]. \tag{2.1}$$

In this expression ψ is the star formation rate and X_p is the fractional abundance of nuclide p in the ISM, the product of which is the rate of sequestration of nuclide p due to new star formation and the ejection rate $E_p(t)$ is the rate at which both enriched and unenriched mass is returned to the ISM by supernovae and stellar winds. The last two terms (in brackets) account for infalling and outflowing gas, i.e. infalling gas adds nuclide p to the system at the rate $f_{\text{in}}X'_p$ and outflowing gas removes it at the rate $f_{\text{out}}X_p$. Often, the infalling gas is assumed to be primordial, in which case $X'_p = 0$ and infall has no effect on the mass of nuclide p in the system.

The ejection rate $E_p(t)$ can be written as

$$E_p(t) = \int_{m(t)}^{m_u} Y_p(m)\psi(t-\tau_m)\phi(m)\ dm, \tag{2.2}$$

where m is the mass of a star with lifetime τ_m, $Y_p(m)$ is the stellar yield of nuclide p for a star of mass m, and $\psi(t-\tau_m)$ is the star formation rate at time of birth of the star of mass m. The stellar yield $Y_p(m)$ is

$$Y_p(m) = (m - m_{\text{r}})\ X_p(t-\tau_m) + y_p(m), \tag{2.3}$$

where $y_p(m)$ is the mass of newly formed and ejected nuclide p, $m - m_{\text{r}}$ is the difference in mass between a star and it's stellar remnant and thus $(m - m_{\text{r}})\ X_p(t-\tau_m)$ accounts for the mass of pristine, unprocessed nuclide p restored to the ISM.

Substituting Equation (2.3) into Equation (2.2) and integrating over the chosen SFR and

IMF yields an integro-differential equation which can be difficult to solve analytically. For presentation purposes the simplifying assumption that all stars with $m < m_\odot$ are immortal and all others die instantly is often made and is known as the "instantaneous recycling approximation" (IRA). By invoking the IRA and neglecting stellar lifetimes τ_m, the return fraction can be defined as

$$R = \int_{m_\odot}^{m_u} (m - m_{\rm r})\phi(m)\ dm, \tag{2.4}$$

which is the total fraction of mass returned to the ISM after each stellar generation. By applying the IRA and substituting Equation (2.4), Equation (2.2) can be rewritten as

$$\begin{aligned} E_p(t) &= \int_{m(t)}^{m_u} Y_p(m)\psi(t)\phi(m)\ dm \\ &= \int_{m(t)}^{m_u} \left[(m - m_{\rm r})X_p(t) + y_p(m)\right]\ \psi(t)\phi(m)\ dm \\ &= RX_p(t)\psi(t) + (1 - R)\rho_p\psi(t), \end{aligned} \tag{2.5}$$

where ρ_p is the galactic yield of nuclide p, equal to the ratio of ejected to sequestered mass of nuclide p integrated over the range of stellar masses assumed to die instantaneously under the IRA and normalized by the IMF, namely

$$\rho_p = \frac{1}{1 - R}\int_{m_\odot}^{m_u} y_p(m)\phi(m)\ dm. \tag{2.6}$$

Substituting Equation (2.5) into Equation (2.1) and rearranging yields an equation which can be solved analytically

$$\frac{d}{dt}(M_{\rm gas}X_p(t)) = -(1 - R)X_p(t)\psi(t) + (1 - R)\rho_p\psi(t) + f_{\rm in}X'_p - f_{\rm out}X_p(t). \tag{2.7}$$

Using the identity $dM_{\rm gas}X_p(t)/dt = M_{\rm gas}\ dX_p(t)/dt + X_p(t)\ dM_{\rm gas}/dt$, Equation (2.7) can be re-written as

$$\begin{aligned} M_{\rm gas}(t)\frac{dX_p(t)}{dt} = &-(1 - R)X_p(t)\psi(t) + (1 - R)\rho_p\psi(t) \\ &+ \left[f_{\rm in}X'_p - f_{\rm out}X_p(t)\right] - X_p(t)\frac{dM_{\rm gas}}{dt}. \end{aligned}$$

Knowing that $dM_{\text{gas}}/dt = -(1-R)\psi(t) + f_{\text{in}} - f_{\text{out}}$, this expression can be simplified to obtain

$$M_{\text{gas}}(t)\frac{dX_p}{dt} = (1-R)\rho_p\psi(t) + f_{\text{in}}(X'_p - X_p(t)) \tag{2.8}$$

where $(1-R)$ is the fraction of mass sequestered in stellar remnants, ρ_p is the IMF-integrated yield of new nuclide p per unit stellar remnant mass, f_{in} is the flux of fresh gas to the Galaxy, and X'_p is the abundance of nuclide p for the infalling material. In this expression $(1-R)\rho_p\psi(t)$ is the mass of newly produced nuclide p ejected from stars into the ISM per unit time. Thus, under the IRA, primary nuclide production is decoupled from stellar metallicity and is proportional to the star formation rate $\psi(t)$ and inversely proportional to the mass of gas remaining in the galaxy. The solution to Equation (2.8) for a simple closed box model where $f_{\text{in}} = 0$ is

$$X_p(t) - X_p(0) = \rho_p \ln\left(\frac{M_{\text{tot}}}{M_{\text{gas}}}\right) = \rho_p \ln\left(\frac{1}{\mu_{\text{gas}}}\right), \tag{2.9}$$

where μ_{gas} is the fraction of total mass that is gas in the system (Prantzos, 2008; Searle and Sargent, 1972; Tinsley and Cameron, 1974). A commonly used parameterization for the decrease in gas in the Galaxy with time is $\mu_{\text{gas}}(t) = \mu_{\text{gas}}(0)\exp(-t/T)$ where T is a characteristic timescale that scales with the terminal age of the Galaxy. In such a case, $X_p(t) - X_p(0) = \rho_p(t/T)$ where μ_{gas} is unity at $t = 0$, showing that the amount of a primary nuclide grows roughly linearly with time. For convenience of presentation in what follows, it is assumed that $X_p(0) = 0$.

2.2.2 Secondary Nuclides

Odd-Z and neutron-rich nuclides are often not accessible via primary nucleosynthetic processes, and production is dependent on presence of primary ”seed” nuclei synthesized in previous stellar generations. As before, the equation for the evolution of mass of a secondary nuclide s with abundance X_s takes the form

$$\frac{d}{dt}(M_{\text{gas}}X_s) = -\psi(t)X_s + E_s(t) + [f_{\text{in}}X'_s - f_{\text{out}}X_s] \tag{2.10}$$

where the first term is the rate of sequestration of nuclide s due to new star formation, the ejection rate $E_s(t)$ is identical to Equation (2.2), and the terms in brackets account for gas entering and exiting the system. Because production rates of secondary nuclides are dependent on the presence of primary seed nuclei, the stellar yield $Y_s(m)$ of nuclide s differs from that seen previously in Equation (2.2), specifically

$$\begin{aligned} Y_s(m) =& (m - m_{\rm r})\ X_s(t - \tau_m) + y_s(m) \\ =& (m - m_{\rm r})\ X_s(t - \tau_m) + \alpha X_p(t - \tau_m), \end{aligned} \tag{2.11}$$

where it is assumed the net yield $y_s(m)$ of nuclide s scales linearly with $X_p(t - \tau_m)$, the abundance of the primary "seed" nuclei present when the star formed, via the proportionality constant α.

By adopting the IRA, the ejection rate $E_s(t)$ of secondary nuclide s can be written

$$\begin{aligned} E_s(t) &= \int_{m(t)}^{m_u} Y_s(m)\psi(t)\phi(m)\ dm \\ &= RX_s(t)\psi(t) + (1 - R)\alpha X_p(t)\psi(t). \end{aligned} \tag{2.12}$$

Notice that in this case the galactic yield ρ_s is equivalent to $\alpha X_p(t)$, and is therefore time dependent. This is not the case for primary nuclide production. Equation 2.10 thus becomes

$$\frac{d}{dt}(M_{\rm gas}X_s) = -(1 - R)X_s(t)\psi(t) + (1 - R)\alpha X_p(t)\psi(t) + [f_{\rm in}X_s'(t) - f_{\rm out}X_s(t)]. \tag{2.13}$$

Following the example set in Equation 2.8, the solution to equation 2.13 can shown to be

$$M_{gas}(t)\frac{dX_s}{dt} = (1 - R)\alpha X_p(t)\psi(t) + f_{\rm in}(X_i'(t) - X_i(t)). \tag{2.14}$$

Again adopting the assumption that $f_{\rm in} = 0$, the solution to equation 2.14 for a simple closed box model (Prantzos, 2008) is

$$X_s(t) - X_s(0) = \alpha X_p(t)\ \ln(\frac{1}{\sigma_{\rm gas}}) = \frac{\alpha}{\rho_p}X_p(t)^2. \tag{2.15}$$

Since X_p, the fractional abundance of primary nuclide p, is expected to vary roughly linearly with time, Equation (2.15) shows that the abundance of the secondary nuclides should vary roughly as t^2 because $X_s = \alpha\rho_p(t/T)^2$. Finally, the ratio of secondary to primary metallicity can be shown to be

$$\frac{X_s}{X_p} = \frac{\alpha X_p(t)}{\rho_p} \propto Z \tag{2.16}$$

where Z is the metallicity. From Equation (2.16), it can be shown that $X_s/X_p = \alpha(t/T)$ and therefore, under the IRA, any given stable isotope system containing members synthesized via both production modes will evolve with time such that ratios of primary to secondary isotopes will rise linearly with time and that the ratio of one secondary isotope to another will remain constant.

The variation in molecular gas surface density across the Galaxy resembles the metallicity variation with R_{GC} shown in Figure 2.1 (Heyer and Dame, 2015) in showing a monotonic increase moving inward from $\approx$ 10 kpc to $\approx$ 5 kpc, and a decrease between $\approx$ 5 kpc and the the Galactic center. This correspondence between metallicity and molecular gas surface density in the Milky Way suggests a link between time-averaged stellar processing and gas density, as suggested by the Schmidt-Kennicutt relationship between star formation rate and gas surface density (Kennicutt and Evans, 2012; Kennicutt, 1998). As with overall metallicity Z, the abundances of primary nuclides of particular interest are also expected to vary with R_{GC}. In a closed system, μ_{gas} is expected to decline towards the Galactic center. Comparisons between the sharp decline in the surface density of stars with increasing R_{GC} (Kent et al., 1991) and the more gradual declines in molecular and total gas surface densities withR_{GC} (Heyer and Dame, 2015) show that μ_{gas} does indeed decrease with smaller R_{GC} in the Milky Way. This is also the case for other, nearby spiral galaxies (Leroy et al., 2008). For illustration purposes, a function for $\mu_{\text{gas}}(R_{\text{GC}})$ with a range of 0 to 1 from the Galactic center to the outer Galactic disk can be written as

$$\mu_{\text{gas}} = 1 - \frac{1}{R_{\text{GC}} + 1} \tag{2.17}$$

where R_{GC} is in kpc. Substituting Equation (2.17) into Equation (2.9) with $X_i(0) = 0$ yields

$$X_p(t) = \rho_p \ \ln\left(\frac{R_{\rm GC}+1}{R_{\rm GC}}\right) \tag{2.18}$$

which reduces to $X_p(t) \approx \rho_p/R_{\rm GC}$ for $R_{\rm GC} \gg 1$ kpc, showing that the relative abundances of primary nuclides should increase towards the Galactic center. Further, by combining Equations (2.16) and (2.18) it is clear that the ratio of secondary nuclides to primary nuclides should also vary inversely with $R_{\rm GC}$

$$\frac{X_s}{X_p} = \frac{\alpha}{R_{\rm GC}}. \tag{2.19}$$

From these closed-system IRA equations, dating back to Tinsley and Cameron (1974) and Tinsley (1975), there is a basis for the expectation that at any given time in the Galaxy, secondary-to-primary isotope ratios should increase towards the Galactic center. A corollary is that two distinct ratios, $X_{s'}/X_p$ and $X_{s''}/X_p$, composed of two distinct secondary nuclides s' and s'' and a single primary nuclide (e.g., $[^{18}\text{O}]/[^{16}\text{O}]$ and $[^{17}\text{O}]/[^{16}\text{O}]$ or $[^{30}\text{Si}]/[^{28}\text{Si}]$ and $[^{29}\text{Si}]/[^{28}\text{Si}]$) will tend to grow in lockstep. The apparent chemical and isotopic "age" of the ISM should increase with decreasing $R_{\rm GC}$ in a manner that mimics the effects of time. For this reason, Galactocentric radius is in principle a proxy for time, and variations in isotope ratios with $R_{\rm GC}$ can be used as models for temporal variations in Galactic isotope ratios.

There are a number of factors that complicate the simple picture developed above. Foremost among them is that the Galactic disk is not a closed system. The effects of infalling gas towards the center of the Galaxy may be evidenced in Figure 2.1 where metallicity is seen to level off or even decline near the Galactic center. Despite these complicating factors, there should be a general relationship between metallicity and secondary/primary stable isotope ratios, and that a trend similar to that shown in Figure 2.1 should also obtain for these isotope ratios as well. If this prediction is verified, it would serve as good evidence that our understanding of the isotopic effects of GCE is reasonable, permitting the extrapolation of stable isotope ratios back in time, for example. Conversely, if a comparable trend is not observed, then the significance of isotope ratio variations with $R_{\rm GC}$, as well as the validity of any inferences made about the time evolution of stable isotope ratios, should be reconsidered.

2.3 Previous Observations

The secondary/primary isotopic abundance ratios of oxygen (e.g., Penzias, 1981b; Wilson et al., 1981) and carbon (e.g., Langer and Penzias, 1990, 1993; Milam et al., 2005; Savage et al., 2002; Wilson et al., 1981; Wilson and Rood, 1994) have have been used extensively to trace variations in the degree of astration across the Galaxy. The primary nuclide ^{12}C is produced during the helium-burning phase by the triple-alpha process (Burbidge et al., 1957) and is the second most abundant non-primordial nuclide in the galaxy (Clayton, 2003). The only other stable isotope of carbon, ^{13}C, is a secondary nucleosynthetic product which requires pre-existing ^{12}C for efficient production (Burbidge et al., 1957). Approximately half of the carbon in the ISM originates from Type II supernovae, while the remainder is produced by intermediate mass (1.5 - 6 $M_{\odot}$) asymptotic giant branch (AGB) stars (Clayton, 2003). Milam et al. (2005) showed that the [^{13}C]/[^{12}C][2] ratios in Galactic molecular clouds increase towards the Galactic center, consistent with the qualitative expectations of GCE. Based on this agreement between data and GCE predictions, the authors suggested that the higher [^{13}C]/[^{12}C] value of $\approx 1/65$ in the present-day ISM (relative to the Solar value of 1/89) could be the consequence of ^{13}C enrichment relative to ^{12}C over the last 4.6 Gyrs.

The oxygen isotope system differs from the carbon system in that it has two stable heavy isotopes, ^{17}O and ^{18}O. The most abundant isotope of oxygen, ^{16}O, is a primary nuclide produced during He burning. The rare isotopes, ^{17}O and ^{18}O, are secondary nuclides. ^{17}O is the daughter product of ^{17}F, which undergoes rapid β^+ decay after being produced as part of the CNO tricycle. Similarly, the majority of ^{18}O is produced by α addition to ^{14}N, via the formation and subsequent β^+ decay of ^{18}F. ^{14}N is in turn produced from ^{12}C during the CNO tricycle. ^{18}O is also produced from ^{17}O via neutron capture (Burbidge et al., 1957).

The existence of two secondary isotopes makes the oxygen system particularly attractive for tracing GCE. Optical depth effects have hampered efforts to determine C^{16}O column densities within sources. However, one can use estimates for the [^{13}C]/[^{12}C] ratio within the

[2]Brackets are used to distinguish atomic abundances from mass abundances. Note that [x]/[y] should not be confused with [x/y], where only the latter is in dex units

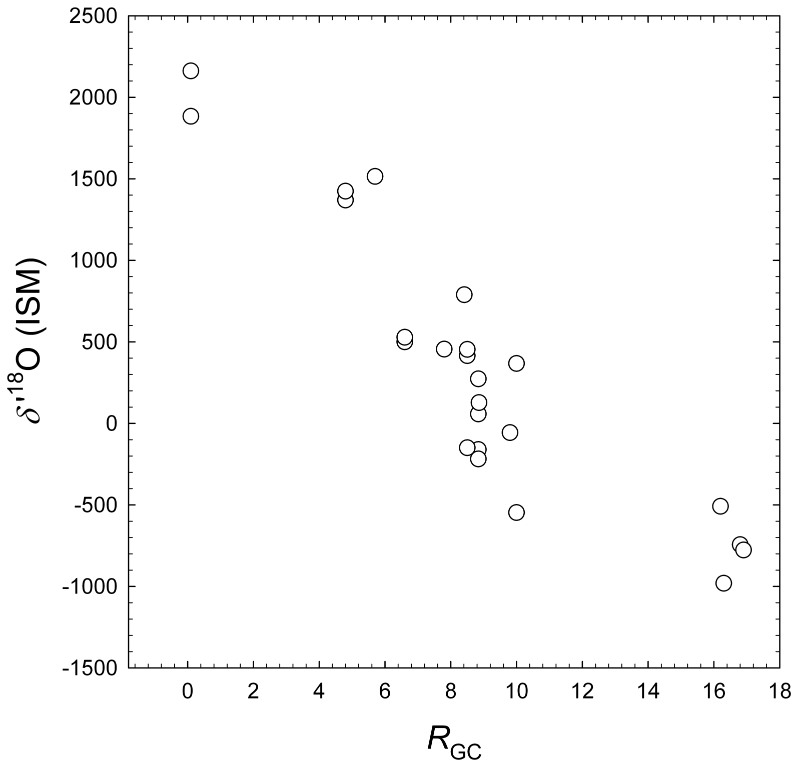


Figure 2.2: $[^{18}\mathrm{O}]/[^{16}\mathrm{O}]$ as $\delta'^{18}\mathrm{O}$ in permil vs Galactocentric radius. References in text.

source to calculate the $\mathrm{C}^{16}\mathrm{O}$ column density from $^{13}\mathrm{C}^{16}\mathrm{O}$ observations. Using this approach, Galactic oxygen isotope abundances can be extrapolated from the $^{13}\mathrm{CO}$, $\mathrm{C}^{18}\mathrm{O}$, and $\mathrm{C}^{17}\mathrm{O}$ column densities reported by Wouterloot et al. (2008) and the $[^{13}\mathrm{CO}]/[^{12}\mathrm{CO}]$ vs. R_{GC} data from Milam et al. (2005). For this and other purposes in this paper, δ' notation is used to compare isotope ratios expressed as permil differences from a reference ratio R_0. That is

$$\delta'^{X}\mathrm{Z} = 10^3 \ln(R/R_0), \tag{2.20}$$

where R is the isotope ratio $[^{X}\mathrm{Z}]/[^{Y}\mathrm{Z}]$ and $^{Y}\mathrm{Z}$ and $^{X}\mathrm{Z}$ are the light and corresponding heavy isotopes of element Z, respectively[3]. The resulting $[^{18}\mathrm{O}]/[^{16}\mathrm{O}]$ ratios, normalized to the ISM value reported by Wilson (1999), vs. R_{GC} are shown in Figure 2.2. These extrapolated data indicate that $[^{18}\mathrm{O}]/[^{16}\mathrm{O}]$ ratios increase linearly with decreasing R_{GC}, in qualitative agreement with the predictions of secondary/primary increases with GCE. However, the observed values in Galactic $[^{18}\mathrm{O}]/[^{16}\mathrm{O}]$ range by greater than a factor of 10, or $> 900\%$ (a

[3]The logarithmic form of the δ notation is used to accommodate the large variations in isotope ratios across the Galaxy.

factor of 10 corresponds to 2300 per mil on the ordinate in Figure 2.2) which exceeds the theoretical predictions of Prantzos et al. (1996) by a factor of ≈ 2 to 3 (Young et al., 2011). Additionally, the trend observed in $[^{18}O]/[^{16}O]$ appears to extend unabated into the Galactic center, contrasting with the "downturn" seen in both the [O/H] and [Fe/H].

The two oxygen secondary/primary isotope ratios can also be used in concert to evaluate the presence or absence of GCE in the oxygen isotopologue data. On a so-called three-isotope plot in which $[^{17}O]/[^{16}O]$ is plotted against $[^{18}O]/[^{16}O]$, both normalized to a suitable reference, the first-order prediction based on GCE is that data representing a range of localities across the Galaxy will define a slope of unity. Quantitative GCE models for the oxygen isotopes are in general agreement with the simplified equations presented above and show that even as $[^{17}O]/[^{16}O]$ and $[^{18}O]/[^{16}O]$ have risen with time, the ratio of the two secondary nuclides, $[^{18}O]/[^{17}O]$, should have been constant after the first billion years (Prantzos et al., 1996; Timmes et al., 1995). This is because both secondary nuclides have a similar dependency on metallicity in these models. Figure 2.3 (after Young et al., 2011) illustrates that the $[^{17}O]/[^{16}O]$ and $[^{18}O]/[^{16}O]$ ratios across the galaxy define a slope in triple-isotope space of 1.11 ± 0.08 (2σ) that is practically indistinguishable from the unity value predicted by closed-system IRA GCE. Also shown in Figure 2.3 are infrared absorption data for young stellar objects that show less of a spread in oxygen isotope ratios, albeit in part because they are from sources near the Solar circle.

The validity of the combined carbon/oxygen data sets has been questioned on the basis that there is good reason to believe that ^{17}O is produced mainly in intermediate-mass stars (Romano and Matteucci, 2003) while ^{18}O is produced in more massive stars. In this case the progenitors of ^{17}O live longer than those of ^{18}O, allowing for deviations from expectations of nearly constant $[^{18}O]/[^{17}O]$ with time (in effect altering α for the two secondary nuclides in Equation 2.19). Nittler and Gaidos (2012) also question the veracity of the $\delta'^{18}O$ vs R_{GC} trend, referring to "chemical" rather than isotopic partitioning to account for varying $[^{13}C^{16}O]/[^{12}C^{18}O]$. It should be noted that both the spatial and spectral resolution in the previous studies limited the ability to detect optical depth effects that might spuriously enhance the recovered $[^{18}O]/[^{16}O]$ and $[^{17}O]/[^{16}O]$ ratios, artificially translating any affected

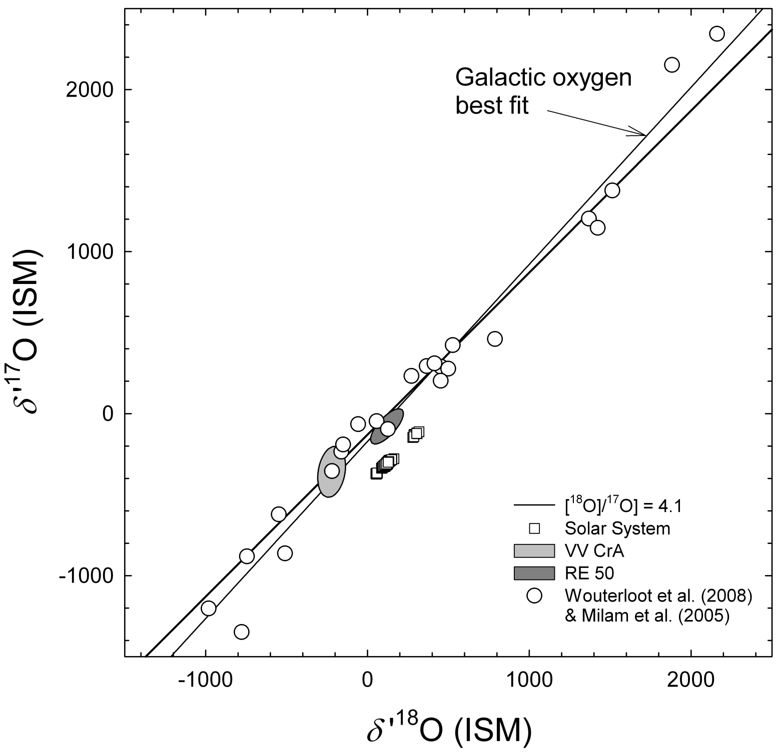


Figure 2.3: Oxygen triple-isotope plot comparing molecular clouds obtained by a combination of radio observations (circles), IR absorption (error ellipses), and the Solar System materials (squares). The best fit line has a slope of 1.11 ±0.08 (2σ). References in text.

sources up a slope-1 trajectory in triple-isotope space. Additionally, some ^{12}C nuclei produced in the He-burning shells of AGB stars are conveyed to the outer envelopes of these stars during convective instability "dredge-up" events. A portion of these newly-synthesized ^{12}C nuclei undergo neutron capture to form ^{13}C. Consequently, a considerable amount of what is effectively primary ^{13}C is created in the He intershell, some of which is then convectively transported to the surface and shed in stellar winds (Gallino et al., 1998; Straniero et al., 1997). The degree to which this effect biases Galactic carbon isotope ratios is not well quantified, and complicates the interpretation of these isotopes in the ISM. For these reasons, the veracity of the oxygen trend as evidence of GCE (Figures (2.2) and (2.3)) is called into question.

CHAPTER 3

Testing GCE using Silicon

While interstellar oxygen isotopes have been extensively studied (Wilson, 1999, e.g.), the same is not true of the other light-element systems having 3 stable-isotopes; ^{24,25,26}Mg and ^{28,29,30}Si. Magnesium is poorly suited to widespread interstellar observations, however silicon is readily observed in molecular clouds at millimeter wavelengths.

A number of silicon-bearing molecular species, including SiC, SiS, SiCN and SiNC, have been detected in the circumstellar envelopes of AGB stars, however the possibility of local nucleosynthesis makes these unsuitable proxies for the average interstellar abundances. SiO is commonly observed to trace shocks in dense, turbulent cloud cores and molecular outflows (Caselli et al., 1997; Martín et al., 2009; Schilke et al., 1997; Ziurys et al., 1989) where it is thought to dominate the gaseous silicon budget and the chances that observational measurements are not representative of the bulk silicon composition are minimized. For this reason, SiO is well suited for probing isotopic GCE. Because silicon is a relatively refractory element and is largely sequestered in silicate dust, SiO column densities are typically modest in comparison to common molecules, such as CO, CS, or HCN, and observed SiO line intensities are similarly modest. As a consequence of requiring relatively dynamic physical conditions, most sources of SiO emission are compact and efficient observation requires large telescopes to achieve favorable beam-filling factors. Fortunately, ^{29}Si and ^{30}Si are relatively abundant (with Solar $[^{28}\text{Si}]/[^{29}\text{Si}] = 19.7$ and $[^{28}\text{Si}]/[^{30}\text{Si}] = 29.8$), allowing the weaker isotopologue lines to be accurately measured with feasible integration times.

The silicon isotope system is largely analogous to that of oxygen, in that it contains one primary and two secondary nuclides. The primary silicon isotope, ^{28}Si, is an alpha process nuclide and is by far the most prevalent, with a Solar abundance of 92.23% (Clayton, 2003).

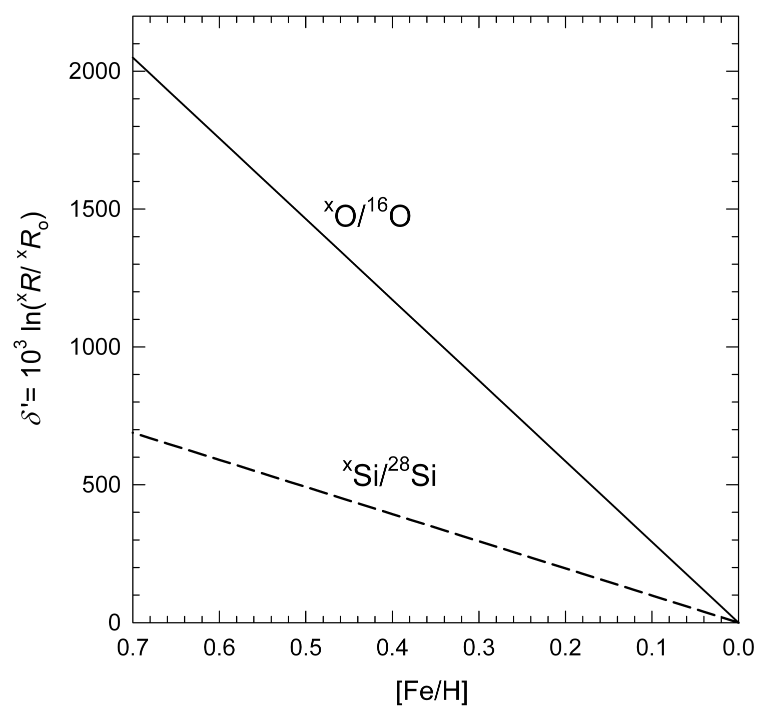


Figure 3.1: Predicted dependence of oxygen and silicon isotope abundance ratios on local stellar [Fe/H], relative to the Solar value (Timmes and Clayton, 1996; Timmes et al., 1995). Secondary to primary isotopic ratio values on the ordinate are expressed as $\delta' = 10^3 \ln(R/R_0)$, where R and R_0 respectively refer to the Galactic values and the initial reference value (see Equation (2.20)).

^{29}Si and ^{30}Si are both secondary, forming largely from ^{25}Mg and ^{26}Mg during neon burning, as well as during core-collapse Type II supernovae. Both rare isotopes also form from ^{28}Si in the He-burning shells of AGB stars. While contributions from He-burning AGB stars could alter local compositions, such a source likely has little effect on the overall isotopic budget of the interstellar medium (Clayton, 2003). GCE models predict that, to first order, the silicon and oxygen isotope ratios should evolve in parallel. Therefore, based on the oxygen data (e.g., Figures 2.2 and 2.3), one expects nearly constant [^{29}Si]/[^{30}Si] across the Galaxy, as well as radial gradients in the [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] ratios that increase with decreasing R_{GC}.

Predictions for the magnitude of the variations in [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] relative to the variations in the oxygen isotope system can be made using the silicon isotope GCE model of Timmes and Clayton (1996) and the oxygen isotope GCE model of Timmes et al. (1995)[1]. The predicted dependencies of isotope ratios on metallicity are $d[^{X}\mathrm{Si}/^{28}\mathrm{Si}]/d[\mathrm{Fe/H}] = 0.43$ and $d[^{X}\mathrm{O}/^{16}\mathrm{O}]/d[\mathrm{Fe/H}] = 1.27$ where X represents the heavy isotopes and all ratios are in dex. If the Galactic center is no greater than ≈ 0.5 dex in [Fe/H], as suggested by the observed metallicities of Quintuplet cluster LBVs (Cunha et al., 2007), then one predicts an increase in $[^{18}\mathrm{O}]/[^{16}\mathrm{O}]$ expressed as $\delta'^{18}\mathrm{O}$ relative to the Solar value of approximately 1500 permil between the Solar circle and the Galactic center. The corresponding increase in $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ expressed as $\delta'^{29}\mathrm{Si}$ is predicted to be ≈ 500 permil (Figure 3.1). As described above, this prediction is similar to, but approximately $3\times$ smaller than, the observed variation for the oxygen isotopes (Wilson, 1999; Young et al., 2011).

Additional motivation for establishing the Galactic distribution of silicon isotopes can be garnered from the $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ isotope abundance ratios found in presolar SiC grains. These grains predate the Sun and are thought to have condensed out of the winds expelled from ancient asymptotic giant branch (AGB) stars. The mainstream SiC grains ($> 90\%$ of all presolar SiC grains) define a spread in $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ (as $\delta'^{29}\mathrm{Si}$) and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ (as $\delta'^{30}\mathrm{Si}$) along a slope of ≈ 1.2 (Figure 3.2). The variation in silicon isotope ratios is an order of magnitude larger than that expected from nucleosynthesis in a single AGB star and it is generally agreed that the spread in $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ predates the AGB parents of these grains (Lugaro et al., 1999, and references therein).

The considerable spread in the presolar SiC grain $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ ratios represents either a manifestation of GCE as sampled by AGB stars with different birth dates, or heterogeneity in the ISM material from which the AGB stars formed. GCE predicts that Solar $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ ratios, representing the ISM when the Sun formed 4.6 Gyr before present, should be larger than the $[^{29}\mathrm{Si}]/[^{28}\mathrm{Si}]$ and $[^{30}\mathrm{Si}]/[^{28}\mathrm{Si}]$ ratios found in presolar SiC grains that predate the Sun, but this is not observed. This apparent excess in $^{28}\mathrm{Si}$ (or depletion in $^{29}\mathrm{Si}$ and $^{30}\mathrm{Si}$) in the Sun is a conundrum. Alexander and Nittler (1999) suggested that the Solar System was enriched in $^{28}\mathrm{Si}$ by supernova ejecta. A model for that

[1]The results from Timmes and Clayton (1996) and Timmes et al. (1995) are typical of other models for $[\mathrm{Fe/H}] \geq$ Solar (Lewis et al., 2013).

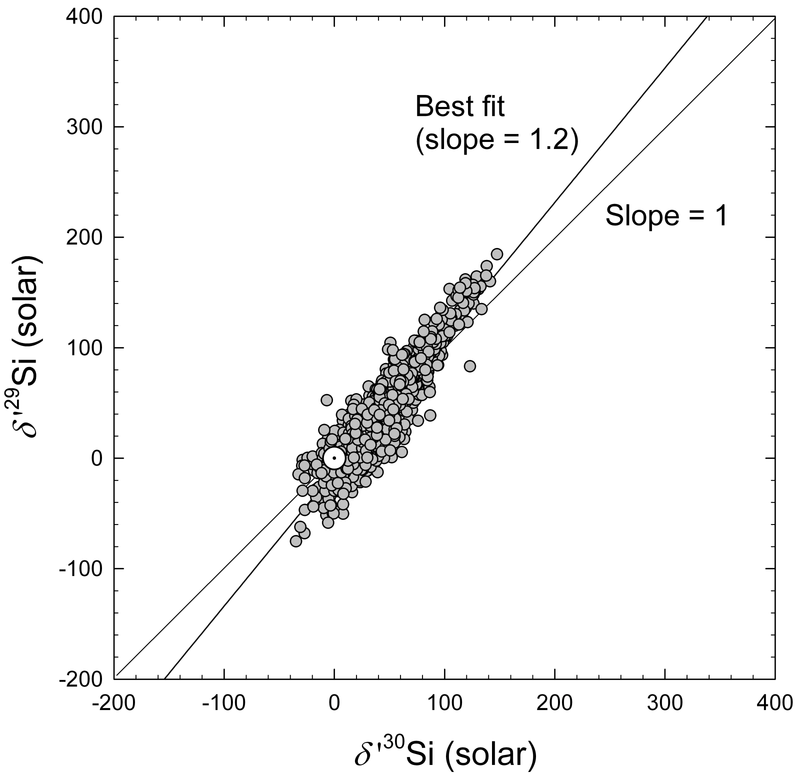


Figure 3.2: Silicon isotope ratios of mainstream presolar SiC grains (grey filled circles) expressed as per mil deviations from the Solar ratios, i.e. δ'^{29}Si vs. δ'^{30}Si (data from Ernst Zinner, pers. comm.). The white circle with the center dot indicates present-day Solar abundances and defines the origin. The best-fit line has a slope of 1.22 $\pm 0.02 (2\sigma)$.

enrichment was given by Young et al. (2011). Alternatively, Lugaro et al. (1999) suggested that the distribution of data in Figure 3.2 can be explained simply by dispersion resulting from incomplete mixing of stellar sources, although this model fails to reproduce correlations between Ti and Si isotope ratios in the SiC grains (Nittler, 2005). More recently, Lewis et al. (2013) used the SiC grain data and GCE models to derive the metallicity [Fe/H] and ages of the SiC parent stars. Their results suggest a distribution in [Fe/H] with a mean near the Solar value and a 1σ error of about 0.2 dex with a skew towards higher [Fe/H]. Their derived range in metallicity is less than that observed in the Solar neighborhood today. Mapping the distribution of Galactic Si isotope ratios as a function of $R_{\rm GC}$ will provide much needed context for the questions raised by the comparison of Si isotope ratios between presolar SiC grains and the Sun.

CHAPTER 4

Observations

Initial observations of the $v = 0$, $J = 1 \rightarrow 0$ transition of the three silicon isotopologues of SiO were carried out at the Robert C. Byrd Green Bank radio telescope (GBT) in May of 2013 (project GBT13A-415). Additionally, several weeks were spent in Green Bank in January and February of 2014 making follow-up observations (project GBT14A-431). Seven sources with known radial distances from the Galactic center and brightness temperatures between 1 and 3 Kelvin were selected (see Table 4.1).

AGFL 5142 ($R_{\mathrm{GC}} = 9.8$ kpc) is a cluster of high-mass protostars (Zhang et al., 2007). DR21(OH) ($R_{\mathrm{GC}} = 7.9$ kpc) is a site of dense molecular clouds within Cygnus X where several OB stars are resident (Duarte-Cabral et al., 2014). L1157 ($R_{\mathrm{GC}} = 8.1$ kpc) is a dark cloud in Cepheus harboring young protostars with chemically active outflows (Nisini et al., 2007). NGC 7538S ($R_{\mathrm{GC}} = 9.3$ kpc) is a high-mass accretion disk candidate comprising a compact H II region surrounding a nascent O star in the Perseus spiral arm (Naranjo-Romero et al., 2012). W51e2 ($R_{\mathrm{GC}} = 6.4$ kpc) is a bright ultracompact H II region in the W51 star-forming region. Hints of bipolar outflows perpendicular to a rotating ionized disk are reported, as is evidence for a newly formed O star or cluster of B stars (Shi et al., 2010). SiO in the Galactic center traces shocked high-velocity molecular cloud gas there. GCM$-0.13 - 0.08$ ($R_{\mathrm{GC}} \lesssim 0.1$ kpc), also known as the 20 km/s cloud, is one of the densest clouds in the Sagittarius A complex (Tsuboi et al., 2011). GCM $0.11 - 0.11$ ($R_{\mathrm{GC}} \lesssim 0.1$ kpc) is also a member of the Sagittarius A complex (Handa et al., 2006).

Data for all three isotopologues of SiO were collected simultaneously using the Q-band receiver and autocorrelation spectrometer backend. The autocorrelation spectrometer accommodated four spectral windows, one for each of the three silicon isotopologues of SiO,

Source	α (J2000)	β (J2000)	Pointing Offset (”,”)	Species	$T_{\rm mb}$ (K)	$\Delta v_{1/2}$ ($\rm km\,s^{-1}$)	$\int T_{\rm mb}\,dv$ ($\rm K\,km\,s^{-1}$)	$V_{\rm LSR}$ ($\rm km\,s^{-1}$)
DR21 (OH)	20:39:01.0	+42:22:50	(0, -5)	^{28}SiO	1.484 ± 0.019	5.14 ± 0.03	9.626 ± 0.025	-4.69 ± 0.17
				^{29}SiO	0.081 ± 0.016	5.40 ± 0.29	0.529 ± 0.016	-4.62 ± 0.17
				^{30}SiO	0.057 ± 0.017	5.02 ± 0.41	0.348 ± 0.016	-4.46 ± 0.17
L1157 B1	20:39:06.4	+68:02:13	(0, 0)	^{28}SiO	3.376 ± 0.019	3.63 ± 0.01	14.080 ± 0.016	1.80 ± 0.17
				^{29}SiO	0.280 ± 0.016	3.19 ± 0.08	1.074 ± 0.013	1.87 ± 0.17
				^{30}SiO	0.189 ± 0.016	3.27 ± 0.10	0.703 ± 0.013	1.76 ± 0.17
NGC 7538 S	23:13:44.8	+61:26:51	(0, -5)	^{28}SiO	1.783 ± 0.022	4.83 ± 0.03	11.823 ± 0.043	-54.22 ± 0.17
				^{29}SiO	0.118 ± 0.020	4.58 ± 0.18	0.729 ± 0.023	-54.21 ± 0.17
				^{30}SiO	0.089 ± 0.020	4.41 ± 0.25	0.522 ± 0.023	-54.19 ± 0.17
AFGL 5142	05:30:45.9	+33:47:56	(+25, -5)	^{28}SiO	0.920 ± 0.013	5.93 ± 0.04	7.467 ± 0.037	-2.71 ± 0.17
				^{29}SiO	0.052 ± 0.013	5.74 ± 0.57	0.401 ± 0.016	-2.41 ± 0.17
				^{30}SiO	0.035 ± 0.012	6.12 ± 0.72	0.289 ± 0.015	-1.88 ± 0.17
W51e2	19:23:42.0	+14:30:00	(+25, +30)	^{28}SiO	2.257 ± 0.014	8.22 ± 0.02	21.620 ± 0.042	-56.30 ± 0.17
				^{29}SiO	0.142 ± 0.011	7.63 ± 0.19	1.264 ± 0.013	-56.33 ± 0.17
				^{30}SiO	0.094 ± 0.012	8.35 ± 0.24	0.856 ± 0.014	-55.98 ± 0.17
GCM 0.11-0.11	17:46:18.0	-28:54:00	(+40, +35)	^{28}SiO	1.791 ± 0.026	19.38 ± 0.14	43.685 ± 0.411	-23.37 ± 0.67
				^{29}SiO	0.168 ± 0.031	16.37 ± 2.48	3.339 ± 0.117	-22.66 ± 0.67
				^{30}SiO	0.106 ± 0.020	17.09 ± 1.01	2.240 ± 0.111	-23.67 ± 0.67
GCM -0.13-0.08	17:45:25.2	-29:05:30	(+180, +70)	^{28}SiO	4.195 ± 0.028	19.88 ± 0.04	91.133 ± 0.510	-17.25 ± 0.67
				^{29}SiO	0.361 ± 0.036	17.45 ± 0.43	6.950 ± 0.129	-16.54 ± 0.67
				^{30}SiO	0.250 ± 0.024	17.13 ± 0.41	4.532 ± 0.090	-17.36 ± 0.67

Table 4.1: List of Sources and Observed SiO v=0, J=1 $\rightarrow$ 0 Emission Lines

and a 'spare' that was put to use in several capacities that will be addressed in subsequent sections. The two Galactic center sources were observed using 200 MHz bandpass windows with 24.4 kHz wide channels yielding $\approx$ 340 m/s resolution, and all other sources utilized 50 MHz bandpass windows with 6.1 kHz channels, yielding $\approx$ 85 m/s resolution. These spectral resolutions translate to resolving powers of approximately 8.8×10^5 and 3.5×10^6 respectively; the emission lines from all sources are well resolved.

Pointing was checked against nearby 7 mm continuum sources every hour, and errors were typically 3 arcseconds or less. All observations were made using in-band frequency switching, and all switching was by 40% of the bandpass at a rate of 2 Hz. System temperatures hovered around $\approx$ 80 K for most observations, but varied from lows of about 70K to highs of 130K at low elevations or in inclement weather. It was found that most sources required approximately 3 hours of integration time to achieve the desired signal-to-noise ratio for the emission line from the rarest isotopologue. Noise temperatures (prior to resampling) on the order of 20mK were achieved in most sources.

Four additional sources were observed in May and June of 2015 (project GBT15A-350), however the project was crippled by a hardware failure at the GBT. The autocorrelator spectrometer backend used during the two previous projects was replaced by the VErsatile GBT Astronomical Spectrometer (VEGAS) at the beginning of the 2015A semester. While VEGAS provides greater flexibility and throughput than the autcorrelator it superseded, hardware faults rendered much of it inoperational soon after it was commissioned. Only four of the eight FFT spectrometers that comprise VEGAS were available for use, and due to the intricacies of how data from the telescope is routed to the spectrometers, the in-band frequency-switching technique used previously could not be reimplemented. The only way to simultaneously observe both polarizations of all three isotopologues without doubling the required integration time was to use an unconventional out-of-band frequency-switching technique wherein two spectral windows were assigned to each emission line. While this technique, dubbed "band swapping" by the Author, was successful in circumventing the limitations imposed by VEGAS, it has severely complicated data reduction. Consequently, isotopic abundance ratios for these new sources are not presented here.

CHAPTER 5

Calibration and Data Reduction

The calibration and reduction of all data reported here were done using a novel suite of IDL programs (the HYDRA software package) written by the Author and verified by consultation with GBT staff astronomers (these functions expand upon the basic data reduction afforded by the GBTIDL software package). The procedures include several vectorized approaches to the calibrations that enhance accuracy and precision of the extracted line profiles.

5.1 Flux Calibration

As a consequence of the sensitive nature of the measurements being made, special attention was paid to flux calibration to ensure that any drift in receiver performance between observations could be corrected. Differences in receiver gain between spectral windows were also of special concern.

The primary concern with the standard approach for calculating system temperatures, $T_{\rm sys}$, and calibration temperatures, $T_{\rm cal}$, is that any information about frequency-dependent gain *within* the bandpass is lost. Although atmospheric opacity and aperture efficiency are largely invariant across 50 MHz and 200 MHz spectral windows, noise diode power output and LO/IF system response are not. Left unaccounted for, these frequency dependencies are an unacceptably large source of potential error. In order to mitigate these effects, the standard calibration protocol has been adapted to account for channel-by-channel variations in the system response by substituting array valued, or *vectorized*, versions of calibration and system temperatures, $T_{\rm cal}(\nu)$ and $T_{\rm sys}(\nu)$, for their standard scalar valued counterparts. Vectorized calibration routines were developed expressly for this survey as part of the HYDRA data pipeline, allowing gain profiles to be determined pixel-by-pixel across the entire bandpass, thereby accommodating any frequency dependence that may be present. Further, gain profiles for each IF, polarization, noise diode state and frequency position were calculated independently to ensure uniform calibration.

The GBT Q-band receiver was calibrated using a noise diode integrated into the primary signal path. The diode was calibrated against nearby radio-loud active galactic nuclei, 3C48, 3C147 or 3C286, at the beginning and end of each observing period. The spectral flux density of the calibrator, $\vec{S}_{\text{source}}$, was calculated using the polynomial expression and coefficients reported by Perley and Butler (2013, 2017) and converted to corrected antenna temperature with the expression

$$T_{\text{a}}^{*}(\nu) = \frac{A_{\text{g}}}{2k_{\text{b}}} \vec{S}_{\nu}\, \vec{\eta}_{\text{a}} \exp\left(\frac{-\vec{\tau}_{\text{z}}}{\sin(\theta)}\right), \tag{5.1}$$

where A_{g} is the geometric collecting area of the antenna, $\vec{\tau}_{\text{z}}$ is the zenith atmospheric opacity estimated from $\vec{\tau}_{\text{z}} = 0.008 + \exp(\sqrt{\vec{\nu}})/8000$ evaluated over the frequency band $\vec{\nu}$ (in GHz), $\vec{\eta}_{\text{a}}$ is aperture efficiency, and θ is the elevation in radians. Aperture efficiency is estimated using Ruze's equation with the GBT-specific peak aperture efficiency of 0.71 and RMS surface accuracy of 390 microns. The calculated source temperature was then used to convert the power output of the noise diode to a calibration temperature profile:

$$T_{\text{cal}}(\nu) = T_{\text{a}}^{*}(\nu) \left[\frac{\text{Src}^{\text{on}} - \text{Src}^{\text{off}} + \text{Sky}^{\text{on}} - \text{Sky}^{\text{off}}}{\text{Src}^{\text{on}} - \text{Sky}^{\text{on}} + \text{Src}^{\text{off}} - \text{Sky}^{\text{off}}}\right]. \tag{5.2}$$

In Equation (5.2) "Src" and "Sky" refer to the source and sky positions and superscripts "on" and "off" refer to the state of the noise diode. Calibration temperatures are obtained for each polarization and frequency position. The flux calibrators were observed for either two or four 30 second integrations followed by an equal number of sky integrations offset by −0.5 degrees in azimuth, and the noise diode calibration temperature for each polarization and frequency position was independently calculated for each of the either four or sixteen possible Src/Sky integration pairs. For more information on data calibration/reduction, please refer to the *HYDRA User's Guide and Cookbook*, located in Appendix 11.

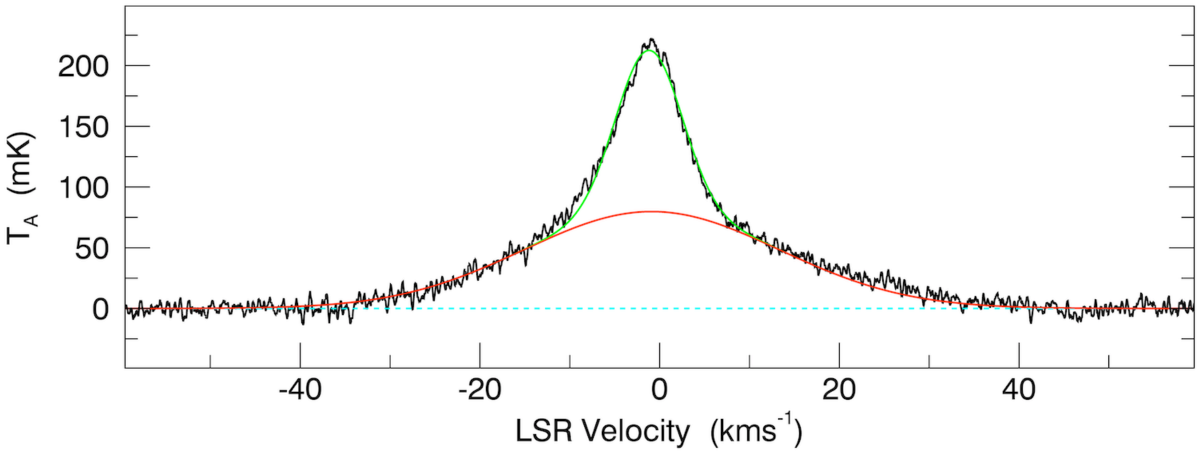


Figure 5.1: The ^{29}SiO emission line from W51e2 with underlying H(83)δ recombination line fitted (smooth curves) for subtraction. Line intensities are shown in antenna temperature in this figure.

5.2 Baselines

The vectorized calibration routine tamed the baselines but did not eliminate all structure. Typical low frequency ($\nu \approx$ bandpass) baselines were fit with low-order polynomials for subtraction. However, differentiating between baseline structure and emission-line structure was challenging in the low-brightness sources DR21(OH) and AFGL 5142. In order to avoid confusing line wings with baselines, all velocities from the baseline fits that lay within ± 3 times the full-width half-maximum (FWHM) of the ^{28}SiO line were omitted. Ultimately, each fit was subject to a Monte Carlo analysis wherein random draws of the boundaries of each region were made, with the standard deviation set to either 5% or 10% of the width of each region (see Section 5.5). Flux-calibrated spectra with baselines subtracted are shown for each of the seven sources in this study in Figure 6.1. The ^{29}SiO and ^{30}SiO line intensities are exaggerated by a factor of 7 for presentation.

5.2.1 Interfering Lines

Extraneous emission lines are seen in most sources, however these extraneous lines generally do not interfere with the SiO lines. Notable exceptions include the six blended 2(0,2) $\rightarrow$

1(0,1) hyperfine lines of formamide (methanamide) between 42385.06 MHz and 42386.68 MHz, which were seen in the ^{30}SiO spectra of both Galactic center sources. The brightness of the formamide line exceeded that of ^{30}SiO in both cases and its effects on the ^{30}SiO lines were removed using the methods described above for baselines. Formamide emission was seen in W51e2 as well, but was rather weak in this source. There was an additional interfering line in W51e2 which appears on the low-velocity wing of the ^{30}SiO line and had to be removed. The poor SNR of the line made identification difficult, although the line is fairly broad and is possibly a blend of the 13(3, 11) $\rightarrow$ 12(4, 8) EA and 13(3, 11) $\rightarrow$ 12(4, 8) AE emission lines of dimethyl ether at 42371.58 MHz and 42372.16 MHz, respectively.

The H(83)δ recombination line in the ^{29}SiO spectrum is also visible in W51e2. The H(83)δ recombination line lies well within 1 MHz of the ^{29}SiO emission line, is thermally broadened, and is easily mistaken as being part of the ^{29}SiO emission line wings (Figure 5.1). Without removal of this overlapping line, the measured [^{29}SiO]/[^{28}SiO] would be in error by over 40%. The H(83)δ recombination line was effectively removed by using the 'spare' IF to observe the nearby and stronger H(53)α recombination line, which was then used as a template profile to fit and subtract the H(83)δ line from the ^{29}SiO spectrum (e.g., Figure 5.1). As a precaution, the H(53)α line was monitored in all other sources, although it was only observed in W51e2.

5.3 Optical Depth Effects

Historically, SiO emission has been assumed to be optically thin (Wolff, 1980) due to the modest brightness of the observed lines. However, Penzias (1981) was quick to demonstrate that SiO thermal emission often contravenes this assumption, and the same was found to be true for this survey. Many studies of interstellar isotope ratios categorize emission lines into one of two groups; optically thin ($\tau_0 \ll 1$) and optically thick ($\tau_0 \gg 1$) (e.g., Adande and Ziurys, 2012; Milam et al., 2005; Savage et al., 2002). Lines are then analyzed in the appropriate limit. This approach has the convenience of simplicity and is a concession to the difficulty in assessing optical depth in radio emission lines (Goldsmith and Langer, 1999).

Many emission lines, however, will not be patently either thick ($\tau_0 \gg 1$) *or* thin ($\tau_0 \ll 1$), and instead are likely to exhibit some finite intermediate values for τ (e.g., Milam et al., 2005; Penzias, 1981a; Savage et al., 2002). This should be especially true for emission from dense gas tracers like SiO, where even moderately bright lines from highly subthermal populations require appreciable optical depths (Shirley, 2015).

Knowing this, the assumption that the myriad of emission lines from dense gas tracers bin neatly into either the optically thin or optically thick limit seems unreasonable. Doing so will likely result in significant errors. Use of the thin limit appears particularly problematic as error grows rapidly with optical depth, reaching $\approx 10\%$ for even a moderate optical depth at line center of $\tau_0 = 0.2$. Optical depth must be addressed directly if the precision required to produce useful isotopic ratio measurements is to be attained. To this end, a novel metric for the direct determination of optical depth was developed as part of this study.

5.3.1 The Shape Parameter

For any given source, optical depth in the emission line of the common isotopologue can be assessed via comparison to the (presumably) thin emission line of one or more rare isotopologues. Both peak intensity T_0 and integrated area W of a spectral line are non-linearly dependent on peak optical depth, with peak intensity exhibiting a stronger dependence than integrated area. This is a consequence of the fact that optical depth varies across the line profile. The line wings will remain thin even as the line center depth increases, and the contribution of optically thin emission desensitizes the integrated area of the line with respect to line center optical depth τ_0. As a result high optical depth in the emission line of the common isotopologue will manifest as broadening relative to the rarer isotopologue lines that is obvious when the emission are scaled by intensity (Figure 5.5b).

By first integrating the emission lines of all isotoplogues through the same velocity range ($\Delta v_r = \pm 3$ FWHM from line center, as defined by the ^{28}SiO line) the lines of the rare isotopologues can then be scaled by the ratio of the common-to-rare integrated main beam temperatures. For example the scaled temperature of the ^{29}SiO emission line is given by

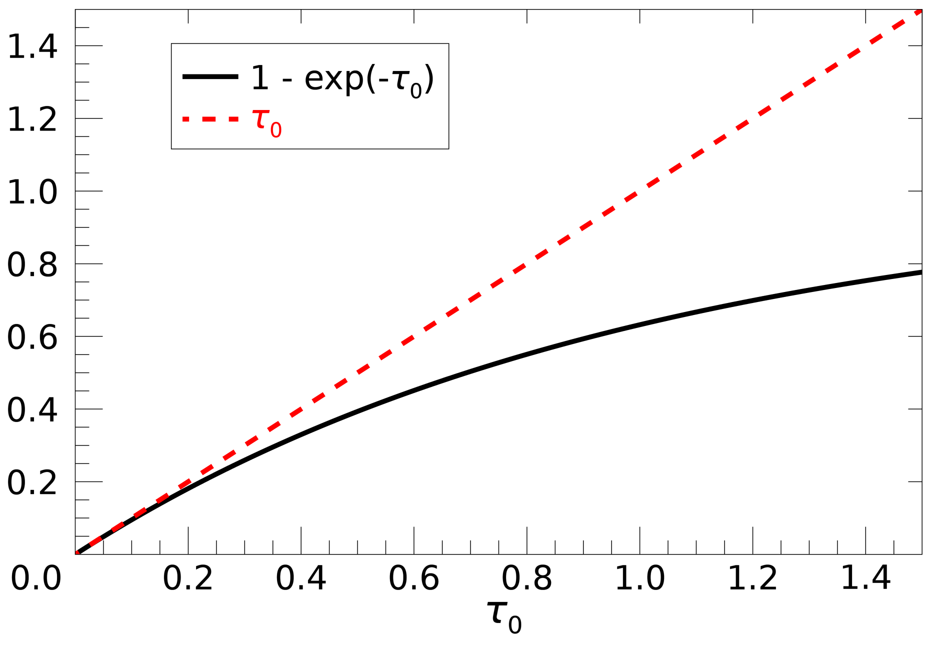


Figure 5.2: The line-center value of the opacity term in the solution to the radiative transfer equation, given as a function of peak optical depth. The black solid line is the exponentiated opacity term from the plane-parallel solution to the radiative transfer equation, i.e., the bracketed term in $I_\nu = B_\nu[1 - \exp(-\tau_0)]$. The red dashed line is the linear opacity term from the optically-thin approximation to the solution, $I_\nu = B_\nu\tau_0$. The ordinate is unitless.

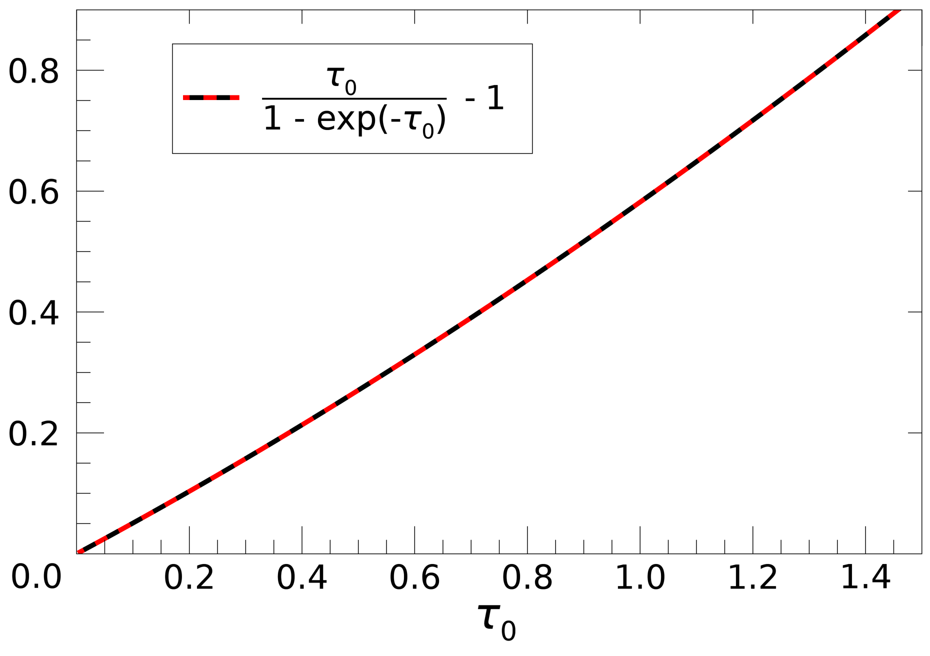


Figure 5.3: The fractional error of the line-center value of the optically-thin approximation of the opacity term relative to the full solution, again as a function of peak optical depth. It is clear that the optically-thin approximation rapidly accumulates error as τ_0 increases.

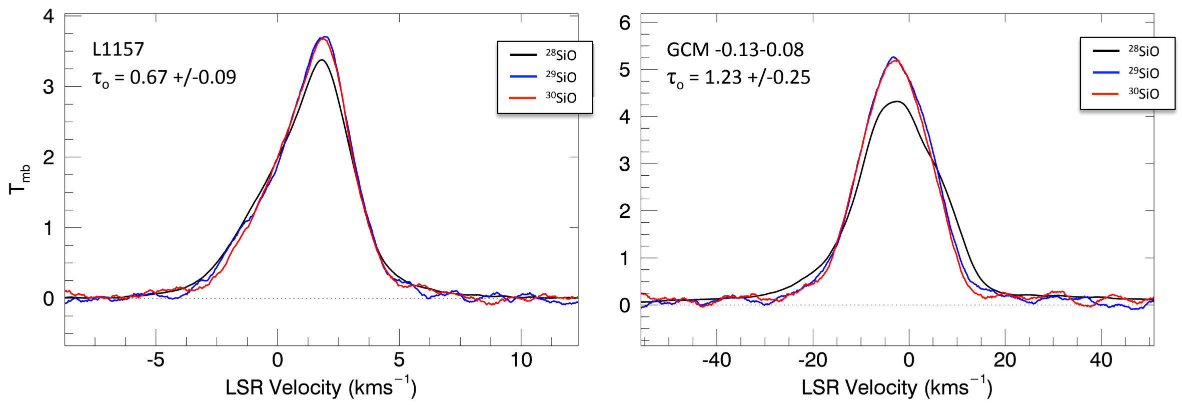


Figure 5.4: Area-scaled emission line profiles for the $v = 0, J = 1 \rightarrow 0$ transitions for the three SiO silicon isotopologues observed in L1157 (left) and GCM-0.13-0.08 (right). The black lines are the ^{28}SiO lines. The red and blue profiles are the ^{29}SiO and ^{30}SiO lines, respectively. Each line has been scaled by integrated intensity relative to the^{28}SiO integrated intensity as in Equation (5.3). The disparities between line-center $T_{\rm mb}$ values for the ^{28}SiO lines and the scaled ^{29}SiO and ^{30}SiO lines are indicative of appreciable optical depths in the ^{28}SiO lines (see text).

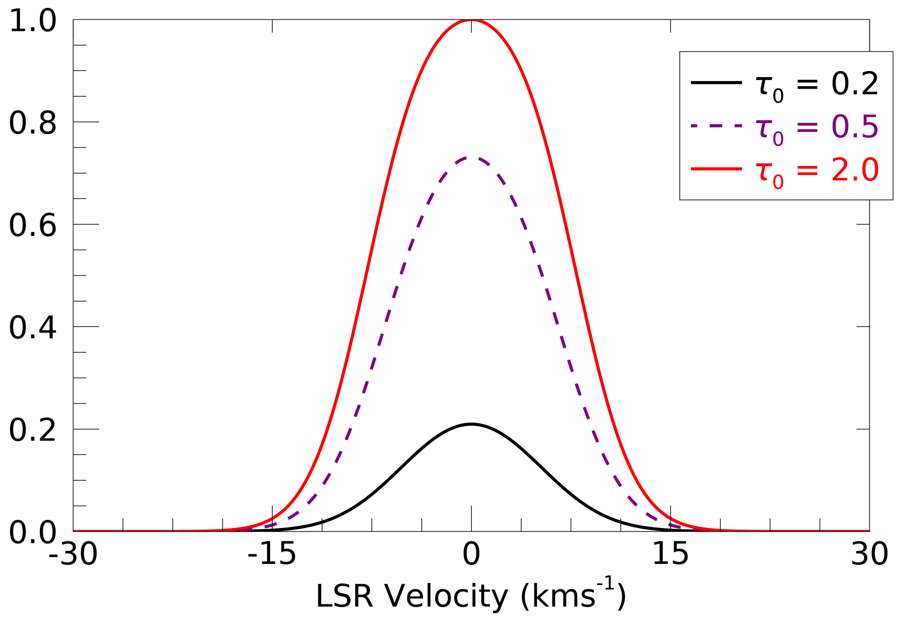
$$T^{^{29}\rm SiO}_{\rm scaled}(v_{\rm r}) = T^{^{29}\rm SiO}_{\rm mb}(v_{\rm r}) \left(\frac{\int T^{^{28}\rm SiO}_{\rm mb}(v_{\rm r})\, dv_{\rm r}}{\int T^{^{29}\rm SiO}_{\rm mb}(v_{\rm r})\, dv_{\rm r}} \right) . \tag{5.3}$$

With this method, optical depth in the ^{28}SiO emission line is determined by analyzing the difference between the ^{28}SiO main beam temperature and ^{29}SiO and/or ^{30}SiO scaled temperatures for the same source under the assumption that the latter is effectively optically thin. Because the scaled ^{29}SiO and ^{30}SiO lines have comparable (presumably low) optical depths based on their normal abundances, any broadening in the ^{28}SiO line is immediately obvious as an apparent deficit in peak temperature at line center when the scaled ^{29}SiO and ^{30}SiO lines are superimposed on the ^{28}SiO line (Figure 5.5c and Figure 5.4).

The ratio of the main beam temperature at line center to the velocity-space integrated main beam temperature, denoted Γ, can be expressed as

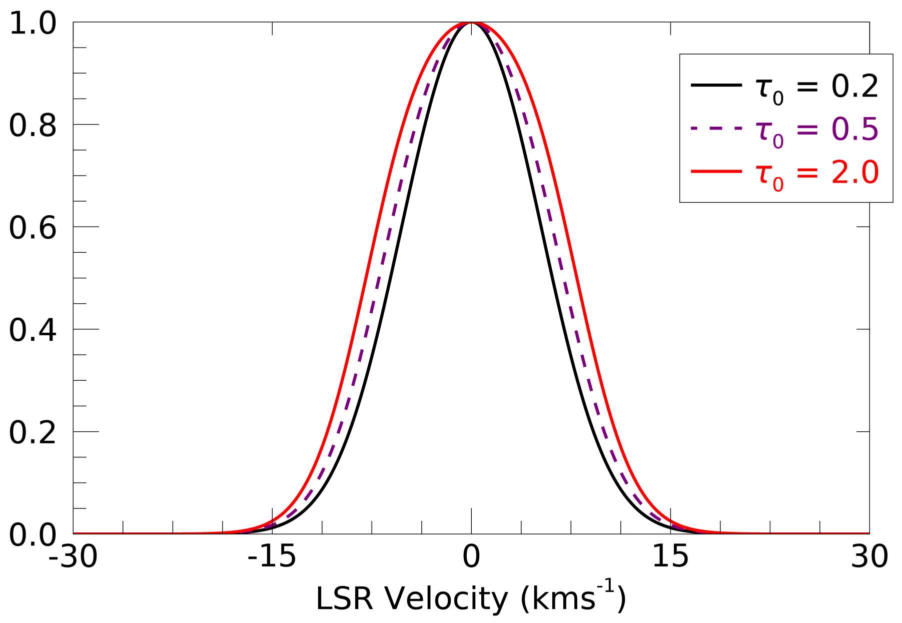
$$\Gamma = \frac{T_{\rm mb}(0)}{\int T_{\rm mb}(v_{\rm r})\, dv_{\rm r}} = \frac{T_{\rm mb}(0)}{W_{\rm mb}} \tag{5.4}$$

where $W_{\rm mb} = \int T_{\rm mb}(v_{\rm r})\, dv_{\rm r}$ is the velocity-space integrated main beam temperature.

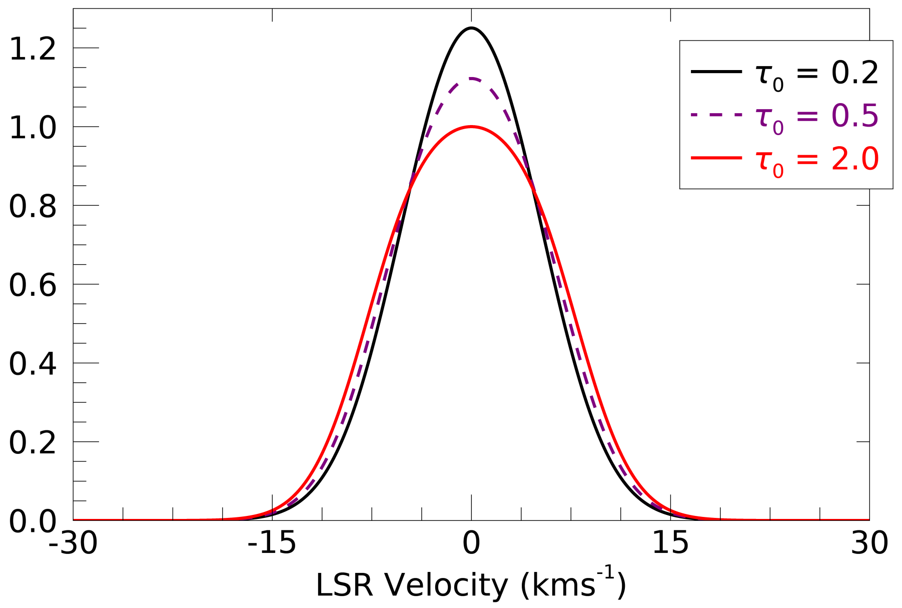


(a) Three synthetic emission lines at various values of τ_0. All three features are normalized to the value of the $\tau_0 = 2.0$ feature at line center. Thus, being of lesser intensity than the $\tau_0 = 2.0$ feature at line center, the $\tau_0 = 0.2$ and $\tau_0 = 0.5$ features take on values < 1.

Figure 5.5: The visible effects of optical depth, seen here in a trio of synthetic line profiles. These plots are non-physical, and are intended only as a visual aid. All three features in Subfigures 5.5a, 5.5b and 5.5c are normalized to the value of the $\tau_0 = 2.0$ feature at line center; thus all ordinates are unitless.



(b) The three emission lines from Subfigure 5.5a, but with the $\tau_0 = 0.2$ and $\tau_0 = 0.5$ features scaled to be equal in *peak intensity* to the $\tau_0 = 2.0$ feature. The $\tau_0 = 0.2$ and $\tau_0 = 0.5$ features are scaled by factors of 4.77 and 1.37, respectively. The broadening effect of increased optical depth is clearly visible.



(c) The three emission lines from Subfigure 5.5a, but with the $\tau_0 = 0.2$ and $\tau_0 = 0.5$ features scaled to be equal in *integrated area* to the $\tau_0 = 2.0$ feature. The $\tau_0 = 0.2$ and $\tau_0 = 0.5$ features are scaled by factors of 5.96 and 1.53, respectively. The difference between the *shape parameter* of each feature is clearly visible.

Figure 5.5: (cont) The visible effects of optical depth. All three subfigures are normalized to the value of the $\tau_0 = 2.0$ feature at line center. All ordinates are unitless.

The value Γ, dubbed the *shape parameter*, is of critical importance as it can be used to quantify optical depths. As optical depth increases, the shape parameter Γ decreases (the profile shape becomes fatter). The optical depth of an emission line can therefore be quantitatively determined by comparing the line shape parameter of the suspected optically thick line (for the abundant isotopologue) with that for a line that is presumed to be optically thin (corresponding to the rare isotopologues). For moderate optical depths, the optical depth at line-center for the optically thick line is a function of the fractional difference between shape parameters for the thick and thin lines:

$$\tau_0 = f\left(\frac{\Gamma^{\text{thin}}}{\Gamma^{\text{thick}}} - 1\right) = f(\Gamma'') \tag{5.5}$$

where the value Γ'' is dubbed the *shape parameter excess*. Evaluation of synthetic data indicates that the empirically-derived proportionality of $d\tau_0/d\Gamma'' = 6.829 - 0.879\ \Gamma'' + 4.698\ \Gamma''^2$ produces results with exceptional accuracy ($\ll 1\%$) up to values of $\tau_0 = 2$ when used in conjunction with Equation (5.5). It should be noted that this polynomial equation was derived from synthetic gaussian lineshapes relative to a perfectly optically-thin line.

A previously derived proportionality constant of ≈ 5 was reported by Monson et al. (2017). The proportionality function presented here, if correct, would push the δ'^{29}Si and δ'^{30}Si values of optically-thick sources (e.g. L1157) closer to the Solar value (see Figures 6.2 and 6.3). Such a shift would likely reduce the difference between the reported values of δ'^{29}Si and δ'^{30}Si in all seven sources to well less than 1σ; effectively eliminating *any* evidence of a gradient in both [^{29}SiO]/[^{28}SiO] and [^{30}SiO]/[^{28}SiO] with respect to R_{GC}. Clearly the disparity between the value given here and the value reported by Monson et al. (2017) is non-trivial.

A concerted effort by the Author to determine the exact nature of the relationship between Γ'' and τ_0 using RADEX is currently underway. However, it seems likely that the difference between the proportionality function reported here and that reported by Monson et al. (2017) is due to a combination of factors. The model used by Monson et al. (2017) to to derive a proportionality constant of ≈ 5 was somewhat oversimplified, and differences in natural linewidth between isotopologues and the effect of the frequency-cubed term in Equation (5.6) were omitted from the modeling presented here. Modeling based on RADEX will be free of these errors and will decisively resolve the issue. RADEX is also being used to evaluate the relationship between Γ'' and τ_0 for values of $\tau_0 > 2$.

All of the line-center optical depths obtained as part of this study are < 1.5 (see Tables 6.1 and 6.2). The process of determining τ_0 using Γ'' was tested using synthetic data and found to be robust against irregular line profiles and even cryptically overlapping velocity components from separate clumps within a complex source. It is worth emphasizing that influences of velocity structure on line shapes are not isotope specific, and forward calculations verify that optical depth effects alone result in the departures from line shape coincidence when normalized to area. A caveat is that there are hypothetical circumstances where one can imagine localized velocity features that affect the rare isotopologues differently than the abundant species, but these will be pathological circumstances.

Another caveat is that, if there are strong gradients in excitation temperature along the line of sight, then an optically thick line for the abundant species will favor the foreground values of excitation temperature for that species only, leading to an error in the abundance ratio. This is a known and important effect for very optically thick lines like those of $^{12}C^{16}O$; indeed, in the absence of a velocity gradient along the line of sight, one is typically observing only the surface layers of a cloud in the most abundant isotopologue of CO. However, for SiO this effect is minimized because of the comparatively modest values of optical depth for the SiO lines. Furthermore, there is little reason to expect strong line-of-sight excitation gradients in the kinds of sources that give rise to SiO emission; the SiO molecules are likely intermixed with the shocks that liberate or form them.

Failures of Equation (5.5), by my observation, require models that invoke rather unlikely circumstances. Further, forward calculations demonstrate further that details of line shapes (e.g., skewness) do not significantly alter the relationship between Γ and line-center optical depth as long as the line profile is not flat-topped.

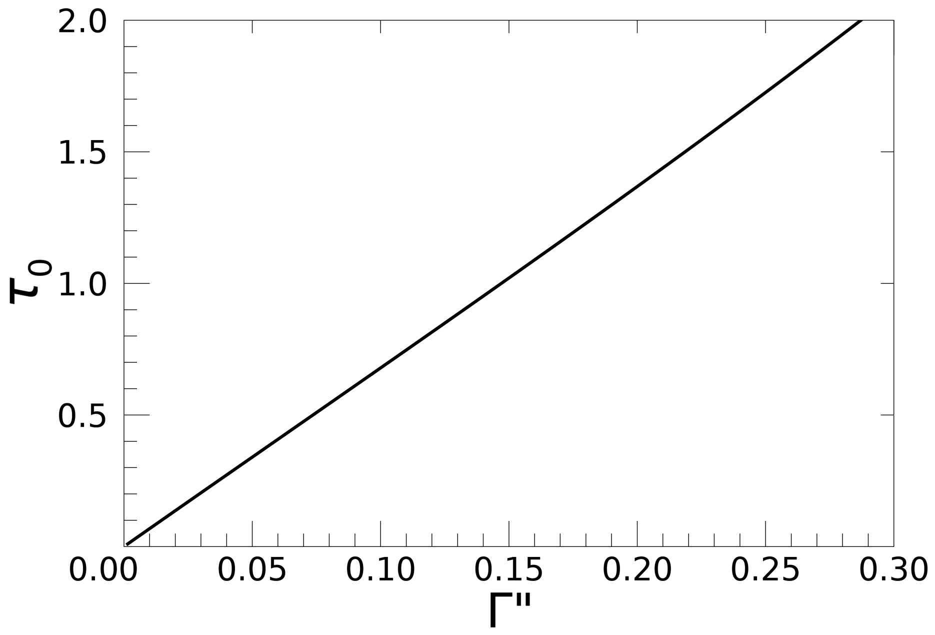


Figure 5.6: The quasilinear relation between optical depth at line center and the shape parameter excess, defined as $\Gamma'' = (\Gamma^{\rm thin}/\Gamma^{\rm thick}) - 1$ (equivalent to Equation (5.5)). The shape parameter excess was calculated using gaussian lineshapes relative to a perfect optically-thin line, i.e. $\tau_0 = 0$.

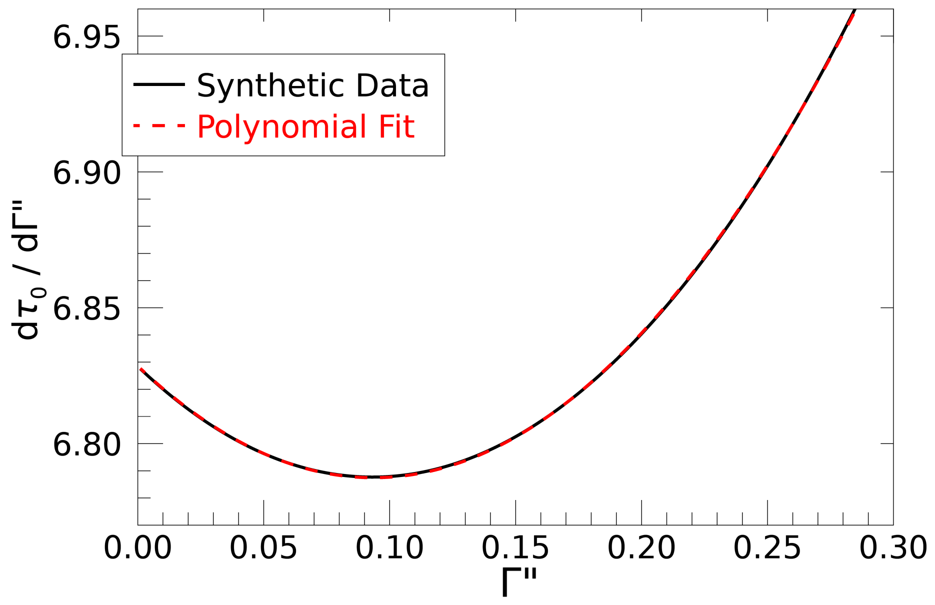


Figure 5.7: The first derivative of τ_0 with respect to Γ'', plotted as a function of Γ''. Like Figure 5.6, the value of Γ'' was calculated using gaussian lineshapes relative to a perfectly optically-thin line. It is clear from the small range of the ordinate that the relation between Γ'' and τ_0 is very nearly linear. The polynomial fit (in red) is given by $d\tau_0/d\Gamma'' = 6.829 - 0.879\ \Gamma'' + 4.698\ \Gamma''^2$.

5.4 Extracting Column Densities from Line Intensities

In order to extract isotopologue ratios from line intensities, one should forgo the Rayleigh-Jeans approximation[1]and express the upper level population column density ratio of secondary (i.e., rare in this application) and primary (abundant in this application) isotopologues N_u^s/N_u^p as

$$\frac{N_u^s}{N_u^p} = \frac{\Lambda^s W^s}{\Lambda^p W^p}\left(\frac{\nu_{u\ell}^p}{\nu_{u\ell}^s}\right)^3\left[\frac{1 - n_\gamma(T_{\rm crf})/n_\gamma^p(T_{\rm ex})}{1 - n_\gamma(T_{\rm crf})/n_\gamma^s(T_{\rm ex})}\right]. \tag{5.6}$$

As in Equation (5.4), W_p and W_s are the integrated line intensities for the primary and secondary silicon isotopologues, p and s respectively. The photon occupation numbers $n_\gamma^p(T_{\rm ex})$ and $n_\gamma^s(T_{\rm ex})$ correspond to the excitation temperatures for isotopologues p and s, and $n_\gamma(T_{\rm crf})$ corresponds to the radiation temperature of the local continuum radiation field. The factors Λ^p and Λ^s correct for optical depth in the emission lines for the two isotopologues and are described in the following section. For a full derivation of Equation (5.6) please refer to Appendix A.2.2.

The term in brackets in Equation (5.6) is a consequence of forgoing the assumption that $n_\gamma(T_{\rm ex}) \gg n_\gamma(T_{\rm crf})$ and allowing $n_\gamma = (n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf}))[1 - \exp(-\tau_\nu)]$ rather than assuming that $n_\gamma \approx n_\gamma(T_{\rm ex})[1 - \exp(-\tau_\nu)]$. However, because of the likelihood for sub-thermal populations of SiO emitters, the excitation temperature $T_{\rm ex}$, and thus $n_\gamma(T_{\rm ex})$, is not known *a priori* and may differ between the three silicon isotopologues of SiO. Further, one should note that the flux contributions from the local continuum radiation field are effectively invariant between isotopologues, but the contributions of the line emission itself are not. While this additional term is undoubtedly of consequence, it cannot be put to direct use when extracting column densities using Equation (5.6) without extensive modeling. The implications of this additional term and the effect it has on measuring isotopologue ratios are discussed at in Section 5.4.2.

[1]The Rayleigh-Jeans approximation is inappropriate in these circumstances; for a highly sub-thermal emission from SiO, $h\nu_0 \approx k_{\rm b}T_{\rm ex}$ and thus the Rayleigh-Jeans approximation does not apply.

5.4.1 Correcting for Finite Optical Depth

The optical-depth correction Λ for measured column densities for ^{28}SiO takes the form

$$\Lambda^{28} = \frac{N^{28}_{\mathrm{corrected}}}{N^{28}_{\mathrm{uncorrected}}} = \frac{\int \tau_{v_{\mathrm{r}}} dv_{\mathrm{r}}}{\int \left(1 - \exp(-\tau_{v_{\mathrm{r}}})\right)\, dv_{\mathrm{r}}}. \tag{5.7}$$

The integrals in this expression are obtained from the optical depths at line center derived using Equation (5.5) and the line profile functions defined by the ^{29}SiO lines (assumed to be optically thin). Although this correction factor was derived independently by the Author, one could use equations 83-85 from Mangum and Shirley (2015) to derive the same expression; however, in the text they instead appeal to an expression analogous to Equation (5.7) attributable to Goldsmith and Langer (1999) that is not correct for application with Equation (5.6). For a full derivation of Equation (5.7), please again refer to Appendix A.2.2.

Equations (5.4), (5.5), and (5.7) are used to determine optical depths for the ^{28}SiO lines for all sources reported here. In all cases the two derived τ_0 values, based on the ^{29}SiO and ^{30}SiO shape parameters, are in agreement within uncertainties; the SNR-weighted average of the two is used to calculate the τ_0 value reported for each source. Values for τ_o differ for the different sources, with values ranging from below detection to slightly greater than unity. The ^{28}SiO lines from DR21(OH) and AFGL 5142 have optical depths below detection, with a noise-limited detection limit of $\tau_0 \approx 0.2$. The T_{mb} values for the ^{28}SiO lines are less than 1K in both sources. The peak T_{mb} of the ^{28}SiO emission line in W51e2 is $\approx$ 3K and is also relatively optically thin, with an estimated optical depth of 0.4. The two Galactic center sources and L1157, by contrast, all show evidence for appreciable optical depth in the main ^{28}SiO emission line, with estimated optical depths of 1.0, 1.2, and 0.7, respectively (Tables 6.1 and 6.2).

5.4.2 Excitation Effects

The bracketed term in Equation (5.6) is included to account for the possibility of isotopologue-specific subthermal excitation effects. Although the flux contribution from the local continuum radiation field, expressed here as $n_\gamma(T_{\rm crf})$, is effectively invariant between isotopologues, feedback in the line radiation field (called line trapping) will have a differential effect on emission any time local thermodynamic equilibrium (LTE) does not obtain.

Though typically ignored when optical depths are low, trapping of line radiation will have an effect on the level populations; the net effect of which is to increase the excitation temperature of any affected transitions relative to the excitation temperature that would occur if the only radiative contribution was the continuum radiation field. Clearly, the magnitude of this effect is proportional to the intensity of the radiation field (both continuum and line contributions) within the emission line profile. Assuming that the continuum radiation field is isothermal, differences in emission line strength, and hence differences in the degree of radiative trapping, will differ between isotopologues. That is to say, although the flux contribution from the local continuum radiation field is effectively invariant between isotopologues, the contribution of the line emission itself is not, and *this will cause their excitation temperatures to diverge.* This conclusion is of critical importance.

Emission from SiO is typically modest in strength, the effect of line trapping on $n_\gamma(T_{\rm ex})$ will be comparably modest. So, for a highly excited population, the increase in excitation as a consequence of line trapping is small enough to have little or no effect observable on emission, even considering $n_\gamma(T_{\rm ex})$ will vary between isotopologues. However, this is not the case for highly sub-thermal populations ($T_{\rm ex} \ll T_{\rm k}$) where the radiation field plays a dominant role in determining the excitation. If $n_\gamma(T_{\rm crf})$ is low (e.g., approaching the cosmic microwave background), even small differences in $n_\gamma(T_{\rm ex})$ can effect large changes in emission line intensity. I.e. line trapping can pump up the isotopologue-dependent excitation temperatures and produce inaccuracies in the derived isotopologue ratios.

The conditions that foster the isotopologue-selective excitation effects can be illustrated using an expression for excitation temperature for a two-level system (e.g. Goldsmith, 1972):

$$T_{\mathrm{ex}} = \frac{\Delta E_{u\ell}}{k_{\mathrm{b}}} \left[\ln \left(\frac{A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}}{\bar{n}_\gamma A_{u\ell} + C_{u\ell}\exp(-\Delta E_{u\ell}/k_{\mathrm{b}}T_{\mathrm{k}})} \right) \right]^{-1}, \tag{5.8}$$

where $A_{u\ell}$ is the Einstein coefficient for spontaneous emission and $C_{u\ell}$ is the collisional de-excitation rate, $\bar{n}_\gamma$ is the radiative contribution to the excitation temperature in the transition, T_{k} is the kinetic temperature, and the other symbols have their usual meanings. A derivation of Equation (5.8) can be found in Appendix A.3.1.

In Equation (5.8), the line-weighted photon occupation number $\bar{n}_\gamma$ is comprised of two separate components; $n_\gamma(T_{\mathrm{crf}})$ and $n_\gamma(T_{\mathrm{line}})$ corresponding to the bacrground continuum radiation and line radiation components respectively. The collisional de-excitation rate depends on the number density of molecules,[2] thus as number density tends to zero, $C_{u\ell} \to 0$, and $T_{\mathrm{ex}} \to (T_{\mathrm{crf}} + T_{\mathrm{line}})$. In this sub-thermal limit, $n_\gamma(T_{\mathrm{crf}})$ and $n_\gamma(T_{\mathrm{line}})$ compete for dominance in determining the excitation temperature T_{ex}.

Recall that for an emission line from an isothermal source, the observed photon occupation number is given by

$$n_\gamma = [n_\gamma(T_{\mathrm{ex}}) - n_\gamma(T_{\mathrm{crf}})](1 - \exp(-\tau_\nu). \tag{5.9}$$

Thus, when the incident continuum radiation field $n_\gamma(T_{\mathrm{crf}})$ is weak (at or just above the cosmic microwave background), even small differences in $n_\gamma(T_{\mathrm{ex}})$, e.g. between isotopologues, constitute large fractional changes in the *difference* between $n_\gamma(T_{\mathrm{ex}})$ and $n_\gamma(T_{\mathrm{crf}})$. This in turn effects significant differences in the photon occupation n_γ of the emergent line. To reiterate, because $n_\gamma \propto n_\gamma(T_{\mathrm{ex}}) - n_\gamma(T_{\mathrm{crf}})$, n_γ becomes increasingly sensitive to small differences in $n_\gamma(T_{\mathrm{ex}})$ as $n_\gamma(T_{\mathrm{ex}})/n_\gamma(T_{\mathrm{crf}}) \to 1$. An analytical expression quantizing this effect was presented in Section 5.6. It is clear to see that this effect is most significant in environments with weak continuum radiation fields, but diminishes rapidly as the intensity of the continuum radiation field increases. Conversely, as the number density tends to infinity and thus $C_{u\ell} \to \infty$, $T_{\mathrm{ex}} \to T_{\mathrm{k}}$ and the system is in LTE and, assuming optical depth is

[2]See Appendix A.3.1

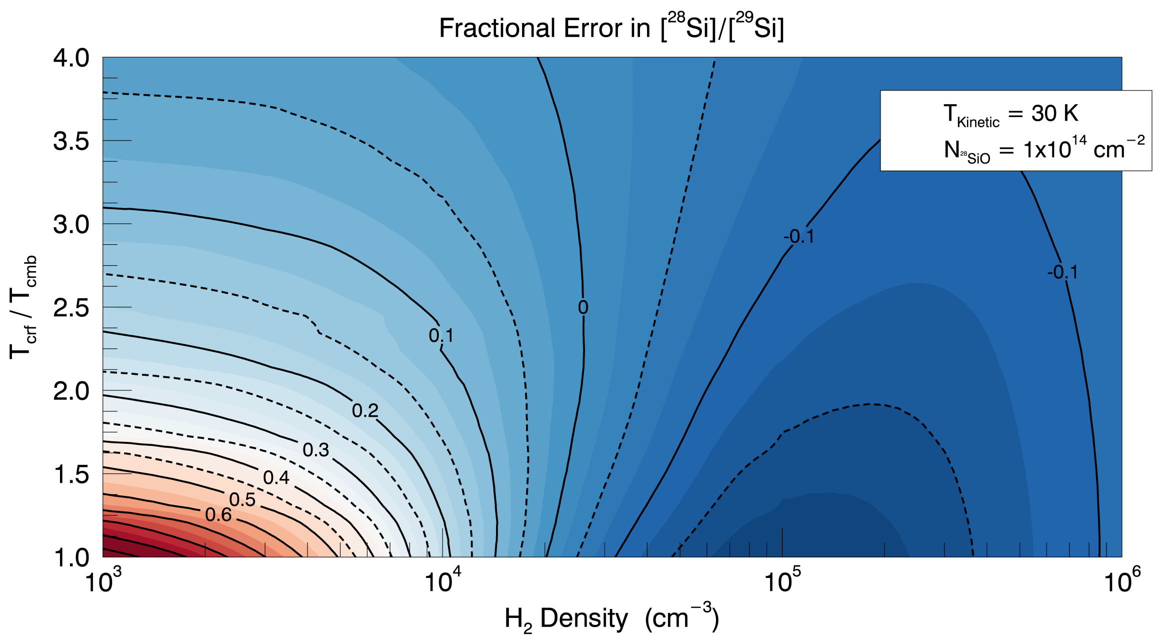


Figure 5.8: Contours of errors in SiO isotopologue ratios obtained from integrated J $= 1 \rightarrow 0$ emission line areas as a function of collision partner number density and the temperature of the incident continuum radiation field, $T_{\rm crf}$, expressed in multiples of the cosmic microwave background temperature, $T_{\rm cmb}$. As indicated in the inset, the kinetic temperature of the gas and the column density of ^{28}SiO are fixed at 30 K and 1×10^{14} respectively. The ratios are fully corrected for optical depth, thus the error is due purely to disparate excitation between the ^{28}SiO and ^{29}SiO isotopologues. Solid contours are fractional differences between the [^{28}Si] / [^{29}Si] extracted from the model and the input parameters (Solar in all cases) in increments of ± 0.1. Dashed contours represent midpoints between solid contours, and are plotted only where the magnitude of the fractional error is < 0.1, as the gradient in the data is comparatively shallow in that region. Radiative transfer calculations were made with RADEX using the large velocity gradient approximation (van der Tak et al., 2007).

not extreme, there are no significant isotopologue-specific effects due to line trapping. In summary, Equation (5.8) shows that sub-thermal excitation and low continuum radiation temperatures facilitate isotopologue-specific emission effects driven by line trapping.

Because the rotational states of SiO are subthermally populated in at least some (likely all) of the observed sources (e.g., Amo-Baladrón et al., 2009; Nisini et al., 2007), and probably in all (Harju et al., 1998), the potential biases in the derived isotopologue ratios attributable to this phenomenon were evaluated. RADEX (van der Tak et al., 2007) was used to constrain the magnitude of error induced by divergent excitation temperatures among isotopologues as a function of H_2 density and continuum radiation field intensity. The large velocity gradient approximation was used for the calculations presented here. Calculations using alternate geometries do not yield appreciably different results from those shown here.

Figure 5.8 shows contours of fractional deviations in measured optical-depth-corrected isotopologue ratios from the true ratios as a function of collision partner number density and the temperature of the local continuum radiation field. The kinetic temperature is assumed to be 30K, but the results are insensitive to the kinetic temperature as long as $T_\mathrm{k} \gtrsim 10$K. The contours illustrate that errors well in excess of 20% are expected in low H_2 density, low continuum flux environments (e.g., for H_2 number densities $n_{\mathrm{H}_2} < 5 \times 10^3$ and T_crf less than twice the CMB) if the excitation effects go unrecognized. Published descriptions of the targets in this study report strong sources of millimeter continuum emission in proximity to the SiO emission sources, typically in the form of either ultra-compact H II regions or winds from nearby high-mass young stellar objects (e.g., Araya et al., 2009; Hunter et al., 1999; Luisi et al., 2016; Zapata et al., 2009). Therefore, the temperatures of the continuum radiation within the observed sources are by all evidence well in excess of the CMB, mitigating isotope-specific excitation effects. Similarly, for the SiO sources reported here $10^4 \lesssim n_{\mathrm{H}_2} \lesssim 10^6$ cm^{-3} and so the environs of these sources correspond to conditions where systematic errors are likely to be < 10% (Figure 5.8), commensurate with the measurement errors. While radiation field effects are an important consideration, they do not appear to be sufficient to significantly alter the isotopologue ratios extracted from the data in this study.

5.5 Evaluation of Uncertainties

In order to account for both measurement uncertainties and the uncertainties imparted by the estimates of optical depth, the entire data reduction pipeline and correction scheme for each source was subjected to a pair of Monte Carlo error analyses. To assess the errors imparted by noise in each spectrum, random draws of each channel in each spectrum were made, where the measured value of each channel used as the mean and the standard deviation set equal to the calculated RMS noise temperature. A total of 500 draws were made for each spectrum, and every trio of draws was fully reduced and corrected for the calculated optical depth. The use of the measured values as means (rather than smoothed values) results in the uncertainties in the derived isotopologue ratios being overestimated by $\approx 5\%$.

Due to the somewhat subjective nature of selecting the region used to fit the baseline of each spectrum, this process was subject to a similar Monte Carlo error analysis. Line free regions of each spectrum were fit by hand, then random draws of the boundaries of each region were made, with the standard deviation set to either 5% or 10% of the width of each region. A total of 400 draws were made for each spectrum. These two analyses were nested in such a way that Monte Carlo analysis of the noise temperature was performed for each random draw of the baseline limits. The result is $200,000$ instances of each of the three SiO lines for each source, each fully reduced and corrected for the calculated optical depth.

The corrections for optical depth in the ^{28}SiO lines (Tables 6.1 and 6.2) generally increase the uncertainties in isotopologue ratios by factors of approximately 2 to 3. Because of the additional uncertainty in the abundant isotologue column densities, the correlation coefficients between the [^{29}SiO]/[^{28}SiO] and [^{30}SiO]/[^{28}SiO] ratios increase from < 0.1 to 0.85 ± 0.2 in all of the sources.

Adapting the Monte Carlo codes to the "band swapped" data obtained in 2015 (AGBT15A-350) has proven troublesome. Combining data from the two spectral windows for each line effectively doubles the baselines in the reduced data, severely complicating the process of fitting and removing them. Additionally, the frequency windows themselves are relatively narrow, and as a result there is little space on either side of the SiO emission line for fitting the baseline structure. Attempts to find acceptable solutions to these issues are ongoing.

CHAPTER 6

Results

A summary of the results is given in Tables 6.1 and 6.2 and shown in Figures 6.2 and 6.3. The uncorrected data exhibit a spread up and down the slope-1 line in Si three-isotope space, anchored by the two Galactic center sources and crudely resembling the predictions from GCE (Figure 6.2). The trend with R_{GC} is broken by the high $[^{29}\mathrm{SiO}]/[^{28}\mathrm{SiO}]$ and $[^{30}\mathrm{SiO}]/[^{28}\mathrm{SiO}]$ ratios for L1157 at Solar R_{GC}. However, correcting for optical depth removes the spread in data, resulting instead in a clustering of the data spanning the range defined by the mainstream SiC presolar grain trend (Figure 6.3). I find, not surprisingly, that optical depths on the order of unity can strongly bias extracted isotope ratios. These results indicate that uncorrected optical depth effects were responsible for the prior evidence of high $[^{29}\mathrm{SiO}]/[^{28}\mathrm{SiO}]$ and $[^{30}\mathrm{SiO}]/[^{28}\mathrm{SiO}]$ ratios in the present-day ISM relative to Solar and meteoritical values (Penzias, 1981a; Wolff, 1980). The prior measurements were suggestive of GCE over the ≈ 4.6 Gyrs since the birth of the Sun and the formation of the presolar SiC grains. These new results suggest instead that silicon isotope ratios have been minimally affected by GCE over this time interval.

Correcting for optical depth removes the evidence for a variation in silicon isotope ratios with R_{GC} (Figure 6.4). Regression of the uncorrected $\delta'^{29}\mathrm{Si}$ values vs R_{GC} gives a negative slope (slope $= -27 \pm 12$ per mil kpc^{-1}, Figure 6.4) while regression of the corrected data yields a slope indistinguishable from zero (slope $= -0.2 \pm 6.8$ per mil kpc^{-1}, Figure 6.4).

It is worth noting that, with the exception of L1157, the sources that exhibit the highest optical depths are also the most distant, namely GCM -0.13-0.08 and GCM 0.11-0.11 which are both at a distance of ≈ 8 kpc. At this distance linear beam size is significant, and as both sources are large ($\approx 10^5 M_\odot$) filaments of the Sagatarius A GMC, the telescope beam likely

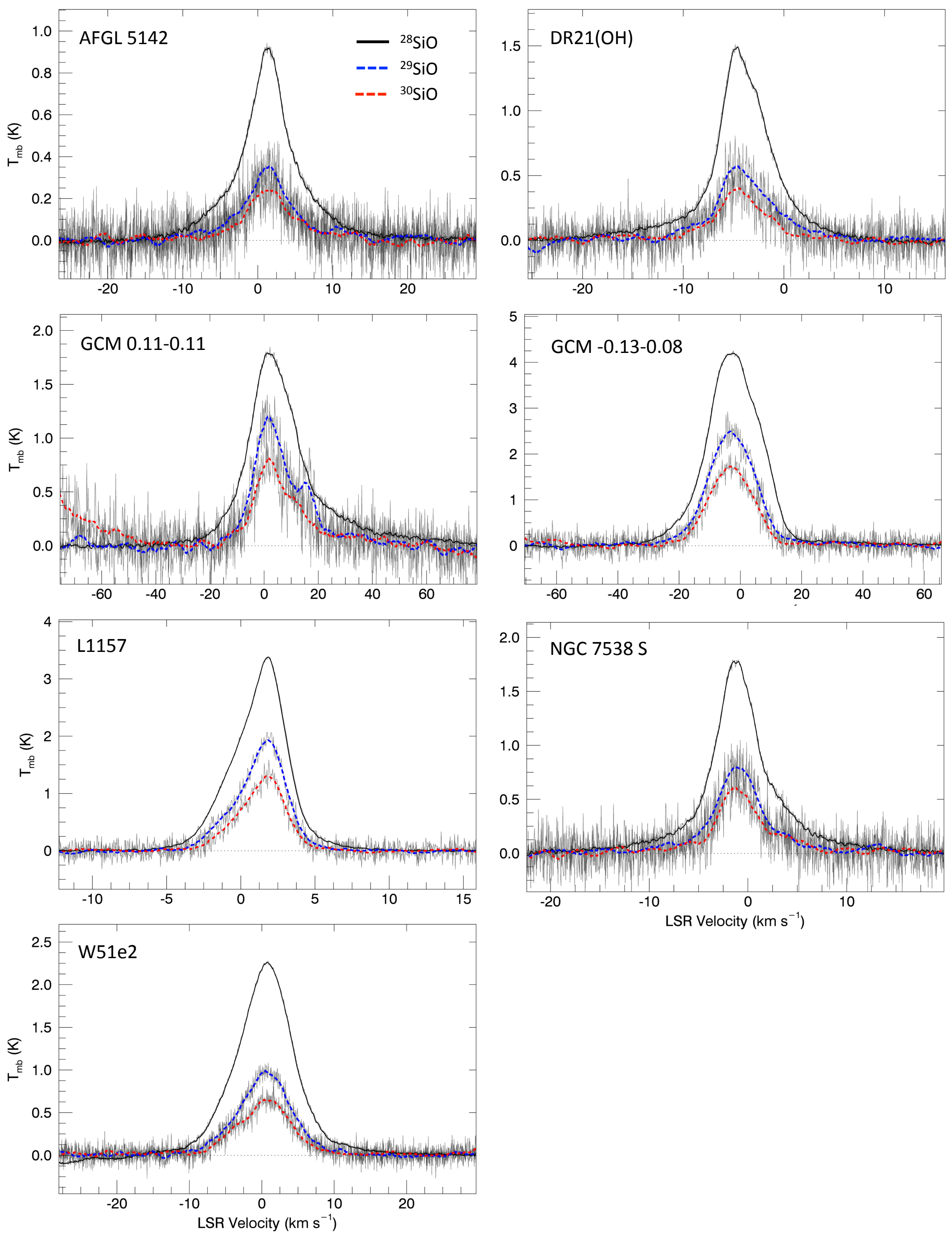


Figure 6.1: Calibrated and baseline-corrected ^{28}SiO, ^{29}SiO, and ^{30}SiO emission lines for the seven sources in this study. The grey lines show the unsmoothed, full-resolution spectra while the solid black, dashed blue, and dashed red lines are the smoothed data for ^{28}SiO, ^{29}SiO, and ^{30}SiO respectively. Main beam temperatures apply to ^{28}SiO while the ^{29}SiO and ^{30}SiO lines are scaled by a factor of 7 for presentation. The baseline at the low-velocity extreme of the ^{30}SiO spectrum for GCM0.11-0.11 is outside the range used to determine the baseline underlying the ^{30}SiO line itself.

			Uncorrected Ratios		Corrected Ratios	
Source	τ_0	Λ_{28}	$[^{28}\mathrm{Si}]/[^{29}\mathrm{Si}]$	$[^{28}\mathrm{Si}]/[^{30}\mathrm{Si}]$	$[^{28}\mathrm{Si}]/[^{29}\mathrm{Si}]$	$[^{28}\mathrm{Si}]/[^{30}\mathrm{Si}]$
DR21 (OH)	0.08 ± 0.21	1.03 ± 0.08	17.53 ± 0.54	25.73 ± 1.18	18.06 ± 1.68	26.52 ± 2.65
L1157 B1	0.67 ± 0.09	1.25 ± 0.04	12.63 ± 0.15	18.61 ± 0.35	15.76 ± 0.57	23.23 ± 0.92
NGC 7538 S	0.49 ± 0.22	1.18 ± 0.08	15.62 ± 0.49	21.10 ± 0.92	18.48 ± 1.69	24.97 ± 2.47
AFGL 5142	0.12 ± 0.26	1.04 ± 0.09	17.95 ± 0.73	24.26 ± 1.32	18.77 ± 2.22	25.37 ± 3.03
W51e2	0.39 ± 0.09	1.14 ± 0.04	16.47 ± 0.17	23.46 ± 0.39	18.83 ± 0.69	26.84 ± 1.03
GCM 0.11-0.11	1.23 ± 0.25	1.47 ± 0.10	12.61 ± 0.46	18.17 ± 0.91	18.61 ± 1.74	26.82 ± 2.90
GCM -0.13-0.08	0.97 ± 0.13	1.37 ± 0.05	12.63 ± 0.24	18.69 ± 0.39	17.27 ± 0.85	25.56 ± 1.31

Table 6.1: The calculated optical depth, correction factor Λ, corrected and uncorrected SiO isotopologue ratios for each source.

			Uncorrected δ' Values		Corrected δ' Values	
Source	τ_0	Λ_{28}	$\delta'^{29}\mathrm{Si}$	$\delta'^{30}\mathrm{Si}$	$\delta'^{29}\mathrm{Si}$	$\delta'^{30}\mathrm{Si}$
DR21 (OH)	0.08 ± 0.21	1.03 ± 0.08	117 ± 31	150 ± 46	90 ± 93	124 ± 99
L1157 B1	0.67 ± 0.09	1.25 ± 0.04	445 ± 12	473 ± 19	223 ± 36	252 ± 39
NGC 7538 S	0.49 ± 0.22	1.18 ± 0.08	232 ± 32	349 ± 44	67 ± 91	184 ± 98
AFGL 5142	0.12 ± 0.26	1.04 ± 0.09	94 ± 40	209 ± 54	55 ± 117	170 ± 119
W51e2	0.39 ± 0.09	1.14 ± 0.04	179 ± 11	242 ± 16	45 ± 36	108 ± 38
GCM 0.11-0.11	1.23 ± 0.25	1.47 ± 0.10	446 ± 36	499 ± 50	61 ± 93	113 ± 108
GCM -0.13-0.08	0.97 ± 0.13	1.37 ± 0.05	444 ± 19	469 ± 21	133 ± 49	157 ± 51

Table 6.2: The same as Table 6.1, but with δ' values in permil for each source in lieu of isotopologue ratios.

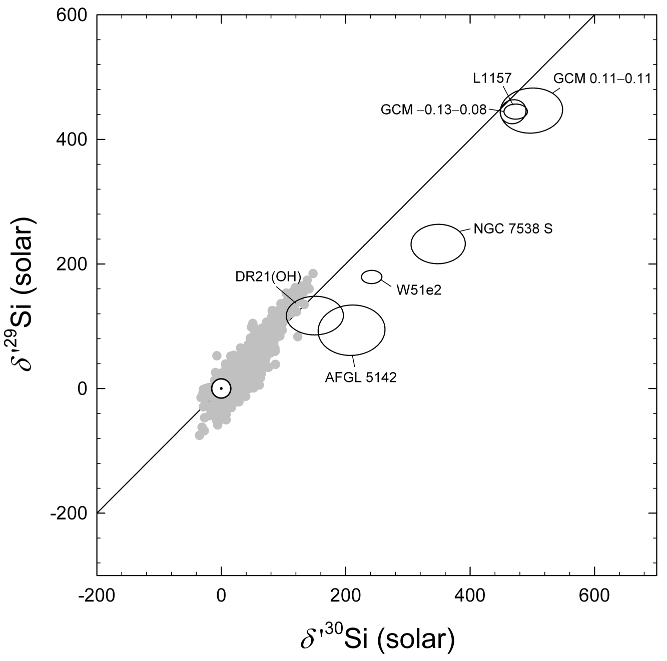


Figure 6.2: Uncorrected SiO silicon isotope abundance ratios for the seven sources observed as part of this survey. Mainstream SiC grain data are shown for reference (grey circles). The solid line is the slope-unity line through the Solar composition. The white circle with dot indicates present-day Solar abundances and defines the origin. Error ellipses are 1σ.

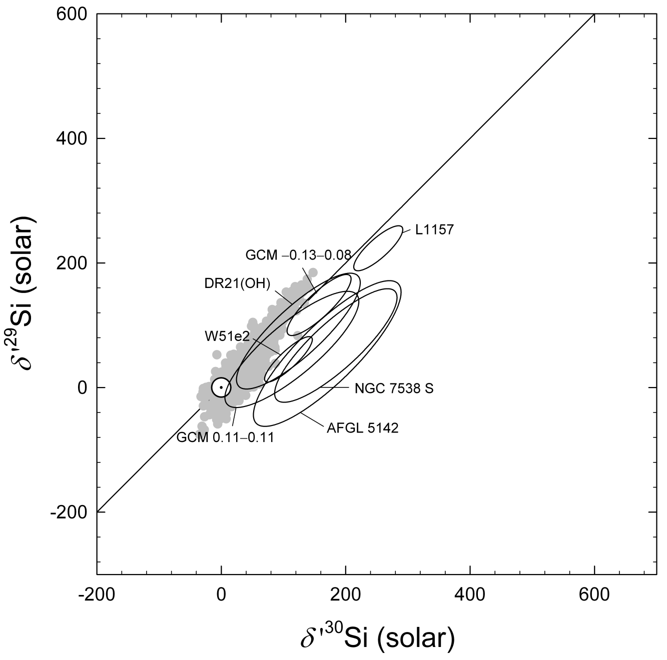


Figure 6.3: SiO silicon isotope ratios after correcting for optical depth effects. Error ellipses are 1σ determined by Monte Carlo simulations including the uncertainty in the optical depth corrections.

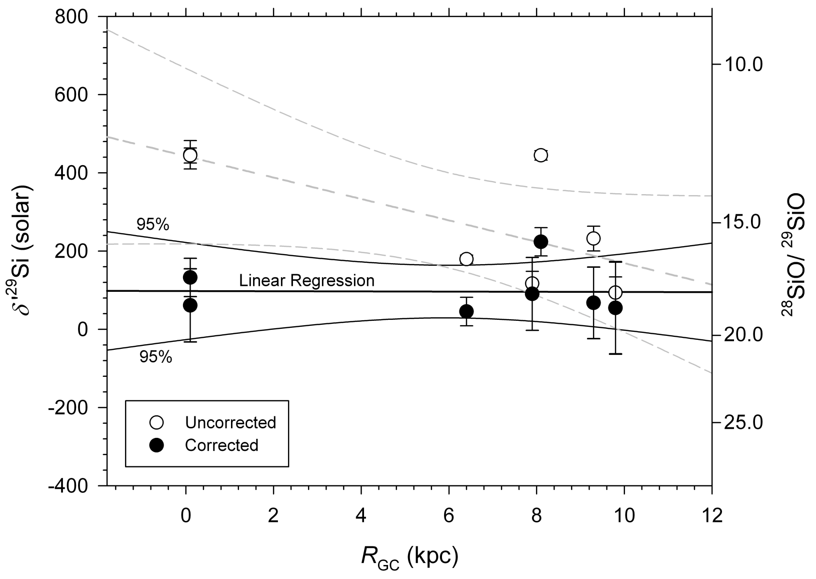


Figure 6.4: δ'^{29}Si in permil vs. Galactocentric distance from this study. Uncorrected data are shown as open symbols. Data corrected for optical depth are shown as black symbols. Error bars are 1σ. Linear regression for the uncorrected data (grey) and corrected data (black) are shown together with 95% confidence bands. The corresponding [^{28}SiO]/[^{29}SiO] ratios are shown on the right-hand ordinate.

encompasses regions of both optically thin and thick gas. As noted by Wouterloot et al. (2008), such effects do not facilitate comparison of sources at differing heliocentric distances and likely constitute a significant contribution to the overall error budget. Attempts to quantify any correlation between heliocentric distance and optical thickness are ongoing.

The mean corrected [^{28}SiO]/[^{29}SiO] ratio for the sources reported here is 17.9 ± 1.1 (1σ) and is 9% lower than the Solar value of 19.7 (i.e., the average measured values are enriched in ^{29}SiO by 97 per mil, relative to the Solar value). The mean of the SiO measurements is slightly further up the slope-1 line in Figure 6.3 than the mean of the presolar SiC grains, although the difference is within 2σ defined by the spread in SiC data (mean [^{28}SiO]/[^{29}SiO] $= 18.9 \pm 0.5$ (1σ) for SiC grains). The spread in Si isotope ratios from $R_{\rm GC} = 10$ kpc to the Galactic center is comparable to the spread in isotope ratios observed in presolar mainstream SiC grains (Figure 6.3) but considerably smaller than predictions based on the apparent variations in oxygen isotope ratios (Figure 3.1). It is not known if the displacement up the slope-1 line in Figure 6.3 is real, or represents some form of systematic error. One possible source of systematic error is discussed in Appendix A.2.1

CHAPTER 7

Discussion

7.1 Secondary/Primary Si Isotope Ratios

The somewhat higher [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] ratios of the present-day ISM relative to Solar values presumably represents GCE over the last 4.6 Gyrs. The finding that there is no resolvable variation in silicon isotope ratios across the Galaxy is important because it conflicts with expectations from oxygen and carbon secondary/primary isotope ratio trends. The implication is that in the present-day Milky Way, stars are forming with similar average silicon isotope ratios regardless of their distance from the Galactic center.

The explanation for the lack of a radial gradient in this isotope system remains elusive. One possibility is mixing by radial gas flows (Tinsley and Larson, 1978). Simulations suggest that spiral arm - bar resonances and infall of gas can result in flattening in metallicity gradients with $R_{\rm GC}$ in both stars and gas on timescales of < 1 Gyr (Cavichia et al., 2014; Minchev et al., 2011). If mixing is the cause of the flat gradient for silicon isotope ratios, it would imply that gradients in metallicity and gradients in other isotopic indicators of GCE have also been at least partially flattened by mixing.

Despite no evidence of a gradient, Galactic silicon isotopic abundances may not be *entirely* static, in so far as the present-day Galaxy appears to be isotopically heavy with respect to both the Sun and the mainstream presolar SiC grains (the latter representing the ISM $\geq$ 4.6 Gyr before present). However the evidence in support of this claim is scant at best; the silicon isotope ratios in the preponderance of presolar SiC grains lie *just* outside of error in some reported sources sources, while well within error of others (see Figure 6.3). Clearly, there is additional work to be done; the ^{30}SiO data suffer from poor signal to noise ratios in many sources, and the precision is insufficient to draw any robust conclusions on the difference between the isotopic composition of silicon in the present-day ISM, and the ISM as it existed $\geq$ 4.6 Gyr before present (as represented by SiC grains). Further, the survey lacks sources in the$R_{\rm GC}$ = 2 to 5 kpc region, a shortcoming that can be corrected with additional observations.

An alternative explanation is a temporal change in the sources of silicon isotopes that is peculiar to silicon. Zinner et al. (2006) reconstructed the GCE of Si isotopes using the measured isotope ratios in Z-type presolar SiC grains and models to filter out the nucleosynthetic effects of the AGB stellar progenitors of this rare class of SiC grains. They oncluded that there was a rapid rise in secondary/primary Si isotope ratios early in the Galaxy followed by a leveling off in the rate of change in these ratios when total metallicity (Z) began to exceed 0.01. These authors suggested that late additions of nearly pure ^{28}Si by Type Ia supernovae, as suggested by Gallino et al. (1994), may have contributed to the slowing in the rise of [^{29}SiO]/[^{28}SiO] and [^{30}SiO]/[^{28}SiO] with metallicity (and time). In this scenario, the addition of ^{28}SiO to the Galaxy was delayed because of the relatively long timescales required for the evolution of Type Ia supernova progenitors (e.g. Tsujimoto et al., 1995). Late addition of ^{28}SiO could have minimized the change in Galactic [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] over time, perhaps explaining the modest difference between the Solar and present-day ISM values.

Suppression of a Si isotope gradient with $R_{\rm GC}$ by a rise in the influence of Type Ia supernovae would require that the relative contribution of ^{28}Si from these products of white-dwarf-bearing binary systems is greater towards the Galactic center, counterbalancing the overall rise in metallicity and secondary isotope formation with decreasing $R_{\rm GC}$. Scannapieco and Bildsten (2005) developed a model for Type Ia formation rate in terms of star formation rate and total stellar mass, implying an overall increase in the rate of Type Ia formation towards the Galactic center. An accelerated decrease in [O/Fe] with increasing [Fe/H] toward the Galactic center is a signature of the influence of Type Ia supernovae owing to the large mass of Fe released in Type Ia events (e.g., Matteucci et al., 2006). It is conceivable that an analogous excess in Type Ia-produced ^{28}Si may exist towards the Galactic center.

7.2 Secondary/Secondary Si Isotope Ratios

The weight of the data for the seven sources is slightly displaced in triple isotope space from the presolar mainstream SiC data, with the former having higher [^{30}Si]/[^{28}Si] for the same [^{29}Si]/[^{28}Si] ratios (i.e., the SiO data lie to the right of the SiC data in Figure 6.3). This displacement, representing a higher [^{30}Si]/[^{29}Si] in SiO than both the Sun and the presolar mainstream SiC grains, could reflect a difference in the GCE of the two secondary silicon isotopes. Presolar SiC grains of types Y and Z have large excesses in [^{30}Si]/[^{29}Si] resulting from neutron capture in low-mass, low-metallicity AGB stars (Zinner et al., 2006). These grains represent a mechanism for altering the ratio of secondary silicon isotopes with time. However, the AGB source of Si is thought to be relatively minor (Clayton, 2003; Timmes and Clayton, 1996) and so the influence of AGB stars in shifting ISM [^{30}SiO]/[^{29}SiO] over time is expected to be limited.

An enhanced [^{30}Si]/[^{29}Si] in interstellar SiO could be indicative of a mass-dependent isotope partitioning (fractionation) because mass-dependent fractionation trends in Figure 6.3 have slopes of approximately 1/2 rather than unity, altering the secondary/secondary [^{30}SiO]/[^{29}SiO] ratios; the offset between the presolar SiC data and the ISM data could be explained if the the ISM SiO experienced mass-dependent heavy isotope enrichment.

SiO is commonly associated with both C-type and J-type shocks in the ISM, where it is produced through non-thermal sputtering processes with heavy neutral species (He, C, O & Fe), vaporization by grain-grain collisions and thermally-driven sublimation or evaporation of silicate grains (Nichols et al., 1995).

Si-bearing species enter the gas phase as either SiO or neutral Si, depending on the grain composition and the production mechanism (Caselli et al., 1997; Martín et al., 2009; Schilke et al., 1997; Ziurys et al., 1989). In the case of sputtering, yields are known to vary with impact energy and are mass dependent; consequently sputtering should result in mass-dependent isotope fractionation in which the heavy isotopes are enriched in the condensed phase residues. The magnitudes of the isotope fractionations associated with sputtering of silicates, which are believed to be the main reservoir of refractory silicon in the ISM, are not well constrained in the environments studied here (Schilke et al., 1997).

Although grain loss is believed to be non-thermal in the environments observed in this study (Caselli et al., 1997; Gusdorf et al., 2008; Schilke et al., 1997), there may be parallels in the isotope systematics of thermal evaporation/sublimation and sputtering given that the rate of the latter depends on a mass-dependent cohesive binding energy barrier. Thermal evaporation or sublimation of condensed silicates is known to cause Si isotope enrichment in the evaporative residues up to a few per cent where the distillation is extreme. These results are well documented from theory, experiments, and observations of meteoritical materials (Shahar and Young, 2007). The effects of partial evaporation of grains would leave the gas depleted in the heavy, secondary Si isotopes and the residual grains enriched in the heavy isotopes with the relative changes in [^{30}Si]/[^{28}Si] ratios being twice those for [^{29}Si]/[^{28}Si] as a consequence of the different vibrational frequencies of ruptured bonds (vibrational frequencies are proportional to the inverse square root of reduced mass). For example, evaporation of 90% of the Si from a typical silicate should yield an increase in [^{29}Si]/[^{28}Si] of $\approx 4\%$ in the residual condensed material and a corresponding increase in [^{30}SiO]/[^{28}SiO] of $\approx 8\%$ (Knight et al., 2009; Richter et al., 2007; Shahar and Young, 2007). This magnitude of fractionation would be sufficient to explain the offset between the SiC and ISM data. However, the sign is wrong for a simple single stage of grain evaporation. Rather than the SiO gas being depleted in the heavy isotopes, the data imply enrichment relative to the older SiC grains (Figure 6.3). If grain evaporation/sublimation is an explanation for the offset between SiC grains and SiO gas in Figure 6.3, it would require extreme distillation by Rayleigh-like processes or multiple discrete steps of partial Si loss so that the observed SiO derives from grains that had a prior history of evaporation and hence heavy isotope enrichment. Naturally, such fractionations are only possible in scenarios where grains are only partially ablated, and would not affect the SiO isotopologue abundance ratios when grains are subject to complete destruction.

Once liberated from grains, via sputtering or otherwise, neutral Si in the gas phase is oxidized to SiO by either molecular oxygen or the hydroxyl radical via (Caselli et al., 1997; Gusdorf et al., 2008; Schilke et al., 1997)

$$\mathrm{Si + O_2 \rightarrow SiO + O} \tag{7.1}$$

$$\mathrm{Si + OH^{\cdot} \rightarrow SiO + H^{\cdot}}. \tag{7.2}$$

The SiO/H_2 abundance ratio in shocked regions is enhanced by up to 10^5 relative to the ambient medium, but quickly declines in the cooling post-shock material. The rates of these gas-phase reactions depend on collisional frequencies, which are in turn proportional to $\mu^{-1/2}$ where μ is the reduced mass of the collisional system. This raises the possibility that the product SiO might be affected by mass-dependent fractionation relative to the parent grains. Here again, the sign of the expected shifts is the opposite of that required to explain the offset between SiC grains and SiO gas in Figure 6.3.

The archetypal destruction pathway of SiO to form SiO_2 is the reaction

$$\mathrm{SiO + OH^{\cdot} \rightarrow SiO_2 + H^{\cdot}} \tag{7.3}$$

occurring in the post-shock gas, where $OH^{\cdot}$ is abundant (Schilke et al., 1997). Similar to the sputtering process, oxidation in the cool post-shock gas has the potential to produce isotope fractionations in SiO. The higher zero-point energy of ^{28}SiO could potentially produce a non-equlibrium, Rayleigh-type fractionation as SiO is oxidized to SiO_2 and condenses into grains. However, even in molecular clouds, the collision frequency between SiO and $OH^{\cdot}$ will be low enough that this effect is likely to be of limited significance.

In all cases, the clustering of the data representing a wide variety of astrophysical environments from the Galactic center to the outer disk makes large differences in mass fractionation effects seem unlikely. The possibility for a decoupling of the growth of the two secondary Si isotopes remains. However, none of these factors could have modified the isotope ratios of SiO sufficiently to alter the conclusion that the variations in [^{29}SiO]/[^{28}SiO] ratios and [^{30}SiO]/[^{28}SiO] ratios across the Galaxy are surprisingly small.

CHAPTER 8

Future Work

I have three immediate observational goals with the GBT. Firstly, I seek to continue observations of the J = 1 $\rightarrow$ 0 rotational transition of the three silicon isotopologues of SiO in order to fill in the large gap of data between the Galactic center and $R_{\rm GC} \approx 5$ kpc. My goal is to establish the degree of silicon isotope variability across the Galaxy and compare this result with existing data for metallicity and for other isotopic systems, namely oxygen and carbon (e.g., Figures 2.3 and 2.2). To this end, additional observations with the GBT were proposed, and accepted with an "A" rating by the GBT time allocation committe, with a total of 18 hours of telescope time granted during the 2018a and 2018b semesters. However, due to the vicissitudes of the dynamic scheduling system at the GBT, these observations were not carried out.

Secondly, it is unknown if there are spatial variations in optical depth and isotopic ratios within each source. SiO emission often arises in hot, star-forming cores where shocks and high temperatures remove the usually refractory SiO molecule from grains. The typical angular size of these hot cores tends to be comparable to the beam size of the GBT, so the GBT observations represent an average over a potentially complex source region. Modeling indicates this complexity does not affect conclusions about isotope abundances in most cases, so long as the lines are optically thin. However, if there is significant variation within extended sources (i.e. those that are larger than the FWHM beam), then it can not be assumed that the isotopic ratios extracted from one location within the source is appropriately representative of the source as a whole. I therefore propose observing multiple locations within an extended source. L1157 seems a perfect candidate, as it is large, bright and is well studied, including observations of the J = 2 $\rightarrow$ 1 transition (Nisini et al., 2007). Fusion of the J $= 2 \rightarrow 1$ data with observations of the J $= 1 \rightarrow 0$ line would also provide good insight into the excitation temperature of the source.

Lastly, I plan to carry out measurements of some of the higher-order rotational transitions of SiO. The motivation to include additional, high-order rotational transitions is twofold; to enhance the completeness and Galactocentric coverage of the survey by including higher excitation sources inaccessible via the J $= 1 \rightarrow 0$ transition, and to constrain the excitation temperature in previously observed sources. The W-Band receiver on the GBT is capable of observing the J $= 2 \rightarrow 1$ line, and data could be reduced with the same pipeline as the J $= 1 \rightarrow 0$ data. Telescope time will also be sought at the Institut de Radioastronomie Millimétrique's 30m telescope on Pico Veleta to observe the J $= 3 \rightarrow 2$ line, as well as the Arizona Radio Observatory's Submillimeter Telescope on Mt. Graham for the J $= 5 \rightarrow 4$ and J $= 4 \rightarrow 3$ lines.

CHAPTER 9

Conclusions

The finding that secondary/primary Si isotope ratios have no detectable variation across the Galaxy within $\approx 20\%$ does not comport with expectations from the large variation in secondary/primary O isotope ratios of $\gtrsim 900\%$. Even when accounting for the prediction that the growth of secondary/primary ratios for Si isotopes should be approximately 1/3 that for O over the same range in metallicity, the observed variation is surprisingly small.

The modest increase in secondary/primary Si isotope ratios and the lack of a significant gradient with Galactocentric distance may be qualitatively consistent with previous suggestions that the increase in secondary/primary silicon isotope ratios has slowed with the increased influence of Type Ia supernovae. This result is in apparent conflict with the hypothesis that Solar Si is substantially and anomalously enriched in ^{28}Si relative to the ISM at the time of the birth of the Solar System (e.g., Alexander and Nittler, 1999; Young et al., 2011). In light of these conclusions, a careful reexamination of the Galactic distribution of oxygen isotopes seems well warranted.

Appendix A

A.1 Preface

The power of interstellar isotope abundance ratios as a diagnostic tool was realized nearly thirty years ago (Frerking et al., 1980; Linke et al., 1977; Penzias, 1981a,b; Wilson et al., 1981; Wolff, 1980). Used in conjunction with the theory of GCE, a knowledge of Galactic isotope abundance ratios would provide a powerful contextual reference for other astrophysical disciplines including nucleosynthesis, the composition of extrasolar planetary systems, as well as the origin and nature of the own Solar system.

Both precision and accuracy of data are of prime importance where isotopic ratios are concerned. Rare isotopes, as their name suggests, comprise only a small fraction of the total atomic abundance; often only single digit percentages. As such, signal to noise ratios for emission lines from rare isotopologues are typically poor and contribute significantly to the error budgets. Measurements of the abundance ratios of the three stable isotopes of silicon by Wolff (1980) and Penzias (1981a) are typical of such early attempts, in which the error budget dominated by noise in the data.

While this is still true, especially for rare isotopologues, modern cryogenic high-electron-mobility transistor (HEMT) amplifiers and superconducting tunnel junction (STJ) mixers provide such exceptional performance that it is possible to lower the noise floor for such observations by over two orders of magnitude (Monson et al., 2017). Measurements of ^{29}Si, [^{28}Si and ^{30}Si emission in the ISM reported by Monson et al. (2017) have RMS noise temperatures similar to those reported by Penzias (1981a) and Wolff (1980), but with frequency resolution $\approx 200\times$ higher. Although statistical error has been greatly reduced, little attention has been given to sources of systematic error. With noise statistics no longer controlling the error budget of isotopic studies and other high-precision spectroscopic measurements, recognizing the effects of these secondary sources of error is paramount.

Examples of such sources of error are examined in the appendices that follow. A number of common simplifying approximations are shown to be considerable sources of error, namely the Raleigh-Jeans approximation and those associated with the optically thin limit. The applicability of Local Thermodynamic Equilibrium (LTE) is also considered, and non-LTE excitation effects are also shown to be of consequence and are addressed in some detail.

Admittedly, stripping all assumptions from the commonly used equations of radiative transfer is, in many ways, a purely academic exercise; *some* assumptions need to be made. Variables like kinetic temperature, collision partner density etc. are often poorly constrained, if at all, and approximations are necessary to make the transfer equation solvable. However it seems prudent that even those approximations that are strictly required be incorporated with a grain of salt, as they will invariably come with *consequences*. The magnitude of the induced error is not especially large in all cases, however as receiver technology advances and researchers claim higher and higher precision in their measurements, a point has been reached when such nuances can no longer be ignored. Although developed as part of a survey to make high-precision, high-accuracy isotope ratio measurements of silicon in the interstellar medium, the techniques discussed herein are equally applicable to Galactic (and extragalactic) oxygen, carbon and nitrogen isotope ratio measurements.

A.2 Calculating Abundance Ratios

A.2.1 Accounting for Beam-Weighting

Following Baars (2007), The total power per unit frequency $\mathcal{P}_\nu$ collected by an antenna with an effective reception pattern $A(\theta, \phi)$ from a source at a large distance with the photon occupation number distribution $n_\gamma(\theta, \phi)$ is given as

$$\mathcal{P}_\nu = \frac{h\nu_{u\ell}^3}{c^2} \iint\limits_{4\pi} n_\gamma(\theta, \phi)\, A(\theta, \phi)\, d\theta d\phi. \tag{A.1}$$

Here θ and ϕ are the respective azimuthal and polar angular displacements from the main beam axis. The reciprocity theorem states that the reception and transmission parameters of an antenna must be linearly related. With a little algebra it can be shown that

$$G(\theta, \phi) = \frac{4\pi\nu^2}{c^2} A(\theta, \phi) \tag{A.2}$$

where $G(\theta, \phi)$ is the antenna gain (Baars, 2007, chap. 5). Like the effective reception pattern $A(\theta, \phi)$, the antenna gain $G(\theta, \phi)$ is also a function of the angular displacement from the main beam axis. Gain is proportional to the power emitted (or received) per steradian from direction (θ, ϕ) divided by the full-sphere average power per steradian, i.e.

$$G(\theta, \phi) = \eta_{\mathrm{r}} \frac{4\pi P_{\mathrm{n}}(\theta, \phi)}{\iint_{4\pi} P_{\mathrm{n}}(\theta, \phi)\, d\theta d\phi} \tag{A.3}$$

where η_{r} is the radiation efficiency of the antenna and $P_{\mathrm{n}}(\theta, \phi)$ is the antenna power pattern, normalized to $P_{\mathrm{n}}(0, 0) = 1$. The integral in the denominator of Equation (A.3) is equivalent to the total solid angle of the antenna Ω_{a}, i.e.

$$\Omega_{\mathrm{a}} = \iint\limits_{4\pi} P_{\mathrm{n}}(\theta, \phi)\, d\theta d\phi. \tag{A.4}$$

Combining Equations (A.2), (A.3) and (A.4) yields

$$A(\theta,\phi) = \left(\frac{\eta_{\mathrm{r}} c^2}{\nu^2 \Omega_{\mathrm{a}}}\right) P_{\mathrm{n}}(\theta,\phi), \tag{A.5}$$

an expression that can be substituted back into Equation (A.1) to give

$$\mathcal{P}_\nu = \frac{\eta_{\mathrm{n}} h \nu_{u\ell}}{\Omega_{\mathrm{a}}} \iint\limits_{4\pi} n_\gamma(\theta,\phi) P_{\mathrm{n}}(\theta,\phi)\, d\theta d\phi. \tag{A.6}$$

If it can be assumed that the emission source is isothermal and well resolved by the antenna main beam (defined as the half-power beam width (HPBW) of the central diffraction spike of the antenna) then the flux density S_ν of the source can be approximated as

$$\begin{aligned} S_\nu &= \frac{2h\nu_{u\ell}^3}{c^2} \iint\limits_{4\pi} n_\gamma(\theta,\phi)\, d\theta d\phi \\ &\approx \frac{2h\nu_{u\ell}^3}{c^2} n_\gamma \iint\limits_{\Omega_{\mathrm{s}}} \psi(\theta,\phi)\, d\theta d\phi \end{aligned} \tag{A.7}$$

where $\psi(\theta,\phi)$ is the normalized photon occupation distribution of the source, and Ω_{s} is the total solid angle subtended by the source (Baars, 2007). Knowing that that the source brightness distribution $\psi(\theta,\phi)$ is normalized, it follows that

$$\Omega_{\mathrm{s}} = \iint\limits_{4\pi} \psi(\theta,\phi)\, d\theta d\phi. \tag{A.8}$$

Substituting Equations (A.7) and (A.7) into Equation (A.6) in conjunction with making use of the Nyquist theorem and the definition of main-beam efficiency, $\eta_{\mathrm{mb}} = T_{\mathrm{a}}/T_{\mathrm{mb}}$, gives

$$T_{\mathrm{mb}} = \frac{\eta_{\mathrm{n}} S_\nu c^2}{2k_{\mathrm{b}} \Omega_{\mathrm{mb}} \nu_{u\ell}^2} \left(\frac{\Omega_\Sigma}{\Omega_{\mathrm{s}}}\right) = \frac{S_\nu c^2}{2k_{\mathrm{b}} \Omega_{\mathrm{mb}} \nu_{u\ell}^2} K \tag{A.9}$$

where Ω_{mb} is the solid angle subtended by the main beam of the antenna and Ω_Σ is the beam-weighted source solid angle

$$\Omega_\Sigma = \iint\limits_{\Omega_{\mathrm{s}}} \psi(\theta,\phi) P_{\mathrm{n}}(\theta,\phi)\, d\theta d\phi \tag{A.10}$$

and the factor $K \equiv \eta_{\mathrm{n}}\Omega_{\Sigma}/\Omega_{\mathrm{s}}$ corrects the main-beam temperature T_{mb} for the weighing of the source photon occupation distribution by the telescope beam.

This correction factor K is often assumed to be unity, as is the case from this point forward. This factor is normalized out when calculating isotopic ratios, so as long as it can be *assumed* to be the same for all three isotopologues, which is unlikely to be the case.

The effect of the beam-weighted solid source angle was not accounted for by Monson et al. (2017) when reducing SiO data; the correction factor K was assumed to be unity, as is typically done. It was argued that K would normalize out when calculating abundance ratios, so long as so as long as it can be *assumed* to be invariant between isotopologues. However this was a short-sighted assumption, and cannot be assumed to be true; Ω_{Σ} is a function of the antenna power pattern $P_{\mathrm{n}}(\theta, \phi)$ which is governed by diffraction and will vary with frequency. Consequently the correction factor K could be a contributing factor to the apparent excess of ^{29}Si and ^{30}Si in the ISM relative Sun and the preponderance of presolar SiC grains reported by Monson et al. (2017). It was suspected by the Author that the systematic enhancement of δ'^{29}Si and δ'^{30}Si in the reported sources was due to a calibration error; although no errors were found, omitting the correction factor for K seems likely to have contributed. Further investigation seems well warranted, and attempts to quantify the effects of beam-weighted source solid angle on abundance ratios are ongoing.

A.2.2 Correcting for Optical Depth

Setting the concept of beam-weighted source solid angle aside for the moment, a relatively straightforward correction factor for optical depth can be derived by assuming that the solid angle subtended by the source is much less than the antenna solid angle (i.e. $\Omega_s \ll \Omega_a$). In this limit K approaches unity, and if it is assumed the source is isothermal and radially uniform and there no other sources in the telescope beam, then the double integral in Equation (A.6) reduces to $\Omega_{\rm s} n_\gamma$, and in analogy to Equation (A.9) the main beam temperature of the telescope can be written

$$T_{\rm mb} = \frac{\mathcal{P}_\nu}{k\eta_{\rm mb}} = \frac{\Omega_{\rm s}}{\Omega_{\rm mb}} \frac{h\nu_{u\ell}}{k} n_\gamma. \tag{A.11}$$

The photon occupation number n_γ in Equation (A.11) can be expressed as a solution to the equation of radiative transfer. For a transition $v = 0, J = u \to \ell$ with an excitation temperature $T_{\rm ex}$ [1], the solution takes the form

$$n_\gamma = \left[n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf})\right] \left(1 - \exp(-\tau_\nu)\right) \tag{A.12}$$

where τ_ν is the optical depth of the line profile as a function of frequency. τ_ν is equal to the integral of the absorption coefficient k_ν along the optical path and is given by

$$\tau_\nu = \int_{s_0}^{s} k_\nu(s')\, {\rm d}s'. \tag{A.13}$$

For evaluating molecular column densities, it is convenient to define the absorption coefficient k_ν in terms of the associated Einstein coefficients. By the definition of these coefficients, agreement with the standard formulation of the equation of radiative transfer is necessitates

$$k_\nu = \frac{h\nu_{u\ell}}{4\pi}(n_\ell B_{\ell u} - n_u B_{u\ell})\phi(\nu) \tag{A.14}$$

[1] See Appendix A.3.1

where g_u and g_ℓ are the degeneracies for the upper and lower states, and n_u and n_ℓ are the fractional level populations for the upper and lower states.

When the source is isothermal along the optical path, the integral in Equation (A.13) becomes strictly proportional to the total column density of the excited state N_u, and the optical depth can be expressed as

$$\tau_\nu = \int_{s_0}^{s} k_\nu(s')\,\mathrm{d}s' = \frac{c^2}{8\pi\nu_{u\ell}^2} N_u A_{u\ell} \left[\frac{n_\ell g_u}{n_u g_\ell} - 1\right] \phi(\nu). \tag{A.15}$$

At this point, it is common to apply the Rayleigh-Jeans approximation. However, for a subthermal population of emitters, $h\nu_{u\ell}/kT_{\rm ex}$ might not be $\ll 1$ and thus the Rayleigh-Jeans approximation may not apply. Avoiding the Rayleigh-Jeans approximation, the main beam temperature $T_{\rm mb}$ can be written in a form that allows for subthermal excitation explicitly.

I start by substituting $n_\ell g_u/n_u g_\ell - 1 = 1/n_\gamma(T_{\rm ex})$ into Equation (A.15), yielding

$$\tau_\nu = \frac{c^2}{8\pi\nu_{u\ell}^2} N_u A_{u\ell} \left(\frac{1}{n_\gamma(T_{\rm ex})}\right) \phi(\nu). \tag{A.16}$$

By combining Equations (A.12) and (A.11), multiplying by τ_ν/τ_ν and substituting Equation (A.16), one obtains

$$T_{\rm mb} = \frac{\Omega_{\rm s}}{\Omega_{\rm mb}} \left(\frac{hc^2 N_u A_{u\ell}}{8\pi k \nu_{u\ell}}\right) \left[\frac{n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf})}{n_\gamma(T_{\rm ex})}\right] \left(\frac{1-\exp(-\tau_\nu)}{\tau_\nu}\right) \phi(\nu). \tag{A.17}$$

Converting this equation to a function of radial velocity $v_{\rm r}$ via the relation $dv_{\rm r}/c = d\nu/\nu_{u\ell}$ and solving for the total column density yields

$$N_u \phi(v_{\rm r}) = \frac{\Omega_{\rm mb}}{\Omega_{\rm s}} \left(\frac{8\pi k \nu_{u\ell}^2}{hc^3 A_{u\ell}}\right) \left[\frac{n_\gamma(T_{\rm ex})}{n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf})}\right] T_{\rm mb} \left(\frac{\tau_{v_{\rm r}}}{1-\exp(-\tau_{v_{\rm r}})}\right). \tag{A.18}$$

Knowing that $\int \phi(v_{\rm r})\, dv_{\rm r}' = 1$, an expression for the total column density in terms of the integrated main beam temperature and optical depth can be obtained by integrating Equation (A.18) over the line profile, yielding

$$N_u = \frac{\Omega_{\rm mb}}{\Omega_{\rm s}} \left(\frac{8\pi k \nu_{u\ell}^2}{hc^3 A_{u\ell}}\right) \left[\frac{n_\gamma(T_{\rm ex})}{n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf})}\right] \int_{\rm line} T_{\rm mb} \left(\frac{\tau_{v_{\rm r}}}{1-\exp(-\tau_{v_{\rm r}})}\right) dv_{\rm r}. \tag{A.19}$$

The photon occupation numbers $n_\gamma(T_{\rm crf})$ and $n_\gamma(T_{\rm ex})$ in Equation (A.19) are effectively invariant across the line profile, and thus left outside the integral. Similarly, the frequency variation across the line profile is sufficiently small that the frequency factor can be evaluated at line center, and left out of the integral.

The optical depth correction factor Λ can be extracted by evaluating Equation (A.19) at finite optical depth in comparison with the optically thin limit. In the optically thin limit, where $\tau_{v_{\rm r}}/(1-\exp(-\tau_{v_{\rm r}})) \to 1$ as $\tau_{v_{\rm r}} \to 0$, Equation (A.19) reduces to

$$N_u^{\tau_{v_{\rm r}}\to 0} = \frac{\Omega_{\rm mb}}{\Omega_{\rm s}} \left(\frac{8\pi k \nu_{u\ell}^2}{hc^3 A_{u\ell}}\right) \left[\frac{n_\gamma(T_{\rm ex})}{n_\gamma(T_{\rm ex}) - n_\gamma(T_{\rm crf})}\right] W, \tag{A.20}$$

where $W = \int T_{\rm mb}\, dv_{\rm r}$ is the integrated main beam temperature. In this case the column density is directly proportional to the integrated line intensity.

In the more realistic case of finite optical depth, where $\tau_{v_{\rm r}} > 0$, Equation (A.19) can be written

$$N_u = \frac{N_u^{\tau_{v_{\rm r}}\to 0}}{W} \int_{\rm line} T_{\rm mb} \left(\frac{\tau_{v_{\rm r}}}{1-\exp(-\tau_{v_{\rm r}})}\right) dv_{\rm r}. \tag{A.21}$$

Note that this is not equivalent to the optically thick limit, where $\tau_{v_{\rm r}}/(1-\exp(-\tau_{v_{\rm r}})) \to \tau_{v_{\rm r}}$ as $\tau_{v_{\rm r}} \to \infty$.

Considering that the thin limit $N_u^{\tau_{v_{\rm r}}\to 0}$ is the ideal case and that N_u is the more general case, their ratio defines the correction factor for optical depth Λ:

$$\frac{N_u}{N_u^{\tau_{v_{\rm r}}\to 0}} = \frac{1}{W} \int_{\rm line} T_{\rm mb} \left(\frac{\tau_{v_{\rm r}}}{1-\exp(-\tau_{v_{\rm r}})}\right) dv_{\rm r} \equiv \Lambda. \tag{A.22}$$

Note that the definition of Λ in Equation (A.22) is equivalent to the ratio $N_u/N_u^{\tau_{v_{\rm r}}\to 0}$ given by Mangum and Shirley (2015). This can be seen by recalling that $T_{\rm mb}$ is a function of $T_{\rm ex}$, as evidenced by Equations (A.11) and (A.12).

Mindful of the definition of W and that $T_{\mathrm{mb}} \propto 1 - \exp(-\tau_{v_{\mathrm{r}}})$, Equation (A.22) can be rewritten as

$$\frac{N_u}{N_u^{\tau_{v_{\mathrm{r}}}\to 0}} = \frac{\int_{\mathrm{line}} \tau_{v_{\mathrm{r}}} dv_{\mathrm{r}}}{\int_{\mathrm{line}} (1 - \exp(-\tau_{v_{\mathrm{r}}}))dv_{\mathrm{r}}} = \Lambda, \tag{A.23}$$

which is the same as Equation (5.7).

Substituting Equation (A.20) into Equation (A.23) reveals a general equation relating column density to integrated main beam temperature:

$$N_u = \Lambda \frac{\Omega_{\mathrm{mb}}}{\Omega_{\mathrm{s}}} \left(\frac{8\pi k \nu_{u\ell}^2}{hc^3 A_{u\ell}}\right) \left[\frac{n_\gamma(T_{\mathrm{ex}})}{n_\gamma(T_{\mathrm{ex}}) - n_\gamma(T_{\mathrm{crf}})}\right] W. \tag{A.24}$$

Recall from Equation (A.11) that $\Omega_{\mathrm{mb}}/\Omega_{\mathrm{s}}$ is the ratio of the solid angle subtended by the main beam of the antenna to that of the source. Unlike Ω_{s}, the size of the main beam of the antenna is frequency dependent and differences in beam size between isotopologues must be accounted for. Utilizing the antenna theorem, $A_{\mathrm{g}}\eta_{\mathrm{a}}\eta_{\mathrm{mb}} = \lambda^2/\Omega_{\mathrm{mb}}$, where η_{a} η_{mb} are the aperture and main beam efficiencies, respectively, the ratio of main beam solid angles $\Omega_{\mathrm{mb}}^s/\Omega_{\mathrm{mb}}^p$ for the same transition between two isotopologues, here denoted p and s (for primary and secondary nuclides, respectively) can now be expressed

$$\frac{\Omega_{\mathrm{mb}}^s}{\Omega_{\mathrm{mb}}^p} = \frac{(\nu_{u\ell}^p)^2 \eta_{\mathrm{a}}^p \eta_{\mathrm{mb}}^p}{(\nu_{u\ell}^s)^2 \eta_{\mathrm{a}}^s \eta_{\mathrm{mb}}^s}. \tag{A.25}$$

The ratio of aperture and main beam efficiencies are both very nearly unity and are safely ignored when the difference between the transition frequencies of isotopologues p and s is small and Equation (A.25) can be reduced to the close approximation

$$\frac{\Omega_{\mathrm{mb}}^s}{\Omega_{\mathrm{mb}}^p} \approx \left(\frac{\nu_{u\ell}^p}{\nu_{u\ell}^s}\right)^2. \tag{A.26}$$

By substituting this expression into Equation (A.24), the column density ratio of the excited states of isotopologues p and s can now be expressed as

$$\frac{N_u^s}{N_u^p} = \frac{\Lambda^s W^s A_{u\ell}^p}{\Lambda^p W^p A_{u\ell}^s} \left[\frac{1 - n_\gamma(T_{\mathrm{crf}})/n_\gamma^p(T_{\mathrm{ex}})}{1 - n_\gamma(T_{\mathrm{crf}})/n_\gamma^s(T_{\mathrm{ex}})}\right]. \tag{A.27}$$

Equation (A.27) can be reduced further by expanding the Einstein A coefficient as

$$A_{ul} = \frac{64\pi^4\nu_{ul}^3}{3hc^3 g_u} \left|\langle\psi_u|\mathbf{R}|\psi_l\rangle\right|^2 , \tag{A.28}$$

where $\left|\langle\psi_u|\mathbf{R}|\psi_l\rangle\right|^2$ is the transition dipole moment matrix element from state vector ψ_u to state vector ψ_l. With this final substitution, assuming the transition dipole moment matrix element is invariant between isotopologues, Equation (A.27) becomes

$$\frac{N_u^s}{N_u^p} = \frac{\Lambda^s W^s}{\Lambda^p W^p} \left(\frac{\nu_{u\ell}^p}{\nu_{u\ell}^s}\right)^3 \left[\frac{1 - n_\gamma(T_{\rm crf})/n_\gamma^p(T_{\rm ex})}{1 - n_\gamma(T_{\rm crf})/n_\gamma^s(T_{\rm ex})}\right] , \tag{A.29}$$

which is Equation (5.6) in the main text, and is the same as Equation (13) in Monson et al. (2017).

A.3 The Principle of Detailed Balance

In cases where LTE does not apply, the specific processes affecting energy level populations must be expressly considered. More succinctly,

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = 0 = \sum_{j\neq i} n_j \sum_k R^k_{ji} - n_i \sum_{j\neq i}\sum_k R^k_{ij} \tag{A.30}$$

where n_i is the number density of state i and R^k_{ij} is the transition probability for the $i \rightarrow j$ via process k (Wilson et al., 2009). Equation (A.30) is aptly named *the principle of detailed balance*. It is implicit that the summed number density of all states remains constant. Since no assumption about the population of state i on any other has been imposed, save that some form of transition between them is permitted, generality is maintained. As such, the principle of detailed balance applies universally.

A.3.1 The Excitation Temperature

Transition probabilities can be formulated for any number of processes affecting state populations. However, Equation (A.30) is a sum of two double series and rapidly becomes intractable as the number of processes considered increases. Fortunately, collisions between molecules and coupling with radiation are the only relevant processes in this context. The balance of a simple two-level system, the populations of which are governed only by collisions and coupling with radiation, can be expressed as a function of the Einstein A, B and C coefficients The upper and lower states are denoted u and ℓ, respectively.

$$\frac{\mathrm{d}n_u}{\mathrm{d}t} = n_\ell \left[B_{\ell u}\bar{U} + C_{\ell u}\right] - \; n_u \left[A_{u\ell} + B_{u\ell}\bar{U} + C_{u\ell}\right], \tag{A.31}$$

where the upper and lower states are denoted u and ℓ, respectively.

The collisional excitation coefficient $C_{\ell u}$, in s^{-1}, is the product of volumetric number density of the colliding species n_c and the collisional rate coefficient $q_{\ell u}$

$$C_{\ell u} = n_c q_{\ell u} = n_c \int_{v_0}^{\infty} f(v) v \sigma_{\ell u} \, dv. \tag{A.32}$$

Here $f(v)$ is the velocity distribution function of the colliding particles, $\sigma_{\ell u}$ is the cross section between the species of interest and the collision parter for the $\ell \rightarrow u$ transition, and v is defined by $(\mu v_0^2)/2 = \Delta E_{u\ell}$ where μ is the reduced mass of the collisional system. Similarly, the rate of de-excitation is given as

$$C_{u\ell} = n_c q_{u\ell} = n_c \int_{0}^{\infty} f(v) v \sigma_{u\ell} \, dv. \tag{A.33}$$

The colliding particles are most often assumed to be either electrons or diatomic hydrogen molecules. However, electrons are not of particular importance in molecular cloud cores, where ionization is low and collision rates with molecular hydrogen outpace collisions with other species by many orders of magnitude.

The net energy density $\bar{U}$ from Equation (A.31) is defined

$$\bar{U} = \frac{1}{c} \iint I_\nu \phi_\nu \, \mathrm{d}\nu \mathrm{d}\Omega = \frac{4\pi}{c} \int J_\nu \phi_\nu \, \mathrm{d}\nu \tag{A.34}$$

where $\phi(\nu)$ is the normalized line profile function. The net energy density $\bar{U}$ is equivalent to the total energy density U, albeit that net energy density is weighted by $\phi(\nu)$. This desensitizes Equation (A.31) to the intensity of the radiation field outside of the line profile; the importance of which is self evident. It is convenient to replace the net energy density $\bar{U}$ with the dimensionless photon occupation number

$$\bar{U} = \frac{c^2}{2h\nu_{u\ell}^3} \int J_\nu \phi_\nu \, d\nu = \frac{8\pi h \nu_{u\ell}^3}{c^3} \bar{n}_\gamma, \tag{A.35}$$

equal to the number of photons per degree of degeneracy per frequency mode within the line profile. Rearranging Equation (A.35), substituting into Equation (A.31) and recalling that $g_\ell B_{\ell u} = g_u B_{u\ell}$ and $A_{u\ell} = (8\pi h \nu_{u\ell}^3 / c^3) B_{u\ell}$, the rate equation reduces to

$$\frac{\mathrm{d}n_u}{\mathrm{d}t} = n_\ell \left[\frac{g_u}{g_\ell}\,\bar{n}_\gamma A_{u\ell} + C_{\ell u}\right] - \; n_u \left[A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}\right], \tag{A.36}$$

the steady-state solution to which is easily shown to be

$$n_u \left[A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}\right] = n_\ell \left[\frac{g_u}{g_\ell}\,\bar{n}_\gamma A_{u\ell} + C_{\ell u}\right].^{2} \tag{A.37}$$

It can be shown the collisional rate coefficients are related by a Boltzmann distribution at the kinetic temperature of the gas (Draine, 2011)

$$\frac{q_{\ell u}}{q_{u\ell}} = \frac{g_u}{g_\ell}\,\exp\left(\frac{-\Delta E_{u\ell}}{k_\mathrm{b} T_\mathrm{k}}\right). \tag{A.38}$$

Using this expression along with Equations (A.32) and (A.33), $C_{u\ell}$ can be substituted for $C_{\ell u}$ in Equation (A.37), yielding an expression exclusively in terms of the radiative and collisional de-excitation rates, i.e.

$$n_u \left[A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}\right] = \frac{n_\ell g_u}{g_\ell}\left[\bar{n}_\gamma A_{u\ell} + C_{u\ell}\,\exp\left(\frac{-\Delta E_{u\ell}}{k_\mathrm{b} T_\mathrm{k}}\right)\right]. \tag{A.39}$$

It is clear to see that a Equation (A.39) can be rearranged and set equal to Boltzmann's factor

$$\frac{A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}}{\bar{n}_\gamma A_{u\ell} + C_{u\ell}\,\exp(-\Delta E_{u\ell}/k_\mathrm{b} T_\mathrm{k})} = \frac{n_\ell g_u}{n_u g_\ell} = \exp\left(\frac{\Delta E_{u\ell}}{k_\mathrm{b} T_\mathrm{ex}}\right) \tag{A.40}$$

where the excitation temperature T_ex is the thermodynamic temperature corresponding to the level populations n_u and n_ℓ. Solving Equation (A.40) for T_ex, one obtains

$$T_\mathrm{ex} = \frac{\Delta E_{u\ell}}{k_\mathrm{b}}\left[\ln\left(\frac{A_{u\ell}(1+\bar{n}_\gamma) + C_{u\ell}}{\bar{n}_\gamma A_{u\ell} + C_{u\ell}\,\exp(-\Delta E_{u\ell}/k_\mathrm{b} T_\mathrm{k})}\right)\right]^{-1} \tag{A.41}$$

which is Equation (5.8) in Section 5.4.2 of the main text, and is equivalent to Equation (19) in Goldsmith (1972).

[2]It is worth noting that the derivations the equations $g_\ell B_{\ell u} = g_u B_{u\ell}$ and $A_{u\ell} = (8\pi h \nu_{u\ell}^3/c^3) B_{u\ell}$ *do* invoke thermodynamic equilibrium, however, the Einstein coefficients are determined solely by the quantum mechanical properties of the atom, hence the ratios hold in general

A.3.2 The Critical Density

The critical density n_{crit} is defined as the volumetric number density of the colliding species such that the radiative and collisional de-excitation rates are equal, i.e.

$$C_{u\ell} = A_{u\ell}(1 + \bar{n}_\gamma) \tag{A.42}$$

which is simply the radiative and collisional terms de-excitation terms from Equation (A.36) set equal. Recalling that the collisional de-excitation coefficient $C_{u\ell}$, in s^{-1}, is the product of volumetric number density of the colliding species ($n_{\rm crit}$ in this instance) and the collisional rate coefficient $q_{u\ell}$, Equation (A.42) can be rewritten

$$n_{\rm crit} q_{u\ell} = n_{\rm crit}\langle v\sigma_{u\ell}\rangle = A_{u\ell}(1 + \bar{n}_\gamma) \tag{A.43}$$

where $\sigma_{\ell u}$ is the cross section between the species of interest and the collision parter for the $\ell \to u$ transition, and v is defined by $\mu v_0^2 = 2\Delta E_{u\ell}$ where μ is the reduced mass of the collisional system. The angle brackets indicate the average over all applicable velocities of the integral term in Equation (A.33). From Equation (A.43), the stimulated-emission-corrected expression for critical density is

$$n_{\rm crit} = \frac{A_{u\ell}(1 + \bar{n}_\gamma)}{\langle v\sigma_{u\ell}\rangle}, \tag{A.44}$$

from which it is readily apparent that molecules with large Einstein A coefficients have correspondingly large critical densities, which is in keeping with expectations. Since Equation (A.44) includes the correction for stimulated emission, it is therefore sensitive to the intensity of ambient radiation within the line profile. Further, as evidenced by Equation (A.35), both Equations (A.44) and (A.40) are sensitive to the degree by which the photon occupation number n_γ varies over the line profile ϕ_ν.

The correction for stimulated emission is typically forgone in reported values of $n_{\rm crit}$, and critical densities are calculated as

$$n_{\text{crit}} = \frac{A_{u\ell}}{\langle v\sigma_{u\ell}\rangle}. \tag{A.45}$$

However for sub-thermal emission lines from dense gas tracers, like those seen in SiO, the effects of stimulated emission are not so easily ignored.

Molecules with large dipole moments couple strongly with radiation, increasing the rate of spontaneous emission and effectively pumping thermal energy out of the gas and into the radiation field, drawing the excitation temperature towards the radiation temperature. This effect dominates the excitation of lines at cm wavelengths. So long as n_γ at the transition frequency remains high (i.e $n_\gamma \geq 1$), rates of stimulated emission are appreciable even in the the $\lambda = 3$ mm window, where most molecules don't thermalize until their critical densities (as defined by Equation (A.45)) are exceed by ≈ 2 orders of magnitude. For molecules with already high critical densities, i.e. SiO, CN^-, HCN, HNC and CS, the effect is enhanced, becoming significant enough boost the effective thermalization density (the density at which LTE can be considered to apply) to values in the 10^7 cm^{-3} to 10^8 cm^{-3} range. While such H_2 densities are not unheard of they are are not typical of molecular clouds. Consequently, sub-thermal excitation is typical for most low J transitions of dense gas tracers in the ISM (Shirley, 2015).

CHAPTER 10

HYDRA User's Guide & Cookbook

Hydra version 5.1

Revision History

- Written by: N.N. Monson (UCLA) 14 August, 2013
- Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
- Version 2.1 written by: N.N. Monson (UCLA). 5 June, 2014
- Version 2.2 written by: N.N. Monson (UCLA). 12 October, 2014
- Version 2.3 written by: N.N. Monson (UCLA). 19 June, 2015
- Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
- Version 3.1 written by: N.N. Monson (UCLA). 9 December, 2016
- Version 4.0 (VEGAS spec) written by: N.N. Monson (UCLA). 30 April, 2017
- Version 4.1 (VEGAS spec) written by: N.N. Monson (UCLA). 17 September, 2017
- Version 4.2 (VEGAS spec) written by: N.N. Monson (UCLA). 30 May, 2018
- Version 5.0 (VEGAS spec) written by: N.N. Monson (UCLA). 23 December, 2018
- Version 5.1 (VEGAS spec) written by: N.N. Monson (UCLA). 11 February, 2019

10.1 User Agreement

The HYDRA program suite and runtime environment is the intellectual property of the original author(s), all rights reserved. The redistribution and use of HYDRA, in any form, with or without modification, is permitted provided that the following conditions are met:

- Any publication in a peer reviewed scientific journal that utilizes a distribution HYDRA to an appreciable degree must include a reference to this program and it's original author(s).
- Redistributions in any form must reproduce the entirety of this usage agreement, the complete revision/modification history (including all authorship information) of each component program, and all documentation and/or other materials provided with the original distribution.
- Neither the name of original author nor the names of any contributors may be used to endorse or promote products derived from HYDRA without first acquiring explicit written permission from the original author.

HYDRA is provided "as is" and any express or implied warranties are disclaimed. In no event shall the author be liable for any form of damages, however caused, that arise in any way out of the use of HYDRA.

CHAPTER 11

The HYDRA Program Suite

11.1 Preface

HYDRA is an IDL runtime environment designed and built for the express purpose of calibrating and reducing Q-band spectral line measurements of the three silicon isotopologues of SiO made with the Greenbank Telescope (GBT). While GBTIDL[1]contains some excellent tools for visualizing and manipulating a data collected with the GBT, the package emphasizes speed and ease of use over precision. HYDRA was written with the opposite intent, and is comprised of two main components: a vectorized high-precision data calibration routine and a Monte Carlo based data reduction pipeline. With this advanced functionality, HYDRA is capable of extracting high-precision and optical depth corrected primary to secondary silicon isotope ratios with fully quantified errors from raw, uncalibrated frequency-switched observations.

This document is intended to serve as a guide to the proper use of HYDRA and it's features. The individual programs that comprise HYDRA are explained in sufficient detail to enable their use. The proper order of execution, syntax and calling sequences are addressed.

As HYDRA was designed with a strong emphasis on precision, it deviates from the 'standard' approach to temperature calibration in several notable ways. Some key aspects of how HYDRA calibrates and reduces data are mentioned and explained explicitly; however the vast majority of the code is not covered in such detail. For additional details and documentation on the inner workings of HYDRA, please refer to the code itself, which is

[1]The National Radio Astronomy Observatory publishes GBTIDL, it's own IDL programs for manipulating and reducing data.

well annotated and most details of its operation are addressed in-situ. As such, anyone interested in implementing HYDRA or any of it's components are encouraged to throughly examine the source code before doing so.

In it's current condition HYDRA is hardcoded to reduce frequency switched SiO data only, however reconfiguring HYDRA to be compatible with different molecules or different observational setups would not be especially difficult. As previously mentioned, The HYDRA source code is well annotated, and includes additional notes in locations where hardcoded values would need to be changed in order to adapt HYDRA to handle data for different observational setups. HYDRA is intended to be freely modified, used and distributed, given the conditions of the User's Agreement are met.

CHAPTER 12

HYDRA Cookbook

12.1 Installing HYDRA

Much like GBTIDL package published by the National Radio Astronomy Observatory, HYDRA operates in a modified IDL runtime environment (RTE). This is required for a number of technical reasons and, beyond the fact that some setup is required, is of little relevance to the end-user. Once properly installed, the HYDRA RTE is automatically configutrf when the initialization procedure, Hydra.pro, is called from within IDL.

Installation is relatively straight-forward. After downloading the HYDRA tar ball, move it to the default IDL workspace directory and unpack it. This is easily done using the desktop environment (e.g. Finder, GNOME, KDE etc.) or via the following terminal commands.

- After downloading the HYDRA tar ball, make a directory named 'hydra' in the default IDL workspace, move the HYDRA tar ball there and unzip it. The directory *must* be named 'hydra' or the HYDRA RTE will fail to initialize.

  ```
  mkdir /users/bruce_lee/idlworkspace/hydra
  cd /users/bruce_lee/downloads
  mv hydra_5.1.0.tgz /users/bruce_lee/idlworkspace/hydra
  cd /users/bruce_lee/idlworkspace/hydra
  tar −zxvf hydra_5.1.0.tgz
  ```

  Note that the directories shown are for illustrative purposes only; they need to be replaced with the names of the appropriate directories on the local machine.

- Unpacking the HYDRA tar ball creates several new directories. Go to the directory "startup" and open Hydra.pro with emacs:

  ```
  cd startup/
  emacs hydra.pro
  ```

- There is a string in Hydra.pro that HYDRA uses to locate it's own installation directory. The variable name is "hydrapath" and is defined on line 47, just below the procedure definition. Set "hydrapath" equal to a string containing the full path to the local installation directory.

  ```
  hydrapath = '/users/bruce_lee/idlworkspace/hydra'
  ```

- Once this is complete, add Hydra.pro to the default IDL path so that it can be called from the IDL workbench. Amend the IDL_PATH preference to include a string containing the full path to the HYDRA startup directory using the PREF_SET command:

  ```
  IDL> Pref_set, 'idl_path', '<idl_default>: $
       +/users/bruce_lee/idlworkspace/hydra/startup', /commit
  ```

- After completing the above steps, test the validity of the installation by restarting IDL and calling Hydra.pro from the console.

  ```
  IDL> .Full_reset_session
  IDL> Hydra
  ```

If initialization is successful, a stylized HYDRA logo will print to the console and the IDL prompt will change from 'IDL>' to 'HYDRA>'.

12.2 Setting Up HYDRA

The HYDRA RTE is initialized by calling Hydra.pro from the IDL console. As noted above, a stylized HYDRA logo will print to the console and the IDL prompt will change from 'IDL>' to 'HYDRA>' if initialization is successful. Once Hydra initializes, there are a few setup steps to completed before any science data can be calibrated or reduced.

12.2.1 Setdir.pro

- **Setdir**, path

Reducing data with HYDRA is a multi-step process, with most steps creating a number of .txt.gz files to which data are written. This is done to preserve computing resources, but also to give the end user the ability to repeat steps with alternate keywords, or even to exit HYDRA without fear of loosing data stored in memory. HYDRA does the lion's share of managing the files it creates, but it does require the user to use Setdir.pro to define the path to a "home" directory (called the working directory) it can write to. Be sure that HYDRA has both read and write privileges to the chosen directory.

HYDRA> **Setdir**, '/users/bruce_lee/myresults/obs_01'

HYDRA records details on it's in the file headers, as well as in in separate reference files that it writes after critical steps. These features are designed to allow any individual data reduction to be fully reproducible, and is invaluable as an investigative tool when examining old or aberrant results. For this reason it is highly recommended that the user give each source object *and* reduction attempt it's own unique directory. If this is not done, any preexisting data or reference files in the specified directory will be overwritten and lost. This may seem trivial, but it is cheap insurance, and highly recommended.

Arguments And Keywords

- PATH: Set the path argument to a string containing the full path to the working directory. Be sure that HYDRA has both read and write privileges to the chosen directory. It is highly recommended that the user give each source object *and* reduction attempt it's own unique directory.

12.2.2 Getfits.pro

- **Getfits**, path [, /BYPASS, /SILENT]

The next step to any reduction is to load the appropriate .sdfits file to memory. HYDRA uses the procedure Getfits.pro to accomplish this. Only one .fits file can be loaded at any given time. If the data to be reduced is spread across multiple .fits files, this can be accommodated but the files must still be loaded individually. Details on how to reduce data spread across multiple observations or .fits files is addressed in the next section, as part of the documentation for Gather.pro. Once an .fits file has been copied to memory, Getfits.pro will print a summary of the copied scan data to the console. This summary data can be displayed again at any time by calling Summary.pro.

Arguments And Keywords

- PATH: Set the path argument to a string containing the full path to the .fits file containing the science data to be reduced. If there are multiple .fits files for each observation (as is the case with the VEGAS spectrometer backend at the GBT; The number of .fits files generated depends on settings but is typically one file per IF) all the files must be in the same directory. HYDRA will automatically switch between .fits file when necessary.

- BYPASS: This keyword is optional. The BYPASS keyword is used to suppress a warning that prints to the console if Getfits.pro is called before Setdir.pro. In normal usage, Setdir.pro *should* be the first HYDRA routine called after startup. This is not critical, but is recommended, as a number of variables are initialized by Setdir.pro that HYDRA uses for housekeeping. The default value is BYPASS = 0 (off).

- **SILENT:** This keyword is optional. The SILENT keyword is used to suppress the output from summary.pro, which is called when Getfits.pro loads a new .fits file to memory. This keyword is used when other HYDRA programs call Getfits.pro, and should not be used when calling Getfits.pro from the console. The default value is SILENT = 0 (off).

12.3 Managing Data With HYDRA

As previously stated, HYDRA does not read data from .fits files directly, but instead reads and writes .txt.gz files from the working directory specified using Setdir.pro. For each .fits file containing science data to be reduced, the scans of the calibrator source and science target contained therein are extracted and written to the working directory using Gather.pro and it's subroutine, Gather.pro.

12.3.1 Gather.pro

- **Gather**, session, ONSCANS=array, ONINTS=list, OFFSCANS=array, OFFINTS=list, SRCSCANS=array, ISO=value{28,29,30} [, /RRL, /JUSTCAL]

Gather.pro locates the required data in the .fits file in memory, and (if required) resamples and averages the raw data before writing it to .txt.gz files in the working directory. In essence, Gather.pro is a fancy wrapper for Extract.pro, which does all the real work. It should never be necessary for the user to call Extract.pro directly, and attempting to do so will almost certainly fail. The calling sequence for Gather.pro has some special requirements; namely that the keywords ONINTS and OFFINTS must each be a LIST.

Arguments And Keywords

- SESSION: The SESSION argument is used to organize data when a source is observed multiple times. The actual value of SESSION is irrelevant, so long as it is a longword and is unique to each observation. For example, consider a source that was observed on two separate occasions; the data for each occasion can be written to the same working directory by calling Gather.pro twice:

  HYDRA −> **Getfits**, '/users/bruce_lee/mydata/orion_kl/obs_0'
  HYDRA −> **Gather**, 1, oncalscans=[10, 12], oncalints=**LIST**([3], [0,1])...
  HYDRA −> **Getfits**, '/users/bruce_lee/mydata/orion_kl/obs_02'
  HYDRA −> **Gather**, 2, oncalscans=[2, 3], oncalints=**LIST**([0,1], [0,1])...

where Getfits.pro is called between calls of gather.pro in order to load the .fits file for the second observation to memory. The data for these two observations would be saved to separate directories labeled “Session_01” and “Session_02” respectively.

- ISO: The ISO keyword is used to specify the atomic mass number of the silicon isotope for which data will be extracted from the .fits file and written to the working directory. Data from multiple values of ISO can not be extracted with a single call of Gather.pro; data must be organized and written to the working directory one value of ISO at a time. The ISO keyword has no default (therefore it must always be specified) and must be a longword scalar. Any other data type will be converted to a longword (if possible). This keyword is overridden by the RRL keyword.

- ONSCANS: The ONSCANS keyword is used to select the scan numbers that will be saved to the working directory. The ONSCANS keyword has no default (therefore it must always be specified) and must be a longword ARRAY. Any other data type will be converted to a longword ARRAY (if possible).

- ONINTS: The ONINTS keyword must be a LIST with one entry per scan specified with the ONSCANS keyword. A LIST is similar to an ARRAY, but unlike an ARRAY a LIST can contain elements of any dimension. The ONINTS and OFFINTS keywords must each be a LIST in order to allow the user to pick individual integrations out of calibration scans. For example, consider two on-source calibration scans (numbers 10 and 12) each with 4 integrations. An ARRAY would not allow one integration to be used out of scan 10, and two out of scan 12; entries in an ARRAY must have the same dimension. A LIST does not have that requirement, so the integrations for scans 10 and 12 can be specified as

  HYDRA −> **Gather**, 1, oncalscans=[10, 12], oncalints=**LIST**([3], [0,1])...

  Most users do not need to use this functionality; even so, ONINTS and OFFINTS must always be a LIST with one entry per scan. The default value for ONINTS is to use the first (and only the first) integration of each scan specified with ONSCANS.

- OFFSCANS: The syntax and function of the OFFSCANS keyword is identical to the ONSCANS keyword, except it is used to specify the scan numbers of the off-source calibration data. The OFFSCANS keyword has no default (therefore it must always be specified) and must be a longword ARRAY. Any other data type will be converted to a longword ARRAY (if possible).

- OFFINTS: The syntax and function of the OFFINTS keyword is identical to the ONINTS keyword, except it is used to specify the integration numbers of the off-source calibration data. If set, the OFFINTS keyword must be a LIST. The default value for OFFINTS is to use the first (and only the first) integration of each scan specified with OFFSCANS.

- SRCSCANS: Similar to the ONSCANS and OFFSCANS keywords, the SRCSCANS keyword is used to specify the science data scans to be excised from the .fits file and saved to the working directory. The SRCSCANS keyword has no default (therefore it must always be specified) and must be a longword ARRAY. Any other data type will be converted to a longword ARRAY (if possible).

- RRL: This keyword is optional. Set the RRL keyword to override the ISO keyword, and direct gather.pro to extract data for the fourth IF. The RRL keyword should only be used if there is concern that the ^{29}SiO emission is contaminated with emission from the H(83)δ radio recombination line. The default value is RRL = 0 (off).

- JUSTCAL: This keyword is optional. Set the JUSTCAL keyword to override the SRCSCANS keyword and cause gather.pro to exit after it has finished writing the specified on-source and off-source calibration data to the working directory. This keyword is useful when the data for a given observation has already been extracted and written to the working directory, but the user wishes to modify or change the calibration data. The default value is JUSTCAL = 0 (off).

12.4 Calibrating Data With HYDRA

As noted in Monson et al. (2017), the primary concern with the standard approach to calculating T_{sys} and T_{cal} is that any information about frequency dependent gain *within* the bandpass is destroyed. Although atmospheric opacity and aperture efficiency are largely invariant across MHz scale spectral windows, noise diode power output and LO/IF system response are not, and this frequency dependence needs to be accounted for when making high precision spectral measurements. To this end, the standard calibration protocol is adapted to account for channel by channel variations in the system response by substituting array valued, or 'vectorized', incarnations of T_{cal} and T_{sys} for their standard scalar valued counterparts.

In order to retain information about frequency-dependent gain within the bandpass, HYDRA implements a fully vectorized calibration routine that calculates gain profiles pixel-by-pixel across the entire bandpass, thereby accommodating any frequency dependence that is present. Further, gain profiles for each IF, polarization, noise diode state and frequency position are calculated independently to ensure uniform calibration.

12.4.1 Dreamcatcher.pro

- **Dreamcatcher**, sesnums, ISO=value{28,29,30} [, FITORDER=integer{1 to 7}, CALFITORDER=integer{1 to 7}, /RRL, /PLOTCAL, /PRINTCOLOR]

Data calibration in HYDRA is handled by a single, comprehensive program that calibrates the noise diodes, calculates the system and main beam temperatures, combines polarizations and frequency positions, and finally averages each scan together to produce a single spectrum for each source. This program, Dreamcatcher.pro, is likely the most important (and largest) HYDRA program. Although the inner-workings of most HYDRA programs are not discussed here[1], the operation of Dreamcatcher.pro is covered in Chapter 13.

[1]For details about the operation of a particular program, users are encouraged to consult that program's source code (which is included with this manual, see Chapter 14). The source code is well annotated, and easily navigable by an experienced IDL user.

Arguments And Keywords

- SESNUMS: The SESNUMS argument specifies the observations that will be calibrated and averaged. Recall that when calling Gather.pro, each individual observation is given a unique integer value via the SESSION argument; these values are passed to Dreamcatcher.pro using the SESNUMS argument. For example, if a source was observed twice and the data from the two observations are given SESSION values of 1 and 2 respectively, the appropriate SESNUMS argument when calling Dreamcatcher.pro is

  HYDRA> **Dreamcatcher**, [1,2], iso=28, fitorder...

  The SESNUMS argument must be a longword ARRAY, even if it is only a single value.

- ISO: The ISO keyword is used to specify the atomic mass number of the silicon isotope for which be read from the working directory and calibrated. Data from multiple values of ISO can not be calibrated with a single call of Dreamcatcher.pro; data must be read from the working directory and calibrated one value of ISO at a time. The ISO keyword has no default (therefore it must always be specified) and must be a longword and a scalar. Any other data type will be converted to a longword (if possible). This keyword is overridden by the RRL keyword.

- RRL: This keyword is optional. This keyword overrides the ISO keyword. Setting the RRL keyword directs dreamcatcher.pro to calibrate data for the fourth IF. This keyword Ssould only be used when there is concern that the ^{29}SiO emission is contaminated with emission from the H(83)δ radio recombination line. The default value is RRL = 0 (off).

- FITORDER: This keyword is optional. The FITORDER keyword is used to set the order of the polynomial fit to the system temperature $T_{\rm sys}(\nu)$ for each integration of each science data scan being calibrated. This is done to avoid introducing unnecessary noise when calculating the antenna temperatures $T_{\rm a}(\nu)$ (see Equation (13.7)). If set, the FITORDER keyword must be a longword and a scalar. The acceptable range of values is $1 \leq$ FITORDER ≤ 7. The default value is FITORDER = 4.

- CALFITORDER: This keyword is optional. The CALFITORDER keyword is used to set the order of the polynomial fit to the diode calibration temperature $T_{\rm cal}(\nu)$ for each integration of each calibration scan being used. This is done to avoid introducing unnecessary noise when calculating the system temperatures $T_{\rm sys}(\nu)$ (see Equation (13.6)). If set, the CALFITORDER keyword must be a longword and a scalar. The acceptable range of values is $1 \leq$ CALFITORDER ≤ 7. The default value is CALFITORDER = 3.

- PLOTCAL: This keyword is optional. The PLOTCAL keyword prompts dreamcatcher.pro to plot and save (as .png files) additional information about the calibration process; namely calculating the diode calibration temperatures. Use of the PLOTCAL keyword is *highly* recommended. Using the PLOTCAL keyword does impose a small time penalty (it takes a couple seconds for each .png image to save). The default value is PLOTCAL = 0 (off).

- PRINTCOLOR: This keyword is optional. The PRINTCOLOR keyword can be used to change the background color of the plots produced by dreamcatcher.pro from black to white. Black is the preferred color, as it is easier to distinguish between colors on busy or crowded plots. The default value is PRINTCOLOR = 0 (off).

12.5 Reducing Data With HYDRA

Utilizing a vector valued calibration routine helps to keep baseline structure to a minimum but does not eliminate it completely. Typical low frequency ($\nu \approx$ IF bandpass) baselines are handled with relative ease, however differentiating between baseline structure and emission line structure can be tedious when lines are weak or system temperatures are high. In most cases, a simple low-order polynomial fit is adequate to fit baselines, provided that any lines in the spectrum are given an adequate birth in order to avoid fitting and subtracting line wings. Extraction of accurate isotopic ratios depends critically on proper baseline removal, and great care should be taken in this respect.

12.5.1 Outerlimits.pro

- **Outerlimits**, ISO=value{28,29,30} [, /RRL, /LOWSNR, /CRAPSHOOT, /ZOOM]

HYDRA has a special GUI as part of Outerlimits.pro that enables emission-free regions of each spectrum to be selected using the mouse and keyboard. Once the GUI has launched, regions can be selected by holding the “CTRL” key and clicking the left mouse button. After selecting the desired regions, exit the GUI by holding the “ALT/CMD” key and again clicking the left mouse button. The boundaries of a fitting region appear as dashed vertical lines once the region has been selected. HYDRA stores the values from the last time Outerlimits.pro was called (if applicable) and plots them as dotted vertical lines to aid in making fine adjustments to the fitting regions. Outerlimits.pro will only accept an even number of boundaries for fitting, and if an odd number have been set when Outerlimits.pro exits, the las boundary will be discarded. NOTE: If possible, position the GUI window such that the IDL console is visible; Outerlimits.pro prints warnings, reminders etc. to the console when the GUI is in use.

Arguments And Keywords

- ISO: The ISO keyword should be set equal to the atomic mass number of the silicon isotope for which data will be read from the working directory and plotted. The ISO keyword has no default (therefore it must always be specified) and must be a longword scalar. Any other data type will be converted to a longword (if possible). This keyword is overridden by the RRL keyword.

- RRL: This keyword is optional. This keyword overrides the ISO keyword. Setting the RRL keyword directs outerlimits.pro to plot data for the fourth IF. Should only be used when there is concern that the ^{29}SiO emission is contaminated with emission from the H(83)δ radio recombination line. The default value is RRL $= 0$ (off).

- LOWSNR: This keyword is optional. Set the LOWSNR keyword to increase the degree to which the spectrum is smoothed before it is plotted. The LOWSNR keyword is useful when fitting noisy spectra from rare isotopologues or weak sources.

- CRAPSHOOT: This keyword is optional. Set to the CRAPSHOOT keyword to drastically increase the degree to which the spectrum is smoothed before it is plotted. Overrides LOWSNR keyword. NOTE: This much smoothing should be avoided whenever possible.

- ZOOM: This keyword is optional. Set the ZOOM keyword to restrict the range of the ordinate when plotting. NOTE: This keyword is buggy; use with caution.

12.5.2 Polybase.pro

- **Polybase**, ISO=value [, /RRL, /LOWSNR, /CRAPSHOOT, /ZOOM]

Once the fitting regions are set with Outerlimits.pro, HYDRA uses Polybase.pro fit and subtracts a polynomial baseline from each spectrum. The user is prompted to define the order of the polynomial fit on the console, after which the fit is calculated and displayed using a GUI similar to that used by Outerlimits.pro. Each polynomial fit can be rejected if it is deemed unsuitable, and the spectrum refitted until an appropriate fit is found. Again, this functionality is controlled with console prompts, so the GUI window must be positioned such that the IDL console is visible. The fitting regions selected for each spectrum using Outerlimits.pro are displayed for convenience. The current polynomial fit is plotted as a solid white line once it has been calculated. Additionally, HYDRA stores the last fit that was calculated (if applicable) and plots it as a dotted white line to aid in making fine adjustments

In most cases, a simple low-order polynomial fit can be used to subtract baselines, provided that any lines in the spectrum are given an adequate birth in order to avoid fitting and subtracting line wings. The fitting process can be severely complicated if portions of emission line wings are included in the fitting regions, especially if lines are weak or system temperatures are high. Care should be taken to avoid this.

Arguments And Keywords

- ISO: The ISO keyword should be set equal to the atomic mass number of the silicon isotope for which data will be read from the working directory and plotted. The ISO keyword has no default (therefore it must always be specified) and must be a longword scalar. Any other data type will be converted to a longword (if possible). This keyword is overridden by the RRL keyword.

- RRL: This keyword is optional. This keyword overrides the ISO keyword. Setting the RRL keyword directs polybase.pro to fit a polynomial to the data for the fourth IF. Should only be used when there is concern that the ^{29}SiO emission is contaminated with emission from the H(83)δ radio recombination line. The default value is RRL $= 0$ (off).

- LOWSNR: This keyword is optional. Set the LOWSNR keyword to increase the degree to which the spectrum is smoothed before it is plotted. The LOWSNR keyword is useful when fitting noisy spectra from rare isotopologues or weak sources.

- CRAPSHOOT: This keyword is optional. Set to the CRAPSHOOT keyword to drastically increase the degree to which the spectrum is smoothed before it is plotted. Overrides LOWSNR keyword. NOTE: This much smoothing should be avoided whenever possible.

- ZOOM: This keyword is optional. Set the ZOOM keyword to restrict the range of the ordinate when plotting. NOTE: This keyword is buggy; use with caution.

12.5.3 Dreamweaver.pro

- **Dreamweaver**, NLOOPS=integer, NSUBLOOPS=integer [, NSIGMA=integer, /KILL_RRL, /LOWSNR, /CRAPSHOOT]%, /PLOTSTUFF]

After the data for all three isotopologues are calibrated, they can be reduced to produce [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si] ratios. HYDRA uses Dreamweaver.pro to both reduce data, as well as to estimate and correct for the effect of optical depth in the ^{28}SiO emission line (see Sections 5.4, 5.4.1 and Appendix A.2.2). In order to account for both measurement uncertainties and the uncertainties imparted by fitting baselines by hand, the entire data reduction pipeline and optical depth correction scheme is built around a pair of nested Monte Carlo error analyses. The "outer" Monte Carlo simulation makes random, normally-distributed draws to the boundaries of each fitting region (set via Outerlimits.pro). The boundaries of each individual region are resampled with $\sigma = 5\%$ of the width of that region. Dreamweaver.pro then fits and subtracts a polynomial baseline from each spectrum using the new fitting regions. The data are then passed to the "inner" Monte Carlo simulation, which calculates the root mean square (RMS) noise temperature $T_{\rm rms}$ for each spectrum, then makes normally distributed random draws of each channel in each spectrum with $\sigma = T_{\rm rms}$ to assess the errors imparted by noise. The use of unsmoothed spectra when resampling the noise results in an $\approx 5\%$ overestimation of the uncertainty in each derived isotopologue ratio. After resampling the noise, Dreamweaver.pro calculates the [^{29}Si]/[^{28}Si] and [^{30}Si]/[^{28}Si], then estimates and corrects for optical depth before proceeding with the next loop. The total number of simulations is equal to the product of the NLOOPS and NSUBLOOPS keywords.

Despite being computationally intensive, Dreamweaver.pro is a relatively fast program with a full reduction taking no more than a few minutes (each loop takes an average of 2.5 milliseconds on the Author's computer). Memory usage is also fairly low, so running Dreamweaver.pro should not pose a problem for most users, even those running HYDRA on underpowered hardware.

Arguments And Keywords

- NLOOPS: This keyword is optional. The NLOOPS keyword is used to specify the number of times the "outer" Monte Carlo simulation will be looped, and thus the number of times the baseline fitting regions will be altered, and a new baseline will be fit and subtracted from each spectrum (this action is performed simultaneously for all three values of ISO). If set, the NLOOPS keyword must be a longword scalar. Any other data type will be converted to a longword (if possible). The default value is NLOOPS = 200.

- NSUBLOOPS: This keyword is optional. The NSUBLOOPS keyword is used to specify the number of times the "inner" Monte Carlo simulation will be looped, and thus the number of times the noise will be resampled and the isotopologue ratios will be calculated before a new baseline will be fit and subtracted from each spectrum (this action is performed simultaneously for all three values of ISO). If set, the NSUBLOOPS keyword must be a longword scalar. Any other data type will be converted to a longword (if possible). The default value is NSUBLOOPS = 200.

- KILL_RRL: This keyword is optional. Setting the KILL_RRL keyword directs Dreamweaver.pro to use reduced data from the fourth IF to fit and remove the H(83)δ emission feature from the ^{29}SiO spectrum. NOTE: This functionality is not currently supported. Better results will be obtained from fitting and removing the interfering H(83)δ emission manually prior to calling Dreamweaver.pro.

- NSIGMA: This keyword is optional. The NSIGMA keyword is used to specify the number of σ widths through which the spectra are integrated when calculating the isotopologue ratios. Use of this keyword is not advised, the default value should be sufficient. If set, the NSIGMA keyword must be a longword scalar. Any other data type will be converted to a longword (if possible). The default value is NSIGMA = 3.

- LOWSNR: This keyword is optional. The LOWSNR keyword is used to increase the degree to which each spectrum is smoothed. Dreamweaver.pro() uses the smoothed spectra for a number of operations related to estimating optical depth, but does not use smoothed spectra to calculate isotopologue ratios. The LOWSNR keyword can help when reducing noisy spectra from weak sources. The default value is LOWSNR = 0 (off).

- CRAPSHOOT: This keyword is optional. Set to the CRAPSHOOT keyword to drastically increase the degree to which the spectrum is smoothed. Dreamweaver.pro() uses the smoothed spectra for a number of operations related to estimating optical depth, but does not use smoothed spectra to calculate isotopologue ratios. The CRAPSHOOT keyword can help when reducing nightmarish spectra from weak sources or sources with interfering lines. Overrides the LOWSNR keyword. NOTE: This much smoothing should be avoided whenever possible. The default value is CRAPSHOOT = 0 (off).

CHAPTER 13

How HYDRA Calibrates Data

In order to retain information about frequency-dependent gain within the bandpass, Dreamcatcher.pro implements a fully vectorized calibration routine that calculates gain profiles pixel-by-pixel across the entire bandpass, thereby accommodating any frequency dependence that may be present. Further, gain profiles for each IF, polarization, noise diode state and frequency position are calculated independently to ensure uniform calibration.

13.1 Calibration Sources

Absolute spectral flux densities for commonly used calibration sources are determined by flux density ratio measurements (typically using WMAP observations of Mars or Venus as the standard) and reported as coefficients to a polynomial expression. Dreamcatcher.pro calculates the spectral flux density $\vec{S}_\nu$ of the calibrator over observed frequency band $\vec{\nu}$ using the polynomial coefficients reported by Perley and Butler (2013, 2017)

$$\log(\vec{S}_\nu) = a_0 + a_1 \log(\vec{\nu}) + a_2[\log(\vec{\nu})]^2 + a_3[\log(\vec{\nu})]^3 ... \tag{13.1}$$

up to a total of six coefficients for some sources. In it's current state, Dreamcatcher.pro is hardcoded to accept a limited number of calibration sources (3C48, 3C147, 3C196, 3C286, 3C295 and 3C380) although this list can be easily expanded.

The spectral flux density of the calibration source is then converted to corrected antenna temperature, $T_\text{a}^*(\nu)$, with the expression

$$T_{\mathrm{a}}^{*}(\nu) = 2.85\ \vec{S}_{\nu}\ \vec{\eta}_{\mathrm{a}} \exp\left(\frac{-\vec{\tau}_{\mathrm{z}}}{\sin(\theta)}\right), \tag{13.2}$$

where $\vec{\tau}_{\mathrm{z}}$ is the zenith atmospheric opacity and $\vec{\eta}_{\mathrm{a}}$ is aperture efficiency, both evaluated over the frequency band $\vec{\nu}$. The elevation of the source, θ, is in radians. The exponential term in this equation is a geometric factor that accounts for the increased airmass when observing at low elevations, and The GBT-specific gain constant of $2.85 = (A_{\mathrm{g}}/2k_{\mathrm{b}})$ accounts for the Boltzmann constant k_{b} and geometric area A_{g} of the telescope. While both aperture efficiency and zenith opacity are seen here implemented as vectors, any variation of these values across a MHz scale spectral window is likely exceedingly small.

The aperture efficiency $\vec{\eta}_{\mathrm{a}}$ is stimated using the Ruze equation, seen here with the GBT-specific peak aperture efficiency of 0.71

$$\vec{\eta}_{\mathrm{a}} = 0.71 \exp\left(\frac{-4\pi\varepsilon\vec{\nu}}{c}\right), \tag{13.3}$$

where ν is the frequency in GHz and the RMS surface accuracy ε is ≈ 390 microns for the GBT. Dreamcatcher.pro uses a simple equation published by the NRAO to estimate zenith opacity $\vec{\tau}_{\mathrm{z}}$ as a function of frequency:

$$\vec{\tau}_{\mathrm{z}} = 0.008 + \frac{\exp(\sqrt{\vec{\nu}}\,)}{8000}. \tag{13.4}$$

It should be noted that the NRAO advises against using this equation for high precision calibration, suggesting instead that zenith opacity be measured directly using tipping operations. However, any variations in either zenith opacity or aperture efficiency are assumed to be the same for all three isotopologues, and are normalized out when calculating isotopic ratios, therefore the accuracy of the equation is not of particular importance in this application.

13.2 The Noise Diodes

Once calculated, the antenna temperature of the source is used to convert the power output of the noise diodes to a vectorized calibration temperature profile

$$T_{\text{cal}}(\nu) = T_{\text{a}}^{*}(\nu)\left[\frac{\text{Src}^{\text{on}} - \text{Src}^{\text{off}} + \text{Sky}^{\text{on}} - \text{Sky}^{\text{off}}}{\text{Src}^{\text{on}} - \text{Sky}^{\text{on}} + \text{Src}^{\text{off}} - \text{Sky}^{\text{off}}}\right], \tag{13.5}$$

where "Src" and "Sky" refer to the *source* and *sky* positions and superscripts "on" and "off" refer to the state of the noise diode. The calibration temperature $T_{\text{cal}}(\nu)$ is calculated separately for each frequency position and polarization.

Note that the formulation of Equation (13.5) is somewhat non-standard, and differs from the GBTIDL calibration guide in that both the *source* and *sky* positions are used to calculate the noise diode calibration temperature. Typically only one position is used. However, assuming the amplifier is linear in it's response, there should be no relevant difference in the the noise diode calibration temperatures between the *source* and *sky* positions. Even in the event that using Equation (13.5) introduced additional error, it would be equal between intermediate frequencies (and thus equal between isotopologues) and wouldnot affect isotope ratio calculations. Short integration times yield a vectorized $T_{\text{cal}}(\nu)$ has a relatively low signal-to-noise ratio; thus the increase in the signal to noise ratio is well worth the additional effort.

Since only large-scale variations in $T_{\text{cal}}(\nu)$ are of concern, Dreamcatcher.pro fits each solution with a low-order polynomial expression, $f(T_{\text{cal}}(\nu))$, which is used in all subsequent calculations. This step retains the relevant information about $T_{\text{cal}}(\nu)$ across the bandpass, while effectively filtering out any high-frequency or transient variations that may introduce additional noise into the science data during calibration. Care must be taken when fitting $T_{\text{cal}}(\nu)$, as any errors made here would be propagated through the rest of the reduction pipeline. Each fit should be examined manually, and the degree of the polynomial used can be controlled using the CALFITORDER keyword when calling Dreamcatcher.pro.

Up to this point a single generalized case of the calibration process has been explicitly demonstrated. It is important to note that for actual observations four independent temperatures are calculated for each IF, one for each polarization and frequency position combination. There is no difference in calibration procedure between the polarizations, however the equations for calibrating the *signal* and *reference* frequency positions differ slightly, so they are shown for clarity from this point forward.

13.3 Calibrating Science Data

With the calibration temperature $T_{\mathrm{cal}}(\nu)$ in hand, the system temperature $T_{\mathrm{sys}}(\nu)$ is calculated for each frequency position, polarization, and IF. Further, $T_{\mathrm{sys}}(\nu)$ is calculated separately for each integration, to capture any time dependent drift in the system response. The system temperatures for the *signal* and *reference* frequency positions are calculated using the equations

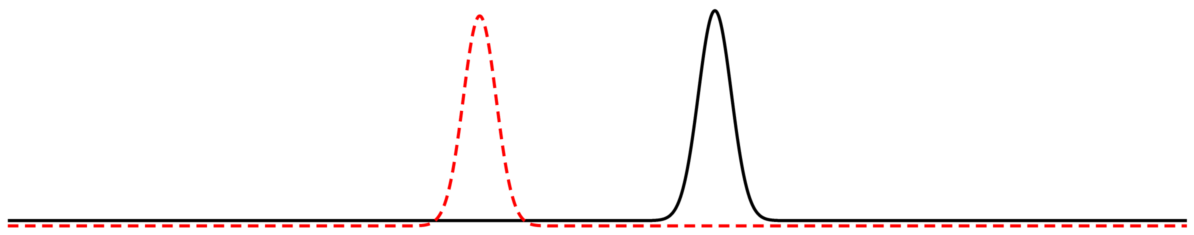
$$\begin{aligned} T_{\mathrm{sys}}^{\mathrm{sig}}(\nu) &= \frac{f(T_{\mathrm{cal}}^{\mathrm{sig}}(\nu))}{2}\left[\frac{\mathrm{Sig}^{\mathrm{on}}+\mathrm{Sig}^{\mathrm{off}}}{\mathrm{Sig}^{\mathrm{on}}-\mathrm{Sig}^{\mathrm{off}}}\right] \\ T_{\mathrm{sys}}^{\mathrm{ref}}(\nu) &= \frac{f(T_{\mathrm{cal}}^{\mathrm{ref}}(\nu))}{2}\left[\frac{\mathrm{Ref}^{\mathrm{on}}+\mathrm{Ref}^{\mathrm{off}}}{\mathrm{Ref}^{\mathrm{on}}-\mathrm{Ref}^{\mathrm{off}}}\right] \end{aligned} \tag{13.6}$$

where “Sig” and “Ref” indicate the *signal* and *reference* frequency positions respectively, and the superscripts refer to the state of the noise diode.

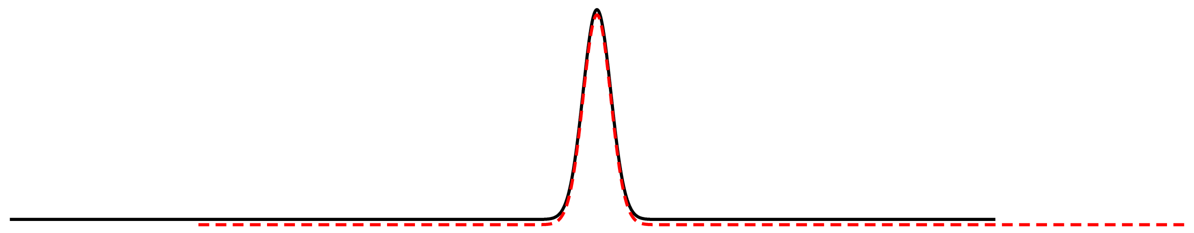
The antenna temperatures $T_a(\nu)$ of the two frequency positions is then calculated via the expressions

$$\begin{aligned} T_{\mathrm{a}}^{\mathrm{sig}}(\nu) &= f(T_{\mathrm{sys}}^{\mathrm{ref}}(\nu))\left[\frac{Sig-Ref}{Ref}\right] \\ T_{\mathrm{a}}^{\mathrm{ref}}(\nu) &= f(T_{\mathrm{sys}}^{\mathrm{sig}}(\nu))\left[\frac{Ref-Sig}{Sig}\right] \end{aligned} \tag{13.7}$$

where $f(T_{\mathrm{sys}}^{\mathrm{ref}}(\nu))$ and $f(T_{\mathrm{sys}}^{\mathrm{sig}}(\nu))$ are polynomial fits to the system temperature profiles $T_{\mathrm{sys}}^{\mathrm{sig}}(\nu)$ and $T_{\mathrm{sys}}^{\mathrm{ref}}(\nu)$ calculated in Equation (13.6). It is the net shape of the system temperature profiles that are valuable, so a low-order polynomial can be used to capture all relevant information, and doing so avoids introducing unnecessary noise when calculating the antenna



(a) Visualized in *channel-space*, the emission features in the *signal* and *reference* spectra are offset by the number of channels corresponding to the frequency offset between the frequency positions.



(b) Visualized in *frequency-space*, the emission features in the *signal* and *reference* spectra are no longer offset and can be properly folded together. The final spectrum is equivalent to the region where both phases overlap.

Figure 13.1: The solid black and dashed red lines represent the the *signal* and *reference* frequency positions, respectively. The small vertical offset is for visual clarity only.

temperatures $T_{\mathrm{a}}^{\mathrm{sig}}(\nu)$ and $T_{\mathrm{a}}^{\mathrm{ref}}(\nu)$ in the steps that follow. The order of these polynomial fits can be controlled with the FITORDER keyword when calling Dreamcatcher.pro.

It is recommended that considerable care be taken at this juncture; Dreamcatcher.pro will plot the polynomial fits to $T_{\mathrm{sys}}^{\mathrm{sig}}(\nu)$ and $T_{\mathrm{sys}}^{\mathrm{ref}}(\nu)$ and one should take care to ensure that the fits between integration are consistent. Fluctuation in the shape of the fit from one integration to the next is a good indication that $T_{\mathrm{sys}}^{\mathrm{sig}}(\nu)$ and $T_{\mathrm{sys}}^{\mathrm{ref}}(\nu)$ are being over-fitted.

At this point, Dreamcatcher.pro displays and saves a number of diagnostic plots, including the calculated antenna temperature profiles for the two frequency positions, after which the frequency positions are combined, or folded, to increase the SNR of the final spectrum by a factor of $\sqrt{2}$, assuming they have equal system temperatures (see Equation (13.8)). Because the bandpass of the *signal* and *reference* frequency positions differ by the frequency offset between them, one of the two profiles must be shifted before they can be properly folded. Dreamcatcher.pro performs this action in *frequency-space*, so no actual shifting is required;

Dreamcatcher.pro instead selects different portions of the *signal* and *reference* spectra such that the selected regions correspond to each other channel-by-channel in *frequency-space*, and discards the remainder (see Figure 13.1).

Typically, the antenna temperature profiles are folded with a simple, unweighted average, and any difference in system temperature between the two frequency positions is ignored. However, the system temperature profiles often differ between the *signal* and *reference* frequency positions, a fact that must be considered when making high precision spectral line measurements. To account for the disparate system temperatures between frequency positions, they are folded using a channel-by-channel weighted mean, where each channel is weighted by the the inverse square of the corresponding system temperature for that channel.

$$T_{\mathrm{a}}(\nu) = \frac{T_{\mathrm{a}}^{\mathrm{sig}}(\nu)(T_{\mathrm{sys}}^{\mathrm{sig}}(\nu))^{-2} + T_{a}^{\mathrm{ref}}(\nu)(T_{\mathrm{sys}}^{\mathrm{ref}}(\nu))^{-2}}{(T_{\mathrm{sys}}^{\mathrm{ref}}(\nu))^{-2} + (T_{\mathrm{sys}}^{\mathrm{sig}}(\nu))^{-2}} \tag{13.8}$$

All subsequent averaging operations between polarizations, integrations, scans and observations are done using the same channel-by-channel weighted mean as shown in Equation (13.8). Finally, the folded antenna temperature is corrected for the zenith opacity in a manner analogous to the calibrator source (Equations (13.2) and (13.4) and converted to main beam temperature via the equation

$$T_{\mathrm{mb}}(\nu) = \frac{T_{\mathrm{a}}(\nu)}{\vec{\eta}_{\mathrm{mb}}} \exp\left(\frac{-\vec{\tau}_{\mathrm{z}}}{\sin(\theta)}\right) \tag{13.9}$$

where $\vec{\eta}_{\mathrm{mb}}$ is the main beam efficiency, which for the GBT, is $\approx 1.3\eta_{\mathrm{a}}$.

CHAPTER 14

HYDRA Source Code

14.1 Hydra.pro

```
;+——————————————————————————————————————+
;>>>>> Ancillary Program(s) <<<<<
;+——————————————————————————————————————+

;+——————————————————————————————————————+
;>>>>> Primary Program <<<<<
;+——————————————————————————————————————+
;+
; NAME:
; Hydra
; HYDRA Version 5.1
;
; PURPOSE:
; –> Initiates the HYDRA runtime.
;
; CALLING SEQUENCE:
; –> Hydra [, /Radpath]
;
; ARGUMENT(S):
; –> None
;
; KEYWORD(S):
; –> None
;
; OPTIONAL KEYWORD(S):
; –> Radpath: Rebuild default shell path to inclide the local
; RADEX installation. Needed to run RADEX simulations in HYDRA.
;
; EXAMPLE(S):
```

```
; -> Hydra, /Radpath
;
; OUTPUT(S):
; -> None
;
; COMMENTS:
; -> Hydra.pro must be modified to include the path to the local
; HYDRA and RADEX installation directory paths.
;
; PROCEDURES/FUNCTIONS CALLED:
; -> None
;-

Pro Hydra, Radpath=radpath
  Compile_opt IDL2
  On_error, 0
  !Except = 1

  ;*** Provide the full filepath to the HYDRA installation directory here ***
  hydrapath = '/Users/Onyx/IDLWorkspace/hydra'

  ;*** Provide the full filepath to the RADEX installation directory here ***
  radexpath = 'ignore me'
  radpath = 0

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 5 June, 2014
  ;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 12 October, 2014
  ;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 19 June, 2015
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 9 December, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 4.1 (V-spec) written by: N.N. Monson (UCLA). 17 September, 2017
  ;>>>>> Version 4.2 (V-spec) written by: N.N. Monson (UCLA). 30 May, 2018
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
  ;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 January, 2019

  ;+-------------------------------------+
  ;>>>>> Usage Agreement <<<<<
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
  ;See HYDRA User's Guide.

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;>>>>> Developer's Notes <<<<<
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;>>>>> Limitations & Known Bugs <<<<<
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 0: − Check argument(s).
  ; − Set keyword defaults.
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Check to see if radpath is set.
  If Keyword_set(radpath) Then Begin
    ;If radpath is set to anything other than 1, change it to 1 to
    ;ensure compatibility with logical operators.
    If radpath Ne 1 Then Begin
      Print, '>>>>> Alert: radpath is a binary keyword'
      Print, '>>>>> setting radpath accordingly ...'
      Print, ''
      radpath = 1
    Endif
  Endif

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 1: − Build HYDRA path cache.
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Print an empty line below the program call on the command line.
  Print, ''
```

```
;Print some stuff...
Print, '>>>>> Initializing HYDRA Runtime...'
Wait, 0.5
Print, '>>>>> Building Path...'

;Add HYDRA installation directory to existing path and rebuild
!Path = Expand_path('+' + Strtrim(hydrapath,2) + ':+' + Strtrim(!Dir, 2))
Path_cache, enable = 1, /rebuild

;Make sure the directory is the correct one,
;i.e. is properly labeled as 'Hydra'. If not, throw an error.
scrap = Strsplit(Strtrim(hydrapath, 2), '/', /extract)
If ~Strcmp(scrap[-1], 'hydra', /fold_case) Then Begin
  Print, '>>>>> Error: Root Directory Invalid'
  Return
Endif

;Make sure directory is not empty.
files = File_search(Strtrim(hydrapath, 2) + '/SiO_Pipeline', $
  '*.pro', /fully_qualify_path)
If files.length Eq 0 Then Begin
  Print, '>>>>> Error: Root Directory Empty'
  Return
Endif

;Pick out all the filenames in the HYDRA directory & compile the files.
;!quiet is set to 0 just to avoid printing everything to the command line.
files = Strsplit(Transpose(files), '/.', /extract)
proname = Strarr(files.length)
For i = 0, files.length-1 Do proname[i] = files[i,-2]
!Quiet=1
Resolve_routine, proname, /either, /compile_full_file
!Quiet=0

;+----------------------------------------+
;SECTION 2: - Include RADEX in shell path cache.
;+----------------------------------------+

;use setenv to insert the locar RADEX installation into the default shell path
If Keyword_set(radpath) Then Begin
```

```
  Setenv, 'PATH=/usr/bin:/bin:/usr/sbin:/sbin:' + Strtrim(radexpath, 2)
  Print, '>>>>> Setting path for RADEX...'
  Print, ''
Endif Else Setenv, 'PATH=/usr/bin:/bin:/usr/sbin:/sbin'

;+----------------------------------------+
;SECTION 3: - Build common variables
;+----------------------------------------+

;Print some stuff...
Print, '>>>>> ...Done'
Print, ''
Print, '>>>>> Initializing Common Block Variables...'

;Initialize common variables.
;The variables that are in these common blocks are here because they need
;to be available to several different HYDRA programs.

;initialize overlord common variable block.
;this variable is used to store .fits file data.

;The summary and sdfdata variables are arrays of structures,
;the size of which changes depending on the data being read.
;This makes them cumbersome to incorporate into the OVERLORD
;structure, so I left them independent.
Common Overlord, Overlord, olord_summary, olord_sdfdata

overlord = {overlord, dirloc: '', $
  sumsize: 0, $
  scanloc: [0], $
  scannum: [0], $
  sdfsize: 0, $
  sdfname: '', $
  sdfpath: ''}

;Initialize RTYPE common variable block.
;This variable is used to store information on array/structure sizes and
;file formatting. Since it will be accessible to all HYDRA programs,
;future changes to sizes and/or formatting will only need to be made here,
;and will be applied uniformly across the whole pipeline.
```

```
Common Rtype, rtype

rtype = { sml: 6000, $
  med: 8192, $
  lrg: 65536, $
  format: 'e17.9', $
  adtypes: ['INT', 'LONG', 'LONG64', 'FLOAT', 'DOUBLE'], $
  aivals: [28,29,30], $
  aifvals: [0,1,2,3] }

;Initialize LSPEX common variable block.
;This variable is used to store information on the baseline fits to the
;reduced spectra. i.e. regions used in the fit, and the polynomial order.
Common Lspex, lspex

lspex = { s28: { xlims: List(), $
  numlims: 0, $
  order: 0 }, $
  s29: { xlims: List(), $
  numlims: 0, $
  order: 0 }, $
  s30: { xlims: List(), $
  numlims: 0, $
  order: 0 }, $
  rrl: { xlims: List(), $
  numlims: 0, $
  order: 0 } }

;Initialize MCSPEX common variable block.
;This variable is used to pass data around during monte-carlo simulations.
Common Mcspex, mcspex

mcspex = { s28: { wild: Dblarr(rtype.sml, /nozero), $
  tame: Dblarr(rtype.sml, /nozero), $
  vlsr: Dblarr(rtype.sml, /nozero), $
  vbin: Dblarr(rtype.sml, /nozero), $
  rmsnt: 0d }, $
  s29: { wild: Dblarr(rtype.sml, /nozero), $
  tame: Dblarr(rtype.sml, /nozero), $
  vlsr: Dblarr(rtype.sml, /nozero), $
  vbin: Dblarr(rtype.sml, /nozero), $
  rmsnt: 0d }, $
  s30: { wild: Dblarr(rtype.sml, /nozero), $
  tame: Dblarr(rtype.sml, /nozero), $
  vlsr: Dblarr(rtype.sml, /nozero), $
  vbin: Dblarr(rtype.sml, /nozero), $
  rmsnt: 0d }, $
  rrl: { wild: Dblarr(rtype.sml, /nozero), $
  tame: Dblarr(rtype.sml, /nozero), $
  vlsr: Dblarr(rtype.sml, /nozero), $
  vbin: Dblarr(rtype.sml, /nozero), $
  rmsnt: 0d } }

;Initialize PSPEX common variable block.
Common Pspex, pspex

;Initialize structure(s) in PSPEX
pspex = { windim: [1900,800], $
  icolors: { white: [0,0,0], $
  black: [255,255,255], $
  teal: [250,0,0], $
  aqua: [250,0,125], $
  green: [250,0,250], $
  puke: [125,0,250], $
  yellow: [0,0,250], $
  orange: [0,125,250], $
  red: [0,250,250], $
  pink: [0,250,125], $
  violet: [0,250,0], $
  purple: [125,250,0], $
  blue: [250,250,0], $
  sky: [250,125,0] }, $
  rcolors: { white: [255,255,255], $
  black: [0,0,0], $
  teal: [0,250,250], $
  aqua: [0,125,250], $
  green: [0,0,250], $
  puke: [125,250,0], $
  yellow: [250,250,0], $
  orange: [250,125,0], $
```

```
    red: [250,0,0], $
    pink: [250,0,125], $
    violet: [250,0,250], $
    purple: [125,0,250], $
    blue: [0,0,250], $
    sky: [0,125,250] } }

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 4: − Print cool banner.
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Change prompt and error message prefix, just for fun.
  !Prompt = 'HYDRA> '
  !Error_state.msg_prefix = '>>>>> '

  Print, '>>>>> ...Done'
  Print, ''

  ;HYDRA logo omitted for brevity

  Print, ''
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, '>>>>> HYDRA Online <<<<<'
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, 'Version 5.1'
  Print, ''
  Print, '>>>>> Please Consult The Supplied User Guide For F.A.Qs'
  Print, ">>>>> Type '.RESET_SESSION' To Exit To Primary Runtime"
  Print, ''
  Wait, 0.5
  Print, '>>>>> End Of Line <<<<<'
  Print, ''

End
```

14.2 Getfits.pro

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME
; Summary
; HYDRA Version 5.1
;
; PURPOSE
; −> Prints a summary of the .sdfits data currently in memory.
;
; CALLING SEQUENCE
; −> Summary
;
; ARGUMENT(S)
; −> None
;
; KEYWORD(S)
; −> None
;
; OPTIONAL KEYWORD(S)
; −> None
;
; EXAMPLE(S)
; −> Summary
;
; OUTPUT(S)
; −> None.
;
; COMMENTS
; −> Operates only within the HYDRA 5.1 RTE.
; −> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro
;
; PROCEDURES/FUNCTIONS CALLED
; −> None.
;−
```

```
Pro Summary
  Compile_opt IDL2
  Common overlord
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
  ;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 January, 2019

  ;+---------------------------------------+
  ;>>>>> Usage Agreement <<<<<
  ;+---------------------------------------+

  ;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
  ;See HYDRA User's Guide.

  ;+---------------------------------------+
  ;>>>>> Developer's Notes <<<<<
  ;+---------------------------------------+

  ;+---------------------------------------+
  ;>>>>> Limitations & Known Bugs <<<<<
  ;+---------------------------------------+

  ;None known.

  ;+---------------------------------------+
  ;SECTION 0: - Check argument(s).
  ; - Set keyword defaults.
  ;+---------------------------------------+

  ;Print an empty line below the program call on the command line.
```

```
Print, ''

;Initialize variable(s).
keymaster = 0

;Ensure OVERLORD has already been created by getfits.pro.
;If not, then throw an error.
If ~ Isa(olord_sdfdata) Then Begin
  Print, '>>>>> Error: No .sdfits file in memory'
  Print, '>>>>> Alert: Please load an .sdfits file before calling summary'
  Print, '' & ++ keymaster
Endif Else Begin
  If Overlord.dirloc Eq '' Then crap = '*** NOT SET ***' $
  Else crap = Overlord.dirloc
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
Endelse

;+-----------------------------------------+
;SECTION 1: - Print .sdfits summary data.
;+-----------------------------------------+

;Use a proxy of the OVERLORD variable 'summary', so nothing is
;inadvertently altered.
sumprox = olord_summary

;Print some stuff.
```

```
  Print, '>>>>> Current working directory: ' + Strtrim(crap, 2)
  Print, ''
  Print, ' scan source int mode'
  Print, '−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−'
  Print, ''

  ;Loop to print the data in overlord.summary for each scan in the .sdfits file.
  For i=0, sumprox.length−1 Do Begin
    Print, sumprox[i].scan, Strtrim(sumprox[i].object,2), sumprox[i].nrec, $
      sumprox[i].mode, format = '(3x,i−3,4x,a−16,5x,i−3,3x,a−8)'
  Endfor

  ;Print some stuff and end.
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
End

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME
; Getfits
; HYDRA Version 5.1
;
; PURPOSE
; −> Load the contents of a .fits file to memory.
;
; CALLING SEQUENCE
; −> Getfits, path [, /BYPASS, /SILENT]
;
; ARGUMENT(S)
; −> Path: Full filepath to the GBT .fits file to be read and stored in memory.
;
; KEYWORD(S)
; −> None
;
; OPTIONAL KEYWORD(S)
; −> Silent (binary): Suppresses print commands & Modifies how summary data
; are stored. Only to be used when other HYDRA programs call getfits.
; Getfits.pro should NOT be called from the command line using this keyword.
; −> Bypass (binary): Suppresses the warning for not yet having set the working
; directory with setdir.pro.
;
; EXAMPLE(S)
; −> getfits, '/Users/Bruce_Lee/GBT/AGBT110_05/AGBT110_05.raw.vegas.D.fits'
;
; OUTPUT(S)
; −> None. Stores all data in the OVERLORD common variable.
;
; COMMENTS
; −> Operates only within the HYDRA 5.1 RTE.
; −> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro prior
;
; PROCEDURES/FUNCTIONS CALLED
; −> summary__define.pro
;-

Pro Getfits, path, Bypass=bypass, Silent=silent
  Compile_opt IDL2
  Common overlord
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 4.0 (V−spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V−spec) written by: N.N. Monson (UCLA). 23 March, 2018
  ;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 11 January, 2019

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;>>>>> Usage Agreement <<<<<
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
```

```
;See HYDRA User's Guide.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Developer's Notes <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Bugs <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;None known.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
If Not Keyword_set(silent) Then Print, ''

;Inialize structure(s) and/or array(s).
keymaster = 0

;Use /bypass to circumvent this check
If ~ Keyword_set(bypass) Then Begin
  ;Check to see if the working directory has been set by setdir.pro
  If Array_equal('', overlord.dirloc) Then Begin
    Print, '>>>>> Alert: Working directory not set'
    Print, ''
  Endif
Endif

;Check if the PATH argument is a string and a scalar.
If Isa(path, /string, /scalar) Then Begin
  ;If it is, ensure the path argument leads to a real file.
  ;If not, then throw an error.
  If ~ File_test(path, /regular) Then Begin
    Print, '>>>>> Error: Invalid path'
    Print, '' & ++ keymaster
  Endif
  ;Chop up the given pathname to determine the file extension.
  ;If the file doesn't have a .fits extension, thow an error.
  dummy = Strsplit(path, '.', /extract)
  If dummy[−1] Ne 'fits' Then Begin
    Print, '>>>>> Error: Improper file extension.'
    Print, '>>>>> Alert: File must be a .fits file'
    Print, '' & ++ keymaster
  Endif

Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Alert: Path argument type = ' + Strtrim(Typename(path),2)
  Print, '>>>>> Alert: Path argument dimension = ' + Strtrim(path.length,2)
  Print, '>>>>> Error: Path argument must be a string and a scalar'
  Print, '' & ++ keymaster
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise continue & print some stuff if /silent is not set.
  If Not Keyword_set(silent) Then Begin
    Print, '>>>>> Alert: Status Green'
    Print, '>>>>> Continuing...'
    Print, ''
    Print, '>>>>> Message: .sdfits file confirmed'
    Print, '>>>>> Message: Initializing data import'
    Print, ''
  Endif
Endelse

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: − Separate the .fits filename from the directory path.
```

```
;+––––––––––––––––––––––––––––––––––––––+

;Getfits should only be called w/ the silent keyword set by extract.pro,
;in which case overlord.sdfname and overlord.sdfpath need to remain unchanged.
;If /silent is not set, extract the name of the actual .sdfits file,
;as well as the directory it is in.
If ~ Keyword_set(silent) Then Begin
  junk = Strsplit(path, '/', /extract)
  morejunk = Strsplit(junk[−1], '.', /extract)

  Overlord.sdfname = morejunk[0]
  Overlord.sdfpath = '/' + Strjoin(junk[0:−2], '/')
Endif

;+––––––––––––––––––––––––––––––––––––––+
;SECTION 2: – Read .fits data using MRDFITS
; – Determine the number location and size of the scans.
;+––––––––––––––––––––––––––––––––––––––+

;Pull data out of .sdfits file using MRDFITS, which is part of
;NASA's IDL Astronomy Users Library. go to idlastro.gsfc.nasa.gov
;for further information & to download.
sdfdata = Mrdfits(Strtrim(filepath, 2), 1, header)

;Fix error in how the .sampler values are assigned by MRDFITS.
sdfdata.sampler = Strtrim(sdfdata.sampler, 2)

;Use a proxy of sdfdata, so nothing is inadvertently altered.
datprox = sdfdata

;Find the number of each scan, and the start location in sdfdata.
scanloc = Where(datprox.scan Ne Shift(datprox.scan, 1))
scannum = datprox[scanloc].scan

;Initialize structure(s) and/or array(s).
num = Lonarr(scanloc.length)

;determine number of corresponding sdfdata entries for each scan,
;and divide by 8 to get the number of integrations per scan.
;note: the datprox.length bit is there only so the size of the
```

```
;last scan can be calculated. See extract.pro.
num = Shift([scanloc, datprox.length] − Shift([scanloc, datprox.length], 1), −1)
num = Temporary(num) / 8

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 3: − Create and fill olord_summary
; − File everything away in OVERLORD.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize structure(s) and/or array(s).
summary = Replicate(Create_struct(name = 'summary'), scanloc.length)

;Fill the summary structure with the relevant data.
For j=0, scanloc.length−1 Do Begin
  summary[j].scan = datprox[scanloc[j]].scan
  summary[j].object = datprox[scanloc[j]].object
  summary[j].nrec = num[j]
  tmode = Strsplit(datprox[scanloc[j]].obsmode, ':', /extract)
  summary[j].mode = tmode[0]
Endfor

;OVERLORD is initialized by hydra.pro and should already exist.
;So, update its values, except overlord.dirloc, which is controlled
;and updated by setdir.pro.
Overlord.sumsize = scanloc.length
Overlord.scanloc = scanloc
Overlord.scannum = scannum
Overlord.sdfsize = datprox.length

;The summary and sdfdata variables are arrays of structures,
;the size of which changes depending on the data being read.
;This makes them cumbersome to incorporate into the OVERLORD
;structure, so I left them independent.
olord_summary = summary
olord_sdfdata = sdfdata

;Print some stuff, call summary and end.
Print, '>>>>> ...Done'
Print, ''
Print, ">>>>> Message: Sdfits data stored as 'olord.sdfdata'"
```

```
  Print, ">>>>> Message: Summary of data stored as 'olord.summary'"
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'

  Summary

End
```

14.3 Setdir.pro

```
;+-------------------------------------------+
;>>>>> Ancillary Program(s) <<<<<
;+-------------------------------------------+

;+-------------------------------------------+
;>>>>> Primary Program <<<<<
;+-------------------------------------------+
;+
; NAME
; Setdir
; HYDRA Version 5.1
;
; PURPOSE
; -> Sets the path to the directory that other HYDRA programs will
; write data to, and read data from. Other HYDRA programs will
; create various subdirectories as required.
;
; CALLING SEQUENCE
; -> setdir, path
;
; ARGUMENT(S)
; -> Path: The full path to the directory that other HYDRA programs
; read and write data to.
;
; KEYWORD(S)
; -> None
;
; OPTIONAL KEYWORD(S)
; -> None
;
; EXAMPLE(S)
; -> setdir, '/Users/Bruce_Lee/Data/Orion_KL'
;
; OUTPUT(S)
; -> None
;
; COMMENTS
; -> Operates only within the HYDRA 5.1 RTE.
```

```
;
; PROCEDURES/FUNCTIONS CALLED:
; -> None
;-

Pro Setdir, path
  Compile_opt IDL2
  Common overlord
  On_error, 1
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
  ;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 January, 2019

  ;+----------------------------------------+
  ;>>>>> Usage Agreement <<<<<
  ;+----------------------------------------+

  ;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
  ;See HYDRA User's Guide.

  ;+----------------------------------------+
  ;>>>>> Developer's Notes <<<<<
  ;+----------------------------------------+

  ;+----------------------------------------+
  ;>>>>> Limitations & Known Bugs <<<<<
  ;+----------------------------------------+

  ;None known.

  ;+----------------------------------------+
  ;SECTION 0: - Check argument(s).
```

```
; – Set keyword defaults.
;+––––––––––––––––––––––––––––––––––––––––––+

;Print an empty line below the program call on the command line.
Print, ''

;Initialize variable(s).
keymaster = 0

;Check if the PATH argument is a string and a scalar.
If Isa(path, /string, /scalar) Then Begin

  ;If it is, ensure the path argument leads to a real directory.
  ;If not, then throw an error.
  If ~ File_test(path, /directory) Then Begin
    Print, '>>>>> Error: Invalid path'
    Print, '' & ++ keymaster
  Endif
Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Alert: Path argument type = '+Strtrim(Typename(path),2)
  Print, '>>>>> Alert: Path argument dimension = '+Strtrim(path.length,2)
  Print, '>>>>> Error: Path argument must be a string and a scalar'
  Print, '' & ++ keymaster
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise, continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
Endelse
```

```
    ;+-----------------------------------------+
    ;SECTION 1: - Save filepath
    ;+-----------------------------------------+

    ;Put path into the OVERLORD structure.
    Overlord.dirloc = path

    ;Print some stuff and end.
    Print, '>>>>> Alert: Read/Write filepath set to: '
    Print, '>>>>> ' + Strtrim(Overlord.dirloc,2)
    Print, ''
    Print, '>>>>> End Of Line <<<<<'
    Print, ''

End
```

14.4 Gather.pro

```
;+————————————————————————————————————+
;>>>>> Ancillary Program(s) <<<<<
;+————————————————————————————————————+

;+————————————————————————————————————+
;>>>>> Primary Program <<<<<
;+————————————————————————————————————+
;+
; NAME:
; Gather
; HYDRA Version 5.1
;
; PURPOSE:
; –> Retreives data from a sdfits file and rewrites them as ascii .dat files.
; It writes separate subdirectories for each source, and will organize
; the output files within their respective subdirectories.
;
; CALLING SEQUENCE:
; –> Gather, session, Onscans=vector, Onints=list, Offscans=vector,
; Offints=list, Srcscans=vector, Iso=integer [, /Rrl, /Justcal]
;
; ARGUMENT(S):
; –> Session: An arbitrary integer assigned to the data being extracted.
; Used toorganize data from different observations.
;
; KEYWORD(S):
; –> Onscans: A longword vector of the calibrator scans to be extracted.
;
; –> Offscans: A longword vector of the reference scans to be extracted.
;
; –> Iso: The mass number of the isotope to be reduced and calibrated.
; This keyword is ignored if the /RRL keyword is set.
; * Acceptable values: '28', '29', '30'.
;
; OPTIONAL KEYWORD(S):
; –> Onints: A list containing the integrations of each calibrator scan
; to be extracted. If not set, the first integration of each scan
; will be used by default.
;
; -> Offints: A list containing the integrations of each calibrator
; reference scan to be extracted. If not set, the first integration
; of each scan will be used by default
;
; -> Rrl (binary): Set to extract rrl data. Overrides ISO keyword.
;
; ->Justcal (binary): Set to extract calibrator and calibrator offset
; data only. Overrides SRCSCANS keyword.
;
; EXAMPLE(S):
; -> Gather, 0, Onscans=[12,14], Onints=List([0,1],[1]), Offscans=[13,15], $
; Offints=List([0,1],[0]), Srcscans=[22,23,24,25], Iso=28
;
; OUTPUT(S):
; -> A single compressed data file for the calibrator and reference scans,
; as well as one per source scan. output files are organised within
; their respective subdirectories.
;
; COMMENTS:
; -> Operates only within the HYDRA 5.1 RTE.
; -> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro
;
; PROCEDURES/FUNCTIONS CALLED:
; -> Extract.pro
; -> Scdata_s__define.pro
;-

Pro Gather, session, Onscans=onscans, Onints=onints, Offscans=offscans, $
  Offints=offints, Srcscans=srcscans, Iso=iso, Rrl=rrl, Justcal=justcal
  Compile_opt IDL2
  Common overlord
  Common rtype
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
```

```
;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 5 June, 2014
;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 12 October, 2014
;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 19 June, 2015
;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 9 December, 2016
;>>>>> Version 4.0 (V−spec) written by: N.N. Monson (UCLA). 30 April, 2017
;>>>>> Version 4.1 (V−spec) written by: N.N. Monson (UCLA). 17 September, 2017
;>>>>> Version 4.2 (V−spec) written by: N.N. Monson (UCLA). 30 May, 2018
;>>>>> Version 5.0 (V−spec) written by: N.N. Monson (UCLA). 23 November, 2018
;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 11 January, 2019

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Usage Agreement <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Developer's Notes <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Bugs <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;None Known.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
Print, ''

;Inialize structure(s) and/or array(s).
```

```
keymaster = 0
callsign = ''
fext = ''
ifreq = 0

;Check to see if the working directory has been set by setdir.pro
If Array_equal('', overlord.dirloc) Then Begin
  Print, '>>>>> Error: Common directory not set'
  Print, '>>>>> Set common directory using Setdir.pro'
  Print, '' & ++ keymaster
Endif

;Ensure the SESSION argument is a scalar and
;is a longword integer, or can be converted to one.
;If SESSION is a funky type, e.g. unsigned, then throw an error.
If Array_equal(0, Isa(session, /scalar)) || $
  Array_equal(1, rtype.adtypes.Contains(Typename(session)), /not_equal) Then Begin
  Print, '>>>>> Alert: Session argument dimension = '+Strtrim(Session.length,2)
  Print, '>>>>> Error: Session argument type = '+Strtrim(Typename(session),2)
  Print, '>>>>> Alert: Session argument must be a longword or floating-point scalar'
  Print, '' & ++ keymaster
Endif Else session = Long(session)

;Check if the ONSCANS keyword it set.
If Keyword_set(onscans) Then Begin

  ;If it is, ensure the ONSCANS keyword is a vector and
  ;is a longword integer, or can be converted to one.
  ;If ONSCANS is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(onscans, /vector)) || $
    Array_equal(1, rtype.adtypes.Contains(Typename(onscans)), /not_equal) Then Begin
    Print, '>>>>> Alert: Onscans keyword type = '+Strtrim(Typename(onscans),2)
    Print, '>>>>> Error: Onscans keyword must be a longword or floating-point vector'
    Print, '' & ++ keymaster
  Endif Else onscans = Long(onscans)
Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Error: Onscans keyword not set'
  Print, '' & ++ keymaster
Endelse
```

```
;Check if the OFFSCANS keyword it set.
If Keyword_set(offscans) Then Begin

  ;If it is, ensure the OFFSCANS keyword is a vector and
  ;is a longword integer, or can be converted to one.
  ;If offscans is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(offscans, /vector)) || $
    Array_equal(1, rtype.adtypes.Contains(Typename(offscans)), /not_equal) Then Begin
    Print, '>>>>> Alert: Offscans keyword type = '+Strtrim(Typename(onscans),2)
    Print, '>>>>> Error: Offscans keyword must be a longword or floating-point vector'
    Print, '' & ++ keymaster
  Endif Else offscans = Long(offscans)
Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Error: Offscans keyword not set'
  Print, '' & ++ keymaster
Endelse

;Make sure the onscans and offscans have the same length.
;If not, then throw an error.
If onscans.length Ne offscans.length Then Begin
  Print, '>>>>> Error: Onscans and offscans keywords must have equal dimiensions'
  Print, '' & ++ keymaster
Endif

;Check if the ONINTS keyword is set
If Keyword_set(onints) Then Begin

  ;If it is, ensure the ONINTS keyword is a list. If not, then throw an error.
  If Typename(onints) Ne 'LIST' Then Begin
    Print, '>>>>> Alert: Onints keyword type = '+Strtrim(Typename(onints),2)
    Print, '>>>>> Error: Onints keyword must be a list'
    Print, '' & ++ keymaster
  Endif

  ;And make sure the ONINTS and ONSCANS keywords have the same length.
  ;If not, then throw an error.
  If onints.length Ne onscans.length Then Begin
    Print, '>>>>> Error: Onscans and onints keywords must have equal dimiensions'
    Print, '' & ++ keymaster
  Endif
Endif Else Begin
  ;Otherwise, set the default value.
  Print, '>>>>> Alert: Onints keyword not set'
  Print, '>>>>> Using default values ...'
  onints = List(Lonarr(onscans.length), /extract)
Endelse

;Check if the OFFINTS keyword is set
If Keyword_set(offints) Then Begin

  ;If it is, ensure the OFFINTS keyword is a list. If not, then throw an error.
  If Typename(offints) Ne 'LIST' Then Begin
    Print, '>>>>> Alert: Offints keyword type = '+Strtrim(Typename(offints),2)
    Print, '>>>>> Error: Offints keyword must be a list'
    Print, '' & ++ keymaster
  Endif

  ;And Make sure the OFFINTS and OFFSCANS keywords have the same length.
  ;If not, then throw an error.
  If offints.length Ne offscans.length Then Begin
    Print, '>>>>> Error: Offscans and offints keywords must have equal dimiensions'
    Print, '' & ++ keymaster
  Endif
Endif Else Begin
  ;Otherwise, set the default value.
  Print, '>>>>> Alert: Offints keyword not set'
  Print, '>>>>> Using default values ...'
  offints = List(Lonarr(offscans.length), /extract)
Endelse

;Check if the /JUSTCAL keyword is set.
If Keyword_set(justcal) Then Begin

  ;If it is, check to see if the /JUSTCAL keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If justcal Ne 1 Then Begin
    Print, '>>>>> Alert: Justcal is a binary keyword'
    Print, '>>>>> Setting justcal accordingly ...'
    Print, '' & justcal = 1
  Endif

  ;Ignore SRCSCANS keyword if the JUSTCAL keyword is set
  If Keyword_set(srcscans) Then Begin
    Print, '>>>>> Alert: Justcal keyword set, ignoring srcscans keyword'
    Print, ''
  Endif
Endif Else Begin
  ;Otherwise, ensure the SRCSCANS keyword is set.
  If Keyword_set(srcscans) Then Begin

    ;If it is, ensure the SRCSCANS keyword is a vector and
    ;is a longword integer, or can be converted to one.
    ;If SRCSCANS is a funky type, e.g. unsigned, then throw an error.
    If Array_equal(0, Isa(srcscans, /vector)) || $
      Array_equal(1, rtype.adtypes.Contains(Typename(srcscans)), /not_equal) Then Begin
      Print, '>>>>> Alert: Srcscans keyword type = '+Strtrim(Typename(srcscans),2)
      Print, '>>>>> Error: Srcscans keyword must be a longword or floating-point vector'
      Print, '' & ++ keymaster
    Endif Else srcscans = Long(srcscans)
  Endif Else Begin
    ;Otherwise, throw an error.
    Print, '>>>>> Error: Srcscans keyword not set'
    Print, '' & ++ keymaster
  Endelse
Endelse

;Check if the /RRL keyword is set.
If Keyword_set(rrl) Then Begin

  ;If it is, check to see if the /RRL keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If rrl Ne 1 Then Begin
    Print, '>>>>> Alert: Rrl is a binary keyword'
    Print, '>>>>> Setting rrl accordingly ...'
    Print, '' & rrl = 1
  Endif

  ;Ignore iso keyword if RRL keyword is set
```

```
 If Keyword_set(iso) Then Begin
   Print, '>>>>> Alert: Rrl keyword set, ignoring iso keyword'
   Print, ''
 Endif

 ;set the callsign, file extension and ifreq values.
 callsign = 'rrl'
 fext = '.raw.vegas.C.fits'
 ifreq = 2

Endif Else Begin
 ;Otherwise, Check if the ISO keyword is set
 If Keyword_set(iso) Then Begin

   ;Ensure the ISO keyword is a scalar and
   ;is a longword integer, or can be converted to one.
   ;If ISO is a funky type, e.g. unsigned, then throw an error.
   If Array_equal(0, Isa(iso, /scalar)) || $
     Array_equal(1, rtype.adtypes.Contains(Typename(iso)), /not_equal) Then Begin
     Print, '>>>>> Alert: Iso keyword dimension = '+Strtrim(iso.length,2)
     Print, '>>>>> Alert: Iso keyword type = '+Strtrim(Typename(iso),2)
     Print, '>>>>> Error: Iso Keyword must be a longword or floating-point scalar'
     Print, '' & ++ keymaster
   Endif Else Begin
     ;Otherwise, convert ISO to a longword
     iso = Long(iso)

     ;And check to see if ISO is in range. If not, then throw an error.
     If Array_equal(iso, rtype.aivals, /not_equal) Then Begin
       Print, '>>>>> Error: Iso keyword out of range'
       Print, '' & ++ keymaster
     Endif Else Begin
       ;If it is, set the file extension and ifreq values.
       ;NOTE: these values are specific to the q-band/VEGAS setup
       ;used to collect SiO data. These values will almost certainly
       ;be different for other setups!
       Case iso Of
         28:Begin
           fext = '.raw.vegas.D.fits'
           ifreq = 0
```

```
          End
          29:Begin
            fext = '.raw.vegas.B.fits'
            ifreq = 1
          End
          30:Begin
            fext = '.raw.vegas.A.fits'
            ifreq = 3
          End
        Endcase

        ;set the callsign
        callsign = Strtrim(iso,2) + 'sio'
      Endelse
    Endelse
  Endif Else Begin
    ;Otherwise, throw an error.
    Print, '>>>>> Error: Iso keyword is not set'
    Print, '' & ++ keymaster
  Endelse
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise, continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
Endelse

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: − Extract calibrator data and header.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;Call getfits to load the correct data to memory.
Getfits, overlord.sdfpath + '/' + overlord.sdfname + fext, /silent

;Print some jazz
Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
Print, ''
Print, '>>>>> Extracting calibrator data ...'
Print, ''

;Inialize variable(s).
advance = 0
icount = 0

;Count the total number of integrations (for all scans) to be extracted.
For jj=0, onscans.length−1 Do icount = Temporary(icount) + onints[jj].length

;Initialize structure(s) and/or array(s).
proncal = Replicate(Create_struct(name = 'scdata_s'), icount)
oncal = Create_struct(name = 'scdata_s')

;Begin loops for the calibrator data.
;Run this loop once for each scan called with onscans.
For master=0, onscans.length−1 Do Begin
  ;Get header and data for the calibrator scans,
  ;All polarizations, frequency positions and calstates.
  numints = 0
  result = Extract(onscans[master], ifreq, numints, onhead, /resample)
  If numints Eq 0 Then Message, '>>>>> Error: Extraction failure'

  ;Print number of integrations
  Print, '>>>>> ' + Strtrim(result.length,2) + ' integration(s) retrieved'
  Print, ''

  ;Take the data out of 'result' and file it away in proncal.
  Foreach pp, onints[master] Do Begin
    proncal[advance] = result[pp]

    ;Print some jazz & Incriment the loop counter
    Print, '>>>>> Keeping integration ' + Strtrim(pp,2)
    ++ advance
  Endforeach
Endfor

;Print some stuff.
Print, '>>>>> Averaging ' + Strtrim(advance,2) + ' (of ' + $
  Strtrim(icount,2) + ' commanded) integrations'
Print, ''

;Average integrations and insert in to the new data structure.
If icount Eq 1 Then Begin
  oncal = proncal
Endif Else Begin
  oncal.sig.left.tube_on = Mean(proncal.sig.left.tube_on, dimension=2, /double)
  oncal.sig.left.tube_off = Mean(proncal.sig.left.tube_off, dimension=2, /double)
  oncal.sig.right.tube_on = Mean(proncal.sig.right.tube_on, dimension=2, /double)
  oncal.sig.right.tube_off = Mean(proncal.sig.right.tube_off, dimension=2, /double)

  oncal.ref.left.tube_on = Mean(proncal.ref.left.tube_on, dimension=2, /double)
  oncal.ref.left.tube_off = Mean(proncal.ref.left.tube_off, dimension=2, /double)
  oncal.ref.right.tube_on = Mean(proncal.ref.right.tube_on, dimension=2, /double)
  oncal.ref.right.tube_off = Mean(proncal.ref.right.tube_off, dimension=2, /double)
Endelse

;+---------------------------------------+
;SECTION 2: - Create directories
; - Write calibrator data to ascii files.
;+---------------------------------------+

;Define path to the folder to save the data to, and create the directory.
rootpath = Strtrim(overlord.dirloc, 2) + '/session_0' + $
  Strtrim(session,2) + '/' + callsign
If ~Keyword_set(justcal) Then File_mkdir, rootpath, /noexpand_path

;Create directory to write onsource calibrator scan data to.
File_mkdir, Strtrim(rootpath, 2) + '/calibrator', /noexpand_path

;Print some jazz
Print, '>>>>> Writing data to .gzip file ...'
Print, ''
```

```
;Write signal phase data to an ascii datafile for the onsource
;calibration scans, noisetube on & off, both polarizations.
Openw, lun, Strtrim(rootpath, 2) + $
  '/calibrator/sig_phase.dat.gzip', /get_lun, /compress
Printf, lun, Strtrim(Systime(), 2)
Printf, lun, [[icount], [N_tags(onhead.sig)]]

;Build & insert standard size header.
Help, onhead.sig, output = output
head = Strsplit(output[1:*], " '", /extract)
Foreach part, head Do Printf, lun, part

;Write structure tags and data.
Printf, lun, [['sig.left.tube_on'], $
  ['sig.left.tube_off'], $
  ['sig.right.tube_on'], $
  ['sig.right.tube_off']]
Printf, lun, Transpose([[oncal.sig.left.tube_on], $
  [oncal.sig.left.tube_off], $
  [oncal.sig.right.tube_on], $
  [oncal.sig.right.tube_off]]), $
  format = '(4' + rtype.format + ')'
Close, lun
Free_lun, lun

;Write reference phase data to an ascii datafile for the onsource
;calibration scans, noisetube on & off, both polarizations.
Openw, lun, Strtrim(rootpath, 2) + $
  '/calibrator/ref_phase.dat.gzip', /get_lun, /compress
Printf, lun, Strtrim(Systime(), 2)
Printf, lun, [[icount], [N_tags(onhead.ref)]]

;Build & insert standard size header.
Help, onhead.ref, output = output
head = Strsplit(output[1:*], " '", /extract)
Foreach part, head Do Printf, lun, part

;Write structure tags and data.
Printf, lun, [['ref.left.tube_on'], $
```

```
   ['ref.left.tube_off'], $
   ['ref.right.tube_on'], $
   ['ref.right.tube_off']]
Printf, lun, Transpose([[oncal.ref.left.tube_on], $
   [oncal.ref.left.tube_off], $
   [oncal.ref.right.tube_on], $
   [oncal.ref.right.tube_off]]), $
   format = '(4' + rtype.format + ')'
Close, lun
Free_lun, lun

;Print some stuff.
Print, '>>>>> ... Done'
Print, ''

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 3: − Extract calibrator offset data and header.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some jazz
Print, '>>>>> Extracting reference data ...'
Print, ''

;Inialize variable(s).
advance = 0
icount = 0

;Count the total number of integrations (for all scans) to be extracted.
For jj=0, offscans.length−1 Do $
   icount = Temporary(icount) + offints[jj].length

;Initialize structure(s) and/or array(s).
proffcal = Replicate(Create_struct(name = 'scdata_s'), icount)
offcal = Create_struct(name = 'scdata_s')

;Begin loops for calibrator reference data import.
;Run this loop once for each scan called with offscans.
For master=0, offscans.length−1 Do Begin
   ;Get header and data for the offsource calibrator scans,
   ;All polarizations, frequency positions and calstates.
```

```
  numints = 0
  result = Extract(offscans[master], ifreq, numints, offhead, /resample)
  If numints Eq 0 Then Message, '>>>>> Error: Extraction failure'

  ;Print number of integrations
  Print, '>>>>> ' + Strtrim(result.length,2) + ' integration(s) retrieved'
  Print, ''

  ;Take the data out of 'result' and file it away in proffcal.
  Foreach pp, offints[master] Do Begin
    proffcal[advance] = result[pp]

    ;Print some jazz & Incriment the loop counter
    Print, '>>>>> Keeping integration ' + Strtrim(pp,2)
    ++ advance
  Endforeach
Endfor

;Print some stuff.
Print, '>>>>> Averaging ' + Strtrim(advance,2) + ' (of ' + $
  Strtrim(icount,2) + ' commanded) integrations'
Print, ''

;Average integrations and insert in to the new data structure.
If icount Eq 1 Then Begin
  offcal = proffcal
Endif Else Begin
  offcal.sig.left.tube_on = Mean(proffcal.sig.left.tube_on, dimension=2, /double)
  offcal.sig.left.tube_off = Mean(proffcal.sig.left.tube_off, dimension=2, /double)
  offcal.sig.right.tube_on = Mean(proffcal.sig.right.tube_on, dimension=2, /double)
  offcal.sig.right.tube_off = Mean(proffcal.sig.right.tube_off, dimension=2, /double)

  offcal.ref.left.tube_on = Mean(proffcal.ref.left.tube_on, dimension=2, /double)
  offcal.ref.left.tube_off = Mean(proffcal.ref.left.tube_off, dimension=2, /double)
  offcal.ref.right.tube_on = Mean(proffcal.ref.right.tube_on, dimension=2, /double)
  offcal.ref.right.tube_off = Mean(proffcal.ref.right.tube_off, dimension=2, /double)
Endelse

;+----------------------------------------+
;SECTION 4: - Create directories
```

```
; − Write calibrator offset data to ascii files.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some jazz
Print, '>>>>> Writing data to .gzip file ...'

;Create directory to write offsource calibration scan data to.
File_mkdir, Strtrim(rootpath, 2) + '/cal_offset', /noexpand_path

;Write signal phase data to an ascii datafile for the offsource
;calibration scans, noisetube on & off, both polarizations.
Openw, lun, Strtrim(rootpath, 2) + $
  '/cal_offset/sig_phase.dat.gzip', /get_lun, /compress
Printf, lun, Strtrim(Systime(), 2)
Printf, lun, [[icount], [N_tags(offcal_head.sig)]]

;Build & insert standard size header.
Help, offcal_head.sig, output = output
head = Strsplit(output[1:*], " ", /extract)
Foreach part, head Do Printf, lun, part

;Write structure tags and data.
Printf, lun, [['sig.left.tube_on'], $
  ['sig.left.tube_off'], $
  ['sig.right.tube_on'], $
  ['sig.right.tube_off']]
Printf, lun, Transpose([[offcal.sig.left.tube_on], $
  [offcal.sig.left.tube_off], $
  [offcal.sig.right.tube_on], $
  [offcal.sig.right.tube_off]]), $
  format = '(4' + rtype.format + ')'
Close, lun
Free_lun, lun

;Write reference phase data to an ascii datafile for the offsource
;calibration scans, noisetube on & off, both polarizations.
Openw, lun, Strtrim(rootpath, 2) + $
  '/cal_offset/ref_phase.dat.gzip', /get_lun, /compress
Printf, lun, Strtrim(Systime(), 2)
Printf, lun, [[icount], [N_tags(offcal_head.ref)]]
```

```
;build & insert standard size header.
Help, offcal_head.ref, output = output
head = Strsplit(output[1:*], " '", /extract)
Foreach part, head Do Printf, lun, part

;Write structure tags and data.
Printf, lun, [['ref.left.tube_on'], $
  ['ref.left.tube_off'], $
  ['ref.right.tube_on'], $
  ['ref.right.tube_off']]
Printf, lun, Transpose([[offcal.ref.left.tube_on], $
  [offcal.ref.left.tube_off], $
  [offcal.ref.right.tube_on], $
  [offcal.ref.right.tube_off]]), $
  format = '(4' + rtype.format + ')'
Close, lun
Free_lun, lun

;Print some stuff
Print, '>>>>> ... Done'
Print, ''
Print, '>>>>> All calibrator data has been extracted'
Print, ''

;If justcal is set, end here and dont write new source data.
If Keyword_set(justcal) Then Begin
  Print, '>>>>> Alert: Justcal keyword is set...'
  Print, '>>>>> No source data has been extracted'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif

;+-----------------------------------------+
;SECTION 5: - Extract sorce data and header.
; - Create directories
; - Write calibrator offset data to ascii files.
;+-----------------------------------------+
;Print some stuff.
Print, '+----------------------------------------+'
Print, ''
Print, '>>>>> Extracting source data ...'
Print, ''

;Create directory to write source scan data to.
File_mkdir, Strtrim(rootpath, 2) + '/source', /noexpand_path

;begin loops for source scan data import and ascii file export.
;Run this loop once for each scan called with srcscans.
For master=0, srcscans.length-1 Do Begin
  ;Get header and data for the source scans.
  ;All polarizations, frequency positions and calstates.
  numints = 0
  source = Extract(srcscans[master], ifreq, numints, shead, /resample)
  If numints Eq 0 Then Message, '>>>>> Error: Extraction failure'

  ;Print some stuff.
  Print, '>>>>> Writing scan ' + Strtrim(srcscans[master],2) $
    + ' data to .gzip file ...'

  ;Write signal phase data to ascii datafile for the source scans,
  ;noisetube on & off, both polarizations.
  Openw, lun, Strtrim(rootpath, 2) + '/source/sigphase_' + $
    Strtrim(master, 2) + '.dat.gzip', /get_lun, /compress
  Printf, lun, Strtrim(Systime(), 2)
  Printf, lun, [[numints], [N_tags(shead.sig)]]

  ;build & insert standard size header.
  Help, shead.sig, output = output
  head = Strsplit(output[1:*], " "", /extract)
  Foreach part, head Do Printf, lun, part

  ;Write structure tags and data.
  Printf, lun, [['sig.left.tube_on'], $
    ['sig.left.tube_off'], $
    ['sig.right.tube_on'], $
    ['sig.right.tube_off']]
  Printf, lun, Transpose([[source.sig.left.tube_on], $
    [source.sig.left.tube_off], $
    [source.sig.right.tube_on], $
    [source.sig.right.tube_off]]), $
    format = '(' + Strtrim(4*numints,2) + rtype.format + ')'
  Close, lun
  Free_lun, lun

  ;Write reference phase data to ascii datafile for the source scans,
  ;noisetube on & off, both polarizations.
  Openw, lun, Strtrim(rootpath, 2) + '/source/refphase_' + $
    Strtrim(master, 2) + '.dat.gzip', /get_lun, /compress
  Printf, lun, Strtrim(Systime(), 2)
  Printf, lun, [[numints], [N_tags(shead.ref)]]

  ;build & insert standard size header.
  Help, shead.ref, output = output
  head = Strsplit(output[1:*], " '", /extract)
  Foreach part, head Do Printf, lun, part

  ;Write structure tags and data.
  Printf, lun, [['ref.left.tube_on'], $
    ['ref.left.tube_off'], $
    ['ref.right.tube_on'], $
    ['ref.right.tube_off']]
  Printf, lun, Transpose([[source.ref.left.tube_on], $
    [source.ref.left.tube_off], $
    [source.ref.right.tube_on], $
    [source.ref.right.tube_off]]), $
    format = '(' + Strtrim(4*numints,2) + rtype.format + ')'
  Close, lun
  Free_lun, lun

  ;Print some stuff
  Print, '>>>>> ... Done'
  Print, ''
Endfor

;Print some stuff and end.
Print, '>>>>> All source data has been extracted'
```

```
  **Print**, ''
  **Print**, '>>>>> End Of Line <<<<<'
  **Print**, ''

**End**
```

14.5 Extract.pro

```
;+--------------------------------------------------+
;>>>>> Ancillary Program(s) <<<<<
;+--------------------------------------------------+

;+--------------------------------------------------+
;>>>>> Primary Program <<<<<
;+--------------------------------------------------+
;+
; NAME:
; Extract
; HYDRA Version 5.1
;
; PURPOSE:
; -> Retreives all data for a single 'FSW' scan from an .sdfits file.
;
; CALLING SEQUENCE:
; -> output = Extract_p(scan, ifreq, numints, header [, /Resample])
;
; ARGUMENT(S):
; -> Scan: Specifies the scan number in the .sdfits file for which data
; will be to be returned.
;
; -> Ifreq: Specifies the intermediate frequency of the data to be returned.
; * Acceptable values: '0', '1', '2', '3'.
;
; -> Numints: Set to pass back a long integer containing the total
; number of integrations in the scan.
;
; -> Header: A single 'schead' structure containing the header pulled
; from the scan in the .sdfits file. Set to pass back header
; information to the calling program.
;
; KEYWORD(S):
; -> None
;
; OPTIONAL KEYWORD(S):
; -> Resample: Binary. Resample the data from 65.6k to 8.2k pixels via
; bilinear interpolation. Requires each integration to be separately
; resampled in a loop, which may incur a time penalty.
;
; EXAMPLE(S):
; -> output = Extract_p(27, 2, integrations, scanhead, /resample)
;
; OUTPUT(S):
; -> A vector of 'scdata_m' structures. The length of the vector is
; determined by the number of integrations in the scan.
;
; COMMENTS:
; -> Operates only within the HYDRA 5.1 RTE.
; -> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro
;
; PROCEDURES/FUNCTIONS CALLED:
; -> scdata_l__define.pro
; -> scdata_m__define.pro
; -> schead__define.pro
;-

Function Extract, scan, ifreq, numints, header, RESAMPLE=resample
  Compile_opt IDL2
  Common overlord
  Common rtype
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 5 June, 2014
  ;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 12 October, 2014
  ;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 19 June, 2015
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 9 December, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 4.1 (V-spec) written by: N.N. Monson (UCLA). 17 September, 2017
  ;>>>>> Version 4.2 (V-spec) written by: N.N. Monson (UCLA). 30 May, 2018
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
```

```
;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 11 January, 2019

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Usage Agreement <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Developer's Notes <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Bugs <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Currently hardcoded to work ONLY with the receiver/spectrometer setup
;used to observe the J= 1−>0 transition of SiO. Using this program w/
;other setups will require some tweaks, namely, changing the array sizes
;(including arrays in structures), and the sampler values.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Inialize structure(s) and/or array(s).
keymaster = 0

;Ensure OVERLORD has already been created by getfits.pro.
;If not, then throw an error.
If ~ Isa(olord_summary) Then Begin
  Print, '>>>>> Error: No .sdfits file in memory'
  Print, '>>>>> Alert: Please load a .sdfits before calling Extract_p.pro'
  Print, '' & ++ keymaster
Endif Else Begin
  ;Use a proxy for olord_summary so nothing is inadvertently altered.
  sumprox = olord_summary
Endelse

;Ensure the SCAN argument is a scalar and
;is a longword integer, or can be converted to one.
;If SCAN is a funky type, e.g. unsigned, then throw an error.
If Array_equal(0, Isa(scan, /scalar)) || $
  Array_equal(1, rtype.adtypes.Contains(Typename(scan)), /not_equal) Then Begin
  Print, '>>>>> Alert: Scan argument dimension = '+Strtrim(scan.length,2)
  Print, '>>>>> Alert: Scan argument type = '+Strtrim(Typename(scan),2)
  Print, '>>>>> Error: Scan argument must be an integer or floating−point scalar'
  Print, '' & ++ keymaster
Endif Else Begin
  ;Otherwise, convert SCAN to a longword.
  scan = Long(scan)

  ;Check that SCAN is within the bounds of the data held in OVERLORD.
  If Array_equal(scan, sumprox.scan, /not_equal) Then Begin
    Print, '>>>>> Error: Scan argument out of range'
    Print, '' & ++ keymaster
  Endif
Endelse

;Ensure the IFREQ argument is a a scalar and
;is a longword integer, or can be converted to one.
;If IFREQ is a funky type, e.g. unsigned, then throw an error.
If Array_equal(0, Isa(ifreq, /scalar)) || $
  Array_equal(1, rtype.adtypes.Contains(Typename(ifreq)), /not_equal) Then Begin
  Print, '>>>>> Alert: Ifreq argument dimension = '+Strtrim(ifreq.length,2)
  Print, '>>>>> Alert: Ifreq argument type = '+Strtrim(Typename(ifreq),2)
  Print, '>>>>> Error: Ifreq argument must be an integer or floating−point scalar'
  Print, '' & ++ keymaster
Endif Else Begin
  ;Otherwise, convert IFREQ to a longword
  ifreq = Long(ifreq)

  ;Check that IFREQ is within the bounds of the data held in OVERLORD.
  ;NOTE: these values are specific to the q−band/VEGAS setup used to collect
  ;SiO data. These values will almost certainly be differentfor other setups!
  If Array_equal(ifreq, rtype.aifvals, /not_equal) Then Begin
```

```
    Print, '>>>>> Error: Ifreq argument out of range'
    Print, '' & ++ keymaster
  Endif
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
Endelse

;+---------------------------------------+
;SECTION 1: -Index data in OVERLORD.
; -Find & copy the indicated scan number.
;+---------------------------------------+

;NOTE: I adjusted how this works, previously I had put an extra
;value at the end of overlord.scanloc to accomodate the last value,
;but this new code should eliminate the need to do that.
;In essence, I'm doing the same thing, but I do it here
;instead of in getfits.pro.

;Locate target scan in the sdfdata structure.
spot = Where(scan Eq overlord.scannum)

;If SCAN is the last in the observation, use the total size of the
;sdfdata structure to define the upper bound of the 'scan' data,
;otherwise use the first entry of the subsequent scan.
If scan Eq overlord.scannum[-1] Then Begin
  endspot = overlord.sdfsize
Endif Else Begin
  endspot = overlord.scanloc[spot+1]
```

```
Endelse

;generate index for the locatinon of all the
;data for SCAN in sdfdata.
iindex = Lindgen(endspot − overlord.scanloc[spot], $
  start=overlord.scanloc[spot])

;Search newly indexed data array for blanked spectra by checking if
;the middle element in each data array (i.e. each integration) is finite.
;If the middle element is not finite, assume the array is blanked.
blindex = Where(Finite(olord_sdfdata[iindex].data[32768]) Ne 1)

;Check if there are any blanked integrations. If there are, remove them.
;Recall there are 8 data arrays per integration
;2 positions, 2 polarizations & 2 cal states)
If Array_equal(blindex, −1, /not_equal) Then Begin
  ;set BLANKEY, for use later.
  blankey = 1

  ;Im assuming the blank spectra is the last in the scan.
  ;NOTE: This likely isn't universally true, but it is for my data.
  ;Other users will need to modify this section of code accordingly.
  skindex = iindex[0:−((8 * Ceil(blindex.length / 8.0)) + 1)]
Endif Else Begin
  ;Otherwise, keep all of IINDEX.
  blankey = 0
  skindex = iindex
Endelse

;Final data index excluding blanked spectra.
;Use a proxy of oolord_sdfdata, so nothing is inadvertently altered.
datprox = olord_sdfdata[skindex]

;Calc. 'numints' keyword value to pass back to calling routine.
;Eight phases per integration: 2 polarizations, 2 cals, 2 frequency positions.
numints = skindex.length / 8d
If numints Mod 1.0 Ne 0.0 Then Begin
  Message, '>>>>> Error: Incorrect number of phases per integration'
Endif Else numints = Long(numints)
```

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 2: − Define 'SAMPLER' values.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Establish .sampler values for different polarizations and IF nums.
;These values were pulled from the .sdfits headers using GBTIDL.
;NOTE: these values are specific to the q−band/VEGAS setup used to
;collect SiO data. These values will almost certainly be different
;for other setups!
Case ifreq Of
  0:sam = ['D1_0', 'D2_0']
  1:sam = ['B1_0', 'B2_0']
  2:sam = ['C1_0', 'C2_0']
  3:sam = ['A1_0', 'A2_0']
Endcase

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 3: − Execute grid search for separate integrations (LL Pol)
; − File data into the appropriate structures.
; − Copy header to structure.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Inialize structure(s) and/or array(s).
master = Replicate(Create_struct(name = 'scdata_l'), numints)
header = Create_struct(name = 'schead')

;Search index and narrow to only lpol, sig position ints. Tube on.
master[*].sig.left.tube_on = datprox[Where((datprox.sig Eq 'T') And $
  (datprox.sampler Eq sam[0]) And (datprox.cal Eq 'T'))].data

;Re−search index and narrow to only lpol, sig position ints. Tube off.
master[*].sig.left.tube_off = datprox[Where((datprox.sig Eq 'T') And $
  (datprox.sampler Eq sam[0]) And(datprox.cal Eq 'F'))].data

;Search index and narrow to only rpol, sig position ints. Tube on.
master[*].sig.right.tube_on = datprox[Where((datprox.sig Eq 'T') And $
  (datprox.sampler Eq sam[1]) And (datprox.cal Eq 'T'))].data

;Re−search index and narrow to only rpol, sig position ints. Tube off.
master[*].sig.right.tube_off = datprox[Where((datprox.sig Eq 'T') And $
  (datprox.sampler Eq sam[1]) And(datprox.cal Eq 'F'))].data

;Inialize structure(s) and/or array(s).
scrapper = Lonarr(numints)

;Find one of the rpol, sig position integrations.
scrapper = Where((datprox.sig Eq 'T') And $
  (datprox.sampler Eq sam[1]) And(datprox.cal Eq 'F'))

;Put its header into a separate header structure.
Struct_assign, datprox[scrapper[0]], header.sig

;Calculate LSR frequency & convert elevation to radians
header.sig.lsrfreq = (((−1d ∗ header.sig.rvsys) / (!Const.c)) + 1d) ∗ header.sig.obsfreq
header.sig.elevatio = header.sig.elevatio ∗ (!Dpi / 180d)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 4: − Execute grid search for separate integrations (RR Pol)
; − File data into the appropriate structures.
; − Copy header to structure.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Search index and narrow to only lpol, ref position ints. Tube on.
master[∗].ref.left.tube_on = datprox[Where((datprox.sig Eq 'F') And $
  (datprox.sampler Eq sam[0]) And (datprox.cal Eq 'T'))].data

;Re−search index and narrow to only lpol, ref position ints. Tube off.
master[∗].ref.left.tube_off = datprox[Where((datprox.sig Eq 'F') And $
  (datprox.sampler Eq sam[0]) And (datprox.cal Eq 'F'))].data

;Search index and narrow to only rpol, ref position ints. Tube on.
master[∗].ref.right.tube_on = datprox[Where((datprox.sig Eq 'F') And $
  (datprox.sampler Eq sam[1]) And (datprox.cal Eq 'T'))].data

;Re−search index and narrow to only rpol, ref position ints. Tube off.
master[∗].ref.right.tube_off = datprox[Where((datprox.sig Eq 'F') And $
  (datprox.sampler Eq sam[1]) And (datprox.cal Eq 'F'))].data

;Inialize structure(s) and/or array(s).
crapper = Lonarr(numints)
```

```
;Find one of the rpol, ref position integrations.
crapper = Where((datprox.sig Eq 'F') And $
  (datprox.sampler Eq sam[1]) And (datprox.cal Eq 'F'))

;And put its header into a separate header structure.
Struct_assign, datprox[crapper[0]], header.ref

;Calculate LSR frequency & convert elevation to radians
header.ref.lsrfreq = (((−1d ∗ header.ref.rvsys) / (!Const.c)) + 1d) $
  ∗ header.ref.obsfreq
header.ref.elevatio = header.ref.elevatio ∗ (!Dpi / 180d)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 5: − Print information on data being returned.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some jazz
Print, ''
Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
Print, ''
Print, '>>>>> Source: ' + Strtrim(header.sig.object, 2)
Print, '>>>>> Accessing scan number ' + Strtrim(header.sig.scan,2) + ' ...'

;If blanked integrations were found, say so.
If Keyword_set(blankey) Then Begin
  Print, '>>>>> Warning: blanked spectra detected!'
  Print, '>>>>> Warning: ' + Strtrim(Ceil(blindex.length / 8.0), 2) + $
    ' integration(s) excluded'
Endif

;Print number of non−blank integrations being returned,
Print, '>>>>> Returning ' + Strtrim(numints,2) + ' integration(s) − all phases'

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 5: − Resample data to 8.2k pixels.
; − Check for any errors & return.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Inialize new variable(s)
```

```
;Inialize structure(s) and/or array(s).
blaster = Replicate(Create_struct(name = 'scdata_m'), numints)
gatekeeper = 0
cryptkeeper = 0

;Resample to 8.2k pixels using linear interpolation.
For ii=0, numints−1 Do Begin
  blaster[ii].sig.left.tube_on = Rebin(master[ii].sig.left.tube_on, rtype.med)
  blaster[ii].sig.left.tube_off = Rebin(master[ii].sig.left.tube_off, rtype.med)
  blaster[ii].sig.right.tube_on = Rebin(master[ii].sig.right.tube_on, rtype.med)
  blaster[ii].sig.right.tube_off = Rebin(master[ii].sig.right.tube_off, rtype.med)

  blaster[ii].ref.left.tube_on = Rebin(master[ii].ref.left.tube_on, rtype.med)
  blaster[ii].ref.left.tube_off = Rebin(master[ii].ref.left.tube_off, rtype.med)
  blaster[ii].ref.right.tube_on = Rebin(master[ii].ref.right.tube_on, rtype.med)
  blaster[ii].ref.right.tube_off = Rebin(master[ii].ref.right.tube_off, rtype.med)
Endfor

;Check to make sure no integrations have been missed or overwritten.
If numints Gt 1 Then Begin
  For ii=0, numints−2 Do Begin
    For jj=ii+1, numints−1 Do Begin
      If Array_equal(blaster[ii].sig.left.tube_on, blaster[jj].sig.left.tube_on) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].sig.left.tube_off, blaster[jj].sig.left.tube_off) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].sig.right.tube_on, blaster[jj].sig.right.tube_on) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].sig.right.tube_off, blaster[jj].sig.right.tube_off) Then $
        ++ gatekeeper

      If Array_equal(blaster[ii].ref.left.tube_on, blaster[jj].ref.left.tube_on) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].ref.left.tube_off, blaster[jj].ref.left.tube_off) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].ref.right.tube_on, blaster[jj].ref.right.tube_on) Then $
        ++ gatekeeper
      If Array_equal(blaster[ii].ref.right.tube_off, blaster[jj].ref.right.tube_off) Then $
        ++ gatekeeper
      If gatekeeper Ne 0 Then Message, '>>>>> Error: Data extraction failure'
    Endfor
  Endfor
Endif

;check each array for nasty little devil pixels!
;Devil pixels are defined as having a value greater than 3x the average of the spectrum.
;If found, they are replaced with the same value as a nearby, non-devil pixel.
;This is a crude way of going about this, but it it's good enough.
f1 = Where(blaster.sig.left.tube_on.Compare(3d * Mean(blaster.sig.left.tube_on)) Eq 1)
If Array_equal(f1, -1, /not_equal) Then $
  blaster.sig.left.tube_on[f1] = blaster.sig.left.tube_on[f1+10] & ++ cryptkeeper

f2 = Where(blaster.sig.left.tube_off.Compare(3d * Mean(blaster.sig.left.tube_off)) Eq 1)
If Array_equal(f2, -1, /not_equal) Then $
  blaster.sig.left.tube_off[f2] = blaster.sig.left.tube_off[f2+10] & ++ cryptkeeper

f3 = Where(blaster.sig.right.tube_on.Compare(3d * Mean(blaster.sig.right.tube_on)) Eq 1)
If Array_equal(f3, -1, /not_equal) Then $
  blaster.sig.right.tube_on[f3] = blaster.sig.right.tube_on[f3+10] & ++ cryptkeeper

f4 = Where(blaster.sig.right.tube_off.Compare(3d * Mean(blaster.sig.right.tube_off)) Eq 1)
If Array_equal(f4, -1, /not_equal) Then $
  blaster.sig.right.tube_off[f4] = blaster.sig.right.tube_off[f4+10] & ++ cryptkeeper

;+--------------------------------------+

f5 = Where(blaster.ref.left.tube_on.Compare(3d * Mean(blaster.ref.left.tube_on)) Eq 1)
If Array_equal(f5, -1, /not_equal) Then $
  blaster.ref.left.tube_on[f5] = blaster.ref.left.tube_on[f5+10] & ++ cryptkeeper

f6 = Where(blaster.ref.left.tube_off.Compare(3d * Mean(blaster.ref.left.tube_off)) Eq 1)
If Array_equal(f6, -1, /not_equal) Then $
  blaster.ref.left.tube_off[f6] = blaster.ref.left.tube_off[f6+10] & ++ cryptkeeper

f7 = Where(blaster.ref.right.tube_on.Compare(3d * Mean(blaster.ref.right.tube_on)) Eq 1)
If Array_equal(f7, -1, /not_equal) Then $
  blaster.ref.right.tube_on[f7] = blaster.ref.right.tube_on[f7+10] & ++ cryptkeeper

f8 = Where(blaster.ref.right.tube_off.Compare(3d * Mean(blaster.ref.right.tube_off)) Eq 1)
```

```
  If Array_equal(f8, −1, /not_equal) Then $
    blaster.ref.right.tube_off[f8] = blaster.ref.right.tube_off[f8+10] & ++ cryptkeeper

  ;Print some jazz
  If cryptkeeper Ne 0 Then Begin
    Print, '>>>>> Alert: Nasty little devil pixels found & replaced!'
  Endif Else Print, '>>>>> Alert: Rejoice! No nasty little devil pixels found!'

  ;return data structure. header is passed back via 'header' argument.
  Return, blaster

End
```

14.6 Dreamcatcher.pro

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME:
; Dreamcatcher
; HYDRA Version 5.1
;
; PURPOSE:
; −> Calibrates the noise diodes using scans of a standard Q−band
; flux calibrator, then reduces the frequency−switched science
; data and applies a vectorized temperature calibration calculated
; using scans of a standard Q−band flux calibrator.
;
; CALLING SEQUENCE:
; −> Dreamcatcher, sesnums, Iso=value [, Fitorder=integer, $
; Calfitorder=integer, /Rrl, /Plotcal, /Printcolor]
;
; ARGUMENT(S):
; −> Sesnums: An integer vector containing the observation number(s)
; to be reduced. These numbers are defined when calling gather.pro.
; If more than oneobservation is called, all data from all observations
; will be reduced independently and then averaged
;
; KEYWORD(S):
; −> Iso: The mass number of the isotope to be reduced and calibrated.
; ∗ Acceptable values: '28', '29', '30'.
;
; OPTIONAL KEYWORD(S):
; −> Plotcal (binary):Produces and saves plots of the vectorized
; calibration temperatures for each scan, polarization and position.
; ∗ note: Enabled by default, set to '0' to disable plotting.
;
```

```
; –> Fitorder: Set to specify the polynomial order used when fitting the
; vectorized system temperatures.
; * Default: 4
; * Acceptable values: any positive integer < 8
;
; –> Calfitorder: Set to specify the polynomial order used when fitting the
; vectorized calibration temperatures.
; * Default: 3
; * Acceptable values: any positive integer < 8
;
; –> Printcolor (binary): Changes the color scheme of the plots produced.
;
; EXAMPLES:
; –> Dreamcatcher, [0,1], Iso=28, Fitorder=3, /Plotcal, /Printcolor
;
; OUTPUTS:
; –> A single compressed data file named "calibrated_**sio.dat.gzip"
; containing the reduced and calibrated science data in the
; directory "**sio/" wher ** is the mass number of the
; isotope set with the 'ISO' keyword.
;
; COMMENTS:
; –> Only operates within the HYDRA 5.1 RTE.
; –> Filepath for the output directory is stored in OVERLORD and must
; be set using setdir.pro
;  –> Currently only calibrates data from 3C286, 3C48 & 3C147
;
; PROCEDURES/FUNCTIONS CALLED:
; –> scdata_m__define.pro
; –> schead__define.pro
; –> redata_s__define.pro
; –> redata_m__define.pro
;–

Pro Dreamcatcher, sesnums, Iso=iso, Rrl=rrl, Plotcal=plotcal, $
  Printcolor=printcolor, Fitorder=fitorder, Calfitorder=calfitorder
  Compile_opt idl2
  Common overlord
  Common rtype
  Common pspex
```

```
On_error, 0
!Except = 1

;Hydra version 5.1
;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 5 June, 2014
;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 12 October, 2014
;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 19 June, 2015
;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 9 December, 2016
;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
;>>>>> Version 4.1 (V-spec) written by: N.N. Monson (UCLA). 17 September, 2017
;>>>>> Version 4.2 (V-spec) written by: N.N. Monson (UCLA). 30 May, 2018
;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 December, 2018
;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 February, 2019

;+----------------------------------------+
;>>>>> Usage Agreement <<<<<
;+----------------------------------------+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+----------------------------------------+
;>>>>> Developer's Notes <<<<<
;+----------------------------------------+

;+----------------------------------------+
;>>>>> Limitations & Known Bugs <<<<<
;+----------------------------------------+

;Currently hardcoded to work ONLY with the receiver/spectrometer setup
;used to observe the J= 1->0 transition of SiO. Using this program w/
;other setups will require some tweaks, namely, changing the array sizes.

;+----------------------------------------+
;SECTION 0: - Check argument(s).
```

```
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
Print, ''

;Ensure input/output filepath had been set, and is in stored
;in OVERLORD. If not, then throw an error.
If Array_equal('', overlord.dirloc) Then Begin
  Print, '>>>>> Error: Common directory not set'
  Print, '>>>>> Set common directory using Setdir.pro'
  Print, '' & ++ keymaster
Endif

;Inialize structure(s) and/or array(s).
keymaster = 0
callsign = ''

;Check if the /RRL keyword is set.
If Keyword_set(rrl) Then Begin

  ;If it is, check to see if the /RRL keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If rrl Ne 1 Then Begin
    Print, '>>>>> Alert: Rrl is a binary keyword'
    Print, '>>>>> Setting rrl accordingly ...'
    Print, '' & rrl = 1
  Endif

  ;Ignore iso keyword if RRL keyword is set
  If Keyword_set(iso) Then Begin
    Print, '>>>>> Alert: Rrl keyword set, ignoring iso keyword'
    Print, ''
  Endif

  ;set the callsign, file extension and ifreq values.
  callsign = 'rrl'
Endif Else Begin
  ;Otherwise, Check if the ISO keyword is set
  If Keyword_set(iso) Then Begin
```

```
    ;Ensure the ISO keyword is a scalar and
    ;is a longword integer, or can be converted to one.
    ;If ISO is a funky type, e.g. unsigned, then throw an error.
    If Array_equal(0, Isa(iso, /scalar)) || $
      Array_equal(1, rtype.adtypes.Contains(Typename(iso)), /not_equal) Then Begin
      Print, '>>>>> Alert: Iso keyword dimension = '+Strtrim(iso.length,2)
      Print, '>>>>> Alert: Iso keyword type = '+Strtrim(Typename(iso),2)
      Print, '>>>>> Error: Iso Keyword must be a longword or floating−point scalar'
      Print, '' & ++ keymaster
    Endif Else Begin
      ;Otherwise, convert ISO to a longword
      iso = Long(iso)

      ;And check to see if ISO is in range. If not, then throw an error.
      If Array_equal(iso, rtype.aivals, /not_equal) Then Begin
        Print, '>>>>> Error: Iso keyword out of range'
        Print, '' & ++ keymaster
      Endif

      ;set the callsign
      callsign = Strtrim(iso,2) + 'sio'
    Endelse
  Endif Else Begin
    ;Otherwise, throw an error.
    Print, '>>>>> Error: Iso keyword is not set'
    Print, '' & ++ keymaster
  Endelse
Endelse

;Check to see if FITORDER keyword has been used
If Keyword_set(fitorder) Then Begin

  ;Ensure the FITORDER keyword is a scalar and
  ;is a longword integer, or can be converted to one.
  ;If FITORDER is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(fitorder, /scalar)) || $
    Array_equal(1, rtype.adtypes.Contains(Typename(fitorder)), /not_equal) Then Begin
    Print, '>>>>> Alert: Fitorder keyword dimension = '+Strtrim(fitorder.length,2)
    Print, '>>>>> Alert: Fitorder keyword type = '+Strtrim(Typename(fitorder),2)
    Print, '>>>>> Error: Fitorder Keyword must be a longword or floating−point scalar'
    Print, '' & ++ keymaster
  Endif Else Begin
    ;Otherwise, convert FITORDER to a longword
    fitorder = Long(fitorder)

    ;And check to see if FITORDER is in range. If not, then throw an error.
    If (fitorder Lt 1) Or (fitorder Gt 7) Then Begin
      Print, '>>>>> Error: Fitorder keyword out of range'
      Print, '' & ++ keymaster
    Endif
  Endelse
  ;Otherwise, set default value
Endif Else fitorder = 4

;Check to see if CALFITORDER keyword has been used
If Keyword_set(calfitorder) Then Begin

  ;Ensure the CALFITORDER keyword is a scalar and
  ;is a longword integer, or can be converted to one.
  ;If CALFITORDER is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(calfitorder, /scalar)) || $
    Array_equal(1, rtype.adtypes.Contains(Typename(calfitorder)), /not_equal) Then Begin
    Print, '>>>>> Alert: Calfitorder keyword dimension = '+Strtrim(calfitorder.length,2)
    Print, '>>>>> Alert: Calfitorder keyword type = '+Strtrim(Typename(calfitorder),2)
    Print, '>>>>> Error: Calfitorder Keyword must be a longword or floating−point scalar'
    Print, '' & ++ keymaster
  Endif Else Begin
    ;Otherwise, convert CALFITORDER to a longword
    calfitorder = Long(calfitorder)

    ;And check to see if CALFITORDER is in range. If not, then throw an error.
    If (calfitorder Lt 1) Or (calfitorder Gt 7) Then Begin
      Print, '>>>>> Error: Calfitorder keyword out of range'
      Print, '' & ++ keymaster
    Endif
  Endelse
  ;Otherwise, set default value
Endif Else fitorder = 3
```

```
;Check if the /PLOTCAL keyword is set.
If Keyword_set(plotcal) Then Begin
  ;If it is, check to see if the /PLOTCAL keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If plotcal Ne 1 Then Begin
    Print, '>>>>> Alert: Plotcal is a binary keyword'
    Print, '>>>>> Setting plotcal accordingly ...'
    Print, '' & plotcal = 1
  Endif
Endif

;Set colors to use when plotting. Inverted colors used by default.
If Keyword_set(printcolor) Then Begin
  pcolors = pspex.rcolors
Endif Else pcolors = pspex.icolors

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
Endelse

;+------------------------------------------+
;SECTION 1: - Create required directories.
; - Initialize structures for loops.
;+------------------------------------------+

;Close any open graphics windows
Purge

;Create new directory to write calibrated data to.
newdir = Strtrim(Overlord.dirloc, 2) + '/' + Strtrim(callsign, 2)
```

```
File_mkdir, newdir, /noexpand_path

;identify number of observations being calibrated
nsessions = sesnums.length

;Initialize structure(s) and/or array(s) for use later in the 'Kappa' loop.
obs_wtmb = Replicate(Create_struct(name = 'redata_s'), nsessions)
obs_wtsfit = Replicate(Create_struct(name = 'redata_s'), nsessions)
obs_twt = Replicate(Create_struct(name = 'redata_s'), nsessions)
keeper = Findgen(rtype.sml)
bigkeeper = Findgen(rtype.med)

;Use 'Kappa' to keep track of which observation is being reduced.
For kappa=0, nsessions−1 Do Begin

  rootpath = Strtrim(Overlord.dirloc, 2) + '/session_0' + $
    Strtrim(sesnums[kappa],2) + '/' + Strtrim(callsign, 2)

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 2: − Import Calibrator Data
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Print some jazz
  Print, ''
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, '>>>>> Importing calibrator data ...'
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, ''
  Print, '>>>>> Indexing onsource spectra ...'

  ;Import data for the on−source calibrator scan as written by Gather.
  ;Repeat twice, once for the reference phase, and once for the signal phase.
  Foreach phase, ['sig', 'ref'] Do Begin
    Openr, lun, Strtrim(rootpath, 2) + '/calibrator/' + $
      Strtrim(phase, 2) + '_phase.dat.gzip', /get_lun, /compress
    time = ''
    Readf, lun, time
    ncalints = 0
    Readf, lun, ncalints
    numtags = 0
```

```
Readf, lun, numtags
header = Strarr(numtags)
Readf, lun, header
tnames = Strarr(4)
Readf, lun, tnames
data = Dblarr(4, rtype.med, /nozero)
Readf, lun, data, format = '(4' + rtype.format + ')'
Close, lun
Free_lun, lun

;Transpose data array
data = Transpose(data)

;Split the header into a string array
header = Strsplit(header, ' ', /extract)

;Make structures to put all onsource calibrator scan data into
;Do the same for the onsource calibrator scan header
If PHASE Eq 'sig' Then Begin
  oncal = Create_struct(name = 'scdata_m')
  oncal_head = Create_struct(name = 'schead')
Endif

;Put the header into it's structure & check for errors.
;Using EXECUTE is a little clunky, but i see no other option
;short of explicitly listing everything.
For i=0, (numtags−1) Do Begin
  exe_chk = Execute('oncal_head.' + Strtrim(phase, 2) + '.' + $
    Strtrim(header[i, 0],2) + ' = header[i, 2]')
  If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
Endfor

;Put the data into it's structure & check for errors.
;Using EXECUTE is a little clunky, but i see no other option
;short of explicitly listing everything.
For i=0, 3 Do Begin
  exe_chk = Execute('oncal.' + Strtrim(tnames[i],2) + ' = data[*, i]')
  If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
Endfor
```

```
Endforeach

;Print some jazz
Print, '>>>>> ...Done'

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 3: − Import Calibrator Offset Data
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some jazz
Print, ''
Print, '>>>>> Indexing offsource spectra ...'

;Import data for the off−source calibrator scan as written by Gather.
;Repeat twice, once for the reference phase, and once for the signal phase.
Foreach phase, ['sig', 'ref'] Do Begin
  Openr, lun, Strtrim(rootpath, 2) + '/cal_offset/' + $
    Strtrim(phase, 2) + '_phase.dat.gzip', /get_lun, /compress
  time = ''
  Readf, lun, time
  ncalints = 0
  Readf, lun, ncalints
  numtags = 0
  Readf, lun, numtags
  header = Strarr(numtags)
  Readf, lun, header
  tnames = Strarr(4)
  Readf, lun, tnames
  data = Dblarr(4, rtype.med, /nozero)
  Readf, lun, data, format = '(4' + rtype.format + ')'
  Close, lun
  Free_lun, lun

  ;Transpose data array
  data = Transpose(data)

  ;Split the header into a string array
  header = Strsplit(header, ' ', /extract)

  ;Make structures to put all offsource calibrator scan data into
  ;Do the same for the offsource calibrator scan header
  If PHASE Eq 'sig' Then Begin
    offcal = Create_struct(name = 'scdata_m')
    offcal_head = Create_struct(name = 'schead')
  Endif

  ;Put the header into it's structure & check for errors.
  ;Using EXECUTE is a little clunky, but i see no other option
  ;short of explicitly listing everything.
  For i=0, (numtags−1) Do Begin
    exe_chk = Execute('offcal_head.' + Strtrim(phase, 2) + '.' + $
      Strtrim(header[i, 0],2) + ' = header[i, 2]')
    If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
  Endfor

  ;Put the data into it's structure & check for errors.
  ;Using EXECUTE is a little clunky, but i see no other option
  ;short of explicitly listing everything.
  For i=0, 3 Do Begin
    exe_chk = Execute('offcal.' + Strtrim(tnames[i],2) + ' = data[*, i]')
    If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
  Endfor
Endforeach

;Print some jazz
Print, '>>>>> ...Done'
Print, ''
Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 4: − Calculate Calibrator Temp
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize structure(s).
vrange = Create_struct(name = 'sigref_m')
flux = Create_struct(name = 'sigref_m')
tau = Create_struct(name = 'sigref_m')
eta = Create_struct(name = 'sigref_m')
tsrc = Create_struct(name = 'sigref_m')
```

```
;Calculate the frequency range covered by the spectral windows
;for both frequency phases.
vrange.sig = [oncal_head.sig.lsrfreq − (oncal_head.sig.bandwid * 0.6d) : $
  oncal_head.sig.lsrfreq + (oncal_head.sig.bandwid * 0.4d) : $
  (oncal_head.sig.bandwid) / rtype.med] / 1d9

vrange.ref = [oncal_head.ref.lsrfreq − (oncal_head.ref.bandwid * 0.4d) : $
  oncal_head.ref.lsrfreq + (oncal_head.ref.bandwid * 0.6d) : $
  (oncal_head.ref.bandwid) / rtype.med] / 1d9

;Calculate calibrator flux.
Case oncal_head.sig.object Of
  '3C286': Begin
    ;3c286 flux density as function of frequency: via Perley & Butler, 2017.
    flux.sig = 10.0d ^ ((1.2481d) + $
      (−0.4507d) * Alog10(vrange.sig) + $
      (−0.1798d) * (Alog10(vrange.sig) ^ 2d) + $
      (0.0357d) * (Alog10(vrange.sig) ^ 3d))

    flux.ref = 10.0d ^ ((1.2481d) + $
      (−0.4507d) * Alog10(vrange.ref) + $
      (−0.1798d) * (Alog10(vrange.ref) ^ 2d) + $
      (0.0357d) * (Alog10(vrange.ref) ^ 3d))
  End
  '3C48': Begin
    ;3c48 flux density as function of frequency: via Perley & Butler, 2017.
    flux.sig = 10.0d ^ ((1.3253d) + $
      (−0.7553d) * Alog10(vrange.sig) + $
      (−0.1914d) * (Alog10(vrange.sig) ^ 2d) + $
      (0.0498d) * (Alog10(vrange.sig) ^ 3d))

    flux.ref = 10.0d ^ ((1.3253d) + $
      (−0.7553d) * Alog10(vrange.ref) + $
      (−0.1914d) * (Alog10(vrange.ref) ^ 2d) + $
      (0.0498d) * (Alog10(vrange.ref) ^ 3d))
  End
  '3C147': Begin
    ;3c147 flux density as function of frequency: via Perley & Butler, 2017.
    flux.sig = 10.0d ^ ((1.4516d) + $
      (−0.6961d) * Alog10(vrange.sig) + $
      (−0.2007d) ∗ (Alog10(vrange.sig) ˆ 2d) + $
      (0.0640d) ∗ (Alog10(vrange.sig) ˆ 3d) + $
      (−0.0464d) ∗ (Alog10(vrange.sig) ˆ 4d) + $
      (0.0289d) ∗ (Alog10(vrange.sig) ˆ 5d))

    flux.ref = 10.0d ˆ ((1.4516d) + $
      (−0.6961d) ∗ Alog10(vrange.ref) + $
      (−0.2007d) ∗ (Alog10(vrange.ref) ˆ 2d) + $
      (0.0640d) ∗ (Alog10(vrange.ref) ˆ 3d) + $
      (−0.0464d) ∗ (Alog10(vrange.ref) ˆ 4d) + $
      (0.0289d) ∗ (Alog10(vrange.ref) ˆ 5d))
  End
Endcase

;Caluclate aperture efficiency and zenith opacity − EXCISED FROM GBTIDL CODE
;Tau is a function of frequency and is ˜ 0.095 for SiO J = 1−0. Eta is ˜ 0.6
;Both calculations are vectorized, although I doubt it matters much
tau.sig = 0.008d + Exp(Sqrt(vrange.sig)) / (8000.0d)
tau.ref = 0.008d + Exp(Sqrt(vrange.ref)) / (8000.0d)

eta.sig = 0.71d ∗ Exp( −((4d ∗ !Dpi ∗ (vrange.sig ∗ 1d9) ∗ 3.9d−4) / (!Const.c)) ˆ 2d)
eta.ref = 0.71d ∗ Exp( −((4d ∗ !Dpi ∗ (vrange.ref ∗ 1d9) ∗ 3.9d−4) / (!Const.c)) ˆ 2d)

;Average elevation data from headers.
;It should all be the same anyway.
elev = Mean([[oncal_head.sig.elevatio], $
  [oncal_head.ref.elevatio],$
  [offcal_head.sig.elevatio], $
  [offcal_head.ref.elevatio]])

;Calculate source temperature
;I dug this out of the NRAO's GBTIDL calibration guide, It should
;be verifible, although I haven't actually double checked this.
tsrc.sig = 2.84d ∗ flux.sig ∗ eta.sig ∗ Exp(−1d ∗ (tau.sig / Sin(elev)))
tsrc.ref = 2.84d ∗ flux.ref ∗ eta.ref ∗ Exp(−1d ∗ (tau.ref / Sin(elev)))

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 5: − Calculate noise tube calibration temperatures.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;Initialize structure(s).
cal = Create_struct(name = 'redata_m')
tcal = Create_struct(name = 'redata_m')

;Calculate vector T_cal & smooth, sig phase, LL pol.
;Use some gaussian smoothing to keep noise down.
cal.sig.left = ((offcal.sig.left.tube_on − offcal.sig.left.tube_off) + $
  (oncal.sig.left.tube_on − oncal.sig.left.tube_off)) / $
  (Gauss_smooth((oncal.sig.left.tube_off − offcal.sig.left.tube_off) + $
  (oncal.sig.left.tube_on − offcal.sig.left.tube_on), 3d, /edge_mirror))

;Fit low order polynomial to smoothed T_cal, sig phase, LL pol.
;keyword calfitorder controls polynomial order. dafault is 3.
result = Poly_fit(bigkeeper[200:−201], cal.sig.left[200:−201], calfitorder, /double)
tcal.sig.left = Poly(bigkeeper, result) ∗ tsrc.sig


;Calculate vector T_cal & smooth, ref phase, LL pol.
;Use some gaussian smoothing to keep noise down.
cal.ref.left = ((offcal.ref.left.tube_on − offcal.ref.left.tube_off) + $
  (oncal.ref.left.tube_on − oncal.ref.left.tube_off)) / $
  (Gauss_smooth((oncal.ref.left.tube_off − offcal.ref.left.tube_off) + $
  (oncal.ref.left.tube_on − offcal.ref.left.tube_on), 3d, /edge_mirror))

;Fit low order polynomial to smoothed T_cal, ref phase, LL pol.
;keyword calfitorder controls polynomial order. dafault is 3.
result = Poly_fit(bigkeeper[200:−201], cal.ref.left[200:−201], calfitorder, /double)
tcal.ref.left = Poly(bigkeeper, result) ∗ tsrc.ref

;Print mean T_cal values, both phases, LL pol.
Print, ''
Print, format = '(">>>>> mean T_cal (Sig − LL Pol) = ", f−15.2)', $
  Mean(tcal.sig.left[277:6276])
Print, format = '(">>>>> mean T_cal (Ref − LL Pol) = ", f−15.2)', $
  Mean(tcal.ref.left[1915:7914])

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Calculate vector T_cal & smooth, sig phase, RR pol.
;Use some gaussian smoothing to keep noise down.
```

```
cal.sig.right = ((offcal.sig.right.tube_on − offcal.sig.right.tube_off) + $
  (oncal.sig.right.tube_on − oncal.sig.right.tube_off)) / $
  (Gauss_smooth((oncal.sig.right.tube_off − offcal.sig.right.tube_off) + $
  (oncal.sig.right.tube_on − offcal.sig.right.tube_on), 3d, /edge_mirror))

;Fit low order polynomial to smoothed T_cal, sig phase, RR pol.
;keyword calfitorder controls polynomial order. dafault is 3.
result = Poly_fit(bigkeeper[200:−201], cal.sig.right[200:−201], calfitorder, /double)
tcal.sig.right = Poly(bigkeeper, result) ∗ tsrc.sig


;Calculate vector T_cal & smooth, ref phase, RR pol.
;Use some gaussian smoothing to keep noise down.
cal.ref.right = ((offcal.ref.right.tube_on − offcal.ref.right.tube_off) + $
  (oncal.ref.right.tube_on − oncal.ref.right.tube_off)) / $
  (Gauss_smooth((oncal.ref.right.tube_off − offcal.ref.right.tube_off) + $
  (oncal.ref.right.tube_on − offcal.ref.right.tube_on), 3d, /edge_mirror))

;Fit low order polynomial to smoothed T_cal, ref phase, RR pol.
;keyword calfitorder controls polynomial order. dafault is 3.
result = Poly_fit(bigkeeper[200:−201], cal.ref.right[200:−201], calfitorder, /double)
tcal.ref.right = Poly(bigkeeper, result) ∗ tsrc.ref

;Print mean T_cal values, both phases, RR pol.
Print, ''
Print, format = '(">>>>> mean T_cal (Sig − RR Pol) = ", f−15.2)', $
  Mean(tcal.sig.right[277:6276])
Print, format = '(">>>>> mean T_cal (Ref − RR Pol) = ", f−15.2)', $
  Mean(tcal.ref.right[1915:7914])

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 5b: − Plot the noise tube calibration data.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

If Keyword_set(plotcal) Then Begin

  ;Close any open graphics windows
  Purge

  ;Sig phase, raw data
```

```
;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot the R & L polarizations together & save as a .png.
;remember the 'sig' and 'ref' indicies are [277:6276] & [1915:7914].
pcs_sl = Plot(offcal.sig.left.tube_off[277:6276], xrange=[0,6d3], xstyle=1, xminor=3, $
  xmajor=7, ymajor=5, yminor=1, title='Sig − LL Pol', ytitle='Counts', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,1], /current)
pcs_sl = Plot(offcal.sig.left.tube_on[277:6276], color=pcolors.pink, layout=[1,2,1], $
  /current, /overplot)
pcs_sl = Plot(oncal.sig.left.tube_off[277:6276], color=pcolors.teal, layout=[1,2,1], $
  /current, /overplot)
pcs_sl = Plot(oncal.sig.left.tube_on[277:6276], color=pcolors.orange, layout=[1,2,1], $
  /current, /overplot)
ax = pcs_sl.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

pcs_sr = Plot(offcal.sig.right.tube_off[277:6276], xrange=[0,6d3], xstyle=1, xminor=3, $
  xmajor=7, ymajor=5, yminor=1, title='Sig − RR Pol', ytitle='Counts', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,2], /current)
pcs_sr = Plot(offcal.sig.right.tube_on[277:6276], color=pcolors.pink, layout=[1,2,2], $
  /current, /overplot)
pcs_sr = Plot(oncal.sig.right.tube_off[277:6276], color=pcolors.teal, layout=[1,2,2], $
  /current, /overplot)
pcs_sr = Plot(oncal.sig.right.tube_on[277:6276], color=pcolors.orange, layout=[1,2,2], $
  /current, /overplot)
ax = pcs_sr.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Save graphic as .png file.
pic_name = Strtrim(newdir,2) + '/raw_' + Strtrim(callsign, 2) + '_sig_run'+ $
  Strtrim(kappa,2) + '.png'
w.Save, Strtrim(pic_name,2), resolution = 100
```

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Ref phase, raw data
;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot the R & L polarizations together & save as a .png.
;remember the 'sig' and 'ref' indicies are [277:6276] & [1915:7914].
pcs_rl = Plot(offcal.ref.left.tube_off[1915:7914], xrange=[0,6d3], xstyle=1, xminor=3, $
  xmajor=7, ymajor=5, yminor=1, title='Ref − LL Pol', ytitle='Counts', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,1], /current)
pcs_rl = Plot(offcal.ref.left.tube_on[1915:7914], color=pcolors.pink, layout=[1,2,1], $
  /current, /overplot)
pcs_rl = Plot(oncal.ref.left.tube_off[1915:7914], color=pcolors.teal, layout=[1,2,1], $
  /current, /overplot)
pcs_rl = Plot(oncal.ref.left.tube_on[1915:7914], color=pcolors.orange, layout=[1,2,1], $
  /current, /overplot)
ax = pcs_rl.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

pcs_rr = Plot(offcal.ref.right.tube_off[1915:7914], xrange=[0,6d3], xstyle=1, xminor=3, $
  xmajor=7, ymajor=5, yminor=1, title='Ref − RR Pol', ytitle='Counts', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,2], /current)
pcs_rr = Plot(offcal.ref.right.tube_on[1915:7914], color=pcolors.pink, layout=[1,2,2], $
  /current, /overplot)
pcs_rr = Plot(oncal.ref.right.tube_off[1915:7914], color=pcolors.teal, layout=[1,2,2], $
  /current, /overplot)
pcs_rr = Plot(oncal.ref.right.tube_on[1915:7914], color=pcolors.orange, layout=[1,2,2], $
  /current, /overplot)
ax = pcs_rr.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Save graphic as .png file.
pic_name = Strtrim(newdir,2) + '/raw_' + Strtrim(callsign, 2) + '_ref_run'+ $
```

```
  Strtrim(kappa,2) + '.png'
 w.Save, Strtrim(pic_name,2), resolution = 100

 ;+--------------------------------------+

 ;Sig & Ref phases, calibration temps
 ;Initialize a graphics window w/ standard dimensions.
 w = Window(dimensions=pspex.windim, background_color=pcolors.black)

 ;Plot the calculated calibration temps together & save as a .png.
 ;remember the 'sig' and 'ref' indicies are [277:6276] & [1915:7914].
 ptc_sb = Plot(tcal.sig.left[277:6276], color=pcolors.puke, thick=2, xrange=[0,6d3], $
  xstyle=1, xminor=3, xmajor=7, yminor=1, title='Sig Position', ytitle='T_cal (K)', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,1], /current)
 ptc_sb = Plot(tcal.sig.right[277:6276], color=pcolors.pink, thick=2, layout=[1,2,1], $
  /current, /overplot)
 ax = ptc_sb.axes
 ax[2].minor = 0.0d
 ax[2].ticklen = ax[2].ticklen / 2.0d
 ax[3].minor = 0.0d
 ax[3].ticklen = ax[3].ticklen / 2.0d

 ptc_rb = Plot(tcal.ref.left[1915:7914], color=pcolors.puke, thick= 2, xrange=[0,6d3], $
  xstyle=1, xminor=3, xmajor=7, yminor=1, title='Ref Position', ytitle='T_cal (K)', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,2], /current)
 ptc_rb = Plot(tcal.ref.right[1915:7914], color=pcolors.pink, thick= 2, layout=[1,2,2], $
  /current, /overplot)
 ax = ptc_rb.axes
 ax[2].minor = 0.0d
 ax[2].ticklen = ax[2].ticklen / 2.0d
 ax[3].minor = 0.0d
 ax[3].ticklen = ax[3].ticklen / 2.0d

 ;Save graphic as .png file.
 pic_name = Strtrim(newdir,2) + '/tcal_' + Strtrim(callsign, 2) + '_run' + $
  Strtrim(kappa,2) + '.png'
 w.Save, Strtrim(pic_name,2), resolution = 100
Endif

;+--------------------------------------+
;SECTION 6: − Import Science Data
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;determine number of scans in the directory to be calibrated
com_chk = File_search(Strtrim(rootpath, 2) + '/source/sigphase_*', count = numscans)

;Initialize structure(s) for use later in 'Master' loop.
scan_wtmb = Replicate(Create_struct(name = 'redata_s'), numscans)
scan_wtsfit = Replicate(Create_struct(name = 'redata_s'), numscans)
scan_twt = Replicate(Create_struct(name = 'redata_s'), numscans)

;Loop once for each 10 min scan.
;Use 'Master' to count the number of loops
For master=0, (numscans−1) Do Begin
  If master Eq 0 Then Begin
    Print, ''
    Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
    Print, '>>>>> source data import: initializing ...'
    Print, ''
  Endif
  Print, '>>>>> message: importing file ' + Strtrim(master+1, 2) + ' of ' + $
    Strtrim(numscans,2) + ' ...'

  ;Import data for the on−source scans as written by Gather.
  ;Repeat twice, once for the reference phase, and once for the signal phase.
  Foreach phase, ['sig', 'ref'] Do Begin
    Openr, lun, Strtrim(rootpath, 2) + '/source/' + Strtrim(phase, 2) + $
      'phase_' + Strtrim(master, 2) + '.dat.gzip', /get_lun, /compress
    time = ''
    Readf, lun, time
    numints = 0
    Readf, lun, numints
    numtags = 0
    Readf, lun, numtags
    header = Strarr(numtags)
    Readf, lun, header
    tnames = Strarr(4)
    Readf, lun, tnames
    data = Dblarr(4*numints, rtype.med)
    Readf, lun, data, format = '(' + Strtrim(4*numints, 2) + rtype.format + ')'
    Close, lun
    Free_lun, lun

    ;Transpose data array
    data = Transpose(data)

    ;Split the header into a string array
    header = Strsplit(header, ' ', /extract)

    ;Make a structure for the scan data in this loop, and
    ;make another for the scan headers in this loop.
    If phase Eq 'sig' Then Begin
      onsource = Replicate(Create_struct(name = 'scdata_m'), numints)
      source_head = Create_struct(name = 'schead')
    Endif

    ;Put header into it's structure & check for errors.
    ;Using EXECUTE is a little clunky, but i see no other option
    ;short of explicitly listing everything.
    For i=0, (numtags−1) Do Begin
      exe_chk = Execute('source_head.' + Strtrim(phase, 2) + '.' + $
        Strtrim(header[i, 0],2) + ' = header[i, 2]')
      If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
    Endfor

    ;Put data into it's structure array & check for errors.
    ;Using EXECUTE is a little clunky, but i see no other option
    ;short of explicitly listing everything.
    For i=0, 3 Do Begin
      exe_chk = Execute('onsource.' + Strtrim(tnames[i],2) + $
        ' = data[*, (i*numints):((i*numints)+(numints−1))]')
      If exe_chk Ne 1 Then Message, '>>>>> Error: Execution failure'
    Endfor
  Endforeach

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 7: − Calculate Vectorized Tsys.
  ;      − both frequency phases, LL pol.
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;Initialize structure(s) and/or array(s)
  proxy = Replicate(Create_struct(name = 'redata_m'), numints)
  tsys = Replicate(Create_struct(name = 'redata_m'), numints)
  tsfit = Replicate(Create_struct(name = 'redata_s'), numints)
  ta = Replicate(Create_struct(name = 'redata_s'), numints)

  ;Generate array of random RGB colors to use when plotting.
  randomclrs = Randomcolors(numints−2)

  ;Initialize a graphics window w/ NONstandard dimensions.
  w = Window(dimensions = pspex.windim ∗ [0.80,1.5], background_color=pcolors.black)

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Initialize structure(s) and/or array(s)
  osleft = Replicate(Create_struct(name = 'sigref_m'), numints)
  tcleft_rb = Replicate(Create_struct(name = 'sigref_m'), numints)

  ;Rebin LL pol calibration temps.
  tcleft_rb.sig = Rebin(tcal.sig.left, rtype.med, numints, /sample)
  tcleft_rb.ref = Rebin(tcal.ref.left, rtype.med, numints, /sample)

  ;Calculate the tube on/off average for both phases
  osleft.sig = (onsource.sig.left.tube_on + onsource.sig.left.tube_off) / 2.0d
  osleft.ref = (onsource.ref.left.tube_on + onsource.ref.left.tube_off) / 2.0d

  ;Calculate vectorized & system temp for the on−source scans.
  tsys.sig.left = (osleft.sig ∗ tcleft_rb.sig) / $
    (onsource.sig.left.tube_on − onsource.sig.left.tube_off)

  tsys.ref.left = (osleft.ref ∗ tcleft_rb.ref) / $
    (onsource.ref.left.tube_on − onsource.ref.left.tube_off)

  ;Fit a low order polynomial to system temp for the on−source scans.
  For ii=0, (numints−1) Do Begin
    result = Poly_fit(bigkeeper[200:−201], tsys[ii].sig.left[200:−201], fitorder, /double)
    tsfit[ii].sig.left = Poly(bigkeeper[277:6276], result)

    result = Poly_fit(bigkeeper[200:−201], tsys[ii].ref.left[200:−201], fitorder, /double)
    tsfit[ii].ref.left = Poly(bigkeeper[1915:7914], result)
```

```
Endfor

;Sig phase, LL pol:
;Calculate appropriate range for the ordinate & round to nearest integer.
center = Mean(Total(tsfit.sig.left ∗ (tsfit.sig.left ˆ (−2d)), 2, /double) / $
  Total(tsfit.sig.left ˆ (−2d), 2, /double))
range = [Round(center)−2.0, Round(center)+2.0]

;Plot polynomial fit to T_sys for 'sig' phase, LL pol.
ptsf_sl = Plot(keeper[200:−201], tsfit[0].sig.left[200:−201], xrange=[0,6d3], $
  yrange=range, xstyle=1, xminor=3, xmajor=7, yminor=1, title='Sig − LL Pol', $
  ytitle='T_sys (K)', margin=[0.075,0.1,0.025,0.125], layout=[2,2,1], /current)

;Use FOR loop to plot all T_sys fits beyond the first.
For ii=1, (numints−1) Do ptsf_sl = Plot(keeper[200:−201], tsfit[ii].sig.left[200:−201], $
  color=randomclrs[∗,ii−1], layout=[2,2,1], /current, /overplot)
ax = ptsf_sl.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Ref phase, LL pol:
;Calculate appropriate range for the ordinate & round to nearest integer.
center = Mean(Total(tsfit.ref.left ∗ (tsfit.ref.left ˆ (−2d)), 2, /double) / $
  Total(tsfit.ref.left ˆ (−2d), 2, /double))
range = [Round(center)−2.0, Round(center)+2.0]

;Plot polynomial fit to T_sys for 'ref' phase, LL pol.
ptsf_rl = Plot(keeper[200:−201], tsfit[0].ref.left[200:−201], xrange=[0,6d3], $
  yrange=range, xstyle=1, xminor=3, xmajor=7, yminor=1, title='Ref − LL Pol', $
  ytitle= 'T_sys (K)', margin=[0.075,0.1,0.025,0.125], layout=[2,2,3], /current)

;Use FOR loop to plot all T_sys fits beyond the first.
For ii=1, (numints−1) Do ptsf_rl = Plot(keeper[200:−201], tsfit[ii].ref.left[200:−201], $
  color=randomclrs[∗,ii−1], layout=[2,2,3], /current, /overplot)
ax = ptsf_rl.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
```

```
ax[3].ticklen = ax[3].ticklen / 2.0d

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 8: − Calculate Vectorized Tsys.
;      − Both frequency phases, RR pol.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize structure(s) and/or array(s)
osright = Replicate(Create_struct(name = 'sigref_m'), numints)
tcright_rb = Replicate(Create_struct(name = 'sigref_m'), numints)

;Rebin RR pol calibration temps.
tcright_rb.sig = Rebin(tcal.sig.right, rtype.med, numints, /sample)
tcright_rb.ref = Rebin(tcal.ref.right, rtype.med, numints, /sample)

;Calculate the tube on/off average for both phases
osright.sig = (onsource.sig.right.tube_on + onsource.sig.right.tube_off) / 2.0d
osright.ref = (onsource.ref.right.tube_on + onsource.ref.right.tube_off) / 2.0d

;Calculate vectorized system temp for the on−source scans.
tsys.sig.right = (osright.sig * tcright_rb.sig) / $
  (onsource.sig.right.tube_on − onsource.sig.right.tube_off)

tsys.ref.right = (osright.ref * tcright_rb.ref) / $
  (onsource.ref.right.tube_on − onsource.ref.right.tube_off)

;Fit a low order polynomial to system temp for the on−source scans.
For ii=0, (numints−1) Do Begin
  result = Poly_fit(bigkeeper[200:−201], tsys[ii].sig.right[200:−201], fitorder, /double)
  tsfit[ii].sig.right = Poly(bigkeeper[277:6276], result)

  result = Poly_fit(bigkeeper[200:−201], tsys[ii].ref.right[200:−201], fitorder, /double)
  tsfit[ii].ref.right = Poly(bigkeeper[1915:7914], result)
Endfor

;Sig phase, RR pol:
;Calculate appropriate range for the ordinate & round to nearest integer.
center = Mean(Total(tsfit.sig.right * (tsfit.sig.right ^ (−2d)), 2, /double) / $
  Total(tsfit.sig.right ^ (−2d), 2, /double))
range = [Round(center)−2.0, Round(center)+2.0]
```

```
;Plot polynomial fit to T_sys for 'sig' phase, RR pol.
ptsf_sr = Plot(keeper[200:−201], tsfit[0].sig.right[200:−201], xrange=[0,6d3], $
  yrange=range, xstyle=1, xminor=3, xmajor=7, yminor=1, title='Sig − RR Pol', $
  ytitle='T_sys (K)', margin=[0.075,0.1,0.025,0.125], layout=[2,2,2], /current)

;Use FOR loop to plot all T_sys fits beyond the first.
For ii=1, (numints−1) Do ptsf_sr = Plot(keeper[200:−201], tsfit[ii].sig.right[200:−201], $
  color=randomclrs[*,ii−1], layout=[2,2,2], /current, /overplot)
ax = ptsf_sr.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Ref phase, RR pol:
;Calculate appropriate range for the ordinate & round to nearest integer.
center = Mean(Total(tsfit.ref.right * (tsfit.ref.right ^ (−2d)), 2, /double) / $
  Total(tsfit.ref.right ^ (−2d), 2, /double))
range = [Round(center)−2.0, Round(center)+2.0]

;Plot polynomial fit to T_sys for 'ref' phase, RR pol.
ptsf_rr = Plot(keeper[200:−201], tsfit[0].ref.right[200:−201], xrange=[0,6d3], $
  yrange=range, xstyle=1, xminor=3, xmajor=7, yminor=1, title='Ref − RR Pol', $
  ytitle='T_sys (K)', margin=[0.075,0.1,0.025,0.125], layout=[2,2,4], /current)

;Use FOR loop to plot all T_sys fits beyond the first.
For ii=1, (numints−1) Do ptsf_rr = Plot(keeper[200:−201], tsfit[ii].ref.right[200:−201], $
  color=randomclrs[*,ii−1], layout=[2,2,4], /current, /overplot)
ax = ptsf_rr.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Save graphic as .png file.
pic_name = Strtrim(newdir,2) + '/Tsys_' + Strtrim(callsign, 2) + '_scan' + $
  Strtrim(master+1, 2) + '_run' + Strtrim(kappa,2) + '.png'
W.Save, Strtrim(pic_name,2), resolution = 100
```

```
;+————————————————————————————————————————+
;SECTION 9: – Calculate Vectorized Ta.
; – Both frequency phases, both polarizations.
;+————————————————————————————————————————+

;If the 100 mHz bandpass is a total of 8192 pixels, then the sig & ref peaks,
;being 20 mHz apart, must be separated by 8192 * (2/10) = 1638.4 (1638)
;pixels, making them @ pixels (8192 +/– 1638) / 2, that is 3277 and 4915,
;respectively.

;The gain in the receiver gets all funky within ~ 300 pixels of the end of
;the bandpass, so, centering the lines in the clipped data means a maximum
;window width of 6k pixels. [277:6276] & [1915:7914].

;I know it looks like I have SIG and REF mixed up here, but it's right.
;It works out, notice below ....

;Calculate vectorized antenna temp for each integration in the scan:
;ll pol, sig phase.
proxy.sig.left = (osleft.sig – osleft.ref) / osleft.ref
ta.sig.left = proxy.sig.left[1915:7914] * tsfit.ref.left

;ll pol, sig phase.
proxy.ref.left = (osleft.ref – osleft.sig) / osleft.sig
ta.ref.left = proxy.ref.left[277:6276] * tsfit.sig.left

;rr pol, sig phase.
proxy.sig.right = (osright.sig – osright.ref) / osright.ref
ta.sig.right = proxy.sig.right[1915:7914] * tsfit.ref.right

;rr pol, ref phase.
proxy.ref.right = (osright.ref – osright.sig) / osright.sig
ta.ref.right = proxy.ref.right[277:6276] * tsfit.sig.right

;+————————————————————————————————————————+
;SECTION 10: – Convert Ta To Tmb & fold
; – Compile and store data for each scan
;+————————————————————————————————————————+

;Average lsrfreq values from the 'sig' & 'ref' position headers.
```

```
;these values should be the same anyway.
lsr_freq = Mean([source_head.sig.lsrfreq, source_head.ref.lsrfreq])

;Calculate airmass & zenith opacity using the Ruze equation.
tau = 0.008d + Exp(Sqrt(lsr_freq / 1d9)) / (8000.0d)
airmass = 1.0d / Sin(source_head.sig.elevatio)

;calc. main beam efficiency, which is 1.37 * aperture efficiency for the GBT,
;or so says the proposers guide. either way its a simple scalar, and is
;frequency independent.
eta_mb = 0.973d * Exp(-1d * ((4d * !Dpi * (lsr_freq) * 3d-4) / (!Const.c)) ^ 2d)
factor = Exp(tau * airmass) / eta_mb

;LL Pol
;Calculate the 'weighted' antenna temp, weighting the phases of each
;integration by the inverse square of the corresponding system temp.
;This is done bin-by-bin, & results are saved to be averaged later
scan_wtmb[master].sig.left = Total((factor * ta.sig.left) * $
  (tsfit.ref.left ^ (-2d)), 2, /double)
scan_wtmb[master].ref.left = Total((factor * ta.ref.left) * $
  (tsfit.sig.left ^ (-2d)), 2, /double)

scan_twt[master].sig.left = Total(tsfit.ref.left ^ (-2d), 2, /double)
scan_twt[master].ref.left = Total(tsfit.sig.left ^ (-2d), 2, /double)

;Calculate the 'weighted' system temp, weighting the phases of each
;integration by the inverse square of itsef?
;This is done bin-by-bin, & results are saved to be averaged later
;This is EXPERIMENTAL, as in it seems right, but I can't prove it is.
; <<<Proceed with CAUTION>>>
scan_wtsfit[master].sig.left = Total(tsfit.ref.left * (tsfit.ref.left ^ (-2d)), 2, /double)
scan_wtsfit[master].ref.left = Total(tsfit.sig.left * (tsfit.sig.left ^ (-2d)), 2, /double)

;RR Pol
;Calculate the 'weighted' antenna temp, weighting the phases of each
;integration by the inverse square of the corresponding system temp.
;This is done bin-by-bin, & results are saved to be averaged later
scan_wtmb[master].sig.right = Total((factor * ta.sig.right) * $
  (tsfit.ref.right ^ (-2d)), 2, /double)
scan_wtmb[master].ref.right = Total((factor * ta.ref.right) * $
      (tsfit.sig.right ^ (−2d)), 2, /double)

    scan_twt[master].sig.right = Total(tsfit.ref.right ^ (−2d), 2, /double)
    scan_twt[master].ref.right = Total(tsfit.sig.right ^ (−2d), 2, /double)

    ;Calculate the 'weighted' system temp, weighting the phases of each
    ;integration by the inverse square of itsef?
    ;This is done bin−by−bin, & results are saved to be averaged later
    ;This is EXPERIMENTAL, as in it seems right, but I can't prove it is.
    ; <<<Proceed with CAUTION>>>
    scan_wtsfit[master].sig.right = Total(tsfit.ref.right * (tsfit.ref.right ^ (−2d)), 2, /double)
    scan_wtsfit[master].ref.right = Total(tsfit.sig.right * (tsfit.sig.right ^ (−2d)), 2, /double)

    ;Print some stuff.
    Print, ''
    Print, format = '(">>>>> weighted mean tsys (Sig − LL Pol) = ", f−15.2)', $
      Mean(scan_wtsfit[master].ref.left / scan_twt[master].ref.left)
    Print, format = '(">>>>> weighted mean tsys (Sig − RR Pol) = ", f−15.2)', $
      Mean(scan_wtsfit[master].ref.right / scan_twt[master].ref.right)
    Print, format = '(">>>>> weighted mean tsys (Ref − LL Pol) = ", f−15.2)', $
      Mean(scan_wtsfit[master].sig.left / scan_twt[master].sig.left)
    Print, format = '(">>>>> weighted mean tsys (Ref − RR Pol) = ", f−15.2)', $
      Mean(scan_wtsfit[master].sig.right / scan_twt[master].sig.right)
    Print, ''
    Print, '>>>>> ...Done'
    Print, ''
  Endfor

  ;Print some stuff.
  Print, '>>>>> source data import: ...Done'
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 11: − Average data for all scans
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;LL Pol
  ;Sum the weight and weighted tmb values for each scan into a single array
  obs_wtmb[kappa].sig.left = Total(scan_wtmb.sig.left, 2, /double)
  obs_wtmb[kappa].ref.left = Total(scan_wtmb.ref.left, 2, /double)
```

```
obs_wtsfit[kappa].sig.left = Total(scan_wtsfit.sig.left, 2, /double)
obs_wtsfit[kappa].ref.left = Total(scan_wtsfit.ref.left, 2, /double)

obs_twt[kappa].sig.left = Total(scan_twt.sig.left, 2, /double)
obs_twt[kappa].ref.left = Total(scan_twt.ref.left, 2, /double)


;RR Pol
;Sum the weight and weighted tmb values for each scan into a single array
obs_wtmb[kappa].sig.right = Total(scan_wtmb.sig.right, 2, /double)
obs_wtmb[kappa].ref.right = Total(scan_wtmb.ref.right, 2, /double)

obs_wtsfit[kappa].sig.right = Total(scan_wtsfit.sig.right, 2, /double)
obs_wtsfit[kappa].ref.right = Total(scan_wtsfit.ref.right, 2, /double)

obs_twt[kappa].sig.right = Total(scan_twt.sig.right, 2, /double)
obs_twt[kappa].ref.right = Total(scan_twt.ref.right, 2, /double)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 12: − Plot the results for each observation
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize array(s)
ptsys = Create_struct(name = 'lrpol_s')

;Calculate system temps for both polarizations.
ptsys.left = Total([[obs_wtsfit[kappa].sig.left],[obs_wtsfit[kappa].ref.left]], 2, /double) $
  / Total([[obs_twt[kappa].sig.left],[obs_twt[kappa].ref.left]],2, /double)
ptsys.right = Total([[obs_wtsfit[kappa].sig.right],[obs_wtsfit[kappa].ref.right]], 2, /double) $
  / Total([[obs_twt[kappa].sig.right],[obs_twt[kappa].ref.right]],2, /double)

;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot the system temps for both polarizations & save as a .png. This is
;purely for reference later on & to provide visual confirmation nothing went haywire.
pts_l = Plot(keeper[200:−201], ptsys.left[200:−201], color=pcolors.puke, xrange=[0,6d3], $
  xstyle=1, xminor=3, xmajor=7, yminor=1, ytitle='T_sys (K)', title='LL Pol', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,1], /current)
```

```
ax = pts_l.axes
ax[2].minor = 0
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0
ax[3].ticklen = ax[3].ticklen / 2.0d


pts_r = Plot(keeper[200:−201], ptsys.right[200:−201], color=pcolors.pink, xrange=[0,6d3], $
  xstyle=1, xminor=3, xmajor=7, yminor=1, ytitle='T_sys (K)', title='RR Pol', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,2], /current)
ax = pts_r.axes
ax[2].minor = 0
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0
ax[3].ticklen = ax[3].ticklen / 2.0d

;Save graphic as .png file.
pic_name = Strtrim(newdir,2) + '/tsys_' + Strtrim(callsign, 2) + '_obs' + $
  Strtrim(kappa,2) + '.png'
w.Save, Strtrim(pic_name,2), resolution = 100

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize array(s)
pta = Create_struct(name = 'lrpol_s')

;Calculate main beam temps for both polarizations.
pta.left = Total([[obs_wtmb[kappa].sig.left],[obs_wtmb[kappa].ref.left]], 2, /double) $
  / Total([[obs_twt[kappa].sig.left],[obs_twt[kappa].ref.left]],2, /double)
pta.right = Total([[obs_wtmb[kappa].sig.right],[obs_wtmb[kappa].ref.right]], 2, /double) $
  / Total([[obs_twt[kappa].sig.right],[obs_twt[kappa].ref.right]],2, /double)

;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot the main beam temps for both polarizations & save as a .png. This is
;purely for reference later on & to provide visual confirmation nothing went haywire.
ptmb_l = Plot(keeper[200:−201], pta.left[200:−201], color=pcolors.puke, xrange=[0,6d3], $
  xstyle=1, xminor=3, xmajor=7, yminor=1, ytitle='T_sys (K)', title='LL Pol', $
  margin=[0.075,0.1,0.025,0.125], layout=[1,2,1], /current)
  ax = ptmb_l.axes
  ax[2].minor = 0
  ax[2].ticklen = ax[2].ticklen / 2.0d
  ax[3].minor = 0
  ax[3].ticklen = ax[3].ticklen / 2.0d

  ptmb_r = Plot(keeper[200:−201], pta.right[200:−201], color=pcolors.pink, xrange=[0,6d3], $
    xstyle=1, xminor=3, xmajor=7, yminor=1, ytitle='T_sys (K)', title='RR Pol', $
    margin=[0.075,0.1,0.025,0.125], layout=[1,2,2], /current)
  ax = ptmb_r.axes
  ax[2].minor = 0
  ax[2].ticklen = ax[2].ticklen / 2.0d
  ax[3].minor = 0
  ax[3].ticklen = ax[3].ticklen / 2.0d

  ;Save graphic as .png file.
  pic_name = Strtrim(newdir,2) + '/tmb_' + Strtrim(callsign, 2) + '_obs' + $
    Strtrim(kappa,2) + '.png'
  w.Save, Strtrim(pic_name,2), resolution = 100

  ;As of version 5.1, the old infrastructure has been remved, in an attempt to clean things
  ;up and prepare the code for publication. A version of the code retaining the old
  ;infrastructure was retained, and named dreamcatcher_unaltered.pro.

  ;I think it is worth averaging things together by 'ref' and 'sig' values, whereas you are
  ;currently doing to via the polarization. It SHOULD be exactly the same, but it might look
  ;nicer, or reveal somthing interesting that might otherwise be lost
Endfor

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 13: − Average data for all observations
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize array(s)
spinal_tmb = Dblarr(rtype.sml, /nozero)
smoo_spinal = Dblarr(rtype.sml, /nozero)

;Average all observations together. the main beam temperature is already weighted
;by the inverse square of the system temperature, so the sum must be normalized by the
;sum of the weights (the inverse square of the system temperature) which is carried
```

```
;out of the loops as the structure 'obs_twt'
spinal_tmb = Total([[obs_wtmb.sig.left],[obs_wtmb.sig.right], $
  [obs_wtmb.ref.left],[obs_wtmb.ref.right]], 2, /double) / $
  Total([[obs_twt.sig.left],[obs_twt.sig.right],$
  [obs_twt.ref.left],[obs_twt.ref.right]], 2, /double)

smoo_spinal = Convol(spinal_tmb, Savgol(40,40,0,3))

;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot the final spectrum.
ptspin_a = Plot(keeper[49:−50], spinal_tmb[49:−50], color=pcolors.pink, xrange=[0,6d3], $
  xmajor=7, xminor=1, yminor=3, font_size=11, ytitle='T_mb (K)', $
  margin=[0.045,0.05,0.015,0.03], /current)
ptspin_a = Plot(keeper[49:−50], smoo_spinal[49:−50], color=pcolors.puke, $
  thick=2, /current, /overplot)
ax = ptspin_a.axes
ax[2].minor = 0.0d
ax[2].ticklen = ax[2].ticklen / 2.0d
ax[3].minor = 0.0d
ax[3].ticklen = ax[3].ticklen / 2.0d

;Save graphic as .png file.
pic_name = Strtrim(newdir,2) + '/finalline_' + Strtrim(callsign, 2) + '.png'
w.Save, Strtrim(pic_name,2), resolution = 100

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 14: − Save data to .dat.gzip file
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize array(s)
tvlsr = Dblarr(rtype.med, /nozero)
vlsr = Dblarr(rtype.sml, /nozero)

;Convert the abscissa values from frequency to velocity
tvlsr = [source_head.sig.lsrfreq − (source_head.sig.bandwid ∗ 0.6d) : $
  source_head.sig.lsrfreq+ (source_head.sig.bandwid ∗ 0.4d) : $
  (source_head.sig.bandwid) / rtype.med]
  tvlsr = !Const.c * (1d − (Temporary(tvlsr)) / (source_head.sig.lsrfreq))
  vlsr = tvlsr[1915:7914] / 1d3

  ;Export data to ASCII files. Build & insert standard size header Repeat
  ;only once, writing all 'ref' phase and 'sig' phase data into a single file.
  Openw, lun, Strtrim(newdir,2) + '/calibrated_' + Strtrim(callsign, 2) + $
    '.dat.gzip', /get_lun, /compress
  Printf, lun, Strtrim(Systime(), 2)
  Printf, lun, [[nsessions], [N_tags(source_head.sig)]]

  Help, source_head.sig, output = pshead
  pphead = Strsplit(pshead[1:−1], " "", /extract)

  Printf, lun, Transpose(pphead)
  Printf, lun, [['vlsr'], ['t_mb']]
  Printf, lun, Transpose([[vlsr], [spinal_tmb]]), format = '(2' + rtype.format + ')'
  Close, lun
  Free_lun, lun

  ;Print some jazz and end.
  Print, ''
  Print, '>>>>> ...Done'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''

End
```

14.7 Outerlimits.pro

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
;Event handler used by the outerlimits.pro GUI.
;−

Function Olmd_uv, owin, x, y, ibutton, keymods, nclicks
  Compile_opt IDL2, hidden
  Common lspex
  Common pspex
  On_error, 2
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 9 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 22 July, 2016
  ;>>>>> Version 4.0 (V−spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V−spec) written by: N.N. Monson (UCLA). 27 November, 2018
  ;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 13 January, 2019

  ;Set STATE variables.
  state = owin.uvalue
  State.mouse = ibutton

  ;Set a few proxy variables, to ensure they aren't unwittingly alterd
  ;in the common variable block
  numlims = State.numlims
  last = State.last
  pline = State.pline

  ;Keymod 2 is the control key, and 8 is the alt/command key.
  Case keymods Of
    ;If the CTRL key and left mouse button are down...
    2:Begin
      If State.mouse Eq 1 Then Begin
```

```
;Check if the mouse position along the abscissa is less than the
;last time (if there was a last time).
If x Gt last Then Begin

  ;If it is, save the current mouse position (in device coordinates) for reuse.
  State.last = x

  ;Get the coordinates of the current and last mouse positions and
  ;convert the coordinates to data values.
  new = pline.Convertcoord(x,y, /device, /to_data)
  new = new[0]
  old = pline.Convertcoord(last,y, /device, /to_data)
  old = old[0]

  ;Set the window created by Outerlimits.pro as the current window,
  ;and plot a vertical line at the current mouse position.
  owin.Setcurrent
  fpfive = Plot([new, new], [State.ymin − Abs(20 * State.ymin), $
    State.ymax + Abs(20 * State.ymax)], color=pcolors.teal, linestyle=2, $
    thick=1.5, /current, /overplot)
  owin.Refresh

  ;Add the current mouse position to the list of previous positions
  ;and increment the loop counter by one.
  State.xarr.Add, new[0], /extract
  ++ State.numlims

  ;Plot each new bounded region as a different color.
  If (numlims Mod 2) Eq 0 Then Begin
    owin.Setcurrent
    fpsix = Plot(State.indep[old:new], State.dep[old:new], color=pcolors.teal, $
      thick=1.5, /overplot, /current)
    owin.Refresh
  Endif
Endif Else Begin
  ;Otherwise, discard it and start over.
  Print, '>>>>> Warning: Entry out of range'
  Print, '>>>>> Discarding last entry...'
Endelse
```

```
    Endif
  End
  ;If the ALT/CMD key and left mouse button are down...
  8:Begin
    ;Check to see if at least one region has been selected.
    If numlims Gt 1 Then Begin

      ;If there are an odd number of limits, then discard the last
      ;one and decrement the loop counter by one.
      If (numlims Mod 2) Eq 1 Then Begin
        Print, '>>>>> Warning: Odd number of limits detected'
        Print, '>>>>> Discarding last entry...'
        State.xarr.Remove, −1
        −− numlims
      Endif

      ;If so, set STATE.done variable to 'y' to exit the fitting GUI.
      Print, '>>>>> Alert: Fitting regions set...'
      State.done = 'y'
    Endif Else Begin
      ;Otherwise, Complain because no regions have been set.
      Print, '>>>>> Error: No fitting regions set'
    Endelse
  End
  ;if any keymod other than 8 or 2 is down, then complain.
  Else: Print, '>>>>> Error: Invalid keymod'
Endcase

;After selecting the fitting regions, save data to lspex,
;so it is available to Polybase.pro and Dreamweaver.pro.
If State.done Eq 'y' Then Begin

  ;Remove any existing items from the .xlims list in lspex,
  ;then store new fit data in the now empty .xlims list
  exe2 = Execute('lspex.'+Strtrim(State.sign,2) + '.xlims.remove, /all')
  exe3 = Execute('lspex.'+Strtrim(State.sign,2) + '.xlims.add, STATE.xarr, /extract')

  ;Copy the total number of limits to the lspex structure.
  exc4 = Execute('lspex.'+Strtrim(State.sign,2) + '.numlims = numlims')
  If ~ Array_equal(1, [exc2, exc3, exc4]) Then Message, '>>>>> Error: execution failure'
```

```
    ;Close the window and print some stuff.
    owin.Close
    Print, ''
    Print, '>>>>> ...Done'
    Print, ''
    Print, '>>>>> End Of Line <<<<<'
    Print, ''

    ;If state.done isn't set to 'y', then save all the state variables back to
    ;the OWIN structure for reuse.
  Endif Else owin.uvalue = state
  Return, 0
End

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME
; Outerlimits
; HYDRA Version 5.1
;
; PURPOSE
; −> Define the regions used to be used by Polybase.pro when fitting and
; subtracting structure in the spectra.
;
; CALLING SEQUENCE
; −> Outerlimits, Iso=value, Dtype=string [, /Rrl, /Lowsnr, /Crapshoot, /Zoom]
;
; ARGUMENT(S)
; −> None.
;
; KEYWORD(S)
; −> Iso: The mass number of the isotope for which data will be loaded and plotted.
; ∗ Acceptable values: '28', '29', '30'.
;
; −> Dtype: A string specifying the type of data to be loaded and plotted.
; ∗ Acceptable values: 'calibrated', 'baselined', 'rectified'
;
```

```
; OPTIONAL KEYWORD(S)
; -> Rrl (binary): Set to load and plot rrl data. Overrides ISO keyword.
;
; -> Zoom (binary): Set to restrict the range of the ordinate when plotting.
;
; -> Lowsnr (binary): Set to increase the degree to which the spectum is smoothed
; before it is plotted. Useful when fitting rare isotopologues or weak sources.
;
; -> Crapshoot (binary): Set to drastically increase the degree to which the
; spectum is smoothed before it is plotted. Overrides LOWSNR keyword.
; NOTE: This much smoothing should be avoided whenever possible.
;
; -> Printcolor (binary): Changes the color scheme of the plots produced.
;
; EXAMPLE(S)
; -> Outerlimits, Iso=29, Dtype='calibrated', /Lowsnr
;
; OUTPUT(S)
; -> None. Stores all data in LSPEX common variable.
;
; COMMENTS
; -> Operates only within the HYDRA 5.1 RTE.
; -> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro
;
; PROCEDURES/FUNCTIONS CALLED
; -> Olmd_Uv.pro
; -> Datatool_s__define.pro
;-

Pro Outerlimits, Iso=iso, Dtype=dtype, Rrl=rrl, Zoom=zoom, $
  Lowsnr=lowsnr, Crapshoot=crapshoot, Printcolor=printcolor
  Compile_opt IDL2
  Common overlord
  Common rtype
  Common lspex
  Common pspex
  On_error, 0
  !Except = 1
```

```
;Hydra version 5.1
;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 January, 2019

;+----------------------------------------+
;>>>>> Usage Agreement <<<<<
;+----------------------------------------+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+----------------------------------------+
;>>>>> Developer's Notes <<<<<
;+----------------------------------------+

;+----------------------------------------+
;>>>>> Limitations & Known Bugs <<<<<
;+----------------------------------------+

;The /ZOOM keyword should be used whenever previously baselined data is
;being plotted, as the the edges of the spectrum often have extreme values.
;If /ZOOM is not set, the default range on the ordinate will be too large
;for the emission features to be seen.

;+----------------------------------------+
;SECTION 0: - Check argument(s).
; - Set keyword defaults.
;+----------------------------------------+

;Print an empty line below the program call on the command line.
Print, ''

;Inialize structure(s) and/or array(s).
keymaster = 0
```

```
callsign = ''
scsign = ''

;Ensure input/output filepath had been set, and is in stored
;in OVERLORD. If not, then throw an error.
If Array_equal('', overlord.dirloc) Then Begin
  Print, '>>>>> Error: Common directory not set'
  Print, '>>>>> Set common directory using Setdir.pro'
  Print, '' & ++ keymaster
Endif

;Check if the /RRL keyword is set
If Keyword_set(RRL) Then Begin

  ;If it is, check to see if the /RRL keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If rrl Ne 1 Then Begin
    Print, '>>>>> Alert: Rrl is a binary keyword'
    Print, '>>>>> Setting rrl accordingly ...'
    Print, '' & RRL = 1
  Endif

  ;Ignore iso keyword if RRL keyword is set
  If Keyword_set(iso) Then Begin
    Print, '>>>>> Alert: Rrl keyword set, ignoring iso keyword'
    Print, ''
  Endif

  ;set callsign(s)
  callsign = 'rrl'
  scsign = 'rrl'

Endif Else Begin
  ;Otherwise, check if the ISO keyword is set
  If Keyword_set(iso) Then Begin

    ;If it is, Ensure the ISO keyword is a scalar and
    ;is a longword integer, or can be converted to one.
    ;If ISO is a funky type, e.g. unsigned, then throw an error.
    If Array_equal(0, Isa(iso, /scalar)) || $
```

```
      Array_equal(1, rtype.adtypes.Contains(Typename(iso)), /not_equal) Then Begin
      Print, '>>>>> Alert: Iso keyword dimension = '+Strtrim(iso.length,2)
      Print, '>>>>> Alert: Iso keyword type = '+Strtrim(Typename(iso),2)
      Print, '>>>>> Error: Iso Keyword must be an integer or floating−point scalar'
      Print, '' & ++ keymaster
    Endif Else Begin
      ;Otherwise, convert ISO to a longword
      iso = Long(iso)

      ;And check to see if ISO is in range. If not, then throw an error.
      If Array_equal(iso, rtype.aivals, /not_equal) Then Begin
        Print, '>>>>> Error: Iso keyword out of range'
        Print, '' & ++ keymaster
      Endif Else Begin

        ;Otherwise, set callsign(s)
        callsign = Strtrim(iso,2) + 'sio'
        scsign = 's' + Strtrim(iso,2)
      Endelse
    Endelse
  Endif Else Begin
    ;Otherwise, throw an error.
    Print, '>>>>> Error: Iso keyword not set'
    Print, '' & ++ keymaster
  Endelse
Endelse

;Check if the DTYPE keyword is set.
If Keyword_set(dtype) Then Begin

  ;Ensure DTYPE is a string and is is an acceptable value.
  ;If not, then throw an error.
  Case dtype Of
    'cal': dtype = 'calibrated'
    'base': dtype = 'baselined'
    'rec': dtype = 'rectified'
    Else: Begin
      If Isa(dtype, /string) Then Begin
        Print, '>>>>> Error: Dtype keyword invalid'
      Endif Else Print, '>>>>> Error: Dtype keyword must be a string'
      Print, '' & ++ keymaster
    End
  Endcase
Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Error: Dtype keyword not set'
  Print, '' & ++ keymaster
Endelse

;Check if the /CRAPSHOOT keyword is set.
If Keyword_set(crapshoot) Then Begin

  ;If it is, check to see if the /CRAPSHOOT keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If crapshoot Ne 1 Then Begin
    Print, '>>>>> Alert: Crapshoot is a binary keyword'
    Print, '>>>>> Setting crapshoot accordingly ...'
    Print, '' & crapshoot = 1
  Endif

  ;Set /LOWSNR. /CRAPSHOOT requires /LOWSNR to be set concurrently.
  lowsnr = 1

Endif Else Begin
  ;Check if the /LOWSNR keyword is set.
  If Keyword_set(lowsnr) Then Begin

    ;If it is, check to see if the /LOWSNR keyword is set to anything other than 1.
    ;If necessary, change it to 1 to ensure compatibility with logical operators.
    If lowsnr Ne 1 Then Begin
      Print, '>>>>> Alert: Lowsnr is a binary keyword'
      Print, '>>>>> Setting lowsnr accordingly ...'
      Print, '' & lowsnr = 1
    Endif
  Endif
Endelse

;Check if the /ZOOM keyword is set.
If Keyword_set(zoom) Then Begin
```

```
  ;If it is, check to see if the /ZOOM keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If Zoom Ne 1 Then Begin
    Print, '>>>>> Alert: Zoom is a binary keyword'
    Print, '>>>>> Setting zoom accordingly ...'
    Print, '' & zoom = 1
  Endif
Endif

;Set colors to use when plotting. Inverted colors used by default.
If Keyword_set(printcolor) Then Begin
  pcolors = pspex.rcolors
Endif Else pcolors = pspex.icolors

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise, close all open graphics windows and continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
  Purge
Endelse

;+---------------------------------------+
;SECTION 1: - Load data
;+---------------------------------------+

;Initialize structure(s) and/or array(s)
raw = Create_struct(name = 'datatool_s')

;Define path to the folder to load data from.
rootpath = Strtrim(Overlord.dirloc,2) + '/' + Strtrim(callsign,2)
```

```
;Print some stuff.
Print, '+--------------------------------------+'
Print, '>>>>> Loading ' + Strtrim(callsign,2) + ' data...'

;Load data for the target spectrum.
Openr, lun, Strtrim(rootpath,2) + '/' + Strtrim(dtype,2) + '_' + $
  Strtrim(callsign,2) + '.dat.gzip', /get_lun, /compress
time = ''
Readf, lun, time
nessions = 0
Readf, lun, nsessions
numtags = 0
Readf, lun, numtags
header = Strarr(numtags)
Readf, lun, header
names = Strarr(2)
Readf, lun, names
data = Dblarr(2, rtype.sml, /nozero)
Readf, lun, data, format = '(2' + rtype.format +')'
Close, lun
Free_lun, lun

;Put the unbaselined tmb data into it's structure.
raw.wild = Transpose(data[1,*])

;Print some stuff
Print, '>>>>> ...Done'
Print, '+--------------------------------------+'
Print, ''

;+--------------------------------------+
;SECTION 2: - Smooth data
; - Calculate plotting range of the ordinate
;+--------------------------------------+

;Initialize structure(s) and/or array(s)
keeper = Dindgen(rtype.sml)

;Smooth the spectrum, and place it in it's structure w/ the unsmoothed data.
;Use keywords to determine how much to smooth the spectrum.
```

```
;Savitzy– Golay smoothing is used, as it preserves peak shapes the best.
If Keyword_set(lowsnr) Then Begin
  If Keyword_set(crapshoot) Then Begin

    ;If /CRAPSHOOT is set, smooth as much as you can get away with.
    ;This much smoothing should be avoided if possible.
    Case callsign Of
      '28sio':slevel = Savgol(45,45,0,3)
      '29sio':slevel = Savgol(65,65,0,3)
      '30sio':slevel = Savgol(75,75,0,3)
      'rrl':slevel = Savgol(100,100,0,3)
    Endcase
  Endif Else Begin

    ;If only /LOWSNR is set, smooth by quite a bit,
    ;but less than if /CRAPSHOOT was set.
    Case callsign Of
      '28sio':slevel = Savgol(40,40,0,4)
      '29sio':slevel = Savgol(60,60,0,4)
      '30sio':slevel = Savgol(70,70,0,4)
      'rrl':slevel = Savgol(100,100,0,3)
    Endcase
  Endelse
  ;Smooth the spectrum.
  raw.tame = Convol(raw.wild, slevel)
Endif Else Begin

  ;If neither keywords are set, smooth by a moderate ammount.
  ;The spectra are always smoothed by at least this much,
  ;even if the SNR is good, it helps with setting the fitting limits.
  Case callsign Of
    '28sio':slevel = Savgol(30,30,0,4)
    '29sio':slevel = Savgol(50,50,0,4)
    '30sio':slevel = Savgol(60,60,0,4)
    'rrl':slevel = Savgol(100,100,0,3)
  Endcase
  ;Smooth the spectrum.
  raw.tame = Convol(raw.wild, slevel)
Endelse
```

```
;Try and calculate some good limits for the ordinate when plotting.
;This technique is a little bit crude, and doesn't always work well.
;Check if the /Zoom keyword is set
If Keyword_set(zoom) Then Begin

  ;If it is, Ignore a large section on each edge of the spectrum and
  ;calculate the limits of the ordinate based on the center of the spectrum.
  m = Mean(raw.tame[1000:−1001])
  ylo = m − (1.2 ∗ (m − Min(raw.tame[1000:−1001])))
  yhi = m + (1.2 ∗ (Max(raw.tame[1000:−1001]) − m))
Endif Else Begin

  ;Otherwise, Adopt a more conservative approach.
  m = Mean(raw.tame[500:−501])
  ylo = m − (2.0 ∗ (m − Min(raw.tame[500:−501])))
  yhi = m + (2.0 ∗ (Max(raw.tame[500:−501]) − m))
Endelse

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 2: − Plot data
; − Initialize fitting GUI
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some stuff.
Print, ">>>>> Launching fitting GUI..."
Print, ''
Print, ">>>>> Alert: 'ctrl' + left click to set regions"
Print, ">>>>> Alert: 'alt/cmd' + left click to finsh"
Print, ''

;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions = pspex.windim, background_color = pcolors.black)

;Plot a dotted line at 0K, just for visual reference.
fpzero = Plot(keeper, Replicate(0d, rtype.sml), color=pcolors.white, thick=1, $
  linestyle=1, /current , /overplot)

;Plot the smoothed spectrum
fpone = Plot(keeper[100:−101], raw.tame[100:−101], color=pcolors.puke, $
  xrange=[0,rtype.sml], yrange=[ylo,yhi], xstyle=1, xmajor=7, xminor=3, ystyle=1, $
  yminor=3, ytitle='Tmb (K)', font_size=11, margin = [0.04,0.05,0.015,0.03], /current)
ax = fpone.axes
ax[2].minor = 0
ax[2].ticklen = ax[2].ticklen / 2d
ax[3].minor = 0
ax[3].ticklen = ax[3].ticklen / 2d

;set olmd_uv as the event handler for the graphics window,
;and set initial vaues for variables.
w.window.mouse_down_handler = 'olmd_uv'
w.window.uvalue={ dep: stmb, $
  indep: keeper, $
  last: 0d, $
  index: 0, $
  mouse: 1, $
  xarr: List(), $
  ymin: ylo, $
  ymax: yhi, $
  pline: fpone, $
  sign: scsign, $
  done: 'n' }

;Check LSPEX and see if there is any limit data stored there.
exe_chk = Execute('limits = lspex.' + scsign + '.xlims.toarray()')
If exe_chk Ne 1 Then Message, '>>>>> Error: execution failure'

;If there is limit data in LSPEX, then plot it for reference when re−fitting data.
If ~ Isa(limits, /null) Then Begin
  For ii=0, limits.length−1 Do Begin

    ;Dont plot anything if it is out of range. This was an issue in previous versions,
    ;but is largely vestigal at this point. Ill leave it, just in case.
    If limits[ii] Lt stmb.length Then Begin

      ;Plot the limits of the previous fit.
      fptwo = Plot([limits[ii], limits[ii]], [ylo−Abs(20∗ylo), yhi+Abs(20∗yhi)], $
        color=pcolors.orange, linestyle=1, thick=2, /current, /overplot)

      ;Plot each new bounded region as a different color, just to make things look nice.
      If (ii Mod 2) Eq 1 Then Begin
```

```
          fpthree = Plot(keeper[limits[ii−1]:limits[ii]], raw.tame[limits[ii−1]:limits[ii]], $
            color=pcolors.pink, linestyle=0, thick=1.5, /current, /overplot)
        Endif
      Endif
    Endfor
  Endif

End
```

14.8 Polybase.pro

```
;+------------------------------------------+
;>>>>> Ancillary Program(s) <<<<<
;+------------------------------------------+

;+------------------------------------------+
;>>>>> Primary Program <<<<<
;+------------------------------------------+
;+
; NAME
; Polybase
; HYDRA Version 5.1
;
; PURPOSE
; -> Fit a polynomial to the regions selected using Outerlimits.pro, and use this
; fit to remove baseline structure from the spectrum.
;
; CALLING SEQUENCE
; -> Polybase, Iso=value, Dtype=string [, /Rrl, /Lowsnr, /Crapshoot, /Zoom]
;
; ARGUMENT(S)
; -> None.
;
; KEYWORD(S)
; -> Iso: The mass number of the isotope for which data will be loaded and plotted.
; * Acceptable values: '28', '29', '30'.
;
; -> Dtype: A string specifying the type of data to be loaded and plotted.
; * Acceptable values: 'calibrated', 'baselined', 'rectified'
;
; OPTIONAL KEYWORD(S)
; -> Rrl (binary): Set to load and plot rrl data. Overrides ISO keyword.
;
; -> Zoom (binary): Set to restrict the range of the ordinate when plotting.
;
; -> Lowsnr (binary): Set to increase the degree to which the spectum is smoothed
; before it is plotted. Useful when fitting rare isotopologues or weak sources.
;
```

```
; -> Crapshoot (binary): Set to drastically increase the degree to which the
; spectum is smoothed before it is plotted. Overrides LOWSNR keyword.
; NOTE: This much smoothing should be avoided whenever possible.
;
; -> Printcolor (binary): Changes the color scheme of the plots produced.
;
; EXAMPLE(S)
; -> Polybase, Iso=29, Dtype='calibrated', /Lowsnr
;
; OUTPUT(S)
; -> A single compressed data file named "Baselined_***.dat.gzip" containing
; the baseline subtracted spectrum.
;
; COMMENTS
; -> Operates only within the HYDRA 5.1 RTE.
; -> Filepath for the input/output directory is stored in OVERLORD
; and must be set using setdir.pro
;
; PROCEDURES/FUNCTIONS CALLED
; -> None
;-

Pro Polybase, Iso=iso, Dtype=dtype, Rrl=rrl, Zoom=zoom, $
  Lowsnr=lowsnr, Crapshoot=crapshoot, Printcolor=printcolor
  Compile_opt IDL2
  Common overlord
  Common rtype
  Common lspex
  Common pspex
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
  ;>>>>> Version 5.1 (V-spec) written by: N.N. Monson (UCLA). 11 January, 2019
```

```
;+-----------------------------------------+
;>>>>> Usage Agreement <<<<<
;+-----------------------------------------+

;;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+-----------------------------------------+
;>>>>> Developer's Notes <<<<<
;+-----------------------------------------+

;+-----------------------------------------+
;>>>>> Limitations & Known Bugs <<<<<
;+-----------------------------------------+

;The /ZOOM keyword should be used whenever previously baselined data is
;being plotted, as the the edges of the spectrum often have extreme values.
;If /ZOOM is not set, the default range on the ordinate will be too large
;for the emission features to be seen.

;+-----------------------------------------+
;SECTION 0: - Check argument(s).
; - Set keyword defaults.
;+-----------------------------------------+

;Print an empty line below the program call on the command line.
Print, ''

;Inialize structure(s) and/or array(s).
keymaster = 0
callsign = ''
scsign = ''

;Ensure input/output filepath had been set, and is in stored
;in OVERLORD. If not, then throw an error.
If Array_equal('', overlord.dirloc) Then Begin
  Print, '>>>>> Error: Common directory not set'
  Print, '>>>>> Set common directory using Setdir.pro'
```

```
 Print, '' & ++ keymaster
Endif

;Check if the /RRL keyword is set
If Keyword_set(RRL) Then Begin

 ;If it is, check to see if the /RRL keyword is set to anything other than 1.
 ;If necessary, change it to 1 to ensure compatibility with logical operators.
 If rrl Ne 1 Then Begin
  Print, '>>>>> Alert: Rrl is a binary keyword'
  Print, '>>>>> Setting rrl accordingly ...'
  Print, '' & RRL = 1
 Endif

 ;Ignore iso keyword if RRL keyword is set
 If Keyword_set(iso) Then Begin
  Print, '>>>>> Alert: Rrl keyword set, ignoring iso keyword'
  Print, ''
 Endif

 ;set callsign(s)
 callsign = 'rrl'
 scsign = 'rrl'

Endif Else Begin
 ;Otherwise, check if the ISO keyword is set
 If Keyword_set(iso) Then Begin

  ;If it is, Ensure the ISO keyword is a scalar and
  ;is a longword integer, or can be converted to one.
  ;If ISO is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(iso, /scalar)) || $
   Array_equal(1, rtype.adtypes.Contains(Typename(iso)), /not_equal) Then Begin
   Print, '>>>>> Alert: Iso keyword dimension = '+Strtrim(iso.length,2)
   Print, '>>>>> Alert: Iso keyword type = '+Strtrim(Typename(iso),2)
   Print, '>>>>> Error: Iso Keyword must be an integer or floating-point scalar'
   Print, '' & ++ keymaster
  Endif Else Begin
   ;Otherwise, convert ISO to a longword
   iso = Long(iso)
```

```
    ;And check to see if ISO is in range. If not, then throw an error.
    If Array_equal(iso, rtype.aivals, /not_equal) Then Begin
      Print, '>>>>> Error: Iso keyword out of range'
      Print, '' & ++ keymaster
    Endif Else Begin

      ;Otherwise, set callsign(s)
      callsign = Strtrim(iso,2) + 'sio'
      scsign = 's' + Strtrim(iso,2)
    Endelse
  Endelse
 Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Error: Iso keyword not set'
  Print, '' & ++ keymaster
 Endelse
Endelse

;Ensure fit region limits have been set, and are stored in LSPEX.
test = Execute('limits = lspex.' + scsign + '.xlims.toarray()')
If Isa(limits, /null) Then Begin
 Print, '>>>>> Error: No fit limits found in memory'
 Print, '>>>>> Set fit limits using Outerlimits.pro'
 Print, '' & ++ keymaster
Endif

;Check if the DTYPE keyword is set.
If Keyword_set(dtype) Then Begin

 ;Ensure DTYPE is a string and is is an acceptable value.
 ;If not, then throw an error.
 Case dtype Of
  'cal': dtype = 'calibrated'
  'base': dtype = 'baselined'
  'rec': dtype = 'rectified'
  Else: Begin
    If Isa(dtype, /string) Then Begin
      Print, '>>>>> Error: Dtype keyword invalid'
    Endif Else Print, '>>>>> Error: Dtype keyword must be a string'
```

```
      Print, '' & ++ keymaster
    End
  Endcase
Endif Else Begin
  ;Otherwise, throw an error.
  Print, '>>>>> Error: Dtype keyword not set'
  Print, '' & ++ keymaster
Endelse

;Check if the /CRAPSHOOT keyword is set.
If Keyword_set(crapshoot) Then Begin

  ;If it is, check to see if the /CRAPSHOOT keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If crapshoot Ne 1 Then Begin
    Print, '>>>>> Alert: Crapshoot is a binary keyword'
    Print, '>>>>> Setting crapshoot accordingly ...'
    Print, '' & crapshoot = 1
  Endif

  ;Set /LOWSNR. /CRAPSHOOT requires /LOWSNR to be set concurrently.
  lowsnr = 1

Endif Else Begin
  ;Check if the /LOWSNR keyword is set.
  If Keyword_set(lowsnr) Then Begin

    ;If it is, check to see if the /LOWSNR keyword is set to anything other than 1.
    ;If necessary, change it to 1 to ensure compatibility with logical operators.
    If lowsnr Ne 1 Then Begin
      Print, '>>>>> Alert: Lowsnr is a binary keyword'
      Print, '>>>>> Setting lowsnr accordingly ...'
      Print, '' & lowsnr = 1
    Endif
  Endif
Endelse

;Check if the /ZOOM keyword is set.
If Keyword_set(zoom) Then Begin
```

```
  ;If it is, check to see if the /ZOOM keyword is set to anything other than 1.
  ;If necessary, change it to 1 to ensure compatibility with logical operators.
  If Zoom Ne 1 Then Begin
    Print, '>>>>> Alert: Zoom is a binary keyword'
    Print, '>>>>> Setting zoom accordingly ...'
    Print, '' & zoom = 1
  Endif
Endif

;Set colors to use when plotting. Inverted colors used by default.
If Keyword_set(printcolor) Then Begin
  pcolors = pspex.rcolors
Endif Else pcolors = pspex.icolors

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: Status Red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise, close all open graphics windows and continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
  Purge
Endelse

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: − Load data
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize structure(s) and/or array(s)
raw = Create_struct(name = 'datatool_s')
bled = Create_struct(name = 'datatool_s')

;Define path to the folder to load data from.
rootpath = Strtrim(Overlord.dirloc,2) + '/' + Strtrim(callsign,2)
```

```
;Print some stuff.
Print, '+----------------------------------------+'
Print, '>>>>> Loading ' + Strtrim(callsign,2) + ' data...'

;Load data for the target spectrum.
Openr, lun, Strtrim(rootpath,2) + '/' + Strtrim(dtype,2) + '_' + $
  Strtrim(callsign,2) + '.dat.gzip', /get_lun, /compress
time = ''
Readf, lun, time
nessions = 0
Readf, lun, nsessions
numtags = 0
Readf, lun, numtags
header = Strarr(numtags)
Readf, lun, header
names = Strarr(2)
Readf, lun, names
data = Dblarr(2, rtype.sml, /nozero)
Readf, lun, data, format = '(2' + rtype.format + ')'
Close, lun
Free_lun, lun

;Put the unbaselined tmb data into it's structure.
raw.wild = Transpose(data[1,*])

;Print some stuff
Print, '>>>>> ...Done'
Print, '+----------------------------------------+'
Print, ''

;+----------------------------------------+
;SECTION 2: - Smooth data
; - Calculate plotting range of the ordinate
;+----------------------------------------+

;Initialize structure(s) and/or array(s)
keeper = Lindgen(rtype.sml)

;Smooth the spectrum, and place it in it's structure w/ the unsmoothed data.
```

```
;Use keywords to determine how much to smooth the spectrum.
;Savitzy−Golay smoothing is used, as it preserves peak shapes the best.
If Keyword_set(lowsnr) Then Begin
 If Keyword_set(crapshoot) Then Begin

   ;If /CRAPSHOOT is set, smooth as much as you can get away with.
   ;This much smoothing should be avoided if possible.
   Case callsign Of
     '28sio':slevel = Savgol(45,45,0,3)
     '29sio':slevel = Savgol(65,65,0,3)
     '30sio':slevel = Savgol(75,75,0,3)
     'rrl':slevel = Savgol(100,100,0,3)
   Endcase
 Endif Else Begin

   ;If /LOWSNR is set, smooth by quite a bit,
   ;but less than if /CRAPSHOOT was set.
   Case callsign Of
     '28sio':slevel = Savgol(40,40,0,4)
     '29sio':slevel = Savgol(60,60,0,4)
     '30sio':slevel = Savgol(70,70,0,4)
     'rrl':slevel = Savgol(100,100,0,3)
   Endcase
 Endelse
 ;Smooth the spectrum.
 raw.tame = Convol(raw.wild, slevel)
Endif Else Begin

 ;If neither keywords are set, smooth by a moderate ammount.
 ;The spectra are always smoothed by at least this much,
 ;even if the SNR is good, it helps with setting the fitting limits.
 Case callsign Of
   '28sio':slevel = Savgol(30,30,0,4)
   '29sio':slevel = Savgol(50,50,0,4)
   '30sio':slevel = Savgol(60,60,0,4)
   'rrl':slevel = Savgol(100,100,0,3)
 Endcase
 ;Smooth the spectrum.
 raw.tame = Convol(raw.wild, slevel)
Endelse
```

```
;Try and calculate some good limits for the ordinate when plotting.
;This technique is a little bit crude, and doesn't always work well.
;Check if the /Zoom keyword is set
If Keyword_set(zoom) Then Begin

   ;If it is, Ignore a large section on each edge of the spectrum and
   ;calculate the limits of the ordinate based on the center of the spectrum.
   m = Mean(raw.tame[1000:−1001])
   ylo = m − (1.2 ∗ (m − Min(raw.tame[1000:−1001])))
   yhi = m + (1.2 ∗ (Max(raw.tame[1000:−1001]) − m))
Endif Else Begin

   ;Otherwise, Adopt a more conservative approach.
   m = Mean(raw.tame[500:−501])
   ylo = m − (2.0 ∗ (m − Min(raw.tame[500:−501])))
   yhi = m + (2.0 ∗ (Max(raw.tame[500:−501]) − m))
Endelse

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 2: − Plot data
; − Initialize fitting GUI
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize variable(s)
;Inialize structure(s) and/or array(s).
limits = List()
nlimits = 0
fit = { indep: List(), $
   dep: List(), $
   order: 0d, $
   poly: List() }
chkpt = { alpha: 0, $
   bravo: 0, $
   charlie: 0 }

;copy limit data out of lspex.
exc1 = Execute('limits = lspex.' + scsign + '.xlims.toarray()')
exc2 = Execute('nlimits = lspex.'+ scsign + '.numlims')
If ˜ Array_equal(1, [exc1, exc2]) Then Message, '>>>>> Error: Execution failure'
```

```
;Use limit data to express limits in terms of dependent and independent variables.
For ii=1, nlimits−1, 2 Do Begin
 fit.indep.Add, keeper[limits[ii−1]:limits[ii]], /extract
 fit.dep.Add, raw.tame[limits[ii−1]:limits[ii]], /extract
Endfor

;Loop over the GUI section until all conditions are met.
;i.e. the fit is within reason, and the user is happy with how everything looks.
While chkpt.alpha Ne 1 Do Begin

 ;Reset values for bravo and charlie for new loop.
 chkpt.bravo = 0
 chkpt.charlie = 0

 ;Initialize a graphics window w/ standard dimensions.
 w = Window(dimensions=pspex.windim, background_color=pcolors.black)

 ;Plot a dotted line at 0K, just for visual reference.
 fpzero = Plot(keeper, Replicate(0.0, rtype.sml), color=pcolors.white, thick=1, $
  linestyle=1, /current , /overplot)

 ;Plot the smoothed spectrum
 fpone = Plot(keeper[100:−101], raw.tame[100:−101], color=pcolors.puke, $
  xrange=[0,rtype.sml], yrange=[ylo,yhi], xstyle=1, xmajor=7, xminor=3, ystyle=1, $
  yminor=3, ytitle='Tmb (K)', font_size=11, margin = [0.04,0.05,0.015,0.03], /current)
 ax = fpone.axes
 ax[2].minor = 0
 ax[2].ticklen = ax[2].ticklen / 2d
 ax[3].minor = 0
 ax[3].ticklen = ax[3].ticklen / 2d

 ;Plot a vertical line at each of the limits set using Outerlimits.
 For ii=0, nlimits−1 Do Begin
  fptwo = Plot([limits[ii], limits[ii]], [ylo−Abs(20*ylo), yhi+Abs(20*yhi)], $
   color=pcolors.teal, linestyle=1, thick=2, /current, /overplot)

  ;Plot each new bounded region as a different color, just to make things look nice.
  If (ii Mod 2) Eq 1 Then Begin
   fpthree = Plot(keeper[limits[ii−1]:limits[ii]], raw.tame[limits[ii−1]:limits[ii]], $
      color=pcolors.pink, linestyle=0, thick=1.5, /current, /overplot)
  Endif
Endfor

;If this isn't the first loop, re-plot the last fit attempt. This helps a lot
;when trying to decide between two different fits.
If Keyword_set(last) Then Begin
  fplast = Plot(Poly(keeper, fit.poly), color=pcolors.white, thick=0.5, $
    linestyle=1, /current , /overplot)
Endif

While chkpt.bravo Ne 1 Do Begin
  ;Initialize variable(s) & query user for polynomial order to use for the fit.
  forder = 0
  Print, ''
  Read, forder, prompt = '>>>>> Enter polynomial order to use (integer): '

  ;Ensure forder is a longword integer, or can be converted to one.
  ;If forder is a funky type, e.g. unsigned, then throw an error.
  If Array_equal(0, Isa(forder, /scalar)) || $
    Array_equal(1, rtype.adtypes.Contains(Typename(forder)), /not_equal) Then Begin
    Print, '>>>>> Warning: Invalid response'

    ;change chkpt value to restart while loop.
    chkpt.bravo = 0
  Endif Else Begin
    fit.order = Long(forder)

    ;If the value of forder is less than zero or greater than 11, reject it.
    If (fit.order Gt 0) And (fit.order Le 11) Then Begin
      chkpt.bravo = 1
    Endif Else Begin
      ;print some stuff & change chkpt value to restart while loop.
      Print, '>>>>> Warning: Polynomial order out of range'
      chkpt.bravo = 0
      fit.order = 0
    Endelse
  Endelse
Endwhile
```

```
;Calculate a polynomial fit to the data, of the order specified above.
fit.poly.Add, Poly_fit(fit.indep.Toarray(), fit.dep.Toarray(), fit.order), /extract

;Overplot the new fit in the existing graphics window
fpnew = Plot(Poly(keeper, fit.poly), color=pcolors.orange, /current, /overplot)

;Set 'last' to one, so that this fit will be replotted if the loop runs again.
last = 1

;Make sure the fit looks good...
While chkpt.charlie Ne 1 Do Begin

  ;Initialize variable(s) & Query user if fit is satisfactory.
  answer = ''
  Print, ''
  Read, answer, prompt = '>>>>> satisfied? y/n/quit: '

  ;Respond accordingly...
  Case 1 Of
    Strmatch(answer, 'y', /FOLD_CASE):Begin
      Print, ''
      Print, '>>>>> Message: Polynomial order accepted'
      Print, '>>>>> Message: Fit accepted'

      ;change chkpt values to exit both while loops.
      chkpt.alpha = 1
      chkpt.charlie = 1

      ;store the fit order in LSPEX
      exe_chk = Execute('lspex.' + scsign + '.order = fit.order')
      If exe_chk Ne 1 Then Message, '>>>>> Error: execution failure'
    End
    Strmatch(answer, 'n', /FOLD_CASE): Begin
      Print, ''
      Print, '>>>>> Message: Restarting GUI...'

      ;delete the fit parameters out of the 'FIT' structure
      fit.poly.Remove, /ALL

      ;close the graphics window
```

```
          w.Close

          ;change chkpt values to restart while loops from the top.
          chkpt.alpha = 0
          chkpt.charlie = 1
        End
        Strmatch(answer, 'quit', /FOLD_CASE): Begin
          Print, '>>>>> Alert: Exiting...'
          Print, ''

          ;close the graphics window & return.
          w.Close
          Return
        End
        Else: Begin
          Print, ''
          Print, '>>>>> Warning: Invalid response'

          ;change chkpt values to restart current while loop.
          chkpt.charlie = 0
        End
      Endcase
    Endwhile
  Endwhile

;Save the current graphics window as a .png file & close the window.
pic_name = Strtrim(rootpath,2) + '/' + Strtrim(callsign, 2) + '_fit.png'
w.Save, Strtrim(pic_name,2), resolution = 100
w.Close

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 2: − Subtract the baseline
; − Replot the data
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print some stuff.
Print, ''
Print, '>>>>> Message: subtracting baseline...'

;Subtract the polynomial fit to the baseline from the unsmoothed data &
```

```
;smooth the baselined data.
bled.wild = raw.wild − Poly(keeper, fit.poly)
bled.tame = Convol(bled.wild, slevel)

;Try and calculate some good limits for the ordinate when plotting.
;This technique is a little bit crude, and doesn't always work well.
m = Mean(bled.tame[1999:−2000])
ylo = m − (2.0 ∗ (m − Min(bled.tame[1999:−2000])))
yhi = m + (2.0 ∗ (Max(bled.tame[1999:−2000]) − m))

;Initialize a graphics window w/ standard dimensions.
w = Window(dimensions=pspex.windim, background_color=pcolors.black)

;Plot a dotted line at 0K, just for visual reference.
bpzero = Plot(keeper, Replicate(0.0, rtype.sml), color=pcolors.white, thick=1, $
  linestyle=1, /current , /overplot)

;Plot the baselined and smoothed spectrum.
bpone = Plot(keeper[100:−101], bled.tame[100:−101], color=pcolors.puke, $
  xrange=[0,rtype.sml], yrange=[ylo,yhi], xstyle=1, xmajor=5, xminor=3, ystyle=1, $
  yminor=3, ytitle='Tmb (K)', font_size=11, margin = [0.04,0.05,0.015,0.03], /current)
ax = bpone.axes
ax[2].minor = 0
ax[2].ticklen = ax[2].ticklen / 2d
ax[3].minor = 0
ax[3].ticklen = ax[3].ticklen / 2d

;Plot a vertical line at each of the limits set using Outerlimits.
For ii=0, nlimits−1 Do Begin
  bptwo = Plot([limits[ii], limits[ii]], [ylo−Abs(20∗ylo), yhi+Abs(20∗yhi)], $
    color=pcolors.teal, linestyle=1, thick=2, /current, /overplot)

  ;Plot each new bounded region as a different color, just to make things look nice.
  If (ii Mod 2) Eq 1 Then Begin
    bpthree = Plot(keeper[limits[ii−1]:limits[ii]], raw.tame[limits[ii−1]:limits[ii]], $
      color=pcolors.pink, linestyle=0, thick=1.5, /current, /overplot)
  Endif
Endfor

;Save the current graphics window as a .png file & close the window.
```

```
    pic_name = Strtrim(rootpath,2) + '/' + Strtrim(callsing,2) + '_newspec.png'
    w.Save, Strtrim(pic_name,2), resolution = 100
    w.Close

    ;Print some stuff & end.
    Print, ''
    Print, '>>>>> End Of Line <<<<<'
    Print, ''

End
```

14.9 Dreamweaver.pro

```
;+--------------------------------------------------+
;>>>>> Ancillary Program(s) <<<<<
;+--------------------------------------------------+

;+--------------------------------------------------+
;>>>>> Primary Program <<<<<
;+--------------------------------------------------+
;+
; NAME:
; Dreamweaver
; HYDRA Version 5.1
;
; PURPOSE:
; ->
;
; CALLING SEQUENCE:
; -> Dreamweaver,
;
; ARGUMENT(S):
; -> None
;
; KEYWORD(S):
; -> None
;
; OPTIONAL KEYWORD(S):
; -> Nloops: number of loops of the outer MC simulation
;
; -> Nslopps: number of loops of the inner MC simulation
;
; -> Nsigma: The distance from line center, in standard deviations, that
;    will be integrated when computing the total area line ratios.
;
;  -> Lowsnr (binary): Set to increase the degree to which the spectum is smoothed
; before it is plotted. Useful when fitting rare isotopologues or weak sources.
;
; -> Crapshoot (binary): Set to drastically increase the degree to which the
; spectum is smoothed before it is plotted. Overrides LOWSNR keyword.
; NOTE: This much smoothing should be avoided whenever possible.
; ->
;
;
; EXAMPLES:
; -> Dreamweaver
;
; OUTPUTS:
; ->
;
; COMMENTS:
; -> Only operates within the HYDRA 5.1 RTE.
; -> Filepath for the output directory is stored in OVERLORD and must
; be set using setdir.pro.
; -> Kill_rrl functionality is not currently supported.
;
; PROCEDURES/FUNCTIONS CALLED:
; -> looper.pro
; -> sloopstruct__define.pro
; -> timer__define.pro
;-
Pro Dreamweaver, Nloops=nloops, nsubloops=nsubloops, Nsigma=nsigma, $
  lowsnr=lowsnr, Crapshoot=crapshoot, Kill_rrl=kill_rrl
  Compile_opt IDL2
  Common overlord
  Common lspex
  Common mcspex
  On_error, 0
  !Except=1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 11 February, 2014
  ;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 15 June, 2014
  ;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 17 October, 2014
  ;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 21 June, 2015
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 25 July, 2016
  ;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 10 December, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 4.1 (V-spec) written by: N.N. Monson (UCLA). 19 September, 2017
```

```
;>>>>> Version 4.2 (V–spec) written by: N.N. Monson (UCLA). 31 May, 2018
;>>>>> Version 5.0 (V–spec) written by: N.N. Monson (UCLA). 25 November, 2018
;>>>>> Version 5.1 (V–spec) written by: N.N. Monson (UCLA). 17 January, 2019

;+––––––––––––––––––––––––––––––––––––––––––+
;>>>>> Usage Agreement <<<<<
;+––––––––––––––––––––––––––––––––––––––––––+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+––––––––––––––––––––––––––––––––––––––––––+
;>>>>> Developer's Notes <<<<<
;+––––––––––––––––––––––––––––––––––––––––––+

;Added automated optical depth estimation and error calculation.

;+––––––––––––––––––––––––––––––––––––––––––+
;>>>>> Limitations & Known Issues <<<<<
;+––––––––––––––––––––––––––––––––––––––––––+

;None known.

;+––––––––––––––––––––––––––––––––––––––––––+
;SECTION 0: – Check argument(s).
; – Set keyword defaults.
;+––––––––––––––––––––––––––––––––––––––––––+


If dirloc Eq '' Then Begin
  Message, '>>>>> error: common directory not set'
Endif

If Not Keyword_set(nloops) Then $
  nloops = 200 $
Else nloops = Long(nloops)

If Not Keyword_set(nsubloops) Then $
```

```
   nsubloops = 200 $
Else nsubloops = Long(nsubloops)

If Not Keyword_set(nsigma) Then $
   nsigma = 3.0 $
Else nsigma = Float(nsigma)

If Keyword_set(crapshoot) Then lowsnr=1




;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: −INITIALIZE 'README' FILE
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Count numer of existing monte−carlo run results,
;and name current run accordingly.
com_chk = File_search(Strtrim(Overlord.dirloc,2) + '/mcrun_*', count = n)
If n Lt 9 Then Begin
   runpath = Strtrim(Overlord.dirloc,2) + '/mcrun_0' + Strtrim(n+1, 2)
Endif Else runpath = Strtrim(Overlord.dirloc,2) + '/mcrun_' + Strtrim(n+1, 2)

;Create new directories for this run.
File_mkdir, runpath, /noexpand_path
File_mkdir, runpath + '/ref', /noexpand_path

;Open a new ascii file to print stuff to.
;This file will contain everyting that is printed to the command line,
;This way, one can go back later and see what was done to
;produce any given set of results.
Openw, lun2, runpath + '/results.txt', /get_lun
Printf, lun2, 'Run: ' + Strtrim(Systime(),2)
Printf, lun2, ' −> Baseline Orders (28,29,30): ' + Strtrim(lspex.order.s28,2) + $
   ', ' + Strtrim(lspex.order.s29,2) + ', ' + Strtrim(lspex.order.s30,2)
Printf, lun2, format = "(' −> Integration Width (sigma): +/− ', f−10.2)", nsigma
If Keyword_set(lowsnr) Then answer1 = 'Yes' Else answer1 = 'No'
Printf, lun2, format = "(' −> Low SNR?: ', A0)", answer1
If Keyword_set(crapshoot) Then answer2 = 'Yes' Else answer2 = 'No'
Printf, lun2, format = "(' −> Crapshoot?: ', A0)", answer2
```

```
;Open a new ascii file to print stuff to.
;This file will contain a 'readers digest' version of the
;information in the results.txt file.
;This way, one can go back later and see what was done to
;produce any given set of results.
Openw, lun3, runpath + '/summary.txt', /get_lun
Printf, lun3, 'Run: ' + Strtrim(Systime(),2)
Printf, lun3, ' -> Baseline Orders (28,29,30): ' + Strtrim(lspex.order.s28,2) + $
  ', ' + Strtrim(lspex.order.s29,2) + ', ' + Strtrim(lspex.order.s30,2)
Printf, lun3, format = "(' -> Integration Width (sigma): +/- ', f-10.2)", nsigma
Printf, lun3, format = "(' -> Low SNR?: ', A0)", answer1
Printf, lun3, format = "(' -> Crapshoot?: ', A0)", answer2

;+-----------------------------------------+
;SECTION 2: - Read in data
; - Pull info out of headers
;+-----------------------------------------+

;Initialize structure(s) and/or array(s)
keeper = Dindgen(rtype.sml)

;Loop once for each isotope of SiO.
Foreach iso, [28,29,30] Do Begin

  ;Define path to the firectory to load isotope data from.
  ;Define the filename too.
  rootpath = Strtrim(Overlord.dirloc,2) + '/' + Strtrim(iso,2) + 'sio'
  rfname = Strtrim(dtype,2) + '_' + Strtrim(iso,2) + 'sio.dat.gzip'

  ;copy the data file to the 'ref' folder created in section 1.
  ;This is so it will be accessable for reference purposes in the future.
  File_copy, Strtrim(rootpath,2) + Strtrim(rfname,2), $
    Strtrim(runpath,2) + '/ref/' + Strtrim(rfname,2)

  ;Identify all the various .png images created by HYDRA in overlord.dirloc
  pics = File_search(Strtrim(rootpath, 2), '*.png', /fully_qualify_path)

  ;Copy all found images to the 'ref' folder too.
  ;Again, this is for reference purposes in the future.
  If ~ Array_equal('', pics) Then Begin
    For ii=0, pics.length−1 Do Begin

      ;split up the path so the image name can be reused.
      cpname = Strsplit(Strtrim(pics[ii],2), '/', /extract)
      File_copy, pics[ii], Strtrim(runpath,2) + '/ref/' + Strtrim(cpname[−1],2)
    Endfor
  Endif

  ;Print some stuff
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, '>>>>> Loading ' + Strtrim(iso, 2) + 'sio data...'

  ;Open data file.
  Openr, lun, Strtrim(rootpath,2) + Strtrim(rfname,2), /get_lun, /compress

  ;Load data for the target spectrum.
  time = ''
  Readf, lun, time
  numsessions = 0
  Readf, lun, numsessions
  ntags = 0
  Readf, lun, ntags
  header = Strarr(ntags)
  Readf, lun, header
  names = Strarr(2)
  Readf, lun, names
  data = Dblarr(2, 6000, /nozero)
  Readf, lun, data, format = '(2e17.9)'
  Close, lun
  Free_lun, lun

  ;Put the tmb and vslr data into the appropriate mcspex structures &
  ;calculate the bin size of each pixel as mean the 'distance' to the pixel
  ;to the left and the pixel to the right.
  Case iso Of
    28:Begin
      mcspex.s28.wild = Transpose(data[1,*])
      mcspex.s28.vlsr = Transpose(data[0,*])
      mcspex.s28.vbin = (Shift(mcspex.s28.vlsr,1)−Shift(mcspex.s28.vlsr,−1)) / 2d
    End
```

```
    29:Begin
      mcspex.s29.wild = Transpose(data[1,*])
      mcspex.s29.vlsr = Transpose(data[0,*])
      mcspex.s29.vbin = (Shift(mcspex.s29.vlsr,1)−Shift(mcspex.s29.vlsr,−1)) / 2d
    End
    30:Begin
      mcspex.s30.wild = Transpose(data[1,*])
      mcspex.s30.vlsr = Transpose(data[0,*])
      mcspex.s30.vbin = (Shift(mcspex.s30.vlsr,1)−Shift(mcspex.s30.vlsr,−1)) / 2d
    End
  Endcase

  ;Print some stuff
  Print, '>>>>> ...Done'
  Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Print, ''
Endforeach

;Set the bins for the first and last last two pixels
;set equal to the neighboring bins.
mcspex.s28.vbin[0] = mcspex.s28.vbin[1]
mcspex.s28.vbin[−1] = mcspex.s28.vbin[−2]

mcspex.s29.vbin[0] = mcspex.s29.vbin[1]
mcspex.s29.vbin[−1] = mcspex.s29.vbin[−2]

mcspex.s30.vbin[0] = mcspex.s30.vbin[1]
mcspex.s30.vbin[−1] = mcspex.s30.vbin[−2]

If Keyword_set(kill_rrl) Then Begin

  ;Define path to the firectory to load the RRL data from. Define the filename too.
  rootpath = Strtrim(Overlord.dirloc,2) + '/rrl'
  rfname = Strtrim(dtype,2) + '_rrl.dat.gzip'

  ;copy the data file to the 'ref' folder created in section 1.
  ;This is so it will be accessable for reference purposes in the future.
  File_copy, Strtrim(rootpath,2) + Strtrim(rfname,2), $
    Strtrim(runpath,2) + '/ref/' + Strtrim(rfname,2)
```

```
;Identify all the various .png images created by HYDRA in overlord.dirloc
pics = File_search(Strtrim(rootpath, 2), '*.png', /fully_qualify_path)

;Copy all found images to the 'ref' folder too.
;Again, this is for reference purposes in the future.
If ~ Array_equal('', pics) Then Begin
  For ii=0, pics.length-1 Do Begin

    ;split up the path so the image name can be reused.
    cpname = Strsplit(Strtrim(pics[ii],2), '/', /extract)
    File_copy, pics[ii], Strtrim(runpath,2) + '/ref/' + Strtrim(cpname[-1],2)
  Endfor
Endif

;Print some stuff
Print, '>>>>> Alert: Kill_rrl keyword set...'
Print, '>>>>> Loading RRL data...'

;Open data file.
Openr, lun, Strtrim(rootpath,2) + Strtrim(rfname,2), /get_lun, /compress

;Load data for the target spectrum.
time = ''
Readf, lun, time
numsessions = 0
Readf, lun, numsessions
ntags = 0
Readf, lun, ntags
header = Strarr(ntags)
Readf, lun, header
names = Strarr(2)
Readf, lun, names
data = Dblarr(2, 6000, /nozero)
Readf, lun, data, format = '(2e17.9)'
Close, lun
Free_lun, lun

;Put the tmb and vslr data into the appropriate mcspex structures.
mcspex.rrl.wild = Transpose(data[1,*])
mcspex.rrl.vlsr = Transpose(data[0,*])
```

```
  ;Bin size of each pixel is calculated as mean the 'distance' to the pixel
  ;to the left and the pixel to the right. Also, set the bins for the first
  ;and last last two pixels set equal to the neighboring bins.
  mcspex.rrl.vbin = (Shift(mcspex.rrl.vlsr,1)−Shift(mcspex.rrl.vlsr,−1)) / 2d
  mcspex.rrl.vbin[0] = mcspex.rrl.vbin[1]
  mcspex.rrl.vbin[−1] = mcspex.rrl.vbin[−2]

  ;Print some stuff
  Print, '>>>>> ...Done'
  Print, ''
  Print, '>>>>> Fitting RRL data...'

  ;Initialize structure(s) and/or array(s)
  rrl = { indep: List(), $
    dep: List(), $
    poly: Dblarr(1, lspex.rrl.order) }

  ;Use limit data to express limits in terms of dependent and independent variables.
  For ii=1, lspex.rrl.nlimits−1, 2 Do Begin
    rrl.indep.Add, keeper[limits[ii−1]:limits[ii]], /extract
    rrl.dep.Add, mcspex.rrl.wild[limits[ii−1]:limits[ii]], /extract
  Endfor

  ;Calculate a polynomial fit to the data, of the order specified using polybase.pro
  rrl.poly = Poly_fit(rrl.indep.Toarray(), rrl.dep.Toarray(), lspex.rrl.order)

  ;Subtract the polynomial fit to the baseline from the unsmoothed data &
  ;smooth the baselined data.
  mcspex.rrl.tame = mcspex.rrl.wild − Poly(keeper, rrl.poly, /double)
  mcspex.rrl.tame = Convol(Temporary(mcspex.rrl.tame), Savgol(100,100,0,3))

  ;Print some stuff.
  Print, '>>>>> ...Done'
  Print, ''
  Print, '>>>>> WARNING: This functionality is not currently supported'
  Print, '>>>>> WARNING: Nothing has been subtracted from the 29−SiO spectrum'
  Print, ''
Endif
```

```
;Calculate the total number of limits
tlims = lspex.s28.numlims + lspex.s29.numlims + lspex.s30.numlims

;Initialize arrays/structures
randarr = Dblarr(nloops, tlims, /no_zero)
rnoise = { mcutility, s28: Dblarr(nloops, lspex.s28.numlims, /no_zero), $
  s29: Dblarr(nloops, lspex.s29.numlims, /no_zero), $
  s30: Dblarr(nloops, lspex.s30.numlims, /no_zero) }
tweaker = Create_struct(name = 'mcutility')
modlims = Create_struct(name = 'mcutility')

;Copy limit data to modlims structure
modlims.s28 = Rebin(Transpose(lspex.s28.xlims.Toarray()), nloops, lspex.s28.numlims)
modlims.s29 = Rebin(Transpose(lspex.s29.xlims.Toarray()), nloops, lspex.s29.numlims)
modlims.s30 = Rebin(Transpose(lspex.s30.xlims.Toarray()), nloops, lspex.s30.numlims)

;Generate random number arrays and assign the correct nunmber for
;each isotopologue (number required dependent on number of limits).
randarr = Randomn(Superseed(5), [nloops, tlims], /double)
rnoise.s28 = randarr[*, 0:lspex.s28.numlims-1]
rnoise.s29 = randarr[*, lspex.s28.numlims:lspex.s28.numlims+lspex.s29.numlims-1]
rnoise.s30 = randarr[*, lspex.s28.numlims+lspex.s29.numlims:-1]


;Determine how much each limit will move, using an array of random numbers,
;the width of each window, and a constant, 0.025 in this case.
;(i.e. the width of fit windows vary by a normally distributed random
;variable, with sigma = 5% of the window width).

;For the 28-sio line:
For ii=1, lspex.s28.numlims-1, 2 Do Begin
  tweaker.s28[*, ii-1:ii] = rnoise.s28[*, ii-1:ii] * 0.025d * $
    (lspex.s28.xlims[ii] - lspex.s28.xlims[ii-1])
Endfor

;For the 29-sio line:
For ii=1, lspex.s29.numlims-1, 2 Do Begin
  tweaker.s29[*, ii-1:ii] = rnoise.s29[*, ii-1:ii] * 0.025d * $
    (lspex.s29.xlims[ii] - lspex.s29.xlims[ii-1])
Endfor
```

```
;For the 30−sio line:
For ii=1, lspex.s30.numlims−1, 2 Do Begin
  tweaker.s30[*, ii−1:ii] = rnoise.s30[*, ii−1:ii] * 0.025d * $
    (lspex.s30.xlims[ii] − lspex.s30.xlims[ii−1])
Endfor

;Move the limits by the ammount calculated above.
modlims.s28 = Temporary(modlims.s28) + tweaker.s28
modlims.s29 = Temporary(modlims.s29) + tweaker.s29
modlims.s30 = Temporary(modlims.s30) + tweaker.s30

;Check to ensure none of the limits have beem moved beyond the range of the data.
;If any instances are detected, reset the values to the edge of the data.

;For the 28−sio line:
llimchk = Where(modlims.s28[*,0] Lt 1)
If llimchk Ne −1 Then Foreach jj, llimchk Do modlims.s28[jj,0] = 1
ulimchk = Where(modlims.s28[*,−1] Gt rtype.sml−1)
If ulimchk Ne −1 Then Foreach jj, ulimchk Do modlims.s28[jj,−1] = rtype.sml−1

;For the 29−sio line:
llimchk = Where(modlims.s29[*,0] Lt 1)
If llimchk Ne −1 Then Foreach jj, llimchk Do modlims.s29[jj,0] = 1
ulimchk = Where(modlims.s29[*,−1] Gt rtype.sml−1)
If ulimchk Ne −1 Then Foreach jj, ulimchk Do modlims.s29[jj,−1] = rtype.sml−1

;For the 30−sio line:
llimchk = Where(modlims.s30[*,0] Lt 1)
If llimchk Ne −1 Then Foreach jj, llimchk Do modlims.s30[jj,0] = 1
ulimchk = Where(modlims.s30[*,−1] Gt rtype.sml−1)
If ulimchk Ne −1 Then Foreach jj, ulimchk Do modlims.s30[jj,−1] = rtype.sml−1


Print, ''
Print, ' −> Sampling from gaussian distribution'
Print, ' −> Random number generation complete'
Print, ''
Print, ' −> Array initialization complete'
Print, ''
```

```
Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
Print, ' Commencing simulation...'
Print, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'


result = Dblarr(nloops*nsubloops, 22, /nozero)
proxy_result = Dblarr(nsubloops, 22, /nozero)

fit = { s28: { indep: List(), $
  dep: List(), $
  poly: Dblarr(1, lspex.s28.order) } , $
  s29: { indep: List(), $
  dep: List(), $
  poly: Dblarr(1, lspex.s29.order) } , $
  s30: { indep: List(), $
  dep: List(), $
  poly: Dblarr(1, lspex.s30.order) } }

;Initialize structure(s) and/or array(s)
time = { loop: { start: 0l, $
  stop: 0l, $
  avg: 0l }, $
  subloop: { start: 0l, $
  stop: 0l, $
  avg: 0l }, $
  rem: 0l, $
  tot: 0l }

;Initialize structure(s) and/or array(s)
result = Replicate(Create_struct(name = 'slstruct'), nsubloops, nloops)
slnoise = Randomn(Superseed(7), [2000, 3, nsubloops, nloops], /double)


time.loop.start = Systime(/seconds)
For loop = 0, nloops−1 Do Begin

  ;Reset all fitting data at the start of the new loop.
  Foreach jj, ['.s28', '.s29', '.s30'] Do Begin
    Foreach ii, ['.dep', '.indep']Do Begin
      exe1 = Execute('fit'+jj+ii+'.remove, /all')
    Endforeach
  Endforeach

  ;For the 28−SiO line, Use limit data in lspex to express limits
  ;in terms of dependent and independent variables.
  For ii=1, lspex.s28.numlims−1, 2 Do Begin
    fit.s28.indep.Add, keeper[modlims.s28[ii−1]:modlims.s28[ii]], /extract
    fit.s28.dep.Add, mcspex.s28.wild[modlims.s28[ii−1]:modlims.s28[ii]], /extract
  Endfor

  ;Fit a new baseline to the 28−SiO spectra.
  ;Save baselined spectrum to the mcspex structure to pass to dreamweaver
  fit.s28.poly = Poly_fit(fit.s28.indep.Toarray(), $
    fit.s28.dep.Toarray(), lspex.s28.order)
  mcspex.s28.tame = mcspex.s28.wild − Poly(keeper, fit.s28.poly)

  ;Calculate the per−pixel rms noise for the spectrum.
  nfit = Poly_fit(keeper[500:999], mcspex.s28.tame[500:999], 5, /double)
  mcspex.s28.rmsnt = Stddev(mcspex.s28.tame[500:999] − $
    Poly(keeper[500:999],fnit), /double)

  ;For the 29−SiO line, Use limit data in lspex to express limits
  ;in terms of dependent and independent variables.
  For ii=1, lspex.s29.numlims−1, 2 Do Begin
    fit.s29.indep.Add, keeper[modlims.s29[ii−1]:modlims.s29[ii]], /extract
    fit.s29.dep.Add, mcspex.s29.wild[modlims.s29[ii−1]:modlims.s29[ii]], /extract
  Endfor

  ;Fit a new baseline to the 29−SiO spectra.
  ;Save baselined spectrum to the mcspex structure to pass to dreamweaver
  fit.s29.poly = Poly_fit(fit.s29.indep.Toarray(), $
    fit.s29.dep.Toarray(), lspex.s29.order)
  mcspex.s29.tame = mcspex.s29.wild − Poly(keeper, fit.s29.poly)

  ;Calculate the per−pixel rms noise for the spectrum.
  nfit = Poly_fit(keeper[500:999], mcspex.s29.tame[500:999], 5, /double)
  mcspex.s29.rmsnt = Stddev(mcspex.s29.tame[500:999] − $
    Poly(keeper[500:999],fnit), /double)

  ;For the 30−SiO line, Use limit data in lspex to express limits
```

```
;in terms of dependent and independent variables.
For ii=1, lspex.s30.numlims−1, 2 Do Begin
  fit.s30.indep.Add, keeper[modlims.s30[ii−1]:modlims.s30[ii]], /extract
  fit.s30.dep.Add, mcspex.s30.wild[modlims.s30[ii−1]:modlims.s30[ii]], /extract
Endfor

;Fit a new baseline to the 30−SiO spectra.
;Save baselined spectrum to the mcspex structure to pass to dreamweaver
fit.s30.poly = Poly_fit(fit.s30.indep.Toarray(), $
  fit.s30.dep.Toarray(), lspex.s30.order)
mcspex.s30.tame = mcspex.s30.wild − Poly(keeper, fit.s30.poly)

;Calculate the per−pixel rms noise for the spectrum.
nfit = Poly_fit(keeper[500:999], mcspex.s30.tame[500:999], 5, /double)
mcspex.s30.rmsnt = Stddev(mcspex.s30.tame[500:999] − $
  Poly(keeper[500:999],fnit), /double)


;This feature is not currently supported... sorry!
;If Keyword_set(kill_rrl) Then LK_FMJ

;Start timing the subloops.
time.subloop.start = Systime(/seconds)

;Run subloops
result[*,loop] = Looper(slnoise=slnoise[*,*,*,loop], nsloops=nsubloops, $
  sig_width=nsigma, pix_shift=pixshift, low_snr=lowsnr, crap_shoot=crapshoot)

;Stop timing the loop and subloop.
time.subloop.stop = Systime(/seconds)
time.loop.stop = time.subloop.stop

;Store results from subloops into a common structure.
result[loop*nsubloops:((loop+1)*nsubloops)−1,*] = proxy_result

;Print a status update every 10 loops
If ((loop+1l) Mod (nloops/10l) Eq 0l) And (loop Ne (nloops−1l)) Then Begin

  ;Calculate loop, subloop, total, and total remaining times.
  time.tot = time.loop.stop − time.loop.start
```

```
    time.loop.avg = time.tot / Double(loop+1)
    time.rem = time.loop.avg * Double(nloops-loop+1)
    time.subloop.avg = (time.subloop.stop - time.subloop.start) / Double(nsubloops)

    ;Print some stuff
    Print, ''
    Print, '+-------------------------------------+'
    Print, ' Loop no. ' + Strtrim(loop+1l,2) + ' of ' + Strtrim(nloops,2) $
      + ' complete'
    Print, '+-------------------------------------+'
    Print, Format = "(' -> Total time elapsed (m): ', d-10.2)", time.tot / 60d
    Print, Format = "(' -> Mean time per loop (s): ', d-10.2)", time.loop.avg
    Print, Format = "(' -> Mean time per subloop (ms): ', d-10.2)", $
      1d3 * time.subloop.avg
    Print, ''
    Print, Format = "(' -> Estimated time to completion (m): ', d-10.2)", $
      time.rem / 60d
    Print, ''
  Endif
Endfor

;+-------------------------------------+
;+-------------------------------------+

;Calculate the net loop, subloop and total times.
time.tot = time.loop.stop - time.loop.start
time.loop.avg = time.tot / Double(nloops)
time.subloop.avg = time.tot / Double(nloops * nsubloops)

;Print some stuff
Print, ''
Print, '+-------------------------------------+'
Print, '+-------------------------------------+'
Print, ''
Print, '+-------------------------------------+'
Print, ' Completed ' + Strtrim(nloops,2) + ' loops, each with ' $
  + Strtrim(nsubloops,2) + ' subloops'
Print, '+-------------------------------------+'
Print, Format = "(' -> Total time elapsed (m): ', f-10.2)", time.tot / 60d
Print, Format = "(' -> Mean time per loop (s): ', f-10.2)", time.loop.avg
```

```
Print, Format = "(' -> Net time per subloop (ms): ', f-10.2)", $
  1d3 * time.subloop.avg

;Calculate the correlation coefficients and the covariance matrix
;for the tau corrected data.
pca_29 = result[*,18] - Mean(result[*,18], /double)
pca_30 = result[*,19] - Mean(result[*,19], /double)
cor_coeff = Correlate(pca_30, pca_29)
matrix = Transpose([[pca_30], [pca_29]])
covmatrix = Correlate(matrix, /covariance, /double)
evals = Eigenql(covmatrix, eigenvectors=evecs, /double)
evals = Sqrt(evals)
theta = Atan(evecs[1,0]/evecs[0,0]) * (360d / (2d * !Dpi))

;Print some stuff
Print, ''
Print, '+--------------------------------------+'
Print, ''
Print, ' -> Tau-corrected Error Ellipse properties: '
Print, ''
Print, Format = "(' -> Correlation Coefficient = ', f-7.3)", cor_coeff
Print, ''
Print, Format = "(' -> Major Axis (1 sigma) = ', d-9.4)", evals[0]
Print, Format = "(' -> Minor Axis (1 sigma) = ', d-9.4)", evals[1]
Print, Format = "(' -> Vector Angle (deg) = ', d-9.4)", theta

;Calculate the correlation coefficients and the covariance matrix
;for the uncorrected data.
upca_29 = result[*,16] - Mean(result[*,16], /double)
upca_30 = result[*,17] - Mean(result[*,17], /double)
ucor_coeff = Correlate(upca_30, upca_29)
umatrix = Transpose([[upca_30], [upca_29]])
ucovmatrix = Correlate(umatrix, /covariance, /double)
uevals = Eigenql(ucovmatrix, eigenvectors=uevecs, /double)
uevals = Sqrt(uevals)
utheta = Atan(uevecs[1,0]/uevecs[0,0]) * (360d / (2d * !Dpi))

;Print more stuff.
Print, ''
Print, '+--------------------------------------+'
```

```
Print, ''
Print, ' −> Uncorrected Error Ellipse properties: '
Print, ''
Print, Format = "(' −> Correlation Coefficient = ', f−7.3)", ucor_coeff
Print, ''
Print, Format = "(' −> Major Axis (1 sigma) = ', d−9.4)", uevals[0]
Print, Format = "(' −> Minor Axis (1 sigma) = ', d−9.4)", uevals[1]
Print, Format = "(' −> Vector Angle (deg) = ', d−9.4)", utheta

;Make and plot a histogram of the corrected delta−29 data.
ch29 = Histogram(result[*,18], locations=ch29locs, binsize=4.0d)
w1 = Window(dimensions = [600, 600], background_color = !Color.white)
p1 = Barplot(ch29locs, ch29, title="$\tau$ Corrected $\delta$'29", /current)

  ;Make and plot a histogram of the corrected delta−30 data.
  ch30 = Histogram(result[*,19], locations=ch30locs, binsize=4.0d)
w2 = Window(dimensions = [600, 600], background_color = !Color.white)
p2 = Barplot(ch30locs, ch30, title="$\tau$ Corrected $\delta$'30", /current)

  ;Make and plot a histogram of the estimated optica depths
  th = Histogram(result[*,20], locations=taulocs, binsize=.02d)
w3 = Window(dimensions = [600, 600], background_color = !Color.white)
p3 = Barplot(taulocs, th, title='Line Center Optical Depth', /current)

;Plot the delta values in triple isotope space.
w4 = Window(dimensions = [1200, 1200], background_color = !Color.white)
p4 = Plot(Findgen(1000)−200, Findgen(1000)−200, color = !Color.black, $
  xrange=[−199.9,799.9], yrange=[−199.9,799.9], xstyle=1, ystyle=1, $
  xminor=1, yminor=1, linestyle = 5, font_size = 30, thick = 2, $
  margin = [0.15, 0.15, 0.05, 0.05], ytitle = "$\delta$' 29", $
  xtitle = "$\delta$' 30", /current)
p4 = Plot(Findgen(1000)−200, Fltarr(1000), color = !Color.black, $
  linestyle = 1, thick = 1, /current, /overplot)
p4 = Plot(Fltarr(1000), Findgen(1000)−200, color = !Color.black, $
  linestyle = 1, thick = 1, /current, /overplot)
ax = p4.axes
ax[2].minor = 0.0
ax[2].ticklen = ax[2].ticklen / 2.0
ax[3].minor = 0.0
ax[3].ticklen = ax[3].ticklen / 2.0
```

```
;Plot the actual results of each simulation
p5 = Plot(result[*,19], result[*,18], color = !Color.black, $
   linestyle = 6, symbol = 'circle', sym_filled = 1, sym_size = 0.2, $
   sym_transparency = 70, sym_thick = 0.7, /current, /overplot)

;Plot the 1 sigma error ellipse.
p6 = Ellipse(Mean(result[*,19]), Mean(result[*,18]), /data, color=!Color.red, $
   major=evals[0], minor=evals[1], theta=theta, $
   thick=2.5, fill_background=0, /current, /overplot)



;Save each of the above plots as a .png file.
;Set !Except=0 so no exceptions are reported.
!Except=0
pic_name1 = Strtrim(name,2) + '/d29_histogram.png'
w1.Save, Strtrim(pic_name1,2), resolution = 150
pic_name2 = Strtrim(name,2) + '/d30_histogram.png'
w2.Save, Strtrim(pic_name2,2), resolution = 150
pic_name3 = Strtrim(name,2) + '/tau_histogram.png'
w3.Save, Strtrim(pic_name3,2), resolution = 150
pic_name4 = Strtrim(name,2) + '/results_tip.png'
w4.Save, Strtrim(pic_name4,2), resolution = 150
!Except=1

;Print a whole mess of stuff to both files.
Foreach ii, [lun2, lun3] Do Begin
   Printf, ii, ''
   Printf, ii, '+-------------------------------------+'
   Printf, ii, ' Completed ' + Strtrim(nloops,2) + ' loops, each with ' $
      + Strtrim(nsubloops,2) + ' subloops'
   Printf, ii, '+-------------------------------------+'
   Printf, ii, ''
   Printf, ii, Format = "(' -> Total time elapsed (m): ', f-10.3)", mtime
   Printf, ii, Format = "(' -> Mean time per loop (s): ', f-10.3)", ltime
   Printf, ii, Format = "(' -> Net time per subloop (ms): ', f-10.3)", sltime
   Printf, ii, ''
   Printf, ii, '+-------------------------------------+'
   Printf, ii, '+-------------------------------------+'
   Printf, ii, ''
```

```
Printf, ii, ' -> Tau-corrected Error Ellipse properties: '
Printf, ii, ''
Printf, ii, ' -> Correlation Coefficient = ' + Strtrim(cor_coeff,2)
Printf, ii, ''
Printf, ii, Format = "(' -> Major Axis (1 sigma) = ', d-9.4)", evals[0]
Printf, ii, Format = "(' -> Minor Axis (1 sigma) = ', d-9.4)", evals[1]
Printf, ii, Format = "(' -> Vector Angle (deg) = ', d-9.4)", theta
Printf, ii, ''
Printf, ii, '+----------------------------------------+'
Printf, ii, ''
Printf, ii, ' -> Uncorrected Error Ellipse properties: '
Printf, ii, ''
Printf, ii, ' -> Correlation Coefficient = ' + Strtrim(ucor_coeff,2)
Printf, ii, ''
Printf, ii, Format = "(' -> Major Axis (1 sigma) = ', d-9.4)", uevals[0]
Printf, ii, Format = "(' -> Minor Axis (1 sigma) = ', d-9.4)", uevals[1]
Printf, ii, Format = "(' -> Vector Angle (deg) = ', d-9.4)", utheta
Printf, ii, ''
Printf, ii, '+----------------------------------------+'
Printf, ii, '+----------------------------------------+'
Printf, ii, ''
Printf, ii, [['Column Key & Stats (mean and stddev)'], [''], $
  ['00 : tmb_28 - ' + Strtrim(Mean(result[*,0], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,0], /double))], $
  ['01 : tmb_29 - ' + Strtrim(Mean(result[*,1], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,1], /double))], $
  ['02 : tmb_30 - ' + Strtrim(Mean(result[*,2], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,2], /double))], $
  ['03 : fwhm_28 - ' + Strtrim(Mean(result[*,3], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,3], /double))], $
  ['04 : fwhm_29 - ' + Strtrim(Mean(result[*,4], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,4], /double))], $
  ['05 : fwhm_30 - ' + Strtrim(Mean(result[*,5], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,5], /double))], $
  ['-- : rmsnt_28 - ' + Strtrim(Mean(result[*,6], /double))], $
  ['-- : rmsnt_29 - ' + Strtrim(Mean(result[*,7], /double))], $
  ['-- : rmsnt_30 - ' + Strtrim(Mean(result[*,8], /double))], $
  ['06 : area_28 - ' + Strtrim(Mean(result[*,9], /double))+'+/- '+ $
  Strtrim(Stddev(result[*,9], /double))], $
  ['07 : area_29 - ' + Strtrim(Mean(result[*,10], /double))+'+/- '+ $
```

```
    Strtrim(Stddev(result[*,10], /double))], $
    ['08 : area_30 − ' + Strtrim(Mean(result[*,11], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,11], /double))], $
    ['09 : ratio_2829 − ' + Strtrim(Mean(result[*,12], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,12], /double))], $
    ['10 : ratio_2830 − ' + Strtrim(Mean(result[*,13], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,13], /double))], $
    ['11 : tcratio_2829 − ' + Strtrim(Mean(result[*,14], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,14], /double))], $
    ['12 : tcratio_2830 − ' + Strtrim(Mean(result[*,15], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,15], /double))], $
    ['13 : d29_solar − ' + Strtrim(Mean(result[*,16], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,16], /double))], $
    ['14 : d30_solar − ' + Strtrim(Mean(result[*,17], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,17], /double))], $
    ['15 : tcd29_solar − ' + Strtrim(Mean(result[*,18], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,18], /double))], $
    ['16 : tcd30_solar − ' + Strtrim(Mean(result[*,19], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,19], /double))], $
    ['17 : mtau − ' + Strtrim(Mean(result[*,20], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,20], /double))], $
    ['18 : corrfac − ' + Strtrim(Mean(result[*,21], /double))+'+/− '+ $
    Strtrim(Stddev(result[*,21], /double))]]
  Printf, ii, ''
  Printf, ii, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Printf, ii, '+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+'
  Printf, ii, ''
Endforeach

;Print the actual data to one of the files.
;Then close both and relese the lun numbers.
Printf, lun2, Transpose([[result[*,0:5]],[result[*,9:21]]]), format = '(19d13.7)'
Close, lun2
Free_lun, lun2
Close, lun3
Free_lun, lun3

;Compress the file with the actual data.
Spawn, 'gzip −f ' + Strtrim(runpath, 2) + '/results.txt'
```

```
  ;Print some jazz and end.
  Print, ''
  Print, ' -> Summary & Results files written to:'
  Print, Strtrim(runpath, 2)
  Print, ''
  Print, '>>>>> Alert: Data reduction complete.'
  Print, ''
  Print, ''
  Print, '>>>>> End of Line <<<<<'
  Print, ''

End
```

14.10 Looper.pro

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
;Computes isotopic ratios from the supplied ascii datafiles.
;−

Function Looper, Slnoise=slnoise, Nsloops=nsloops, Fit_pix=fit_pix, $
  Sig_width=sig_width, Low_snr=low_snr, Crap_shoot=crap_shoot
  Compile_opt IDL2
  Common overlord
  Common mcspex
  Common pspex
  Common rtype
  On_error, 0
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 11 February, 2014
  ;>>>>> Version 2.1 written by: N.N. Monson (UCLA). 15 June, 2014
  ;>>>>> Version 2.2 written by: N.N. Monson (UCLA). 17 October, 2014
  ;>>>>> Version 2.3 written by: N.N. Monson (UCLA). 21 June, 2015
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 25 July, 2016
  ;>>>>> Version 3.1 written by: N.N. Monson (UCLA). 10 December, 2016
  ;>>>>> Version 4.0 (V−spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 4.1 (V−spec) written by: N.N. Monson (UCLA). 19 September, 2017
  ;>>>>> Version 4.2 (V−spec) written by: N.N. Monson (UCLA). 31 May, 2018
  ;>>>>> Version 5.0 (V−spec) written by: N.N. Monson (UCLA). 25 November, 2018
  ;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 17 January, 2019

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;>>>>> Usage Agreement <<<<<
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Developer's Notes <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Issues <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;None known.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;
If Keyword_set(fit_pix) Then Begin
  If (fit_pix Lt 50l) Or (fit_pix Gt 1000l) Then Begin
    Message, '>>>>> error: keyword fit_pix set out of range'
  Endif Else fit_pix = fit_pix
Endif Else fit_pix = 300l

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: − Copy data out of mcspex
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

kurtz = { s28: { tmb: Dblarr(rtype.sml/3, /nozero), $
  vlsr: Dblarr(rtype.sml/3, /nozero), $
  vbin: Dblarr(rtype.sml/3, /nozero), $
  rmsnt: 0d }, $
  s29: { tmb: Dblarr(rtype.sml/3, /nozero), $
  vlsr: Dblarr(rtype.sml/3, /nozero), $
  vbin: Dblarr(rtype.sml/3, /nozero), $
  rmsnt: 0d }, $
  s30: { tmb: Dblarr(rtype.sml/3, /nozero), $
  vlsr: Dblarr(rtype.sml/3, /nozero), $
  vbin: Dblarr(rtype.sml/3, /nozero), $
  rmsnt: 0d } }

;Copy tmb data out of mcspex to kurtz.
kurtz.s28.tmb = mcspex.s28.tame[2000:3999]
kurtz.s29.tmb = mcspex.s29.tame[2000:3999]
kurtz.s30.tmb = mcspex.s30.tame[2000:3999]

;Copy the rms noise temp data out of mcspex
kurtz.s28.rmsnt = mcspex.s28.rmsnt
kurtz.s29.rmsnt = mcspex.s29.rmsnt
kurtz.s30.rmsnt = mcspex.s30.rmsnt

;Copy the velocity and velocity bin data out of mcspex
kurtz.s28.vlsr = mcspex.s28.vlsr[2000:3999]
kurtz.s29.vlsr = mcspex.s29.vlsr[2000:3999]
kurtz.s30.vlsr = mcspex.s30.vlsr[2000:3999]

kurtz.s28.vbin = mcspex.s28.vbin[2000:3999]
kurtz.s29.vbin = mcspex.s29.vbin[2000:3999]
kurtz.s30.vbin = mcspex.s30.vbin[2000:3999]

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 2: − Loop MC simulation
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Initialize structure(s) and/or array(s)
pix_space = Dindgen(rtype.sml/3)
sldata = Replicate(Create_struct(name = 'slstruct'), nsloops)

For ii = 0l, nsloops−1l Do Begin

  ;Loop this section 3 times, once for each isotopologue!
  Foreach iso, [28,29,30] Do Begin
    Case iso Of
      28:Begin
        ;Resample the 28−SiO spectrum, using the RMS noise temp
        sldata[ii].s28.tmb = kurtz.s28.tmb + (slnoise[*,0,ii] * kurtz.s28.rmsnt)
```

```
;Determine level of smoothing, based on keywords.
If Keyword_set(low_snr) Then Begin
  slevel = Savgol(30,30,0,4, /double)
Endif Else slevel = Savgol(20,20,0,4, /double)

;Apply smoothing to the resampled spectrum.
sldata[ii].s28.stmb = Convol(sldata[ii].s28.tmb, slevel)

;Convert the antenna temperature to velocity space
sldata[ii].s28.rstmb = sldata[ii].s28.tmb * kurtz.s28.vbin

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Locate the point in the spectrum w/ the greatest magnitude
;(I assume this point corresponds to the SiO peak) & Establish the
;region over which to fit a gaussian to the peak.
;This is done to help parameterize the main isotope line in velocity space.
;WARNING: This section may need revising to get a proper
;sampling of the peak for fitting and plotting i.e. to ensure
;the whole peak is included, but neighboring peaks/noise are omitted.
sldata[ii].s28.tbeam = Max(sldata[ii].s28.stmb[1000−fit_pix:1000+fit_pix], $
  sldata[ii].s28.ploc)
peak_28 = Max(sldata[ii].s28.stmb[500 − fit_pix : 500 + fit_pix], pix_loc_28)

;Set fit boundaries
sldata[ii].s28.lfit = sldata[ii].s28.ploc + 1000 − (2 * fit_pix)
sldata[ii].s28.hfit = sldata[ii].s28.ploc + 1000

;Calculate sigma width
result = Gaussfit(sldata[ii].s28.vlsr[sldata[ii].s28.lfit:sldata[ii].s28.hfit], $
  sldata[ii].s28.tmb[sldata[ii].s28.lfit:sldata[ii].s28.hfit], f_terms, nterms = 3)
sigma = f_terms[2]

;Establish boundaries (in terms of sigma widths) for the integration window.
sldata[ii].s28.wxmin = f_terms[1] − (losig * sigma)
sldata[ii].s28.wxmax = f_terms[1] + (hisig * sigma)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Redundant steps, to make sure all is done equally between isotopologues
```

```
  ;Find 28si peak center & equivalent boundaries (in terms of 28sio sigma widths)
  ;for the integration window & get temperature at the identified peak center.
  sldata[ii].s28.bound = Where((sldata[ii].s28.vlsr Gt sldata[ii].s28.wxmin) $
    And (sldata[ii].s28.vlsr Lt sldata[ii].s28.wxmax))

  sldata[ii].s28.tbeam = Max(sldata[ii].s28.stmb[sldata[ii].s28.bound], $
    sldata[ii].s28.ploc)
  sldata[ii].s28.ploc = sldata[ii].s28.ploc + sldata[ii].s28.bound.Min()
  sldata[ii].s28.lcenter = sldata[ii].s28.vlsr[sldata[ii].s28.ploc]

  ;Establish boundaries (in terms of sigma widths) for the integration window.
  sldata[ii].s28.wxmin = sldata[ii].s28.lcenter − (sig_width ∗ sigma)
  sldata[ii].s28.wxmax = sldata[ii].s28.lcenter + (sig_width ∗ sigma)

  ;calculate the FWHM width of the line
  sldata[ii].s28.hml = Min(Where(sldata[ii].s28.stmb[sldata[ii].s28.bound] $
    Gt sldata[ii].s28.tbeam/2d) + sldata[ii].s28.bound.Min())
  sldata[ii].s28.hmh = Max(Where(sldata[ii].s28.stmb[sldata[ii].s28.bound] $
    Gt sldata[ii].s28.tbeam/2d) + sldata[ii].s28.bound.Min())

  ;calculate the FWHM width of the line in velocity−space
  sldata[ii].s28.vsfwhm = sldata[ii].s28.vlsr[sldata[ii].s28.hml] − $
    sldata[ii].s28.vlsr[sldata[ii].s28.hmh]

  ;Convert the integration window bounds to velocity space &
  ;calculate the area under the re−binned line
  junk = Min(Abs(sldata[ii].s28.vlsr − sldata[ii].s28.wxmin), sldata[ii].s28.wblo)
  junk = Min(Abs(sldata[ii].s28.vlsr − sldata[ii].s28.wxmax), sldata[ii].s28.wbhi)

  sldata[ii].s28.area = Total(sldata[ii].s28.rstmb[sldata[ii].s28.wbhi:$
    sldata[ii].s28.wblo], /double)

End
29:Begin
  ;Resample the 29−SiO spectrum, using the RMS noise temp
  sldata[ii].s29.tmb = kurtz.s29.tmb + (slnoise[∗,1,ii] ∗ kurtz.s29.rmsnt)

  ;Determine level of smoothing, based on keywords.
  If Keyword_set(low_snr) Then Begin
    If Keyword_set(crap_shoot) Then Begin
    slevel = Savgol(40,40,0,3, /double)
  Endif Else slevel = Savgol(40,40,0,4, /double)
Endif Else slevel = Savgol(30,30,0,4, /double)

;Apply smoothing to the resampled spectrum.
sldata[ii].s29.stmb = Convol(sldata[ii].s29.tmb, slevel)

;Convert the antenna temperature to velocity space
sldata[ii].s29.rstmb = sldata[ii].s29.tmb * kurtz.s29.vbin

;Find 29si peak center & equivalent boundaries (in terms of 28sio sigma widths)
;for the integration window & get temperature at the identified peak center.
sldata[ii].s29.bound = Where((sldata[ii].s29.vlsr Gt sldata[ii].s28.wxmin) $
  And (sldata[ii].s29.vlsr Lt sldata[ii].s28.wxmax))

sldata[ii].s29.tbeam = Max(sldata[ii].s29.stmb[sldata[ii].s29.bound], $
  sldata[ii].s29.ploc)
sldata[ii].s29.ploc = sldata[ii].s29.ploc + sldata[ii].s29.bound.Min()
sldata[ii].s29.lcenter = sldata[ii].s29.vlsr[sldata[ii].s29.ploc]

;Establish boundaries (in terms of sigma widths) for the integration window.
sldata[ii].s29.wxmin = sldata[ii].s29.lcenter − (sig_width * sigma)
sldata[ii].s29.wxmax = sldata[ii].s29.lcenter + (sig_width * sigma)

;calculate the FWHM width of the line
sldata[ii].s29.hml = Min(Where(sldata[ii].s29.stmb[sldata[ii].s29.bound] $
  Gt sldata[ii].s29.tbeam/2d) + sldata[ii].s29.bound.Min())
sldata[ii].s29.hmh = Max(Where(sldata[ii].s29.stmb[sldata[ii].s29.bound] $
  Gt sldata[ii].s29.tbeam/2d) + sldata[ii].s29.bound.Min())

;calculate the FWHM width of the line in velocity−space
sldata[ii].s29.vsfwhm = sldata[ii].s29.vlsr[sldata[ii].s29.hml] − $
  sldata[ii].s29.vlsr[sldata[ii].s28.hmh]

;Convert the integration window bounds to velocity space &
;calculate the area under the re−binned line
junk = Min(Abs(sldata[ii].s29.vlsr − sldata[ii].s29.wxmin), sldata[ii].s29.wblo)
junk = Min(Abs(sldata[ii].s29.vlsr − sldata[ii].s29.wxmax), sldata[ii].s29.wbhi)

sldata[ii].s29.area = Total(sldata[ii].s29.rstmb[sldata[ii].s29.wbhi:$
    sldata[ii].s29.wblo], /double)

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Set fit boundaries
  sldata[ii].s29.lfit = sldata[ii].s29.ploc − fit_pix
  sldata[ii].s29.hfit = sldata[ii].s29.ploc + fit_pix

  ;Fit and re−fit the profile in velocity space
  junk = Gaussfit(sldata[ii].s29.vlsr[sldata[ii].s29.lfit:sldata[ii].s29.hfit], $
    sldata[ii].s29.stmb[sldata[ii].s29.lfit:sldata[ii].s29.hfit], guess_terms, nterms = 3)
  junk = Gaussfit(sldata[ii].s29.vlsr[sldata[ii].s29.lfit:sldata[ii].s29.hfit], $
    sldata[ii].s29.tmb[sldata[ii].s29.lfit:sldata[ii].s29.hfit], vs_terms, $
    estimates = guess_terms, nterms = 3)

  ;Fit and re−fit the profile in pixel space
  junk = Gaussfit(pix_space[sldata[ii].s29.lfit:sldata[ii].s29.hfit], $
    sldata[ii].s29.stmb[sldata[ii].s29.lfit:sldata[ii].s29.hfit], guess_terms, nterms = 3)
  junk = Gaussfit(pix_space[sldata[ii].s29.lfit:sldata[ii].s29.hfit], $
    sldata[ii].s29.tmb[sldata[ii].s29.lfit:sldata[ii].s29.hfit], pix_terms, $
    estimates = guess_terms, nterms = 3)

End
30:Begin
  ;Resample the 30−SiO spectrum, using the RMS noise temp
  sldata[ii].s30.tmb = kurtz.s30.tmb + (slnoise[*,2,ii] * kurtz.s30.rmsnt)

  ;Determine level of smoothing, based on keywords.
  If Keyword_set(low_snr) Then Begin
    If Keyword_set(crap_shoot) Then Begin
      slevel = Savgol(50,50,0,3, /double)
    Endif Else slevel = Savgol(50,50,0,4, /double)
  Endif Else slevel = Savgol(40,40,0,4, /double)

  ;Apply smoothing to the resampled spectrum.
  sldata[ii].s30.stmb = Convol(sldata[ii].s30.tmb, slevel)

  ;Convert the antenna temperature to velocity space
  sldata[ii].s30.rstmb = sldata[ii].s30.tmb * kurtz.s30.vbin
```

```
      ;Find 30si peak center & equivalent boundaries (in terms of 30sio sigma widths)
      ;for the integration window & get temperature at the identified peak center.
      sldata[ii].s30.bound = Where((sldata[ii].s30.vlsr Gt sldata[ii].s28.wxmin) $
        And (sldata[ii].s30.vlsr Lt sldata[ii].s28.wxmax))

      sldata[ii].s30.tbeam = Max(sldata[ii].s30.stmb[sldata[ii].s30.bound], $
        sldata[ii].s30.ploc)
      sldata[ii].s30.ploc = sldata[ii].s30.ploc + sldata[ii].s30.bound.Min()
      sldata[ii].s30.lcenter = sldata[ii].s30.vlsr[sldata[ii].s30.ploc]

      ;Establish boundaries (in terms of sigma widths) for the integration window.
      sldata[ii].s30.wxmin = sldata[ii].s30.lcenter − (sig_width ∗ sigma)
      sldata[ii].s30.wxmax = sldata[ii].s30.lcenter + (sig_width ∗ sigma)

      ;calculate the FWHM width of the line
      sldata[ii].s30.hml = Min(Where(sldata[ii].s30.stmb[sldata[ii].s30.bound] $
        Gt sldata[ii].s30.tbeam/2d) + sldata[ii].s30.bound.Min())
      sldata[ii].s30.hmh = Max(Where(sldata[ii].s30.stmb[sldata[ii].s30.bound] $
        Gt sldata[ii].s30.tbeam/2d) + sldata[ii].s30.bound.Min())

      ;calculate the FWHM width of the line in velocity−space
      sldata[ii].s30.vsfwhm = sldata[ii].s30.vlsr[sldata[ii].s30.hml] − $
        sldata[ii].s30.vlsr[sldata[ii].s28.hmh]

      ;Convert the integration window bounds to velocity space &
      ;calculate the area under the re−binned line
      junk = Min(Abs(sldata[ii].s30.vlsr − sldata[ii].s30.wxmin), sldata[ii].s30.wblo)
      junk = Min(Abs(sldata[ii].s30.vlsr − sldata[ii].s30.wxmax), sldata[ii].s30.wbhi)

      sldata[ii].s30.area = Total(sldata[ii].s30.rstmb[sldata[ii].s30.wbhi:$
        sldata[ii].s30.wblo], /double)
    End
  Endcase
Endforeach

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 4: − Calculate integrated area ratios.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Calculate the 28Si/29Si and 28Si/30Si ratios using the velocity−space areas.
```

```
sldata[ii].s29.pratio = sldata[ii].s28.area / sldata[ii].s29.area
sldata[ii].s30.pratio = sldata[ii].s28.area / sldata[ii].s30.area

;Impose frequency correction factor (GHz).
sldata[ii].s29.ratio = ((4.287982d / 4.342385d) ^ 3d) * sldata[ii].s29.pratio
sldata[ii].s30.ratio = ((4.237334d / 4.342385d) ^ 3d) * sldata[ii].s30.pratio

;Convert the corrected area ratio to delta values.
;solar Si values from Lodders (2003)
sldata[ii].s29.delta = 1000.0d * Alog(19.6939d / sldata[ii].s29.ratio)
sldata[ii].s30.delta = 1000.0d * Alog(29.8753d/ sldata[ii].s30.ratio)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 5: − Estimate line profile and tau values
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Calculate line profile, using gaussian fit to 29−SiO line
pixgauss = Gaussian_function(pix_terms[2], maximum=pix_terms[0], width=1000, /double)
vs_gphi = pixgauss / Total(pixgauss * kurtz.s29.vbin, /double)

;Scale lines by the area ratios
sldata[ii].s28.scaled = sldata[ii].s28.stmb
sldata[ii].s29.scaled = sldata[ii].s29.stmb * sldata[ii].s29.pratio
sldata[ii].s30.scaled = sldata[ii].s30.stmb * sldata[ii].s29.pratio

;Calculate gamma values using scaled lines
sldata[ii].s29.gamma = Mean(sldata[ii].s29.scaled[sldata[ii].s29.ploc−15: $
  sldata[ii].s29.ploc+15] / sldata[ii].s28.scaled[sldata[ii].s28.ploc−15: $
  sldata[ii].s28.ploc+15], /double)
sldata[ii].s30.gamma = Mean(sldata[ii].s30.scaled[sldata[ii].s30.ploc−15: $
  sldata[ii].s30.ploc+15] / sldata[ii].s28.scaled[sldata[ii].s28.ploc−15: $
  sldata[ii].s28.ploc+15], /double)

;Calculate tau using both thin line profiles.
sldata[ii].s29.tau = 5d * (4.342376d10 / (100.0d *!Const.c)) * $
  (sldata[ii].s29.gamma − 1d)
sldata[ii].s30.tau = 5d * (4.342376d10 / (100.0d *!Const.c)) * $
  (sldata[ii].s30.gamma − 1d)

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
    ;SECTION 6: − Calculate correction for tau and apply.
    ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

    sldata[ii].s29.weight = (sldata[ii].s29.tbeam/kurtz.s29.rmsnt)^2d
    sldata[ii].s30.weight = (sldata[ii].s30.tbeam/kurtz.s30.rmsnt)^2d

    ;Calculate the mean value of tau.
    sldata[ii].mtau = ((sldata[ii].s29.weight * tau29) + (sldata[ii].s30.weight * tau30)) / $
      (sldata[ii].s29.weight + sldata[ii].s30.weight)



    ;Estimate tau_v using phi
    tau_v = sldata[ii].mtau * Sqrt(2d * !Dpi) * vs_terms[2] * vs_gphi

    ;Calculate correction factor
    sldata[ii].cfactor = Total(tau_v * kurtz.s28.vbin, /double) / $
      Total((1d − Exp(−1d * tau_v)) * kurtz.s28.vbin, /double)

    ;Correct the 28Si/29Si ratio for optical depth
    sldata[ii].s29.tcratio = sldata[ii].s29.ratio * sldata[ii].cfactor

    ;Convert the corrected area ratio to delta values.
    ;solar Si values from Lodders (2003)
    sldata[ii].s29.tcdelta = 1000.0d * Alog(19.6939d / sldata[ii].s29.tcratio)

    ;Correct the 28Si/30Si ratio for optical depth
    sldata[ii].s30.tcratio = sldata[ii].s30.ratio * sldata[ii].cfactor

    ;Convert the corrected area ratio to delta values.
    ;solar Si values from Lodders (2003)
    sldata[ii].s30.tcdelta = 1000.0d * Alog(29.8753d / sldata[ii].s30.tcratio)

  Endfor
  Return, sldata

End
```

14.11 Accessory Programs And Definitions

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME:
; Superseed
; HYDRA Version 3.1
;
; PURPOSE:
; −> Generate a quasi−random longword vector usingthe SYSTIME
; function. Allows multiple seed vectors to be generated
; quickly and ensures a high degree of statistical
; independence between repeated calls of RANDOMN or
; RANDOMU during Monte−Carlo simulations.
;
; CALLING SEQUENCE:
; −> output = Superseed(seedsize)
;
; ARGUMENT(S):
; −> Seedsize: The length of the quasi−random longword
; seed vector to be generated and returned.
;
; KEYWORD(S):
; −> None
;
; OPTIONAL KEYWORD(S):
; −> None
;
; EXAMPLES:
; −> output = Superseed(5)
;
; OUTPUTS:
; −> A single quasi−random, longword vector
; containing 'seedsize' entries.
;
; COMMENTS:
; -> Do not use SUPERSEED to pass a seed to RANDOMN or RANDOMU
; that is the same type and dimension as the output array.
; IDL will assume that this is a previous seed and will
; corrupt the random sequence.
;
; PROCEDURES/FUNCTIONS CALLED:
; -> None
;-

Function Superseed, seedsize
  Compile_opt IDL2
  Common rtype
  On_error, 1
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA). 5 May, 2018.

  ;+---------------------------------------+
  ;>>>>> Usage Agreement <<<<<
  ;+---------------------------------------+

  ;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
  ;See HYDRA User's Guide.

  ;+---------------------------------------+
  ;>>>>> Developer's Notes <<<<<
  ;+---------------------------------------+

  ;+---------------------------------------+
  ;>>>>> Limitations & Known Bugs <<<<<
  ;+---------------------------------------+

  ;None known.
```

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
Print, ''

;Ensure the seedsize argument is a scalar and
;is a longword integer, or can be converted to one.
;If seedsize is a funky type, e.g. unsigned, then throw an error.
If Array_equal(0, Isa(seedsize, /scalar)) || $
  Array_equal(1, rtype.adtypes.Contains(Typename(seedsize)), /not_equal) Then Begin
  Print, '>>>>> Alert: Seedsize argument dimension = '+Strtrim(seedsize.length,2)
  Print, '>>>>> Alert: Seedsize argument type = '+Strtrim(Typename(seedsize),2)
  Print, '>>>>> Error: Seedsize argument must be an integer or floating−point scalar'
  Print, '' & ++ keymaster
Endif Else Begin
  seedsize = Long(seedsize)

  ;Make sure SUPERSEED is nonzero.
  If ~(seedsize Gt 0) Then Begin
    Print, '>>>>> Error: Seedsize argument must be non−zero'
    Print, '' & ++ keymaster
  Endif
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: status red'
  Print, '>>>>> Returning...'
  Print, ''
  Print, '>>>>> End Of Line <<<<<'
  Print, ''
  Retall
Endif Else Begin
  ;Otherwise, continue.
  Print, '>>>>> Alert: Status Green'
  Print, '>>>>> Continuing...'
  Print, ''
  Endelse

  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
  ;SECTION 1: − Generate seed(s).
  ;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

  ;Generate the seed vector, with nelements given by the value of 'seedsize'
  ;the system time (in seconds) is used to try and ensure the seed is different
  ;every time SUPERSEED is called.
  seed = Long(((Systime(/seconds)^2d) * ((Dindgen(seedsize)+2d)^2d)) $
    Mod (2d^(31d) − 1d))

  ;Return the seed.
  Return, seed

End


;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Ancillary Program(s) <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
; NAME:
; Randomcolors
; HYDRA Version 5.1
;
; PURPOSE:
; −> Generate random 8−bit RGB color vectors.
;
; CALLING SEQUENCE:
; −> output = Randomcolors(ncolors)
;
; ARGUMENT(S):
; −> Ncolors: The integer number of random, 8−bit
; RGB color vectors to be returned.
;
```

```
; KEYWORD(S):
; -> None
;
; OPTIONAL KEYWORD(S):
; -> None
;
; EXAMPLES:
; -> output = Randomcolors(20)
;
; OUTPUTS:
; -> A single longword integer matrix with one 8-bit,
; 3 element RGB vector in each row.
;
; COMMENTS:
; -> None
;
; PROCEDURES/FUNCTIONS CALLED:
; -> Superseed.pro
;-

Function Randomcolors, ncolors
  Compile_opt IDL2
  Common rtype
  On_error, 1
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA). 18 May, 2018.

  ;+-----------------------------------------+
  ;>>>>> Usage Agreement <<<<<
  ;+-----------------------------------------+

  ;;Copyright (C) 2019, N.N. Monson

  ;Usage Agreement omitted for brevity.
  ;See HYDRA User's Guide.

  ;+-----------------------------------------+
  ;>>>>> Developer's Notes <<<<<
```

```
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Bugs <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;None known.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
Print, ''

;Ensure the NCOLORS argument is a scalar and
;is a longword integer, or can be converted to one.
;If ncolors is a funky type, e.g. unsigned, then throw an error.
If Array_equal(0, Isa(ncolors, /scalar)) || $
  Array_equal(1, rtype.adtypes.Contains(Typename(ncolors)), /not_equal) Then Begin
  Print, '>>>>> Alert: Ncolors argument dimension = '+Strtrim(ncolors.length,2)
  Print, '>>>>> Alert: Ncolors argument type = '+Strtrim(Typename(ncolors),2)
  Print, '>>>>> Error: Ncolors argument must be an integer or floating−point scalar'
  Print, '' & ++ keymaster
Endif Else Begin
  ncolors = Long(ncolors)

  ;Make sure NCOLORS is nonzero.
  If ~(ncolors Gt 0) Then Begin
    Print, '>>>>> Error: Ncolors argument must be non−zero'
    Print, '' & ++ keymaster
  Endif
Endelse

;Print error message and return if anything went wrong.
If keymaster Ne 0 Then Begin
  Print, '>>>>> Alert: status red'
  Print, '>>>>> Returning...'
  Print, ''
```

```
    Print, '>>>>> End Of Line <<<<<'
    Print, ''
    Retall
  Endif Else Begin
    ;Otherwise, continue.
    Print, '>>>>> Alert: Status Green'
    Print, '>>>>> Continuing...'
    Print, ''
  Endelse

  ;+-----------------------------------------+
  ;SECTION 1: - Generate 3 by N array of random 8-bit integers.
  ;+-----------------------------------------+

  ;Initialize array.
  rcolors = Fltarr(3, ncolors-1, /nozero)

  ;Generate a 3 x NCOLORS array of uniformly distributed
  ;random numbers and convert to 8-bit integer RGB values.
  rcolors = Randomu(Superseed(5), 3, ncolors-1)
  rcolors = Long(255.0 * Temporary(rcolors))

  ;Return color array.
  Return, rcolors

End


;+-----------------------------------------+
;>>>>> Ancillary Program(s) <<<<<
;+-----------------------------------------+

;+-----------------------------------------+
;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
; NAME:
; Purge
; HYDRA Version 5.1
;
```

```
; PURPOSE:
; -> Closes all open IDL graphics windows.
;
; CALLING SEQUENCE:
; -> Purge
;
; ARGUMENT(S):
; -> None
;
; KEYWORD(S):
; -> None
;
; OPTIONAL KEYWORD(S):
; -> None
;
; EXAMPLE(S):
; -> Purge
;
; OUTPUT(S):
; -> None
;
; COMMENTS:
; -> Closes all open graphics windows.
; Thats it! Short & sweet.
;
; PROCEDURES/FUNCTIONS CALLED:
; -> None
;-

Pro Purge
  Compile_opt IDL2
  On_error, 1
  !Except = 1

  ;Hydra version 5.1
  ;>>>>> Written by: N.N. Monson (UCLA) 14 August, 2013
  ;>>>>> Version 2.0 written by: N.N. Monson (UCLA). 7 February, 2014
  ;>>>>> Version 3.0 written by: N.N. Monson (UCLA). 21 July, 2016
  ;>>>>> Version 4.0 (V-spec) written by: N.N. Monson (UCLA). 30 April, 2017
  ;>>>>> Version 5.0 (V-spec) written by: N.N. Monson (UCLA). 23 November, 2018
```

```
;>>>>> Version 5.1 (V−spec) written by: N.N. Monson (UCLA). 11 January, 2019

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Usage Agreement <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Copyright (C) 2019, N.N. Monson

;Usage Agreement omitted for brevity.
;See HYDRA User's Guide.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Developer's Notes <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Limitations & Known Bugs <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;None known.

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 0: − Check argument(s).
; − Set keyword defaults.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;SECTION 1: − Close all open graphics windows.
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+

;Print an empty line below the program call on the command line.
Print, ''

;Print some stuff
Print, '>>>>> Alert: purging graphics buffers'
Print, ''

W = Getwindows()
Foreach I, W Do I.close
```

```
  Print, '>>>>> End Of Line <<<<<'
  Print, ''

End
```

Definitions

```
;+-----------------------------------------+
;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
;Defines the 'Slstruct' structure.
;-
Pro Slstruct__define
  Compile_opt idl2

  structure = { slstruct, $ ;begin tags
    mtau: 0d, $
    cfactor: 0d, $
    s28: $
    { tmb: Dblarr(rtype.sml/3, /nozero), $
    stmb: Dblarr(rtype.sml/3, /nozero), $
    rstmb: Dblarr(rtype.sml/3, /nozero), $
    scaled: Dblarr(rtype.sml/3, /nozero), $
    tbeam: 0d, $
    fwhm: 0d, $
    bound: List(), $
    ploc: 0d, $
    lfit: 0d, $
    hfit: 0d, $
    wxmin: 0d, $
    wxmax: 0d, $
    wblo: 0d, $
    wbhi: 0d, $
    area: 0d }, $
    s29: $
    { tmb: Dblarr(rtype.sml/3, /nozero), $
    stmb: Dblarr(rtype.sml/3, /nozero), $
    rstmb: Dblarr(rtype.sml/3, /nozero), $
    tbeam: 0d, $
```

```
fwhm: 0d, $
bound: List(), $
ploc: 0d, $
lfit: 0d, $
hfit: 0d, $
wxmin: 0d, $
wxmax: 0d, $
wblo: 0d, $
wbhi: 0d, $
area: 0d, $
scaled: Dblarr(rtype.sml/3, /nozero), $
gamma: 0d, $
tau: 0d, $
weight: 0d, $
delta: 0d, $
tcdelta: 0d, $
pratio: 0d, $
ratio: 0d, $
tcratio: 0d }, $
s30: $
{ tmb: Dblarr(rtype.sml/3, /nozero), $
stmb: Dblarr(rtype.sml/3, /nozero), $
rstmb: Dblarr(rtype.sml/3, /nozero), $
scaled: Dblarr(rtype.sml/3, /nozero), $
gamma: 0d, $
tbeam: 0d, $
fwhm: 0d, $
bound: List(), $
ploc: 0d, $
lfit: 0d, $
hfit: 0d, $
wxmin: 0d, $
wxmax: 0d, $
wblo: 0d, $
wbhi: 0d, $
area: 0d, $
scaled: Dblarr(rtype.sml/3, /nozero), $
gamma: 0d, $
tau: 0d, $
weight: 0d, $
```

```
    delta: 0d, $
    tcdelta: 0d, $
    pratio: 0d, $
    ratio: 0d, $
    tcratio: 0d } }

End

w
;+-----------------------------------------+
;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
;Defines the 'scdata_l' structure.
;-
Pro Scdata_l__define
  Compile_opt IDL2
  Common rtype

  noise_tube = { $ ;begin tags
    tube_on: Dblarr(rtype.lrg, /nozero), $
    tube_off: Dblarr(rtype.lrg, /nozero) }

  polarization = { $ ;begin tags
    left: noise_tube, $
    right: noise_tube }

  structure = { scdata_l, $ ;begin tags
    sig: polarization, $
    ref: polarization }

End


;+-----------------------------------------+
;;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
;Defines the 'scdata_m' structure.
;-
Pro Scdata_m__define
```

```
  Compile_opt IDL2
  Common rtype

  noise_tube = { $ ;begin tags
    tube_on: Dblarr(rtype.med, /nozero), $
    tube_off: Dblarr(rtype.med, /nozero) }

  polarization = { $ ;begin tags
    left: noise_tube, $
    right: noise_tube }

  structure = { scdata_m, $ ;begin tags
    sig: polarization, $
    ref: polarization }

End


;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;>>>>> Primary Program <<<<<
;+−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
;+
;Defines the 'schead' structure.
;−
Pro Schead__define
  Compile_opt IDL2

  headtags = { $ ;begin sdfits tags
    object: '', $
    bandwid: 0d, $
    date_obs: '', $
    duration: 0d, $
    exposure: 0d, $
    tsys: 0d, $
    tdim7: '', $
    tunit7: '', $
    ctype1: '', $
    crval1: 0d, $
    crpix1: 0d, $
    cdelt1: 0d, $
```

```
ctype2: ”, $
crval2: 0d, $
ctype3: ”, $
crval3: 0d, $
crval4: Fix(0), $
observer: ”, $
obsid: ”, $
scan: 0, $
obsmode: ”, $
frontend: ”, $
tcal: 0.0, $
veldef: ”, $
vframe: 0d, $
rvsys: 0d, $
obsfreq: 0d, $
lsrfreq:0d, $
lst: 0d, $
azimuth: 0d, $
elevatio: 0d, $
tambient: 0d, $
pressure: 0d, $
humidity: 0d, $
restfreq: 0d, $
freqres: 0d, $
equinox: 0d, $
radesys: ”, $
trgtlong: 0d, $
trgtlat: 0d, $
sampler: ”, $
feed: Fix(0), $
srfeed: Fix(0), $
feedxoff: 0d, $
feedeoff: 0d, $
subref_state: Fix(0), $
sideband: ”, $
procseqn: Fix(0), $
procsize: Fix(0), $
laston: 0, $
lastoff: 0, $
timestamp: ”, $
```

```
    velocity: 0d, $
    zerochan: 0.0, $
    caltype: '', $
    ifnum: Fix(0), $
    plnum: Fix(0) }

  structure = { schead, $ ;begin tags
    sig: headtags, $
    ref: headtags }

End


;+-----------------------------------------+
;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
;Defines the 'redata_m' structure.
;-
Pro Redata_m__define
  Compile_opt IDL2
  Common rtype

  polarization = { $ ;begin tags
    left: Dblarr(rtype.med, /nozero), $
    right: Dblarr(rtype.med, /nozero) }

  structure = { redata_m, $ ;begin tags
    sig: polarization, $
    ref: polarization }

End


;+-----------------------------------------+
;>>>>> Primary Program <<<<<
;+-----------------------------------------+
;+
;Defines the 'redata_s' structure.
;-
Pro Redata_s__define
```

```
  Compile_opt IDL2
  Common rtype

  polarization = { $ ;begin tags
    left: Dblarr(rtype.sml, /nozero), $
    right: Dblarr(rtype.sml, /nozero) }

  structure = { redata_s, $ ;begin tags
    sig: polarization, $
    ref: polarization }

End


;+–––––––––––––––––––––––––––––––––––––––––+
;>>>>> Primary Program <<<<<
;+–––––––––––––––––––––––––––––––––––––––––+
;+
;Defines the 'lrpol_m' structure.
;–
Pro Lrpol_m__define
  Compile_opt IDL2
  Common rtype

  polarization = { lrpol_m, $ ;begin tags
    left: Dblarr(rtype.med, /nozero), $
    right: Dblarr(rtype.med, /nozero) }

End


;+–––––––––––––––––––––––––––––––––––––––––+
;>>>>> Primary Program <<<<<
;+–––––––––––––––––––––––––––––––––––––––––+
;+
;Defines the 'lrpol_s' structure.
;–
Pro Lrpol_s__define
  Compile_opt IDL2
  Common rtype
  polarization = { lrpol_s, $ ;begin tags
    left: Dblarr(rtype.sml, /nozero), $
    right: Dblarr(rtype.sml, /nozero) }

End


;+----------------------------------------+
;>>>>> Primary Program <<<<<
;+----------------------------------------+
;+
;Defines the 'sigref_m' structure.
;-
Pro Sigref_m__define
  Compile_opt IDL2
  Common rtype

  fposition = { sigref_m, $ ;begin tags
    sig: Dblarr(rtype.med, /nozero), $
    ref: Dblarr(rtype.med, /nozero) }

End


;+----------------------------------------+
;>>>>> Primary Program <<<<<
;+----------------------------------------+
;+
;Defines the 'sigref_s' structure.
;-
Pro Sigref_s__define
  Compile_opt IDL2
  Common rtype

  fposition = { sigref_s, $ ;begin tags
    sig: Dblarr(rtype.sml, /nozero), $
    ref: Dblarr(rtype.sml, /nozero) }

End
```

```
;+---------------------------------------+
;>>>>> Primary Program <<<<<
;+---------------------------------------+
;+
;Defines the 'datatool_s' structure.
;-
Pro Datatool_s__define
  Compile_opt IDL2
  Common rtype

  structure = { datatool_s,$ ;begin tags
    wild: Dblarr(rtype.sml, /nozero), $
    tame: Dblarr(rtype.sml, /nozero), $
    vlsr: Dblarr(rtype.sml, /nozero) }

End


;+---------------------------------------+
;>>>>> Primary Program <<<<<
;+---------------------------------------+
;+
;Defines the 'Summary' structure.
;-
Pro Summary__define
  Compile_opt idl2

  structure = { summary, $ ;begin tags
    scan: 0l, $
    object: '', $
    nrec: 0l, $
    mode: '' }

End
```

Bibliography

Adande, G. R. and Ziurys, L. M. Millimeter-wave Observations of CN and HNC and Their ^{15}N Isotopologues: A New Evaluation of the ^{14}N/^{15}N Ratio across the Galaxy. *The Astrophysical Journal*, 744:194, January 2012. doi:10.1088/0004-637X/744/2/194.

Alexander, C. M. O. and Nittler, L. R. The Galactic Evolution of Si, Ti, and O Isotopic Ratios. *The Astrophysical Journal*, 519:222–235, July 1999. doi:10.1086/307340.

Amo-Baladrón, M. A., Martín-Pintado, J., Morris, M. R., Muno, M. P., and Rodríguez-Fernández, N. J. SiO Emission as a Tracer of X-Ray Dominated Chemistry in the Galactic Center. *The Astrophysical Journal*, 694:943–950, April 2009. doi:10.1088/0004-637X/694/2/943.

Andrievsky, S. M., Bersier, D., Kovtyukh, V. V., Luck, R. E., Maciel, W. J., Lépine, J. R. D., and Beletsky, Y. V. Using Cepheids to determine the galactic abundance gradient. II. Towards the galactic center. *Astronomy and Astrophysics*, 384:140–144, March 2002a. doi:10.1051/0004-6361:20020016.

Andrievsky, S. M., Kovtyukh, V. V., Luck, R. E., Lépine, J. R. D., Bersier, D., Maciel, W. J., Barbuy, B., Klochkova, V. G., Panchuk, V. E., and Karpischek, R. U. Using Cepheids to determine the galactic abundance gradient. I. The solar neighbourhood. *Astronomy and Astrophysics*, 381:32–50, January 2002b. doi:10.1051/0004-6361:20011488.

Andrievsky, S. M., Kovtyukh, V. V., Luck, R. E., Lépine, J. R. D., Maciel, W. J., and Beletsky, Y. V. Using Cepheids to determine the galactic abundance gradient. III. First results for the outer disc. *Astronomy and Astrophysics*, 392:491–499, September 2002c. doi:10.1051/0004-6361:20021035.

Araya, E. D., Kurtz, S., Hofner, P., and Linz, H. Radio Continuum and Methanol Observations of DR21(OH). *The Astrophysical Journal*, 698:1321–1329, June 2009. doi:10.1088/0004-637X/698/2/1321.

Baars, J. W. M. *The Paraboloidal Reflector Antenna in Radio Astronomy and Communication*, volume 348 of *Astrophysics and Space Science Library*. Springer-Verlag, 1st edition, 2007. ISBN 978-0-387-69733-8. doi:10.1007/978-0-387-69734-5.

Balser, D. S., Rood, R. T., Bania, T. M., and Anderson, L. D. H II Region Metallicity Distribution in the Milky Way Disk. *The Astrophysical Journal*, 738:27, September 2011. doi:10.1088/0004-637X/738/1/27.

Burbidge, E. M., Burbidge, G. R., Fowler, W. A., and Hoyle, F. Synthesis of the Elements in Stars. *Reviews of Modern Physics*, 29:547–650, 1957. doi:10.1103/RevModPhys.29.547.

Caselli, P., Hartquist, T. W., and Havnes, O. Grain-grain collisions and sputtering in oblique C-type shocks. *Astronomy and Astrophysics*, 322:296–301, June 1997.

Cavichia, O., Mollá, M., Costa, R. D. D., and Maciel, W. J. The role of the Galactic bar in the chemical evolution of the Milky Way. *Monthly Notices of the Royal Astronomical Society*, 437:3688–3701, February 2014. doi:10.1093/mnras/stt2164.

Clayton, D. D. Galactic chemical evolution and nucleocosmochronology - Standard model with terminated infall. *The Astrophysical Journal*, 285:411–425, October 1984. doi:10.1086/162518.

Clayton, D. D. *Handbook of Isotopes in the Cosmos*. Cambridge University Press, September 2003. ISBN 0521823811.

Clayton, D. D. and Pantelaki, I. Secondary metallicity in analytic models of chemical evolution of galaxies. *The Astrophysical Journal*, 307:441–448, August 1986. doi:10.1086/164433.

Cunha, K., Sellgren, K., Smith, V. V., Ramirez, S. V., Blum, R. D., and Terndrup, D. M. Chemical Abundances of Luminous Cool Stars in the Galactic Center from High-Resolution Infrared Spectroscopy. *The Astrophysical Journal*, 669:1011–1023, November 2007. doi:10.1086/521813.

Davies, B., Origlia, L., Kudritzki, R.-P., Figer, D. F., Rich, R. M., and Najarro, F. The Chemical Abundances in the Galactic Center from the Atmospheres of Red Supergiants. *The Astrophysical Journal*, 694:46–55, March 2009. doi:10.1088/0004-637X/694/1/46.

Draine, B. T. *Physics of the Interstellar and Intergalactic Medium*. Princeton Series in Astrophysics. Princeton Univ. Press, 1st edition, 2011. ISBN 978-0-691-12214-4.

Duarte-Cabral, A., Bontemps, S., Motte, F., Gusdorf, A., Csengeri, T., Schneider, N., and Louvet, F. SiO emission from low- and high-velocity shocks in Cygnus-X massive dense clumps. *Astronomy and Astrophysics*, 570:A1, October 2014. doi:10.1051/0004-6361/201423677.

Frerking, M. A., Wilson, R. W., Linke, R. A., and Wannier, P. G. Isotopic abundance ratios in interstellar carbon monosulfide. *The Astrophysical Journal*, 240:65–73, August 1980. doi:10.1086/158207.

Gallino, R., Raiteri, C. M., Busso, M., and Matteucci, F. The puzzle of silicon, titanium, and magnesium anomalies in meteoritic silicon carbide grains. *The Astrophysical Journal*, 430:858–869, August 1994. doi:10.1086/174457.

Gallino, R., Arlandini, C., Busso, M., Lugaro, M., Travaglio, C., Straniero, O., Chieffi, A., and Limongi, M. Evolution and Nucleosynthesis in Low-Mass Asymptotic Giant Branch Stars. II. Neutron Capture and the S-Process. *The Astrophysical Journal*, 497:388–403, April 1998. doi:10.1086/305437.

Goldsmith, P. F. Collisional Excitation of Carbon Monoxide in Interstellar Clouds. *The Astrophysical Journal*, 176:597, September 1972. doi:10.1086/151661.

Goldsmith, P. F. and Langer, W. D. Population Diagram Analysis of Molecular Line Emission. *The Astrophysical Journal*, 517:209–225, May 1999. doi:10.1086/307195.

Gusdorf, A., Cabrit, S., Flower, D. R., and Pineau Des Forêts, G. SiO line emission from C-type shock waves: interstellar jets and outflows. *Astronomy and Astrophysics*, 482: 809–829, May 2008. doi:10.1051/0004-6361:20078900.

Handa, T., Sakano, M., Naito, S., Hiramatsu, M., and Tsuboi, M. Thermal SiO and $H^{13}CO^+$ Line Observations of the Dense Molecular Cloud G0.11-0.11 in the Galactic Center Region. *The Astrophysical Journal*, 636:261–266, January 2006. doi:10.1086/497881.

Harju, J., Lehtinen, K., Booth, R. S., and Zinchenko, I. A survey of SiO emission towards interstellar masers. I. SiO line characteristics. *Astronomy and Astrophysics Supplement Series*, 132:211–231, October 1998. doi:10.1051/aas:1998448.

Henkel, C., Asiri, H., Ao, Y., Aalto, S., Danielson, A. L. R., Papadopoulos, P. P., García-Burillo, S., Aladro, R., Impellizzeri, C. M. V., Mauersberger, R., Martín, S., and Harada, N. Carbon and oxygen isotope ratios in starburst galaxies: New data from NGC 253 and Mrk 231 and their implications. *Astronomy and Astrophysics*, 565:A3, May 2014. doi:10.1051/0004-6361/201322962.

Heyer, M. and Dame, T. M. Molecular Clouds in the Milky Way. *Annual Review of Astronomy and Astrophysics*, 53:583–629, August 2015. doi:10.1146/annurev-astro-082214-122324.

Hunter, T. R., Testi, L., Zhang, Q., and Sridharan, T. K. Molecular Jets and H_2O Masers in the AFGL 5142 Hot Core. *The Astronomical Journal*, 118:477–487, July 1999. doi:10.1086/300936.

Kennicutt, R. C. and Evans, N. J. Star Formation in the Milky Way and Nearby Galaxies. *Annual Review of Astronomy and Astrophysics*, 50:531–608, September 2012. doi:10.1146/annurev-astro-081811-125610.

Kennicutt, R. C. Jr. The Global Schmidt Law in Star-forming Galaxies. *The Astrophysical Journal*, 498:541–552, May 1998. doi:10.1086/305588.

Kent, S. M., Dame, T. M., and Fazio, G. Galactic structure from the Spacelab infrared telescope. II - Luminosity models of the Milky Way. *The Astrophysical Journal*, 378: 131–138, September 1991. doi:10.1086/170413.

Knight, K. B., Kita, N. T., Mendybaev, R. A., Richter, F. M., Davis, A. M., and Valley, J. W. Silicon isotopic fractionation of CAI-like vacuum evaporation residues. *Geochimica et Cosmochimica Acta*, 73:6390–6401, October 2009. doi:10.1016/j.gca.2009.07.008.

Kobayashi, C., Umeda, H., Nomoto, K., Tominaga, N., and Ohkubo, T. Galactic Chemical Evolution: Carbon through Zinc. *The Astrophysical Journal*, 653:1145–1171, December 2006. doi:10.1086/508914.

Kobayashi, C., Karakas, A. I., and Umeda, H. The evolution of isotope ratios in the Milky Way Galaxy. *Monthly Notices of the Royal Astronomical Society*, 414:3231–3250, July 2011. doi:10.1111/j.1365-2966.2011.18621.x.

Langer, W. D. and Penzias, A. A. C-12/C-13 isotope ratio across the Galaxy from observations of C-13/O-18 in molecular clouds. *The Astrophysical Journal*, 357:477–492, July 1990. doi:10.1086/168935.

Langer, W. D. and Penzias, A. A. (C-12)/(C-13) isotope ratio in the local interstellar medium from observations of (C-13)(O-18) in molecular clouds. *The Astrophysical Journal*, 408: 539–547, May 1993. doi:10.1086/172611.

Leroy, A. K., Walter, F., Brinks, E., Bigiel, F., de Blok, W. J. G., Madore, B., and Thornley, M. D. The Star Formation Efficiency in Nearby Galaxies: Measuring Where Gas Forms Stars Effectively. *The Astronomical Journal*, 136:2782–2845, December 2008. doi:10.1088/0004-6256/136/6/2782.

Lewis, K. M., Lugaro, M., Gibson, B. K., and Pilkington, K. Decoding the Message from Meteoritic Stardust Silicon Carbide Grains. *The Astrophysical Journal Letters*, 768:L19, May 2013. doi:10.1088/2041-8205/768/1/L19.

Linke, R. A., Goldsmith, P. F., Wannier, P. G., Wilson, R. W., and Penzias, A. A. Isotopic abundance variations in interstellar HCN. *The Astrophysical Journal*, 214:50–59, May 1977. doi:10.1086/155229.

Luck, R. E., Kovtyukh, V. V., and Andrievsky, S. M. The Distribution of the Elements in the Galactic Disk. *The Astronomical Journal*, 132:902–918, August 2006. doi:10.1086/505687.

Lugaro, M., Zinner, E., Gallino, R., and Amari, S. Si Isotopic Ratios in Mainstream Presolar SIC Grains Revisited. *The Astrophysical Journal*, 527:369–394, December 1999. doi:10.1086/308078.

Lugaro, M., Gallino, R., Amari, S., Zinner, E., and Nittler, L. R. The effect of heterogeneities in the interstellar medium on the CNO isotopic ratios of presolar SiC and corundum grains. *Nuclear Physics A*, 718:419–421, May 2003. doi:10.1016/S0375-9474(03)00819-4.

Luisi, M., Anderson, L. D., Balser, D. S., Bania, T. M., and Wenger, T. V. HII Region Ionization of the Interstellar Medium: A Case Study of NGC 7538. *ArXiv e-prints*, May 2016.

Mangum, J. G. and Shirley, Y. L. How to Calculate Molecular Column Density. *Publications of the Astronomical Society of the Pacific*, 127:266, March 2015. doi:10.1086/680323.

Martín, S., Martín-Pintado, J., and Mauersberger, R. HNCO Abundances in Galaxies: Tracing the Evolutionary State of Starbursts. *The Astrophysical Journal*, 694:610–617, March 2009. doi:10.1088/0004-637X/694/1/610.

Martín, S., Aladro, R., Martín-Pintado, J., and Mauersberger, R. A large $^{12}C/^{13}C$ isotopic ratio in M 82 and NGC 253. *Astronomy and Astrophysics*, 522:A62, November 2010. doi:10.1051/0004-6361/201014972.

Matteucci, F., Panagia, N., Pipino, A., Mannucci, F., Recchi, S., and Della Valle, M. A new formulation of the Type Ia supernova rate and its consequences on galactic chemical evolution. *Monthly Notices of the Royal Astronomical Society*, 372:265–275, October 2006. doi:10.1111/j.1365-2966.2006.10848.x.

Milam, S. N., Savage, C., Brewster, M. A., Ziurys, L. M., and Wyckoff, S. The ^{12}C/^{13}C Isotope Gradient Derived from Millimeter Transitions of CN: The Case for Galactic Chemical Evolution. *The Astrophysical Journal*, 634:1126–1132, December 2005. doi:10.1086/497123.

Minchev, I., Famaey, B., Combes, F., Di Matteo, P., Mouhcine, M., and Wozniak, H. Radial migration in galactic disks caused by resonance overlap of multiple patterns: Self-consistent simulations. *Astronomy and Astrophysics*, 527:A147, March 2011. doi:10.1051/0004-6361/201015139.

Monson, N. N., Morris, M. R., and Young, E. D. Uniform Silicon Isotope Ratios Across the Milky Way Galaxy. *The Astrophysical Journal*, 839:123, April 2017. doi:10.3847/1538-4357/aa67e6.

Najarro, F., Figer, D. F., Hillier, D. J., Geballe, T. R., and Kudritzki, R. P. Metallicity in the Galactic Center: The Quintuplet Cluster. *The Astrophysical Journal*, 691:1816–1827, February 2009. doi:10.1088/0004-637X/691/2/1816.

Naranjo-Romero, R., Zapata, L. A., Vázquez-Semadeni, E., Takahashi, S., Palau, A., and Schilke, P. From Dusty Filaments to Massive Stars: The Case of NGC 7538 S. *The Astrophysical Journal*, 757:58, September 2012. doi:10.1088/0004-637X/757/1/58.

Nichols, R. H. Jr., Wasserburg, G. J., and Grimley, R. T. Evaporation of Forsterite: Identification of Gas-phase Species Via Knudsen Cell Mass Spectrometry. In *Lunar and Planetary Science Conference*, volume 26 of *Lunar and Planetary Science Conference*, March 1995.

Nisini, B., Codella, C., Giannini, T., Santiago Garcia, J., Richer, J. S., Bachiller, R., and Tafalla, M. Warm SiO gas in molecular bullets associated with protostellar outflows. *Astronomy and Astrophysics*, 462:163–172, January 2007. doi:10.1051/0004-6361:20065621.

Nittler, L. R. Constraints on Heterogeneous Galactic Chemical Evolution from Meteoritic Stardust. *The Astrophysical Journal*, 618:281–296, January 2005. doi:10.1086/425892.

Nittler, L. R. and Gaidos, E. Galactic chemical evolution and the oxygen isotopic composition of the solar system. *Meteoritics and Planetary Science*, 47:2031–2048, December 2012. doi:10.1111/j.1945-5100.2012.01410.x.

Pedicelli, S., Bono, G., Lemasle, B., François, P., Groenewegen, M., Lub, J., Pel, J. W., Laney, D., Piersimoni, A., Romaniello, M., Buonanno, R., Caputo, F., Cassisi, S., Castelli, F., Leurini, S., Pietrinferni, A., Primas, F., and Pritchard, J. On the metallicity gradient of the Galactic disk. *Astronomy and Astrophysics*, 504:81–86, September 2009. doi:10.1051/0004-6361/200912504.

Penzias, A. A. On the relative abundances of silicon isotopes in the interstellar medium. *The Astrophysical Journal*, 249:513–517, October 1981a. doi:10.1086/159310.

Penzias, A. A. The isotopic abundances of interstellar oxygen. *The Astrophysical Journal*, 249:518–523, October 1981b. doi:10.1086/159311.

Perley, R. A. and Butler, B. J. An Accurate Flux Density Scale from 1 to 50 GHz. *The Astrophysical Journal Supplement Series*, 204(2):19, Feb 2013. doi:10.1088/0067-0049/204/2/19.

Perley, R. A. and Butler, B. J. An Accurate Flux Density Scale from 50 MHz to 50 GHz. *The Astrophysical Journal Supplement Series*, 230(1):7, May 2017. doi:10.3847/1538-4365/aa6df9.

Prantzos, N. An Introduction to Galactic Chemical Evolution. In Charbonnel, C. and Zahn, J.-P., editors, *EAS Publications Series*, volume 32 of *EAS Publications Series*, pages 311–356, November 2008. doi:10.1051/eas:0832009.

Prantzos, N., Aubert, O., and Audouze, J. Evolution of the carbon and oxygen isotopes in the Galaxy. *Astronomy and Astrophysics*, 309:760–774, May 1996.

Richter, F. M., Janney, P. E., Mendybaev, R. A., Davis, A. M., and Wadhwa, M. Elemental and isotopic fractionation of Type B CAI-like liquids by evaporation. *Geochimica et Cosmochimica Acta*, 71:5544–5564, November 2007. doi:10.1016/j.gca.2007.09.005.

Riquelme, D., Amo-Baladrón, M. A., Martín-Pintado, J., Mauersberger, R., Martín, S., and Bronfman, L. Tracing gas accretion in the Galactic center using isotopic ratios. *Astronomy and Astrophysics*, 523:A51, November 2010a. doi:10.1051/0004-6361/201015008.

Riquelme, D., Bronfman, L., Mauersberger, R., May, J., and Wilson, T. L. A survey of the Galactic center region in HCO^+, $H^{13}CO^+$, and SiO. *Astronomy and Astrophysics*, 523: A45, November 2010b. doi:10.1051/0004-6361/200913359.

Romano, D. and Matteucci, F. Nova nucleosynthesis and Galactic evolution of the CNO isotopes. *Monthly Notices of the Royal Astronomical Society*, 342:185–198, June 2003. doi:10.1046/j.1365-8711.2003.06526.x.

Savage, C., Apponi, A. J., Ziurys, L. M., and Wyckoff, S. Galactic $^{12}C/^{13}C$ Ratios from Millimeter-Wave Observations of Interstellar CN. *The Astrophysical Journal*, 578:211–223, October 2002. doi:10.1086/342468.

Scannapieco, E. and Bildsten, L. The Type Ia Supernova Rate. *The Astrophysical Journal Letters*, 629:L85–L88, August 2005. doi:10.1086/452632.

Schilke, P., Walmsley, C. M., Pineau des Forets, G., and Flower, D. R. SiO production in interstellar shocks. *Astronomy and Astrophysics*, 321:293–304, May 1997.

Searle, L. and Sargent, W. L. W. Inferences from the Composition of Two Dwarf Blue Galaxies. *The Astrophysical Journal*, 173:25, April 1972. doi:10.1086/151398.

Shahar, A. and Young, E. D. Astrophysics of CAI formation as revealed by silicon isotope LA-MC-ICPMS of an igneous CAI. *Earth and Planetary Science Letters*, 257:497–510, May 2007. doi:10.1016/j.epsl.2007.03.012.

Shi, H., Zhao, J.-H., and Han, J. L. Nature of W51e2: Massive Cores at Different Phases of Star Formation. *The Astrophysical Journal*, 710:843–852, February 2010. doi:10.1088/0004-637X/710/1/843.

Shirley, Y. L. The Critical Density and the Effective Excitation Density of Commonly Observed Molecular Dense Gas Tracers. *Publications of the Astronomical Society of the Pacific*, 127:299, March 2015. doi:10.1086/680342.

Spite, M., Cayrel, R., Hill, V., Spite, F., François, P., Plez, B., Bonifacio, P., Molaro, P., Depagne, E., Andersen, J., Barbuy, B., Beers, T. C., Nordström, B., and Primas, F. First stars IX - Mixing in extremely metal-poor giants. Variation of the ^{12}C/^{13}C, [Na/Mg] and [Al/Mg] ratios. *Astronomy and Astrophysics*, 455:291–301, August 2006. doi:10.1051/0004-6361:20065209.

Straniero, O., Chieffi, A., Limongi, M., Busso, M., Gallino, R., and Arlandini, C. Evolution and Nucleosynthesis in Low-Mass Asymptotic Giant Branch Stars. I. Formation of Population I Carbon Stars. *The Astrophysical Journal*, 478:332–339, March 1997.

Timmes, F. X. and Clayton, D. D. Galactic Evolution of Silicon Isotopes: Application to Presolar SiC Grains from Meteorites. *The Astrophysical Journal*, 472:723, December 1996. doi:10.1086/178102.

Timmes, F. X., Woosley, S. E., and Weaver, T. A. Galactic chemical evolution: Hydrogen through zinc. *The Astrophysical Journal Supplement Series*, 98:617–658, June 1995. doi:10.1086/192172.

Tinsley, B. M. Nucleochronology and Chemical Evolution. *The Astrophysical Journal*, 198: 145–150, May 1975. doi:10.1086/153586.

Tinsley, B. M. and Cameron, A. G. W. Possible influence of comets on the chemical evolution of the galaxy. *Astrophysics and Space Science*, 31:31–35, November 1974. doi:10.1007/BF00642598.

Tinsley, B. M. and Larson, R. B. Chemical evolution and the formation of galactic disks. *The Astrophysical Journal*, 221:554–561, April 1978. doi:10.1086/156056.

Tsuboi, M., Tadaki, K.-I., Miyazaki, A., and Handa, T. Sagittarius A Molecular Cloud Complex in H^{13}CO^{+} and Thermal SiO Emission Lines. *Publications of the Astronomical Society of Japan*, 63:763–794, August 2011. doi:10.1093/pasj/63.4.763.

Tsujimoto, T., Nomoto, K., Yoshii, Y., Hashimoto, M., Yanagida, S., and Thielemann, F.-K. Relative frequencies of Type Ia and Type II supernovae in the chemical evolution of the Galaxy, LMC and SMC. *Monthly Notices of the Royal Astronomical Society*, 277:945–958, December 1995. doi:10.1093/mnras/277.3.945.

van der Tak, F. F. S., Black, J. H., Schöier, F. L., Jansen, D. J., and van Dishoeck, E. F. A computer program for fast non-LTE analysis of interstellar line spectra. With diagnostic plots to interpret observed line intensity ratios. *Astronomy and Astrophysics*, 468:627–635, June 2007. doi:10.1051/0004-6361:20066820.

Wilson, R. W., Langer, W. D., and Goldsmith, P. F. A determination of the carbon and oxygen isotopic ratios in the local interstellar medium. *The Astrophysical Journal Letters*, 243:L47–L52, January 1981. doi:10.1086/183440.

Wilson, T. L. Isotopes in the interstellar medium and circumstellar envelopes. *Reports on Progress in Physics*, 62:143–185, February 1999. doi:10.1088/0034-4885/62/2/002.

Wilson, T. L. and Rood, R. Abundances in the Interstellar Medium. *Annual Review of Astronomy and Astrophysics*, 32:191–226, 1994. doi:10.1146/annurev.aa.32.090194.001203.

Wilson, T. L., Rohlfs, K., and Hüttemeister, S. *Tools of Radio Astronomy*. Astronomy and Astrophysics Library. Springer-Verlag, 5th edition, 2009. ISBN 78-3-540-85121-9. doi:10.1007/978-3-540-85122-6.

Wolff, R. S. The relative abundances of Si-28, Si-29, and Si-30 in the interstellar medium. *The Astrophysical Journal*, 242:1005–1012, December 1980. doi:10.1086/158531.

Wouterloot, J. G. A., Henkel, C., Brand, J., and Davis, G. R. Galactic interstellar ^{18}O/{ˆ17}O ratios - a radial gradient? *Astronomy and Astrophysics*, 487:237–246, August 2008. doi:10.1051/0004-6361:20078156.

Young, E. D., Gounelle, M., Smith, R. L., Morris, M. R., and Pontoppidan, K. M. Astronomical Oxygen Isotopic Evidence for Supernova Enrichment of the Solar System Birth Environment by Propagating Star Formation. *The Astrophysical Journal*, 729:43, March 2011. doi:10.1088/0004-637X/729/1/43.

Zapata, L. A., Menten, K., Reid, M., and Beuther, H. An Extensive, Sensitive Search for SiO Masers in High- and Intermediate-Mass Star-Forming Regions. *The Astrophysical Journal*, 691:332–341, January 2009. doi:10.1088/0004-637X/691/1/332.

Zhang, Q., Hunter, T. R., Beuther, H., Sridharan, T. K., Liu, S.-Y., Su, Y.-N., Chen, H.-R., and Chen, Y. Multiple Jets from the High-Mass (Proto)stellar Cluster AFGL 5142. *The Astrophysical Journal*, 658:1152–1163, April 2007. doi:10.1086/511381.

Zinner, E., Nittler, L. R., Gallino, R., Karakas, A. I., Lugaro, M., Straniero, O., and Lattanzio, J. C. Silicon and Carbon Isotopic Ratios in AGB Stars: SiC Grain Data, Models, and the Galactic Evolution of the Si Isotopes. *The Astrophysical Journal*, 650:350–373, October 2006. doi:10.1086/506957.

Ziurys, L. M., Friberg, P., and Irvine, W. M. Interstellar SiO as a tracer of high-temperature chemistry. *The Astrophysical Journal*, 343:201–207, August 1989. doi:10.1086/167696.