RESULTS OF A MICRO PULSE DIFFERENTIAL ABSORPTION LIDAR FOR TEMPERATURE PROFILING AND ANALYSIS CODE

by

Owen Daniel Cruikshank

A professional paper submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Optics

MONTANA STATE UNIVERSITY
Bozeman, Montana

April 2021

ACKNOWLEDGEMENTS

Data presented here come from the ARM SGP site from instruments built by engineers at the National Center for Atmospheric Research (NCAR). I would also like to thank Robert Stillwell for help setting up the micropulse differential absorption lidar (MPD or MPDIAL) instrument at Montana State University (MSU), LabView instrument control programming and help with MATLAB data ingesting. Luke Colberg for help with keeping the MSU MPD running, launching Radiosondes at MSU, and MATLAB backscatter ratio code. The optical remote sensor lab (ORSL) run by Dr. Joe Shaw for surface weather station measurements. My advisor Dr. Kevin Repasky for initial MPD temperature retrieval MATLAB modeling.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES – CONTINUED

Figure Page

LIST OF FIGURES – CONTINUED

Figure Page

ABSTRACT

Thermodynamic profiling of the lower troposphere is necessary for the study of weather and climate. The micropulse DIAL (differential absorption lidar), or MPD, presented here is designed to fill the need. The MPD is eye-safe and can run autonomously for continuous measurements compared to technologies with similar measurement capabilities like Raman lidar. Using a temperature-sensitive absorption line of O_2, the MPD system can measure the absorption of O_2 in the lower troposphere as a function of range and convert that measurement to temperature as a function of range. This process relies on a perturbative correction to the absorption retrieval to account for the fact that the O_2 absorption spectral linewidth is similar to the molecular Rayleigh scattering linewidth. An ancillary measurement of the ratio of aerosol backscatter to molecular backscatter is required for the correction. The integrated high spectral resolution lidar (HSRL) uses a heated potassium vapor notch filter to make the aerosol-to-molecular ratio measurement. An analysis program in MATLAB was written to take in raw lidar data and produce a temperature product of range and time. Results presented from a campaign at the Atmospheric Radiation Measurements program Southern Great Plains site in Oklahoma in spring 2019 show temperature comparisons with radiosonde measurements with a mean difference between radiosonde and MPD measurements of $-1.1\,\mathrm{K}$ and a standard deviation of $2.7\,\mathrm{K}$. Further results from an instrument on the Montana State University campus in Bozeman and at the National Center for Atmospheric Research in Boulder, Colorado have shown that the MPD instrument can produce measurements autonomously for periods of weeks to months.

1

INTRODUCTION

Thermodynamic profiling, including absolute humidity and temperature, of the lower troposphere is important for weather forecasting. The need for thermodynamic profiling has been expressed by the National Research Council [1, 2] as well a report to the National Science Foundation and National Weather Service [3]. Huge observational gaps exist with respect to thermodynamic profiling in the lower troposphere. Many instruments and techniques are available to fill the needs for lower tropospheric measurements, including passive and active sensing systems but low-cost lidar systems have been identified for their potential to address the observational thermodynamic profiling [4].

Raman lidar systems are an active remote sensing technology that can deliver a full thermodynamic profile of the lower troposphere [5, 6]. The water vapor mixing ratio can be measured using vibrational inelastic Raman scattering and temperature can be measured using rotational inelastic Raman scattering. Temperature measurements with Raman lidar use a ratio of two wavelengths of inelastic rotational transitions of N_2 and O_2. The ratio of rotational transitions is a relative measurement and also depends on the overlap function between the transmitter and receiver so calibration must be done with radiosonde measurements. Raman lidar water vapor mixing ratio measurements are based on the ratio of vibrational inelastic scattering between water vapor and N_2. Since the measurement is a ratio and dependent on lidar overlap the measurements must be regularly calibrated with radiosonde measurements [7]. However, implementations of Raman systems can come with challenges. The small Raman scattering cross section in the atmosphere requires high power lasers that can be complex and do not favor unattended operation or eye safety. Raman systems usually have high initial and maintenance costs and must be calibrated

using radiosondes, making them unsuited for network deployment.

Differential absorption lidar (DIAL) is a technique that utilizes more efficient elastic scattering. This means that lower-power semiconductor lasers can be used and provide the benefits of high stability and low cost. Semiconductor-based lasers and amplifiers are also commercially available and can be wavelength tuned and cover the near infrared spectral region containing many gas absorption features of interest. DIAL uses the differential absorption between two closely spaced wavelengths to calculate the absorption of a specific molecule in the atmosphere. The laser wavelengths are chosen based on the absorption spectrum of the molecule to be measured. The online wavelength is placed on a narrow absorption line of the species and the offline wavelength is placed just off of the absorption line so that the only difference between the two wavelengths in the atmosphere results from the absorption due to the molecular species of interest. The laser spectrum must have a narrow bandwidth and high frequency stability for accurate DIAL measurements. Semiconductor micropulse DIAL systems can be eye-safe and can provide the needed frequency stability for long-term unattended operation. Large telescope area and low power transmit pulses ensure eye safety. DIAL systems do not need calibration with radiosondes because the DIAL technique is a direct measurement of absorption, allowing recovery of the gas concentration. Water vapor number density profiling with diode-laser-based micropulse DIAL systems has been proven to be reliable [8, 9, 10, 11, 12]. Temperature profiling with DIAL has only recently been proven to be possible [13, 14].

Theopold and Bösenberg have demonstrated that atmospheric temperature measurements are theoretically possible using a temperature-sensitive oxygen (O_2) absorption line [15]. However, in practice, the measurement of temperature is difficult. The temperature must be found by using an iterative process from an initial seed temperature profile and the measured O_2 absorption. Large errors in the measured absorption lead to large errors in temperature. Errors in the absorption measurement come from broadening in wavelength

of the backscattered lidar signal. The spectrum of the backscattered light from aerosols is nearly identical to the spectrum of the laser transmitter. The backscattered light from molecules is spectrally broadened to a width comparable to the the width of the absorption line itself, about 1 GHz. Therefore, temperature measurements using DIAL are sensitive to the ratio of aerosol to molecular backscatter. The O_2 absorption spectrum and a model of molecular backscatter is shown in 3.2. In this paper, the ratio of total backscatter to molecular backscatter will be referred to as the backscatter ratio (BSR). Without the ability to measure the BSR, early measurements of temperature in the lower troposphere have resulted in errors on the order of 10 K [16]. Atmospheric models have been used to prove that measurements with accuracies of ± 2 K are possible with knowledge of the BSR [13]. The DIAL instrument has no knowledge of the backscatter lineshape so a purtabative solution to the DIAL equation is used to apply a correction factor, dependent on the BSR and backscatter models, to what the DIAL instrument measures. The BSR is measured with a high spectral resolution lidar that uses a narrow potassium vapor spectral notch filter to filter out the aerosol backscatter. Using a combination of a molecular channel with the potassium filter and a combined channel without the filter the BSR can be calculated.

This paper will present a DIAL instrument and measurements of the temperature profile in the lower troposphere using a mask for signal-to-noise ratio and clouds. The masks are used to remove data that are not reliable for the temperature retrieval. A combination of DIAL to measure the temperature-sensitive O_2 absorption line and a high spectral resolution lidar (HSRL) to measure the BSR are used together with the same instrument. The instrument also includes a water vapor DIAL system to measure the water vapor mixing ratio and increase the accuracy of the temperature measurement.

My contribution to this work is developing the data analysis program to process raw data from the lidar system and ground weather station data into a final temperature product. This program is currently written in MATLAB and can be run on a typical workstation

desktop computer, though the processing of multiple days at once requires a large amount of memory. The other contribution is the construction of a combination O_2 DIAL and HSRL system on the Montana State University campus. This instrument has been shown to be working and is being used as a test bed for methods of increasing performance.

THEORY

Perturbative DIAL correction

The first step in the temperature retrieval is to use the perturbative solution to the DIAL equation to retrieve the O_2 absorption coefficient. Once the O_2 absorption profile is retrieved, the temperature profile is then estimated. The perturbative solution requires an initial guess at the temperature profile. For calculations presented in this paper, the initial guess at the temperature profile will consist of the surface temperature and a simple moist adiabatic lapse rate of 6.5 K/km. Furthermore, completion of the retrieval of the O_2 absorption coefficient requires a model of the molecular backscatter profile and the retrieval of the aerosol backscatter coefficient based on the DLB-HSRL data. The following is a summary of theory presented in Repasky et al. [13] and Bunn et al. [17].

The standard DIAL equation calculates the molecular absorption coefficient at the online wavelength as a function of range. Using the lidar returns at the online wavelength (N_1) and the offline wavelength (N_2) the absorption coefficient is calculated as

$$\alpha_{0th}(r) = \alpha_{m,2}(r) - ln\left(\frac{N_1(r+\Delta r)N_2(r)}{N_1(r)N_2(r+\Delta r)}\right). \tag{2.1}$$

The offline absorption coefficient is represented by $\alpha_{m,2}$ which is small and is approximated using the assumed temperature profile. The range difference, Δr, is the pulse length of the lidar or the averaging length if the lidar returns are averaged.

The perturbative DIAL absorption described in Bunn [17] contains two correction terms to the standard DIAL equation. Using the absorption coefficient from the standard DIAL equation as the zeroth order absorption

$$\alpha_{m,1}(r) = \alpha_{0th}(r) + \Delta\alpha_{1st}(r) + \Delta\alpha_{2nd}(r), \tag{2.2}$$

represents the final result of the second order perturbative DIAL process.

The first- and second-order correction terms account for the spectral broadening of the molecularly scattered light and rely on the normalized backscatter lineshape, $g_x(\nu, r)$, with the combination of the molecular backscatter lineshape and the aerosol backscatter lineshape. The equation is written as

$$g_x(\nu, r) = \frac{1}{1 - BSR} h_x(\nu) + \frac{1}{BSR}(h_x(\nu) \otimes l(\nu, r)) \tag{2.3}$$

where $h_x(\nu)$ is the laser lineshape, $l(\nu, r)$ is the molecular backscatter broadened lineshape, and the subscript x is either 1 for the on-line wavelength or 2 for the off-line wavelength. For the work described in this paper, the laser lineshape is assumed to be a delta function. The molecular backcatter is represented by a Doppler-broadened lineshape [18] based on the average molecular mass of the atmospheric molecules with a Brullion approximation correction [16]. The normalized backscatter lineshape is known from the HSRL retrieval of the backscatter ratio and is defined as $BSR = \frac{\beta_a + \beta_m}{\beta_m}$. Here, β_a is the aerosol backscatter coefficient and β_m is the molecular backscatter coefficient model which uses the broadened lineshape model.

The first-order correction to the absorption coefficient may be written [17]

$$\Delta\alpha_{1st}(r) = \frac{1}{2}(\alpha_{0th}(r)\Delta W_{1st}(r) + \Delta G_{1st,1}(r) - \Delta G_{1st,2}(r)), \tag{2.4}$$

where

$$\Delta W_{1st}(r) = \frac{\int \zeta_1(\nu, r)[1 - f(\nu, r)]d\nu}{\int \zeta_1(\nu, r)d\nu} \tag{2.5}$$

and

$$\Delta G_{1st,x}(r) = \frac{\int \eta_x(\nu, r)d\nu}{\int \zeta_x(\nu, r)d\nu}, \tag{2.6}$$

where $f(\nu, r)$ is the molecular absorption lineshape with a maximum value of 1 at line center.

The terms $\eta_x(\nu, r)$ and $\zeta_x(\nu, r)$ are given by

$$\eta_x(\nu, r) = \frac{dg_x(\nu, r)}{dr} E(\nu) T_{m0th,x}(\nu, r) \tag{2.7}$$

and

$$\zeta_x(\nu, r) = g_x(\nu, r) E(\nu) T_{m0th,x}(\nu, r). \tag{2.8}$$

where $E(\nu)$ is the normalized lineshape of the optical filter transmission in the receiver of the DIAL instrument, $T_{m0th,1}(\nu, r) = exp\left(-\int_0^r \alpha_{0th}(r')f(\nu, r')dr'\right)$ and $T_{m0th,2}(\nu, r) = exp\left(-\int_0^r \alpha_{m,2}(r')f(\nu, r')dr'\right)$. It should be noted that in regions where the aerosol backscatter coefficient is small and changing rapidly, care must be taken in estimating $\frac{dg_x(\nu,r)}{dr}$ since the instantaneous derivative may be different than an estimate of this derivative based on a difference between points separated by a finite Δr.

The second-order correction to the absorption coefficient is found from [17]

$$\Delta\alpha_{2nd}(r) = \frac{1}{2}(\Delta\alpha_{1st}(r)\Delta W_{1st}(r) + \alpha_{0th}(r)\Delta W_{2nd}(r) + \Delta G_{2nd,1}(r) - \Delta G_{2nd,2}(r)), \tag{2.9}$$

where

$$\begin{aligned}\Delta W_{2nd}(r) = \Bigg[&\frac{\int \zeta_1(\nu, r)[1 - f(\nu, r)]d\nu \int \zeta_1(\nu, r)[1 - f(\nu, r)][1 - T_{m1st,1}(\nu, r)]d\nu}{\int \zeta_1(\nu, r)[1 - f(\nu, r)][1 - T_{m1st,1}(\nu, r)]d\nu} \\ &- \frac{\int \zeta_1(\nu, r)[1 - f(\nu, r)][1 - T_{m1st,1}(\nu, r)]d\nu}{\int \zeta_1(\nu, r)d\nu}\Bigg]\end{aligned} \tag{2.10}$$

and

$$\begin{aligned}\Delta G_{2nd,x}(r) = \Bigg[&\frac{\int \eta_x(\nu, r)d\nu}{\int \zeta_x(\nu, r)d\nu} \frac{\int \zeta_x(\nu, r)[1 - T_{m1st,x}(\nu, r)]d\nu}{\int \zeta_x(\nu, r)d\nu} \\ &- \frac{\int \eta_x(\nu, r)[1 - T_{m1st,x}(\nu, r)]d\nu}{\int \zeta_x(\nu, r)d\nu}\Bigg],\end{aligned} \tag{2.11}$$

with $T_{m1st,1}(\nu, r) = exp\left(-\int_0^r \Delta\alpha_{1st}(r')f(\nu, r')dr'\right)$ and $T_{m1st,2}(\nu, r) = 1$. The first-order correction to the offline atmospheric transmission resulting from molecular absorption, $T_{m1st,2}(\nu, r) = 1$, is assumed with the realization that the zeroth-order transmission is minimally affected by the offline molecular absorption and thus the first-order offline molecular absorption will have no impact on the offline atmospheric transmission resulting from molecular absorption. With the assumption that $T_{m1st,2}(\nu, r) = 1$, $\Delta G_{2nd,2}(r) = 0$ implying that only $\Delta G_{2nd,2}(r)$ need be considered.

Iterative temperature retrieval

The iterative temperature retrieval starts with the initial guess at a temperature profile with a simple lapse rate. The same temperature guess is used for the perturbative absorption retrieval. The temperature guess and the retrieved perturbative absorption are used to estimate a new temperature profile. The process iterates using the new temperature profile as the guess until the guess profile matches the updated temperature profile. The following is a summary of the theory presented in Repasky et al. [13].

To calculate the updated temperature profile, the O_2 absorption coefficient profile must be calculated from the initial guess as [15]

$$\alpha_{O_2}(r) = S(T_0)\frac{T_0}{T(r)}exp\left[\frac{\epsilon}{k_b T_0} - \frac{\epsilon}{k_b T(r)}\right] g(\nu - \nu_0, r)n_{O_2}(r), \tag{2.12}$$

where $S(T_0)$ is the line strength at the temperature T_0, ϵ is the ground state energy, k_b is the Boltzmann constant, $g(\nu - \nu_0, r)$ is the absorption lineshape function, $n_{O_2}(r)$ is the O_2 number density, and $T(r)$ is the temperature profile. The line parameters are taken from the HITRAN database [19]. The O_2 number density is dependent on Loschmidt's number,

$n_L(r)$, and is a function of both temperature and pressure, where

$$n_{O_2}(r) = q_{O_2}(n_L(r) - n_{wv}(r)) = q_{O_2}(1 - q_{wv}(r))n_L(r). \tag{2.13}$$

The atmospheric mixing ratios are q_{O_2} and $q_{wv}(r)$, which represent O_2 and H_2O, respectively. The water vapor mixing ratio, $q_{wv}(r)$, is provided by the retrieval from the water vapor MPD. Loschmidt's number can be written $n_L(r) = \frac{P(r)}{k_b T(r)}$, where the pressure is found using the barometric formula $P(r) = P_s \left(\frac{T_s}{T_s + L_r}\right)^{\frac{\epsilon_0 M}{RL}}$. In the barometric formula, g_0 is the acceleration due to gravity, M is the molar mass of air, R is the universal gas constant and L is the lapse rate in degrees per meter of the temperature guess. Using the above relationships, the absorption coefficient may be written as

$$\alpha_{O_2}(r) = \frac{S(T_0)T_0 P_s exp\left(\frac{\epsilon}{k_b T_0}\right)}{k_b T_s^{-\frac{\gamma}{L}}} g(\nu - \nu_0, r) q_{O_2}(1 - q_{wv}(r)) T(r)^{\frac{-\gamma}{L} - 2} exp\left(\frac{-\epsilon}{k_b T(r)}\right), \tag{2.14}$$

where $\gamma = \frac{g_0 M}{R}$. The updated temperature profile may be written as

$$T_{i+1}(r) = T_i(r) + \Delta T(r) = T_i(r)\left(1 + \frac{\Delta T(r)}{T_i(r)}\right), \tag{2.15}$$

where $T_i(r)$ is the initial guess at the temperature profile and $\frac{\Delta T(r)}{T_i(r)}$ is assumed to be small. The two temperature-dependent terms on the right-hand side of Eq. 2.14 can be expanded to first order such that $T_{i+1}(r)^{\frac{-\gamma}{L} - 2} \approx T_i(r)^{\frac{-\gamma}{L} - 2}\left(1 + \left(1 + (\frac{-\gamma}{L} - 2)\frac{\Delta T(r)}{T_i(r)}\right)\right)$ and $exp\left(\frac{-\epsilon}{k_b T_{i+1}(r)}\right) \approx exp\left(\frac{-\epsilon}{k_b T_i(r)}\right)\left(1 + \frac{\epsilon}{k_b T_i^2(r)\Delta T(r)}\right)$. Substituting these approximations into Eq. 2.14 and keeping terms to first order in $\frac{\Delta T(r)}{T_i(r)}$, one can show that

$$\Delta T(r) = \frac{\alpha_{O_2}(r) - C_1 C_2(r) g(\nu - \nu_0, r) q_{O_2}(1 - q_{wv}(r))}{C_1 C_2(r) C_3(r) g(\nu - \nu_0, r) q_{O_2}(1 - q_{wv}(r))}, \tag{2.16}$$

where

$$C_1 = \frac{S(T_0)T_0P_s exp\left(\frac{\epsilon}{k_bT_0}\right)}{k_bT_s^{-\frac{\gamma}{L}}}, \tag{2.17}$$

$$C_2 = T_i(r)^{\frac{-\gamma}{L}-2} exp\left(\frac{-\epsilon}{k_bT_i(r)}\right), \tag{2.18}$$

and

$$C_3(r) = \frac{\frac{-\gamma}{L}-2}{T_i(r)} + \frac{\epsilon}{k_bT_i^2(r)}. \tag{2.19}$$

Calculating $\Delta T(r)$ allows the initial temperature profile guess to be updated. Using a linear least-squares fit on the updated temperature profile allows for an estimate of the lapse rate associated with the updated temperature profile, which is necessary to complete the next iteration of the temperature profile update.

Modeling results for realistic atmospheric models has been done in Respasky et. al. [13]. In general, the temperature retrieval maintains a temperature deviation of less than $\pm 1\,\mathrm{K}$ in areas with higher aerosol backscatter coefficients. However, when the atmospheric scattering is dominated by molecular scatter, the temperature retrieval results in a larger temperature deviation.

HSRL

This DLB-HSRL uses the online and offline O_2 DIAL wavelengths (specified in Table 3.1). The offline DIAL wavelength is on an absorption line of potassium (KD_1). Using the DIAL wavelengths eliminates the need for a third laser for a separate rubidium or iodine HSRL. Another benefit of this configuration is that the online wavelength can be used for efficiency and overlap correction between the molecular and combined channels.

The HSRL data can be processed using a process developed for rubidium HSRL [20].

The signal in the combined channel is described by

$$S_c(r) = K\frac{\eta_c(r)}{r^2}[\beta_m(r) + \beta_a(r)]exp[-2\int_0^r \alpha(\zeta)d\zeta] + B_c, \tag{2.20}$$

where $S_c(r)$ is the measured combined channel signal as a function of range, K is an arbitrary constant for power and efficiency terms, $\eta_c(r)$ is the combined channel geometric overlap function, $\beta_a(r)$ is the molecular backscatter coefficient, $\beta_a(r)$ is the aerosol or cloud backscatter coefficient, $\alpha(r)$ is the total absorption coefficient and B_c is the combined channel background counts.

The molecular channel is described by

$$S_m(r) = K\frac{1}{G_m}\frac{\eta_m(r)}{r^2}[C_{mm}(r)\beta_m(r) + C_{am}\beta_a(r)]exp[-2\int_0^r \alpha(\zeta)d\zeta] + B_m, \tag{2.21}$$

where $S_m(r)$ is the measured molecular channel signal, $1/G_m$ is the difference in efficiency between the combined and molecular channels, $\eta_m(r)$ is the molecular channel geometric overlap function, C_{am} is the aerosol-to-molecular crosstalk representing the amount of narrowband light leakage through the potassium vapor optical notch filter and B_m is the molecular channel background counts. The term $C_{mm}(r)$ is the efficiency of the molecular return through the molecular channel filter. Since the spectral shape of the molecular backscatter varies with temperature and pressure, $C_{mm}(r)$ is calculated as a function of range based on the models shown in Section 2.

Using the online signal, the geometric overlap between the molecular and combined channels can be measured as

$$\frac{\eta_m}{\eta_c} = \tilde{G}_m^{on}\frac{(\tilde{S}_m^{on}(r) - \tilde{B}_m^{on})}{(\tilde{S}^{on}(r)_c - \tilde{B}_c^{on})} = O(r), \tag{2.22}$$

where the superscript (on) represents the factors at the online DIAL wavelength. The right

side of the equation is defined as an overlap correction factor $O(r)$.

To compare the two signals we can integrate the measured constants and efficiencies into new signal terms.

$$\hat{S}_c(r) = \tilde{S}_c(r) - \tilde{B}_c, \tag{2.23}$$

and

$$\hat{S}_m(r) = \frac{\tilde{G}_m(\tilde{S}_m(r) - \tilde{B}_m) - \tilde{C}_{am}O(r)(\tilde{S}_C(r) - \tilde{B}_c)}{\tilde{C}_{mm}(r) - \tilde{C}_{am}} \tag{2.24}$$

represent the combined and molecular signals that can be used to calculate the aerosol and molecular backscatter.

The backscatter coefficient of atmospheric aerosols is calculated as

$$\tilde{\beta}_a(r) = [\frac{\hat{S}_C(r)}{\hat{S}_m(r)} - 1]\tilde{\beta}_m(r), \tag{2.25}$$

where $\tilde{\beta}_a(r)$ is the estimated aerosol backscatter coefficient and $\tilde{\beta}_m(r)$ is the estimated molecular backscatter coefficient. The molecular backscatter coefficient is estimated from pressure and temperature profiles using [21]

$$\tilde{\beta}_m(r) = 5.45 \times 10^{-32} \frac{P(r)}{k_B T(r)} \left(\frac{550nm}{\lambda}\right)^4 m^{-1} sr^{-1}, \tag{2.26}$$

where $P(r)$ is the pressure in Pa, $T(r)$ is the temperature in K, λ is the wavelength of the laser in nm and k_B is the Boltzmann constant.

The backscatter ratio (BSR) is defined as the ratio of total backscatter (molecular and aerosol), β_t, to molecular backscatter, β_m, and is obtained from HSRL observations using equation 2.26 and equation 2.25

$$BSR(r) = \frac{\beta_t}{\beta_m} = \frac{\beta_a(r) + \beta_m(r)}{\beta_m(r)}. \tag{2.27}$$

Masking

Some data are not usable for temperature retrievals due to low signal or large backscatter signal from clouds causing an inaccurate DIAL absorption measurement. Masking the data so that incorrect data is not used in the temperature retrieval is important. A signal to noise (SNR) mask is used to mask out data that is below an SNR threshold. As altitude increases, the signal drops away so that at a certain altitude the signal to noise ratio is so low that data is unusable. The main source of noise in our system is Poisson noise due to photon counting. The signal to noise ratio for Poisson noise is calculated as

$$SNR(r) = \sqrt{N(r)}, \tag{2.28}$$

where $N(r)$ is the number of photons counted at a certain altitude. A mask is used to mask out data above the point after the SNR drops below a threshold on an order less than ten.

Data is also unusable when there are clouds in the dataset. Clouds cause a large spike in the photon counting returns and could saturate the photon counting modules. One way clouds can be detected is using a rolling standard deviation window [22]. First the return photons counts are range corrected, $N_{corrected}(r) = N(r) \times r^2$. Each bin is assigned a value of the standard deviation of all the bins around it in range and time. If the standard deviation is above a certain value, the bin is classified as a cloud and is masked out of the data. All data above the clouds are also masked. The thresholds for standard deviation mask is set manually by using a range-corrected profile without clouds (data are range corrected by multiplying by the square of the range to compensate for inverse-range-squared measurement dependence).

METHODS

DIAL Instrument

The new instrument is be based on an existing architecture, starting from a water vapor DIAL system, and has been optimized for performance and autonomous operation. The latest modifications to the DIAL instument include fiber coupled DBR lasers for ease of setup and autonomous operation, spectrally narrow optical filters and etalon for increased signal to background ratio, and a near-field receiver for higher signal at ranges below 1.5 km.

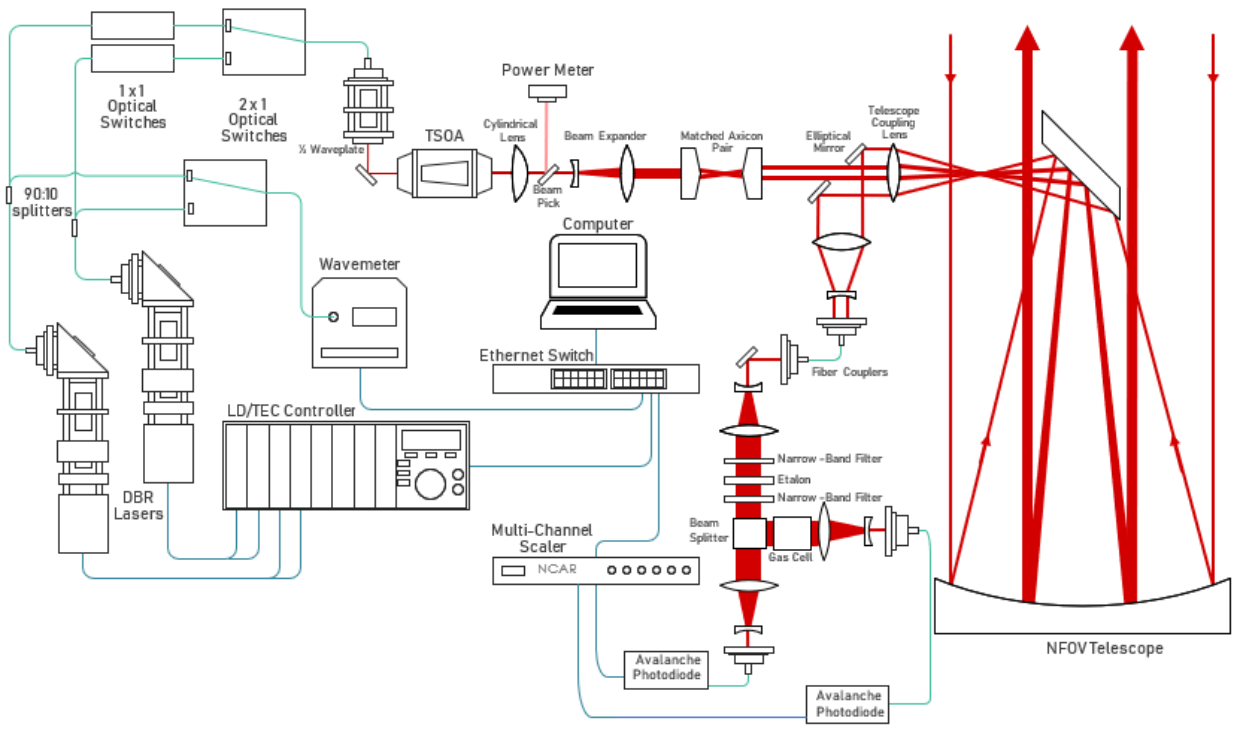


Figure 3.1: Schematic of O_2 DIAL and HSRL instrument. Blue lines indicate electrical power and signal wires. Grey lines indicate fiber optics. Red lines indicate free space beam lines. The WV DIAL system is not shown but would be a copy of the transmitter with different wavelength lasers and a beam combiner before the axicon pair. The WV receiver would include a beamsplitter before the filtering and include its own filters, etalon, and avalanche photo diode.

A schematic of the instrument is shown in Figure 3.1 and specifications are listed in

Table 3.1. The transmitter starts with two distributed brag reflector (DBR) lasers. The lasers are at a wavelength of 770 nm and are sourced from Photodigm. The laser light then passes trough a single mode fiber splitter with a 10% tap. The tap is directed into a wavemeter and a 2x1 switch controlled by a computer used to wavelength lock the lasers using their temperature. The main laser light passes through 1x1 and 2x1 switches to switch the wavelength of the transmitted light and isolate the other wavelength. The light is collimated from the fibers and into the tapered semiconductor optical amplifier (TSOA). The TSOA is the pulsed amplifier generating the transmitted signal that is seeded by the DBR lasers. A cylindrical lens collumates the output of the TSOA. The beam then passes through a matched axicon pair to create an annular mode with a hole in the center. The hole in the beam is so that the transmitted power goes around the secondary mirror of the Newtonian telescope. A coupling lens is matched to the $f/3$ telescope so that the transmitter beam covers the inside of the telescope (with a hole in the transmitted beam where it would have otherwise relfected back from the secondary mirror). A transmitter for the water vapor wavelengths (not shown in the figure) is copied before the amplifier.

The backscattered photons are collected using the outside ring of the telescope. A mirror with a 10 mm diameter hole drilled in the middle collects the received light and the transmitted beam goes through the hole. All collected light is coupled into a 105 μm core mulitmode fiber in the first stage and directed into a second stage containing filtering elements. There are two sections for water vapor and oxygen wavelengths. Each section has two narrow band filters and an etalon to filter background light. The etalons are tuned so that the online and offline wavelengths are one free spectral range apart. In the oxygen receiver a 70/30 beamsplitter directs 70% of the light into a heated potassium vapor cell in order to filter out the aerosol backscatter returns and transmit the spectrally broad molecular backscatter returns into a multimode fiber and an avalanche photodiode for photon counting. The other side of the beamsplitter is coupled into a multimode fiber and into a separate

avalanche photodiode. The transmitted spectrum measurement is shown in Figure 3.2.

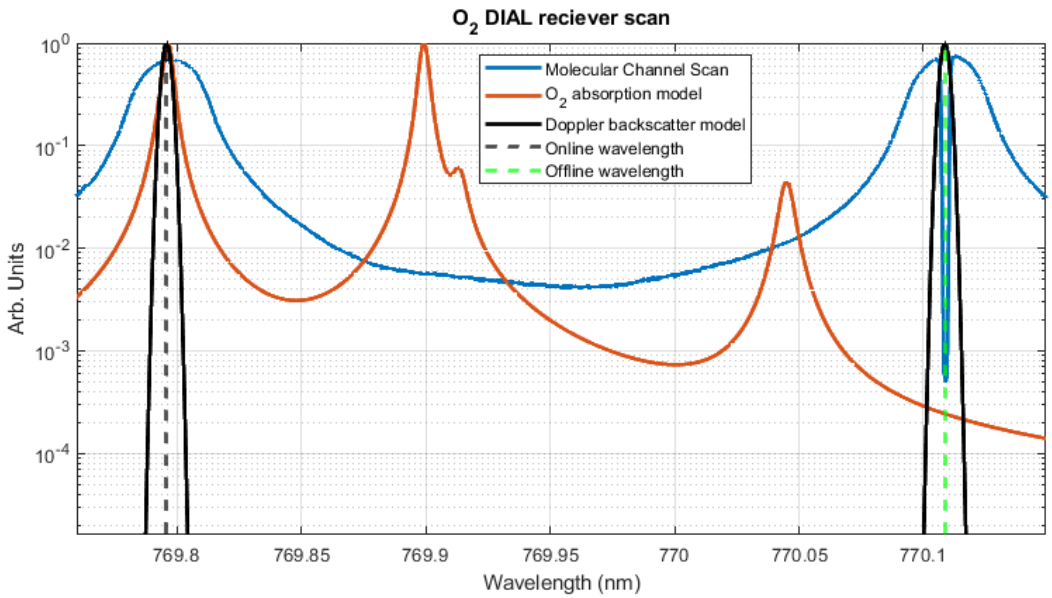


Figure 3.2: Section of the calculated O_2 absorption spectrum in red with the calculated Doppler backscatter spectrum in black and measured receiver transmission spectrum of the molecular channel in blue. The online and offline laser wavelengths are represented by the dashed vertical lines. The O_2 absorption cross section is taken at 300 K. The combined channel absorption spectrum is identical to the molecular channel without the KD_1 absorption line shown as the large dip at the offline wavelength. The online wavelength is chosen at the line center of the 769.7958 nm O_2 absorption line and the offline wavelength is chosen at the line center of the KD_1 line at 770.1085 nm.

MATLAB program

All data are processed in MATLAB. The inputs to the program are the photon counts in time and range for each of the four lidar channels, which are the online and offline wavelengths that each go though the potassium filter to filter out aerosol returns (molecular channel) and the channel that does not go through the potassium filter (combined channel). The other inputs are the surface temperature and pressure as a function of time from a collocated weather station to create an initial atmospheric model. The inputs that do not change with time are the line parameters of the O_2 absorption line and a spectral scan of the receiver

Table 3.1: Specifications for the combined O2 DIAL and HSRL system presented in Fig. 3.1. Abbreviations used are: FSR = free spectral range and NBF = narrow band filter.

Transmitter	Specification	Reciever	Specification
Wavelength	769.7958 nm (online)	Etalon Finesse	15.43
	770.1085 nm (online)	Etalon FSR	157.90 GHz
Output Power	30 mW(combined)	NBF Bandwidth	13 nm, 1 nm
Linewidth	<1 MHz	NBF out-of-band blocking	$10^{-6}, 10^{-4}$
Locking Tolerance	5×10^{-5} nm	NBF transmission	90%, 70%
Spectral Purity	99.9%	Potassium cell length	72 nm
Repetition Rate	7 kHz(combined)	Potassium cell temperature	378 K

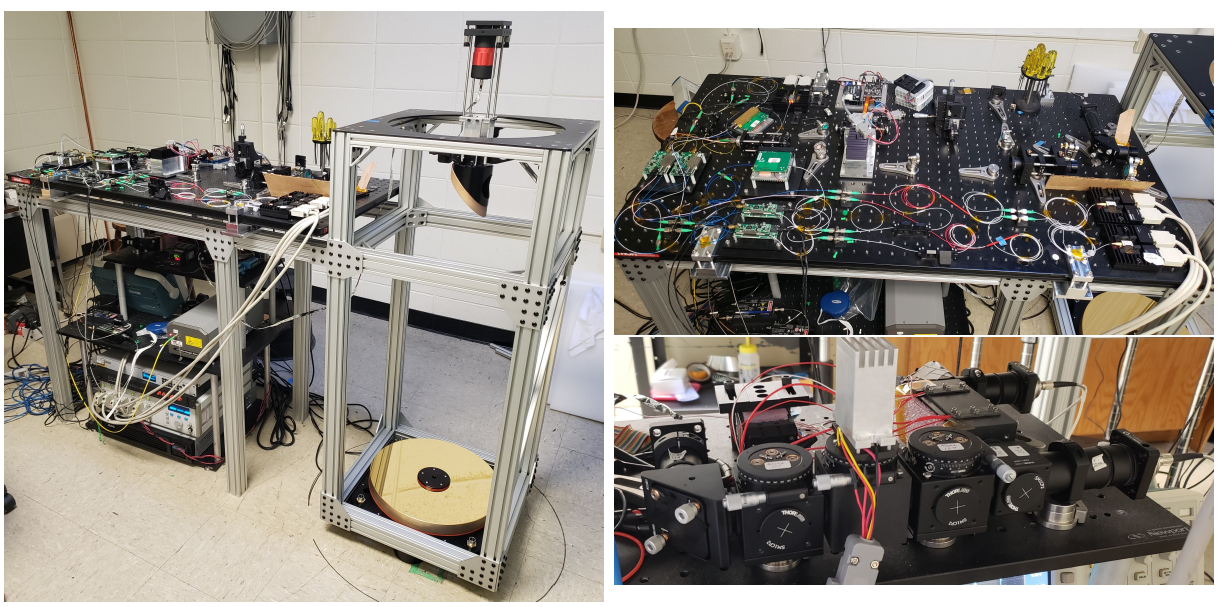


Figure 3.3: Left picture shows a picture of the full MPD instrument excluding the computer. The Upper right picture shows the transmitter optics and the first stage receiver. The lower right picture shows the second stage filtering optics with the two interference filter, temperature controlled etalon, and Potassium vapor oven.

transmission. At first the program loads in data to be processed from files and all set to the same time grid. The background is subtracted from the photon data by using an averaged value of the photon returns at sufficient range. The photon data are then filtered using a 2D rectangular filter in 30 minutes of time and 300 m in range which equates to 30 time bins and 8 range bins. A cloud mask and SNR mask of the data is created using methods described earlier. The backscatter ratio is calculated using method described earlier. The O_2 absorption spectrum for each bin in range and time is calculated using the atmospheric model and the O2 line parameters from HITRAN using the equations for the Voight profile. This is the most computationally intensive part of the program so principal component analysis is used to speed up computation time [23]. The purtabative absorption is calculated from the method in the theory section. The final second-order absorption is cut above the point (500 m) where data are unreliable at low altitudes due to the lidar overlap function being very low and having a high slope. The absorption data are replaced with the calculated O_2 absorption from temperature and pressure taken at the ground station and an interpolation up to the cut point. The absorption is smoothed using reliable data from the SNR and cloud mask. Finally the second-order absorption is used to calculate the temperature profile using the method described earlier.

RESULTS AND DISCUSSION

Results from ARM SGP

Data has been collected from a O_2 MPD instrument in April 2019 at the Atmospheric Radiation Measurement (ARM) Southern Great Plains (SGP) site run by the Department of Energy. The results of the temperature measurements are presented. This is some of the first experimental results from an O_2 MPD temperature profiling system. Collocated radiosondes were launched periodically for validation of retrieved temperature profiles. The data represented over time are shown in Figure 4.1 where the color represents the retrieved temperature, the white areas represent data below the signal to noise threshold and data around clouds that are not usable, and the black areas represent clouds. Radiosonde launches are represented by the black vertical lines at the time of launch.

At the ARM site, four radiosondes were launched per day. The sonde data are interpolated to the DIAL bins in rage with a height of 37.5 m. During the time period of data shown, there were 26 sonde launches, 13 sondes with no clouds, 8 with clouds, and 5 with data that are not considered due to very low clouds or during periods when the instrument was being aligned. A plot of the sonde vs DIAL temperature is shown in Figure 4.2a with a histogram of the difference between the two in Figure 4.2b. The mean of the difference between the sonde measurement and the DIAL measurement is -1.1 K and the standard deviation is 2.7 K. This could be improved with updates to the instrument to increase signal to background ratio and adding a near-field channel to use data at ranges below 1 km. The position of the mean difference indicates that the pertabative absorption correction is not increasing as much as it should.

Selected profiles of a sonde launch compared to DIAL retrievals are shown in Figures 4.3 and 4.4. Figure 4.3 shows the calculated absorption from sonde measurement,s as well as the three absorption steps of the perturbative retrieval, including the zeroth-order

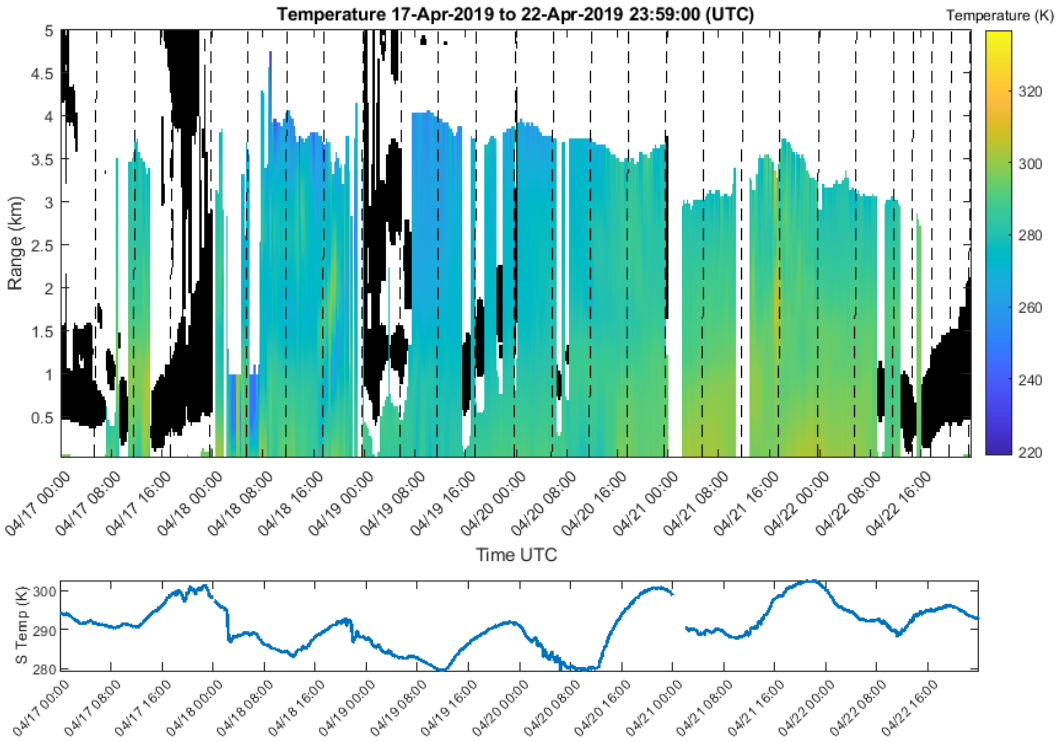
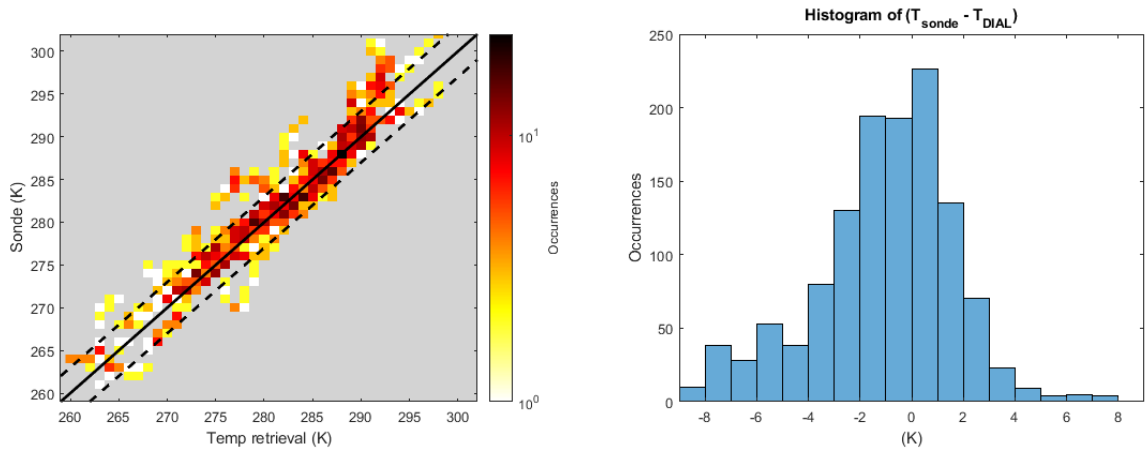


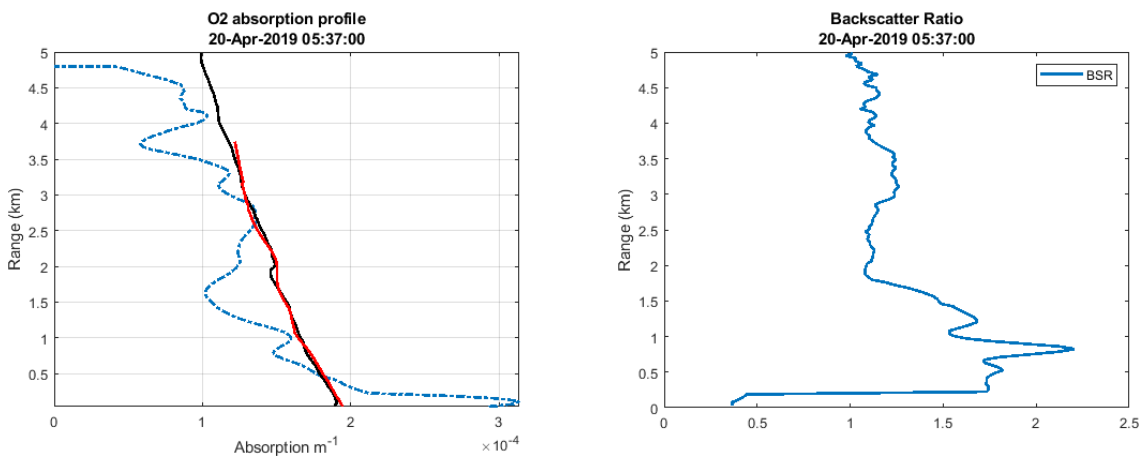
Figure 4.1: Temperature profile time-series for data taken at the ARM SGP site from April 17-22, 2020. Dates and times are based on UTC. Areas masked in black are where clouds are detected. Areas in white are were data are below a signal to a noise ratio threshold. Dashed Vertical lines correspond to radiosonde launches. The lower plot shows surface temperature measurements from a co-located weather station.



(a) (b)

Figure 4.2: Comparison of measured radiosonde temperature and temperature retrieved by the O_2 DIAL and HSRL system. Data was taken at the ARM site from April 17-22, 2020. The color axis is the number of occurrences in each square bin with dimensions of $1K$. The center line corresponds to a 1:1 relation between radiosonde and DIAL. The two side lines represent a $\pm 2K$ difference. On the right are the same data represented as a Histogram of the differences between the radiosonde temperature measurements and the DIAL temperature measurements with bin widths of 1 K.

absorption, the absorption with the perturbative correction and the final absorption used for the temperature retrieval. The measured profile of backscatter ratio is included as well. At the point of high gradient of the BSR in range there is a dip in the zeroth-order absorption. The perturbative absorption process is shown to correct the dip in absorption. Figure 4.4 shows a final temperature profile and a sonde measurement as well as the difference between the two. Most of the temperature errors occur in low-altitude regions and regions with high temperature gradient.

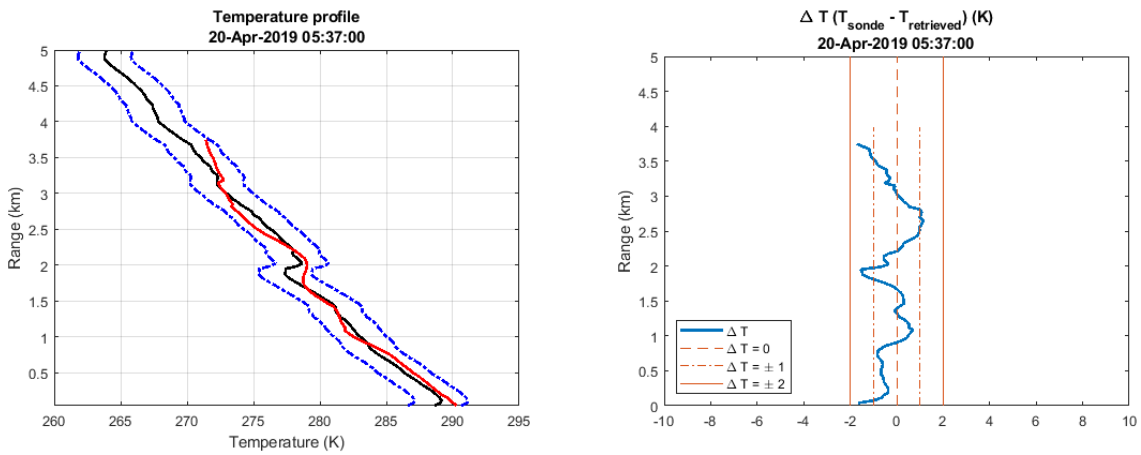


(a) (b)

Figure 4.3: On the left is a profile of O_2 absorption coefficient. The zeroth-order absorption coefficient is represented by the dashed blue line. The red line represents the result of the perturbative correction process and corrects for the error in the zeroth-order profile. The red line represents the final absorption profile used to calculate temperature with the values below 1 km cut to the value of surface temperature and pressure and with smoothing. Finally, the black line represents the O_2 absorption calculated from the sonde measurements of temperature, pressure, and water vapor. On the right is the backscatter ratio (BSR) profile retrieved from the HSRL.

Results from MSU campus

Data were also collected from the MPD instrument located on Montana State University in Cobleigh hall. The O_2 MPD system was set up in March of 2020 and has since been



(a) (b)

Figure 4.4: On the left is a comparison of a retrieved temperature profile in red and a temperature profile measured by a radiosonde in black. The blue dashed lines on the right and left of the radiosonde temperature representing $\pm 2\,\mathrm{K}$. On the right is the difference between the radiosonde temperature and the retrieved temperature in blue. Vertical lines represent $1\,\mathrm{K}$ and $2\,\mathrm{K}$ difference.

running almost continually. The narrow-band spectral filters and etalon were replaced in July 2020 with narrower versions to reduce background. Radiosonde data were collected from a collocated radiosonde station on the roof of Cobleigh hall. Surface weather data were collected from a collocated weather station on the roof of Cobleigh hall run by the Optical Remote Sensor Laboratory (ORSL) on campus [24]. A near-field channel was added to the lidar in late September 2020. Large amounts of aerosol returns from wildfires in California and across the northwest were present in the data from August and early September.

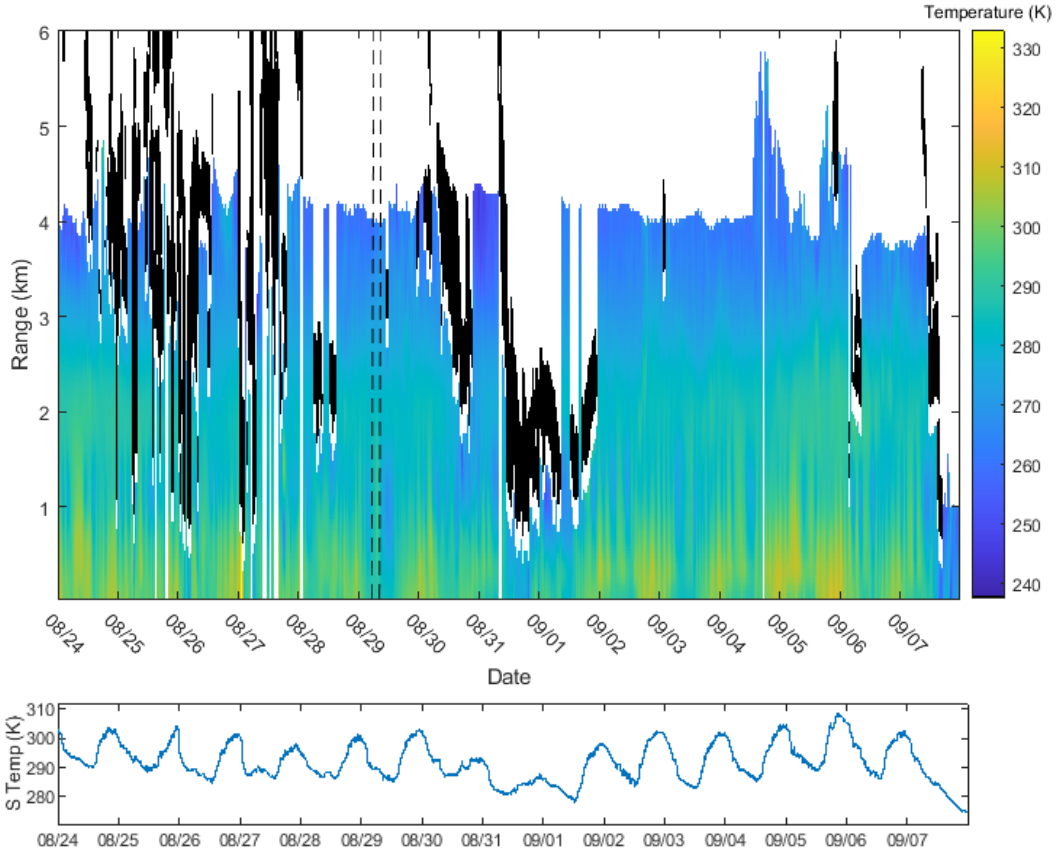
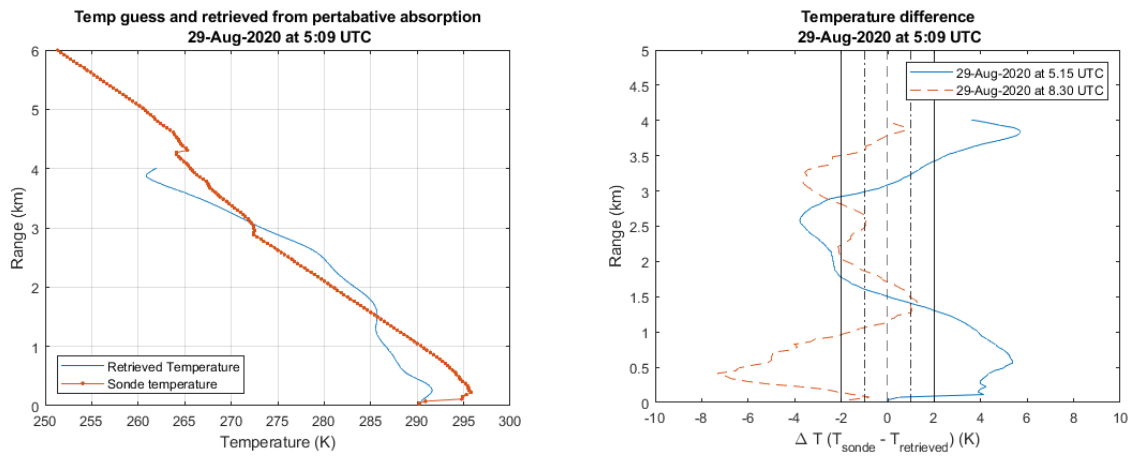
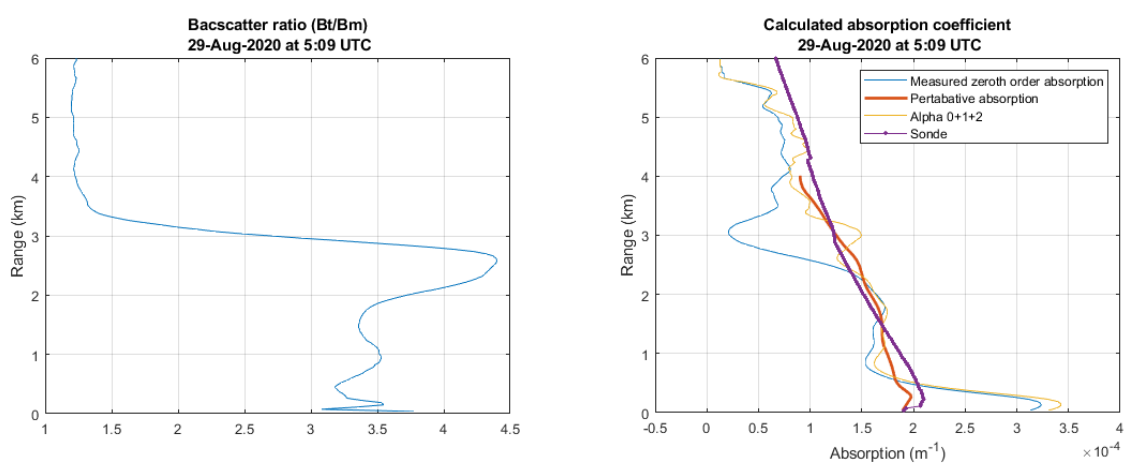


Figure 4.5: Temperature profile for late August to early September 2020. Areas masked in black are where clouds are detected. Areas in white are were data are below a signal to a noise ratio threshold. The two vertical lines indicate two radiosonde launches used as comparisons in Figure 4.6. The lower plot shows the surface temperature taken at a co-located weather station run by the MSU ORSL.



(a) (b)

Figure 4.6: On the left is a comparison of a retrieved temperature profile and a temperature profile measured by a radiosonde. On the right is the difference between the radiosonde temperature and the retrieved temperature. Two radiosonde profiles are represented. The vertical lines represent $\pm 1\,\mathrm{K}$ and $\pm 2\,\mathrm{K}$ difference.



(a) (b)

Figure 4.7: On the left is a a retrieved BSR profile. On the right is the retrieved absorption with zeroth order, corrected, and measured from a radiosonde.

Results from NCAR in Boulder, CO

Another O_2 MPD instrument is being run by NCAR in Boulder, CO. This instrument is similar to the current instrument at MSU and includes the WV MPD as well. It also includes a near-field receiver. A temperature profile over multiple days is shown below in Figure 4.8. The smoke from California wildfires is at a lower altitude than the data taken in Bozeman and shows in the data starting on the third of September.

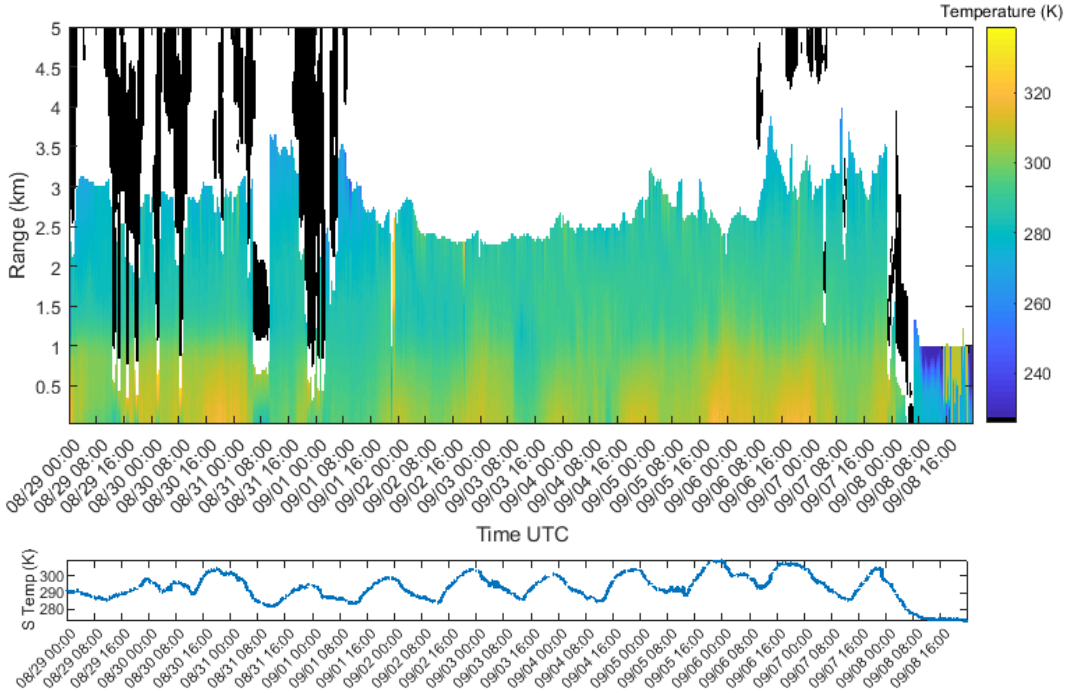


Figure 4.8: Temperature profile for late August to early September 2020. Areas masked in black are where clouds are detected. Areas in white are were data are below a signal to a noise ratio threshold. The lower plot shows the surface temperature measured by a weather station.

CONCLUSION

Micro pulse DIAL (MPD) can provide thermodynamic atmospheric profiling measurements on a continuous basis. Continuous measurements up to 4 km can be made autonomously. This is the main advantage of MPD over other methods of thermodynamic profiling. Radiosondes cannot take continuous measurements and Raman lidar's complex and high-power lasers have eye safety and autonomous concerns. The combined instrument also offers a measurement of the aerosol backscatter that can be used to find the boundary layer.

In this paper, a micro pulse DIAL instrument capable of measuring temperature has been demonstrated. Validation of data taken with radiosondes at the SGP ARM campaign have shown that this instrument is capable of accurate measurements with a bias of -1.1 K and a standard deviation of 2.7 K. Continuous data also shows that this instrument has the potential for autonomous operation for weeks to months.

Future work will focus on improving the temperature retrieval. Improvements to the instrument will increase the reliability and long-term autonomous operation. Near-field improvements with the near-field telescope will improve the retrieval at lower ranges. The analysis program will be optimized to run in real time.

REFERENCES CITED

[1] National Research Council (U.S.). Observing weather and climate from the ground up: A nationwide network of networks. *The National Academies Press*, 2009.

[2] National Research Council (U.S.). Committee on progress and priorities of u.s. weather research and research to- operations activities. *When weather matters: science and services to meet critical societal needs (National Academies Press*, 2010.

[3] R. E. Carbone, R. J. Serafin, R. M. Hoff, R. M. Hardesty, F. Carr, T. Weckwerth, S. Koch, A. Benedetti, S. Crewell, D. Cimini, D. Turner, W. Feltz, B. Demoz, V. Wulfmeyer, D. Sisterson, T. Ackerman, F. Fabry, and K. Knupp. Thermodynamic profiling technologies workshop report to the national science foundation and the national weather service. *NCAR Technical Note NCAR/TN-488 + STR*, 2012.

[4] V. Wulfmeyer, R. M. Hardesty, D. D. Turner, A. Behrendt, M. P. Cadeddu, P. DiGirolamo, P. Schlüssel, J. Van Baelen, and F. Zus. A review of the remote sensing of lower tropospheric thermodynamic profiles and its indispensable role for the understanding and the simulation of water and energy cycles. *Rev. Geophys.*, 53(3):819–895, 2015.

[5] D.A. Whiteman. Examination of the traditional raman lidar technique. i. evaluating the temperature-dependent lidar equations. *Appl. Opt.*, 42(15):2571–2592, 2003.

[6] D.A. Whiteman. Examination of the traditional raman lidar technique. i. evaluating the ratios for water vapor and aerosols. *Appl. Opt.*, 42(15):2593–2608, 2003.

[7] R. Newsom and C. Sivaraman. Raman lidar water vapor mixing ratio and temperature value-added products doe/sc-arm-tr-218. Technical report, US Department of Energy, November 2018.

[8] A. R. Nehrir, K. S. Repasky, and J. L. Carlsten. Eye-safe diode-laser-based micropulse differential absorption lidar (dial) for water vapor profiling in the lower troposphere. *J. Atmos. Oceanic Technol.*, 28(2):1373–147, 2011.

[9] A. R. Nehrir, K. S. Repasky, and J. L. Carlsten. Micropulse water vapor differential absorption lidar: transmitter design and performance. *Opt. Express*, 20(2):25137–25151, 2012.

[10] A. R. Nehrir, K. S. Repasky, J. L. Carlsten, M. D. Obland, and J. A. Shaw. Water vapor profiling using a widely tunable, amplified diode-laser-based differential absorption lidar (dial). *J. Atmos. Oceanic Technol.*, 26(4):733–745, 2009.

[11] K. S. Repasky, D. Moen, S. Spuler, A. R. Nehrir, and J. L. Carlsten. Progress towards an autonomous field deployable diode-laser-based differential absorption lidar (dial) for profiling water vapor in the lower troposphere. *Remote Sens.*, 5(12):6241–6259, 2013.

[12] S. Spuler, K. S. Repasky, B. Morley, D. Moen, M. Hayman, and A. R. Nehrir. Field-deployable diode-laser-based differential absorption lidar (dial) for profiling water vapor. *Atmos. Meas. Tech*, 8(3):1073–1087, 2015.

[13] Kevin S. Repasky, Catharine E. Bunn, Matthew Hayman, Robert A. Stillwell, and Scott M. Spuler. Modeling the performance of a diode laser-based (DLB) micro-pulse differential absorption lidar (MPD) for temperature profiling in the lower troposphere. *Optics Express*, 27(23):33543–33563, November 2019.

[14] Robert A. Stillwell, Scott M. Spuler, Matthew Hayman, Kevin S. Repasky, and Catharine E. Bunn. Demonstration of a combined differential absorption and high spectral resolution lidar for profiling atmospheric temperature. *Optics Express*, 28(1):71–93, January 2020.

[15] F. A. Theopold and J. Bösenberg. Differential absorption lidar measurements of atmospheric temperature profiles: Theory and experiment. *Journal of Atmospheric and Oceanic Technology*, 10(2):165–179, 1993.

[16] Jens Bösenberg. Ground-based differentail absroption lidar for water-vapor and temperature profiling: methodology. *Applied Optics*, 37(18):3845–3860, June 1998.

[17] Catharine E. Bunn, Kevin S. Repasky, Matthew Hayman, Robert A. Stillwell, and Scott M. Spuler. Pertabative solution to the two-component atmosphere DIAL equation for improving the accuracy of the retrieved absorption coefficient. *Applied Optics*, 57(16):4440–4450, June 2018.

[18] S. T. Shipley, D. H. Tracy, E. W. Eloranta, J. T. Trauger, J. T. Sroga, F. L. Roesler, and J. A. Weinman. High spectral resolution lidar to measure optical scattering properties of atmospheric aerosols. 1: Theory and instrumentation. *Applied Optics*, 22(23):3716–3724, 1983.

[19] I.E. Gordon, L.S. Rothman, C. Hill, R.V. Kochanov, Y. Tan, P.F. Bernath, M. Birk, V. Boudon, A. Campargue, K.V. Chance, B.J. Drouin, J.-M. Flaud, R.R. Gamache, J.T. Hodges, D. Jacquemart, V.I. Perevalov, A. Perrin, K.P. Shine, M.-A.H. Smith, J. Tennyson, G.C. Toon, H. Tran, V.G. Tyuterev, A. Barbe, A.G. Csaszar, V.M. Devi, T. Furtenbacher, J.J. Harrison, J.-M. Hartmann, A. Jolly, T.J. Johnson, T. Karman, I. Kleiner, A.A. Kyuberis, J. Loos, O.M. Lyulin, S.T. Massie, S.N. Mikhailenko, N. Moazzen-Ahmadi, H.S.P. Muller, O.V. Naumenko, A.V. Nikitin, O.L. Polyansky, M. Rey, M. Rotger, S.W. Sharpe, K. Sung, E. Starikova, S.A. Tashkun, J. Vander Auwera, G. Wagner, J. Wilzewski, P. Wcislo, S. Yu, and E.J. Zak. The hitran2016 molecular spectroscopic database. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 2017.

[20] Matthew Hayman and Scott Spuler. Demonstration of a diode-laser-based high spectral resolution lidar (HSRL) for quatitative profiling of clouds and aerosols. *Optics Express*, 25(24):A1096–A1110, November 2017.

[21] R. M. Measures. *Laser Remote Sensing*. Wiley-Interscience, 1984.

[22] Ioannis Binietoglou, Giuseppe D'Amico, Holger Baars, Livo Belegante, and Eleni Marinou. A methodology for cloud masking uncalibrated lidar signals. volume 176. EPJ Web of Conferences, 2018.

[23] Matthew Hayman, Robert A. Stillwell, and Scott M. Spuler. Fast computation of absorption spectra for lidar data processing using principal component analysis. *Optics Letters*, 44(8):1900–1903, 2020.

[24] J. Shaw. Montana State University Optical Remote Sensor Laboratory weather station. `https://www.montana.edu/orsl/weather.html`. Accessed: 2021-04-13.

APPENDICES

APPENDIX A

EXAMPLE CODE

This work used the following MATLAB code to produce a final temperature profile using data from the DIAL instrument.

```
%Reading data, calculating pertabative absorption, and iterative
%temperature for O2 DIAL data

%Owen Cruikshank
%February 16, 2019

clear all

%%
%====================
%==== Constants =====
%====================
g0 = 9.80665;          %[m/s^2] Gravitational acceleration
M_air = 0.0289644;     %[kg/mol] Molar mass of Earth's air
R = 8.3144598;         %[J/(mol*K)] Universal gas constant

c = 2.99792458E8;              %[m/s] Speed of light
kb = 1.38065E-23;              %[J/K][m^2 kg s-2 K-1] Boltzman's constant
h = 6.626E-34;                 %[Js] Planck's constant
mo2 = 5.314E-26;               %[kg] Mass O2 molecule
mWV = 2.9915e-26;              %[kg] Mass H2O molecule
m_air = 4.792E-26;              %[kg] Mass of air
q_O2 = .2095;                  %[unitless] O2 atmospheric mixing ratio
No = 2.47937E25;               %[1/m^3] Loschmidt's number  (referenced to 296 K and 1 atm)
%%

disp('Reading in files')
%====================
%=== Reading Data ===
%====================


date_begin = datetime(2020,9,2);
date_end   = datetime(2020,9,2);
span_days = date_begin:date_end;          % Days in set [datetime]


cd ../
pathMATLAB = [pwd '\Data\NCAR Boulder Data\MatlabPreload\'];
pathPython = [pwd '\Data\NCAR Boulder Data\Python\'];
pathSonde = [pwd '\Data\NCAR Boulder Data\'];
cd ../../../
homepath = pwd;
cd( '.\OneDrive - Montana State University - Bozeman\Research\O2 DIAL\analysis')

[Data] = loadNCARBoulderData(span_days,pathMATLAB);
[DataPython] = loadNCARBoulderDataPython(span_days,pathPython);
DataSonde = loadNCARBoulderDataSonde(span_days,pathSonde);


T_sgp_raw = DataSonde.temperature;
rm_sgp = DataSonde.rm;
P_sgp_raw = DataSonde.pressure;
datetime_sgp_cell = DataSonde.date;
t_single_sgp = DataSonde.date_ts;


ts_raw_o2_on = Data.Lidar.Interp.O2OfflineComb.TimeStamp' * 60*60;
ts_raw_o2_off = ts_raw_o2_on;

bins = 49;

o2on_raw = Data.Lidar.Interp.O2OnlineComb.Data' /bins;
o2off_raw = Data.Lidar.Interp.O2OfflineComb.Data' /bins;
O2_Offline_Molecular_Detector_Backscatter_Channel_Raw_Data = Data.Lidar.Interp.O2OfflineMol.Data' /bins;
O2_Online_Molecular_Detector_Backscatter_Channel_Raw_Data = Data.Lidar.Interp.O2OnlineMol.Data' /bins;
rm_raw = 250e-9*(0:length(o2on_raw(:,1))-1)*c/2;
rm_raw = rm_raw' - 46.8;


range_Backscatter_Ratio = DataPython.range;
time_Backscatter_Ratio = DataPython.time;
Backscatter_Ratio = DataPython.Backscatter_Ratio;

Absolute_Humidity = DataPython.Absolute_Humidity;
Absolute_Humidity_mask = DataPython.Absolute_Humidity_mask;
time_Absolute_Humidity = DataPython.time;
range_Absolute_Humidity = DataPython.range;

time_Temperature = DataPython.time;
range_Temperature = DataPython.range;
Temperature = DataPython.Temperature_Model;
Surface_Temperature_HSRL = DataPython.Surface_Temperature;
```

```
time_Pressure = DataPython.time;
range_Pressure = DataPython.range;
Pressure = DataPython.Pressure_Model;
Surface_Pressure_HSRL = DataPython.Surface_Pressure;

O2Offline_Wavelength = Data.TimeSeries.Laser.O2Offline.WavelengthActual;
O2Online_Wavelength = Data.TimeSeries.Laser.O2Online.WavelengthActual;


%%

disp('Creating time and range vectors')
% ====================
% === Time Vectors ===
% ====================
t_start=ts_raw_o2_on(1);
 t_end = ts_raw_o2_on(end); %[s] Ending time
 t_step = 60;                                   %[s] Time step
ts = 0:t_step:86340;
 thr = ts / 60 / 60;                            %[hr] Time vector
 i_time = length(ts);                           %[none] length of time vector

%%
%Fill in nans
[y,m,d]=ymd(span_days(1));
date_ts_N = datetime(y,m,d,0,0,ts);
[hours,minute,sec] = hms(date_ts_N);

ts_D_NLogical = ~((hours == 0) & (minute == 0));

ts_D_N = ts(ts_D_NLogical);

date_ts = date_ts_N;

index = 1;
for j = 1:length(date_ts_N)
    if index<=length(date_ts)
        if isequal(datenum(date_ts_N(j)),datenum(date_ts(index)))
            mask_nodata(:,j) = ones(133,1);
            index = index+1;
        else
            mask_nodata(:,j) = zeros(133,1);
        end
    else
        mask_nodata(:,j) = zeros(133,1);
    end
end

%%


% Minutes to average data over
t_avg = 31;
% Range oversample
oversample = 9;

% ====================
% === Range Vectors ===
% ====================
%create range vector from length of data bins
nsPerBin = double(250);                                 %[ns] Convert data to double
NBins = double(560);                                    %[none] Convert data to double
rangeBin = (c * nsPerBin(1)*10^-9)/2;                   %[m] Create range bin size from speed of light and bin
      time over 2
rm_raw_o2 = rangeBin:rangeBin:NBins(1)*rangeBin;        %[m] Create range vector
rm_raw_o2 = rm_raw_o2(:);                               %[m] Convert range vector to column vector
r_max = 5000;                                           %[m] Max range
rm = rm_raw_o2(rm_raw_o2<=r_max);                       %[m] Shorten range vector
rkm = rm./1000;                                         %[km] Range vector
i_range = length(rm);                                   %[none] Size of range vector


%%
% SGP Radiosonde Data
% Can't interpolate with array with repeating values, so use Kevin's sonde preparation function
for i = 1:numel(T_sgp_raw)
    if ~isempty(T_sgp_raw{i}) & ~isnan(T_sgp_raw{i})
        % Subtract first range value (site elevation) from whole vector
        rm_sgp{i} = rm_sgp{i} - rm_sgp{i}(1);
        % Collect radiosonde surface measurements
        T_sgp_surf(i) = T_sgp_raw{i}(1);
        P_sgp_surf(i) = P_sgp_raw{i}(1);
        % Convert datetimes from cells to vector
        datetime_sgp(i) = datetime_sgp_cell{i};
        % Custom interpolation function
        [T_sgp_int{i},P_sgp_int{i},rm_sgp_int{i}] = interp_sonde(T_sgp_raw{i},P_sgp_raw{i},rm_sgp{i},
            rangeBin);
        T_sgp(:,i) = T_sgp_int{i}(1:length(rm));
```

```
        P_sgp(:,i) = P_sgp_int{i}(1:length(rm));

        %interp sonde time
        [rm_sgp{i},IA,IC] = unique(rm_sgp{i});

        sonde_time(1:length(rm),i) = interp1(rm_sgp{i},t_single_sgp{i}(IA),rm)';
        %sonde_time = sonde_time';

                %Find index of sonde in time vector
         for j = 1:length((rm))
            [~, data_col_real(j,i)]=min(abs(sonde_time(j,i)-date_ts_N));
         end
    else
        T_sgp = [];
        P_sgp = [];
        datetime_sgp=datetime([],[],[]);
        sonde_time = [];
        data_col_real = [];

    end
end
T_real = T_sgp;
Patm_real = P_sgp;
datetime_real = datetime_sgp;
i_time_real = length(datetime_real);

% Find time lapsed since beginning datetime
time_lapsed = datetime_real - date_begin;
[hrs_real,mins_real,sec_real] = hms(time_lapsed);

%%
% Count dead time correction
nsPerBin = 250e-9;
summedBins = 14000/2;
%summedBins = 560;
% Deadtime for apd SPCM-780-12-FC
deadTime = 22e-9;
o2on_countRate = o2on_raw /(nsPerBin * summedBins);
o2on_correctionFactor = 1./(1 - deadTime*o2on_countRate);
o2on_raw_corrected = o2on_correctionFactor .* o2on_raw;
o2off_countRate = o2off_raw /(nsPerBin * summedBins);
o2off_correctionFactor = 1./(1 - deadTime*o2off_countRate);
o2off_raw_corrected = o2off_correctionFactor .* o2off_raw;

%%
% O2 Counts
% Interpolating counts to match new time vector
o2on_intp = interp2(ts_raw_o2_on,rm_raw,o2on_raw_corrected,ts,rm_raw_o2,'nearest␣␣nan');
o2off_intp = interp2(ts_raw_o2_off,rm_raw,o2off_raw_corrected,ts,rm_raw_o2,'nearest␣␣nan');
o2off_intp_mol = interp2(ts_raw_o2_off,rm_raw,double(
     O2_Offline_Molecular_Detector_Backscatter_Channel_Raw_Data),ts,rm_raw_o2,'nearest␣␣nan');
o2on_intp_mol = interp2(ts_raw_o2_off,rm_raw,double(
     O2_Online_Molecular_Detector_Backscatter_Channel_Raw_Data),ts,rm_raw_o2,'nearest␣␣nan');



% ======================
% === O2 Backscatter ===
% ======================
% --- O2 Background Subtraction ---
% Online
bg_o2on = mean(o2on_intp(end-150:end,:),1,'omitnan␣');% Take mean of last data points
o2on_bgsub = o2on_intp - bg_o2on;        % Background subtracted
o2on_bgsub(o2on_bgsub < 0) = 0;          % Minimum of zero
o2on_noise = o2on_bgsub(1:i_range,:);    % Shorten to length of range vector

% Offline
bg_o2off = mean(o2off_intp(end-150:end,:),1,'omitnan␣');% Take mean of last data points
o2off_bgsub = o2off_intp - bg_o2off;       % Background subtracted
o2off_bgsub(o2off_bgsub < 0) = 0;          % Minimum of zero
o2off_noise = o2off_bgsub(1:i_range,:);    % Shorten to length of range vector

% Online
bg_o2on_mol = mean(o2on_intp_mol(end-150:end,:),1,'omitnan␣');% Take mean of last data points
o2on_bgsub_mol = o2on_intp_mol - bg_o2on_mol;        % Background subtracted
o2on_bgsub_mol(o2on_bgsub_mol < 0) = 0;          % Minimum of zero
o2on_noise_mol = o2on_bgsub_mol(1:i_range,:);    % Shorten to length of range vector

% Offline
bg_o2off_mol = mean(o2off_intp_mol(end-150:end,:),1,'omitnan␣');% Take mean of last data points
o2off_bgsub_mol = o2off_intp_mol - bg_o2off_mol;        % Background subtracted
o2off_bgsub_mol(o2off_bgsub_mol < 0) = 0;          % Minimum of zero
o2off_noise_mol = o2off_bgsub_mol(1:i_range,:);    % Shorten to length of range vector
%%

% --- O2 Filtering ---
% Moving average in time and range
% Replicate edges to prevent problems caused by zero padding
```

```
% Even kernel causes data to shift by half step -- interpolate back to original time & range vectors
k = ones(oversample,t_avg)./(oversample*t_avg);      % Kernel

% Online
o2on = filter2(k,o2on_noise,'same');
o2on = fillmissing(o2on,'nearest',1); % Fill in NaNs in dimension 1
o2on = fillmissing(o2on,'nearest',2); % Fill in NaNs in dimension 2

% Offline
o2off = filter2(k,o2off_noise,'same');
o2off = fillmissing(o2off,'nearest',1); % Fill in NaNs in dimension 1
o2off = fillmissing(o2off,'nearest',2); % Fill in NaNs in dimension 2

o2on_mol = filter2(k,o2on_noise_mol,'same');
o2on_mol = fillmissing(o2on_mol,'nearest',1); % Fill in NaNs in dimension 1
o2on_mol = fillmissing(o2on_mol,'nearest',2); % Fill in NaNs in dimension 2

% Offline
o2off_mol = filter2(k,o2off_noise_mol,'same');
o2off_mol = fillmissing(o2off_mol,'nearest',1); % Fill in NaNs in dimension 1
o2off_mol = fillmissing(o2off_mol,'nearest',2); % Fill in NaNs in dimension 2
%%

%=====================
%= Backscatter ratio =
%=====================
%make backscatter ratio the same dimensions as data
BSR = interp2(double(time_Backscatter_Ratio),double(range_Backscatter_Ratio),double(Backscatter_Ratio),ts,
    rm,'nearest',nan); % interpolate backscatter ratio to range and time to match other data

BSR = fillmissing(BSR,'nearest',1);
BSR = fillmissing(BSR,'nearest',2);

%==============
%= Water Vapor =
%==============

% Profile
Absolute_Humidity = Absolute_Humidity.*(1-double(Absolute_Humidity_mask));
Absolute_Humidity(Absolute_Humidity<0) = 0;

%make water vapor the same dimentions as data
abs_humid = interp2(double(time_Absolute_Humidity),double(range_Absolute_Humidity),double(
    Absolute_Humidity),ts,rm,'nearest',nan); % interpolate water vapor to range and time to match other
    data
abs_humid = fillmissing(abs_humid,'nearest',1);
abs_humid = fillmissing(abs_humid,'nearest',2);


WV_intp = abs_humid / 1000 / mWV;          %[molecules/m^3] water vapor number density
WV = WV_intp;


%%

%=====================
%= Cloud and SNR mask =
%=====================
disp('Calculating masks')

cloud_p_point = 18;
SNR_threshold = 4;
 %Boulder data
SD_threshold = 3*10^8;
[SNRm , cloud_SDm_above, cloud_SDm,o2on_SNR] = mask_O2(o2on,o2off,rm,ts,cloud_p_point,SNR_threshold,
    SD_threshold,oversample,t_avg);



%%
disp('Calculating model')

% Calculating temperature and pressure model

Ts = interp1(time_Temperature,Surface_Temperature_HSRL,ts,'nearest',nan); %[K] Surface temperature
Ps = interp1(time_Pressure,Surface_Pressure_HSRL,ts,'nearest',nan); %[atm] Absolute surface pressure

lapseRate = -6.5;                                   %[K/km] Guess adiabatic lapse rate  typically -6.5 up
    to 10km
lapseRate = lapseRate / 1000;                       %[K/m]

%T = Ts + lapseRate .* rm;                           %[K] (1 x r) Temperature model as a function of r

T = interp2(double(time_Temperature),double(range_Temperature),double(Temperature),ts,rm,'nearest'); %
    Interpolate Temperature model to range and time to match other data
T = fillmissing(T,'nearest',1);
```

```
T = fillmissing(T,'nearest␣',2);

P = interp2(double(time_Pressure),double(range_Pressure),double(Pressure),ts,rm,'nearest␣'); % Interpolate
    pressure model to range and time to match other data
P = fillmissing(P,'nearest␣',1);
P = fillmissing(P,'nearest␣',2);


O2Online_Wavelength = interp1(time_Pressure,O2Online_Wavelength,ts);
O2Offline_Wavelength = interp1(time_Pressure,O2Offline_Wavelength,ts);

lambda_online = 769.7958;          %[nm] Updated online wavelength
lambda_offline = 770.1085;         %[nm] Updated offline wavelength
nu_online = 10^7./lambda_online;                    %[cm-1] Online wavenumber
nu_offline = 10^7./lambda_offline;                  %[cm-1] Offline wavenumber

nu01 = nu_online(1);                                   %[cm-1] Set center of scan to
nuMin = nu01-0.334;                                 %[cm-1] Scan lower bound
nuMax = nu01+0.334;                                 %[cm-1] Scan upper bound
nuBin = 0.00222;                                    %[cm-1] Scan increment
nu_scan = (nuMin:nuBin:nuMax);                      %[cm-1](1 x nu) Scan vector
i_scan = length(nu_scan);                           %[none] length of scan vector

lambda_scan = 10^7./nu_scan;                        %[nm](1 x lambda) Scan vector
f_scan = nu_scan * c * 100;                         %[Hz]( x f) Scan vector

nu_scan_3D_short = permute(nu_scan, [3 1 2]);       %[cm-1] putting scan in third dimension
lambda_scan_3D_short = 10^7./nu_scan_3D_short;
i_scan_3D_short = length(nu_scan_3D_short);         %[none] length of scan vector

del_nu = nu_scan_3D_short-nu_online;                %[1/cm] difference from center
del_f = del_nu*100*c*10^-9;                         %[GHz] difference from center


[~,online_index] = min(abs(nu_online - nu_scan_3D_short),[],3);%finding index of online wavenumber
[~,offline_index] = min(abs(nu_offline - nu_scan_3D_short),[],3);%finding index of online wavenumber


absorption(:,:,:) = absorption_O2_770_model(T,P,nu_online(1,1),WV); %[m-1] Function to calculate
    theoretical absorption
absorption_off(:,:,:) = absorption_O2_770_model(T,P,nu_offline(1,1),WV); %[m-1] Function to calculate
    theoretical absorption

%Calculating absorption due to radiosonde measurements
if ~isempty(sonde_time)
    for i=1:numel(sonde_time(1,:))
            if isdatetime(sonde_time(1,i))
                %absorption_sonde{i} = diag(absorption_O2_770_model(T_real(:,i),Patm_real(:,i),nu_online(
                    data_col_real(:,i)),WV(:,data_col_real(:,i)))); %[m-1] Function to calculate
                    theoretical absorption
                absorption_sonde{i} = diag(absorption_O2_770_model(T_real(:,i),Patm_real(:,i),nu_online(1)
                    ,WV(:,data_col_real(:,i)))); %[m-1] Function to calculate theoretical absorption
            else
                absorption_sonde{i} = nan(i_range,1);
                T_real = nan(i_range,1);
                Patm_real = nan(i_range,1);
            end
    end
else
      absorption_sonde{1} = nan(i_range,1);
      T_real = nan(i_range,1);
      Patm_real = nan(i_range,1);
end

%%

%==========================
%= Pertabative absorption =
%==========================

disp('Calculating␣absorption␣')


% === Zeroth Order ===
ind_r_lo = 1:i_range-oversample;                                        % High range vector
ind_r_hi = 1+oversample:i_range;                                        % Low range vector
ln_o2 = log((o2on(ind_r_lo,:).*o2off(ind_r_hi,:))./(o2on(ind_r_hi,:).*o2off(ind_r_lo,:))); % Natural log
    of counts

% Zeroth order term
alpha_0_raw = ln_o2./2./(rangeBin*oversample);                          %[1/m]


    disp('SG␣derivative␣')
    int_der = -log(o2on)+log(o2off);
    tic
    [~,g] = sgolay(2,oversample);
```

```
    parfor j=1:i_time
        alpha_0(:,j) = conv(int_der(:,j), factorial(1)/(-rangeBin)^1 * g(:,2), 'same␣')/2;
    end
    toc


    alpha_0(alpha_0==Inf | alpha_0==-Inf)=0;
    alpha_0 = fillmissing(alpha_0,'nearest␣');


%first order
fsr_O2 = 157.9;%[GHz] etalon free spectral range
finesse_O2 = 15.43;%etalon finesse

% ======================================
% === Normalized Etalon Transmission ===
% ======================================
% --- O2 channel ---
fsr_O2_nm = lambda_online.^2 * fsr_O2 / c;                          %[nm] Free spectral range converted to
    nm
delta_phase = 2 * pi * lambda_online.^2 ./ fsr_O2_nm ./ lambda_scan_3D_short; %[radians] Phase difference
    between orders
delta_phase = delta_phase - 2 * pi * lambda_online ./ fsr_O2_nm;     %[radians] Shifting phase so that
    zero is at the online wavelength
F_O2 = 1 ./ (sin(pi/2/finesse_O2)^2);                               %[none] Coefficient of finesse
T_etalon = 1 ./ (1 + F_O2 * sin(delta_phase/2).^2);                 %[none] Etalon transmission as a
    function of wavelength
%%
% Calculating purtabative absorption
altitude = .314;
[alpha_1, alpha_2] = pertAbsorption(alpha_0, T_etalon, T, P, rm,ts,rkm,m_air, nu_online, nu_scan_3D_short,
    nuBin,BSR,ind_r_lo,ind_r_hi,WV,online_index,i_range,i_time,i_scan_3D_short,rangeBin,oversample,t_avg,c
    ,kb,altitude);

% === Total alpha ===
alpha_total_raw = alpha_0 + alpha_1 + alpha_2;

% Force total alpha to its modeled surface value
[~,cut] = min(abs(rm-500));              % Index where rm is closest to chosen value
alpha_total = [absorption(1,:); NaN((cut - 1),i_time); alpha_total_raw(cut:end,:)];
alpha_total = fillmissing(alpha_total,'linear␣');
alpha_total = alpha_total(2:end,:);      % Remove surface value since rm starts at del_r

%apply SNR mask
alpha_0m = alpha_0 .* SNRm;
alpha_0m(alpha_0m == 0) = NaN;                    % Replace mask with NaNs

alpha_totalm = alpha_total .* SNRm .* cloud_SDm_above;
alpha_totalm(alpha_totalm <= 0) = NaN;            % Replace mask with NaNs

for i = 1:i_time
    alpha_totalm(:,i) = smooth(alpha_totalm(:,i), 4*oversample+1);
end

alpha_totalm = alpha_totalm .* SNRm .* cloud_SDm_above;
alpha_totalm(alpha_totalm <= 0) = NaN;            % Replace mask with NaNs

alpha_0m = alpha_0m .* SNRm .* cloud_SDm_above;
alpha_0m(alpha_0m <= 0) = NaN;            % Replace mask with NaNs

%%
disp('Temp␣ retrieval␣ function␣')
tic
[T_final_test,L_fit_sm_test,Ts_fit,Patm_final,mean_lapse_rate,exclusion] =  temperatureRetrieval(T,ts,rm,P
    ,WV,nu_online,alpha_totalm,SNRm,cloud_SDm_above);
toc
T_finalm = T_final_test .* SNRm .* cloud_SDm_above .* mask_nodata;
T_finalm(T_finalm <= 0) = NaN;


% Calculating pertabative absorption of O2
% author: Owen Cruikshank
% date: 11/30/2020

function [alpha_1, alpha_2] = pertAbsorption(alpha_0, T_etalon, T, P, rm,ts,rkm,m_air, nu_online,
    nu_scan_3D_short,nuBin,BSR,ind_r_lo,ind_r_hi,WV,online_index,i_range,i_time,i_scan_3D_short,rangeBin,
    oversample,t_avg,c,kb,altitude)

%%
%Calculate RB spectrum by PCA
[doppler_O2_ret] = RB_O2_770_PCA(T,P,nu_scan_3D_short);
    %%

    % --- Backscatter Lineshape g ---
    g1_m = 1./BSR .* doppler_O2_ret ;%.*nuBin*100;                      %[m] Molecular backscatter
        lineshape
    g1_a = zeros(i_range,i_time,i_scan_3D_short);                     % Initalize aerosol lineshape
    for i = 1:i_time
```

```
        g1_a(:,i,online_index(1)) = (1 - 1./BSR(:,i))/ nuBin / 100 ; %[m] aerosol backscatter lineshape
    end
    g1 = g1_a + g1_m;                                                    %[m] Combined backscatter
        lineshape
    g1_check = trapz(g1,3).*nuBin*100;                                  %[none] Check if integral of g1 is
        normalized to 1

    disp('SG derivative')
    [~,g] = sgolay(2,oversample);
    parfor j=1:i_time
        for k=1:i_scan_3D_short
            dg1_dr(:,j,k) = conv(g1(:,j,k), factorial(1)/(-rangeBin)^1 * g(:,2), 'same');
        end
    end


    %%
    disp('PCA absorption')
    [absorption_f,cross_section] = absorption_O2_770_PCA(T,P,nu_scan_3D_short(1,1,:),WV);

    %%
    %Create lineshape function
    f = nan(size(absorption_f));
    for i = 1:i_time
        f(:,i,:) = absorption_f(:,i,:) ./ absorption_f(:,i,online_index(1));
    end

     %%
    % --- Zeroth Order Transmission ---
    Tm0 = exp(-cumtrapz(rm,alpha_0.*f,1));        %[none] Zeroth order transmission


    % Integrand terms
    % Online
    zeta = g1.*T_etalon;                          %[m]
    eta = dg1_dr.*T_etalon;                       %[none]

    % Integrated terms
    % Online
    zeta_int = trapz(zeta.*Tm0,3)*nuBin*100;                  %[none]
    eta_int = trapz(eta.*Tm0,3)*nuBin*100;                    %[1/m]
    zeta_ls_int = trapz(zeta.*Tm0.*(1-f),3)*nuBin*100;        %[none]
    % Offline
    zeta2_int = trapz(zeta,3)*nuBin*100;                      %[none]
    eta2_int = trapz(eta,3)*nuBin*100;                        %[1/m]

    % === First Order ===
    W1 = zeta_ls_int./zeta_int;                   %[none]
    G1 = eta_int./zeta_int - eta2_int./zeta2_int;%[1/m]

    alpha_1_raw = 0.5.*(alpha_0.*W1 + G1);        %[1/m]

    % Moving average

   k = ones(oversample,t_avg)./(oversample*t_avg);
   t_step = ts(2)-ts(1);
   rangeBin = rm(2)-rm(1);
   alpha_1 = alpha_1_raw;


    % --- First Order Transmission Tm1 ---
    Tm1 = exp(-cumtrapz(rm,oversample.*alpha_1.*f,1));        %[none] First order transmission

    % === Second Order ===
    % Integrated terms
    zeta_Tm1_int = trapz(zeta.*Tm0.*(1-Tm1),3)*nuBin*100;                  %[none]
    eta_Tm1_int = trapz(eta.*Tm0.*(1-Tm1),3)*nuBin*100;                    %[1/m]
    zeta_ls_Tm1_int = trapz(zeta.*Tm0.*(1-Tm1).*(1-f),3)*nuBin*100;        %[none]

    W2 = (zeta_ls_int.*zeta_ls_Tm1_int./(zeta_int.^2)) - (zeta_ls_Tm1_int./zeta_int);    %[none]
    G2 = (eta_int.*zeta_Tm1_int./(zeta_int.^2)) - (eta_Tm1_int./zeta_int);               %[1/m]

    alpha_2_raw = 0.5.*(alpha_1.*W1 + alpha_0.*W2 + G2);      %[1/m]
   alpha_2=alpha_2_raw;


end

function [T_final,Lapse,Ts_fit,P_final,mean_lapse_rate,exclusion] = temperatureRetrieval(T,ts,rm,P,WV,
    nu_scan,alpha_O2,SNRm,cloud_SDm_above)
%File: temperatureRetrieval.m
%Date: 03/16/2020
%Author: Owen Cruikshank
%Inputs:
%   -T:[K] scalar or (range x time) vector of atmopheric temperature as a function
%    of range
%   -P:[atm] scalar or (range x time) vector of atmopheric pressure as a function
%    of range
```

```
%   -ts:[s] (1 x time) vector of time dimension
%   -rm:[m] (range x 1) vector of range dimension
%   -WV:[molecules/m^3] scalar or (range x time) vector of water vapor
%   number density
%   -nu_scan:[1/cm] wavenumber of online laser. Dimentions: (1 x 1 x wavenumber)
%   -alpha_O2:[1/m] (range x time) calculated absorption from data
%   -SNRm:[none] (range x time) calculated SNR mask
%   -cloud_SDm_above:[none] (range x time) calculated cloud mask
%
%Outputs:
%   -T_final: [K] (range x time x iteration) Final calculated temperature profile
%   -Lapse: [K/m] (range x time x iteration) Fitted lapse rate
%   -Ts_fit: [K] (range x time x iteration) Fitted surface temperature
%   -P_final: [atm] (range x time x iteration) Final calculated pressure
%   -mean_lapse_rate: [atm] (range x time x iteration) Mean lapse rate for
%   whole day
%   -exclusion: [none] (range x time x iteration) Points excluded from lapse rate fit


%==Constants
g0 = 9.80665;                   %[m/s/s]Gravitational acceleration
M_air = 0.0289644;              %[kg/mol] Molar mass of air
q_O2 = .2095;                   %[unitless] O2 atmospheric mixing ratio
kB = 1.38065E-23;               %[J/K] Boltzman's constant
%No = 2.47937E25;                % Loschmidt's number [1/m^3] (referenced to 296 K and 1 atm)
N_A = 6.02214076e23;            %[1/mol] Avagadro's number
R = kB * N_A;                   %[J/K/mol] universal gas constant
h = 6.626E-34;                  %[J s] Planck's constant
c = 2.99792458E8;               %[m/s] speed of light

%==O2 line parameters
S0_O2 = 4.889e-26;              %[cm-1/molec cm-2] line strength
S0_O2 = S0_O2 / 100;            %[m molecule-1] absoption line strength at T=296K
E_lower = 1420.7631;            %[cm-1] ground state energy
E_lower = E_lower * 100;        %[m-1] ground state energy
ep = E_lower*h*c;               %[J]

T0 = 296;                       %[K] reference temperature

L_t_window = 30;                %[min] time moving average
L_r_window = 8;                 %[x37.5m] range moving average
L_k = ones(L_r_window,L_t_window)./(L_r_window*L_t_window); %lapse rate kernel


loop = 20;%number of times to do iterative temperature retrieval loop

del_r = rm(2)-rm(1);            %[m]set range difference
Ts = T(1,:);                    %[K] surface temp
Ps = P(1,:);                    %[atm] surface press

gamma = g0 * M_air / R;         %[K/m]gravity molar mass of air and gas constant

%Set inital temperature guess
Tg = T;                         %[K]

%Preallocate memory
deltaT = zeros(length(rm),length(ts),loop);
T_ret = zeros(length(rm),length(ts),loop);
exponent = zeros(length(rm),length(ts),loop);
P_ii = zeros(length(rm),length(ts));
Lapse = zeros(length(rm),length(ts),loop);
Ts_fit = zeros(length(rm),length(ts),loop);
exclusion = zeros(length(rm),length(ts),loop);

logicalExc = true(length(rm),length(ts));

starting_lapse_rate = -0.0065;%[K/m] typical lapse rate

mean_lapse_rate = ones(1,1,1);
mean_lapse_rate(:,:,:) = starting_lapse_rate;%[K/m] variable to set to for unfitted points

%Fit lapse rate bounds
lower_alt_threshold = 1000;  %[m] only preform a fit above
upper_lapse_bound = -0.004;  %[K/m]
lower_lapse_bound = -0.010;  %[K/m]

upper_Ts_bound = Ts + 20;%[K]
lower_Ts_bound = Ts - 20;%[K]

%Set fittype, reduces computation time by a lot outside loop
polyf = fittype('poly1');

fprintf('Loop ')
for i = 1:loop
    fprintf(' %d ', i)
    %calculate lapse rate
    for time = 1:length(ts)
        % Set fit exclusion zones
```

```
    exclusion(:,time,i) = rm<lower_alt_threshold | SNRm(:,time)==0 | cloud_SDm_above(:,time) ~= 1;
    % Do not fit if the temperature guess is all nans
    if isnan(Tg(:,time))
        % Set fits to mean lapse rates
        Lapse(:,time,i) = mean_lapse_rate(:,:,1);%set lapserate to default
        Ts_fit(:,time,i) = Ts(time);

    %make sure there are enough points to fit
    elseif sum(1-exclusion(:,time,i)) > 20

        %Custom least squares for speed
        %initalize logical and V
        if i==1
            logicalExc(:,time) = ~logical(exclusion(:,time,i));
        end
        V = [rm(logicalExc(:,time)) ones(length(rm(logicalExc(:,time))),1)];
        %calculate polynomial coefficients
        p = V\Tg(logicalExc(:,time),time);

        Lapse(:,time,i) = p(1);
        Ts_fit(:,time,i) = p(2);

        %
        limits = [-10 -3]*10^-3;

        % Keep lapse rate within limits
        if Lapse(:,time,i) <lower_lapse_bound
            Lapse(:,time,i) = lower_lapse_bound;
        elseif Lapse(:,time,i) >upper_lapse_bound
            Lapse(:,time,i) = upper_lapse_bound;
        end

    %if not enough points, set the first datapoints to the mean lapse rate
    elseif time <= L_t_window/2
        disp('time␣ <=L_t_window')
        Lapse(:,time,i) = mean_lapse_rate(:,:,1);%set lapserate to default
        Ts_fit(:,time,i) = Ts(time);
    %if not enough points set lapse to mean of lastfew means
    else
        Lapse(:,time,i) = mean(Lapse(:,time-L_t_window/2:time-1,i),2);
        Ts_fit(:,time,i) = Ts(time);

     end
end

%calclate total mean
mean_lapse_rate(:,:,1) = mean(Lapse(1,:,i),2);

% === Pressure Profile ===
P_i = Ps;
T_i = Tg;
L_i = Lapse(:,:,i);

for j = 1:length(rm)
    P_ii(j,:) = P_i.*(T_i(j,:)./(T_i(j,:) + L_i(j,:).*del_r)).^(gamma./L_i(j,:));
    P_i = P_ii(j,:);
end

Pg = [Ps; P_ii(1:end-1,:)];%concatenate arrays to set ground pressure to measured value

%calculate air number density
n_L = Pg .* 101325 / kB ./ Tg;          %[1/m^3] Loschmidt's number; with pressure converted to Pa
q_WV = WV ./ n_L;                       %[unitless] mixing ratio of water vapor based from water vapor
      number density
q = q_O2 .* (1 - q_WV);

%update lineshape function
[~,~,g] = absorption_O2_770_model(Tg,Pg,nu_scan,WV);%[m] lineshape function

%Calculate Coefficients
exponent(:,:,i) = gamma ./ Lapse(:,:,i);
C1 = S0_O2 * T0 * (Pg*101325) * exp(ep/kB/T0) ./ (kB * Tg .^(-exponent(:,:,i)));%[K/(molec*m^2)]
     pressure converted to Pa
C2 = Tg .^ (-exponent(:,:,i) - 2) .* exp(-ep/kB./Tg);%[K^-1]
C3 = (-exponent(:,:,i) - 2) ./ Tg + ep./(kB.*Tg.^2);%[K^-1]
%Calculate change in temperature from last
deltaT(:,:,i) = (alpha_O2 - C1 .* C2 .* g .* q) ./ (C1 .* C2 .* C3 .* g .* q); %[K] calculate a change
      in temperatre

% Limit deltaT to plus or minus 2 K
deltaT(deltaT > 2) = 2;
deltaT(deltaT < -2) = -2;

% Update temperature profile guress
T_ret(:,:,i) = Tg + deltaT(:,:,i);
Tg = fillmissing(T_ret(:,:,i),'nearest␣␣1);
Tg = fillmissing(Tg,'nearest␣␣2);
```

```
    %Fill data points above cloud and snr mask with lapse rate values
    tempT = Ts_fit(:,:,i) + rm.* Lapse(:,:,i);
    Tg(SNRm(:,:)==0 | cloud_SDm_above(:,:) ~= 1) = tempT(SNRm(:,:)==0 | cloud_SDm_above(:,:) ~= 1);

end

fprintf('\n')
% Final temperature and pressure
T_final = Tg;
P_final = Ps .* (Ts./(Ts+Lapse(:,:,end).*(rm-60))).^(0.0341623./Lapse(:,:,end));

exclusion = exclusion(:,:,end);
end

function absorption = absorption_O2_770_model_wavenumber(T,P,nu_Range,WV)
%File: absorption_O2_770_model_wavenumber.m
%Date: 02/28/2020
%Author: Owen Cruikshank
%Inputs:
%   -T:[K] scalar or (range x time) vector of atmopheric temperature as a function
%    of range
%   -P:[atm] scalar or (range x time) vector of atmopheric pressure as a function
%    of range
%   -nu_Range:[1/cm] wavenumber scan. Dimentions: (1 x 1 x wavenumber)
%   -WV:[1/m^3] water vapor number density. Dimentions: (range x time)
%
%Outputs:
%   -absorption: [1/m] the atmopheric absorption of O2 at the input
%    wavenumber arround the 780nm line, dimensions (range x time x wavenumber)
%   -sigma: [m^2] The voight absorption cross section of O2 at the input
%    wavenumber arround the 780nm line, dimensions (range x time x wavenumber)
%   -f: [m] Absorption lineshape function, dimensions (range x time x wavnumber)

c = 2.99792458E8;                       %[m/s] speed of light
kB = 1.38065E-23;                       %[J/K] Boltzman's constant
h = 6.62607004E-34;                     %[Js] Planck's constant

q_O2 = .2095;                           %[unitless] O2 atmospheric mixing ratio

mo2 = 5.313525282632483E-26;            % Mass O2 molecule [kg]

N_o2 = ((P*101325)./(kB*T)-WV) * q_O2;      %[molecule/m^3](t x r)O2 number density from atmopsheric number
      density and O2 mixing ratio

%reference T and P
T0 = 296;                               %[K]
P0 = 1.0;                               %[atm]

parameters = fopen('02_line_parameters.out␣␣,'r');   %open file containing HITRAN information
fmt = ['%1d␣%1d␣%f␣%e␣%e␣%f␣%f␣%f␣%f␣%f␣];          %format of HITRAN file
O2_parameters =fscanf(parameters,fmt,[10 inf]);      %place HITRAN parameters in vector a
fclose(parameters);                                  %close file
O2_parameters = O2_parameters';                      %transpose matrix to correct format

[rL, tL] = size(T);                                  %length of range vector x length of time vector
absorption = zeros(rL,tL,length(nu_Range));%preallocate
integralV = zeros(rL,tL,length(nu_Range));%preallocate

nu_Range = nu_Range * 100;                           %change nu_Range from [1/cm] to [1/m]

strength_threshold = 1*10^(-27);                     %[cm / molecule] line strength threshold

t = -10:.2:10;                                       %Relative freqency to integrate over
t_4D = permute(t,[4 3 1 2]);                         %shift t to put it in third dimestion

for i = 1:length(O2_parameters)                      %loop over all line parameters

    nu_O2 = O2_parameters(i,3);                      %[1/cm]
    nu_O2 = nu_O2 * 100;                             %[1/m] absoption wavenumber
    S0_O2 = O2_parameters(i,4);                      %[cm/molecule] line strength
    S0_O2 = S0_O2 / 100;                             %[m/molecule] absoption line strength at T=296K
    gamma_L = O2_parameters(i,6);                    %[1/cm] linewidth
    gamma_L = gamma_L * 100;                         %[1/m] Air (Lorentz) broadened linewidth
    n_air = O2_parameters(i,9);                      %[unitless] linewidth temperature dependence
    E_lower = O2_parameters(i,8);                    %[1/cm] ground state energy
    E_lower = E_lower * 100;                         %[1/m] ground state energy
    gamma_D = O2_parameters(i,7);                    %[1/cm]self (Doppler) broadened linewidth
    gamma_D = gamma_D * 100;                         %[1/m]
    delta_air = O2_parameters(i,10);                 %[1/cm/atm] pressure shift induce by air, at p=1atm
    delta_air = delta_air * 100;                     %[1/m/atm]

    if nu_O2 >= nu_Range(:,:,1) && nu_O2 <= nu_Range(:,:,end) && S0_O2 * 100 > strength_threshold %Do not
        compute if line is out of specified wavnumber range

        nuShifted = nu_O2 + delta_air .* P;          %[1/m] (r x t) Shift line center based on atmopheric
            pressure

        %temperature shifted line strength
```

```
        a_o2 = S0_O2.*(T0./T);
        c_o2 = exp(h.*c./kB.*((1./T0)-(1./T)).*E_lower);
        ST_O2 = a_o2.*c_o2;                                  %[m/molecule](t x r) O2 line strength
            adjusted for temperature shift from T0
        gamma_L_T = gamma_L * (P/P0).*((T0./T).^n_air);      %[1/m](t x r) Lorentz lineshape adjusted
            for temperature and pressure shift
        gamma_D_T = (nuShifted/c).*sqrt(2*kB*T*log(2)/mo2); %[1/m](t x r) Dopper lineshape due to
            temperature

        %voight lineshape
        x = ((nu_Range-nuShifted)./gamma_D_T) * sqrt(log(2));   %[none](t x r x nu)
        y = (gamma_L_T./gamma_D_T) * sqrt(log(2));              %[none](t x r)
        K = (ST_O2./gamma_D_T) * sqrt(log(2)/pi);               %[m^2 / molecule (t x r)

        parfor time_i = 1:tL                                        % Loop over time vector
            integralV(:,time_i,:) = trapz(t,exp(-t_4D.^2)./(y(:,time_i).^2 + (x(:,time_i,:)-t_4D).^2),4);
                                                %[none] integrate over t
        end

        absorption = absorption + (y/pi).*integralV .* K .* N_o2; %[1/m](t x r x nu)absorption coefficeint
            of oxygen in the atmosphere
    end
end
end

function [SNRm , cloud_SDm_above, cloud_SDm,o2on_SNR] = mask_O2(o2on,o2off,rm,ts,cloud_p_point,
    SNR_threshold,SD_threshold,oversample,t_avg)
%File: mask_O2.m
%Date: 03/16/2020
%Author: Owen Cruikshank
%Inputs:
%   -o2on:[photons] (range x time) vector of online return photon counts
%   -o2off:[photons] (range x time) vector of offline return photon counts
%   -rm:[m] (range x 1) vector of range dimension
%   -ts:[s] (1 x time) vector of time dimension
%   -cloud_p_point:[hr] point to plot cloud plots. If 0 will not plot.
%   -SNR_threshold: Signal to noise rato threshold for SNR mask
%   -SD_threshold: Standard deviation threshold for cloud mask
%   -oversample: Number of range bin averaging
%   -t_avg: Number of time bin averaging
%
%Outputs:
%   -SNRm:[none] (range x time) calculated SNR mask
%   -cloud_SDm_above:[none] (range x time) calculated cloud mask
%   -cloud_SDm:[none] (range x time) calculated cloud mask with only clouds
%   -o2on_SNR:[none] (range x time) signal to noise ratio for online counts


% ===================
% === SNR O2 mask ===
% ===================

o2on_SNR = sqrt(o2on);                      % Calculate shot noise SNR
o2on_SNRm = ones(size(o2on_SNR));           % Initialize matrix
o2on_SNRm(o2on_SNR < SNR_threshold & rm > 1000) = 0;% Replace low SNR points with zeros

o2off_SNR = sqrt(o2off);
o2off_SNRm = ones(size(o2off_SNR));
o2off_SNRm(o2off_SNR < SNR_threshold & rm > 1000) = 0;

SNRm = o2off_SNRm .* o2on_SNRm;             %Final combination SNR matrix mask with 0 in places of low snr


% ===================
% === Cloud O2 mask ===
% ===================

%Range corrected returns
o2on_range_corrected = o2on.*rm.^2;
o2off_range_corrected = o2off.*rm.^2;

%standard deviation
%stdNeiborhood = true(7,29);                                % NxN window arround each point
stdNeiborhood = true(oversample,t_avg);                                    % NxN window arround each point

o2on_SD = stdfilt(o2on_range_corrected,stdNeiborhood);  % Use a square standard deviation filter on the
    range corrected return to find a standard deviation
cloud_SDm_on = ones(size(o2on));                        % Initialize mask matrix
cloud_SDm_on(o2on_SD > SD_threshold) = 0;                   % Set points with a standard deviation
    greater than the threshold to 0 inicating a cloudy area

o2off_SD = stdfilt(o2off_range_corrected,stdNeiborhood);
cloud_SDm_off = ones(size(o2on));
cloud_SDm_off(o2off_SD > SD_threshold) = 0;

cloud_SDm = cloud_SDm_off .* cloud_SDm_on;              % Combine on and offline cloud masks
```

```
%Set values above the cloud to -1
cloud_SDm_above = cloud_SDm;
diff_cloud_SDm = diff(cloud_SDm);                                   % Calculate difference along range axis
diff_SNRm = diff(SNRm);

[cloud_index_row,cloud_index_col] = find(diff_cloud_SDm);    % Find non-zero values in difference
if ~isempty(cloud_index_row)                                  % Check if any clouds exist
    cloud_index = zeros(1,length(ts));                            % Initalize cloud range index matrix

    index_matrix = ones(length(rm),length(ts)) .* (1:length(rm))'; % Create matrix to match range indecies

    % Set range index value to index of cloud
    cloud_index(cloud_index_col(1))=cloud_index_row(1);
    for i = 2:length(cloud_index_col)
        if cloud_index_col(i)~=cloud_index_col(i-1)
            cloud_index(cloud_index_col(i))=cloud_index_row(i);
        end
    end
    cloud_SDm_above(index_matrix > cloud_index - oversample & cloud_index ~= 0)=-1; % Set all values above
         cloud index to -1
    cloud_SDm_above = cloud_SDm_above .* cloud_SDm;                     % Create final matrix where clouds
         are represended by zeros and all data above represented by -1
end

[SNR_index_row,SNR_index_col] = find(diff_SNRm);    % Find non-zero values in difference
if ~isempty(SNR_index_row)                                    % Check if any clouds exist
    SNR_index = zeros(1,length(ts));                              % Initalize cloud range index matrix
SNR_index = ones(1,length(ts))*length(rm);

    index_matrix = ones(length(rm),length(ts)) .* (1:length(rm))'; % Create matrix to match range indecies

    % Set range index value to index of cloud
    SNR_index(SNR_index_col(1))=SNR_index_row(1);
    for i = 2:length(SNR_index_col)
        if SNR_index_col(i)~=SNR_index_col(i-1)
            SNR_index(SNR_index_col(i))=SNR_index_row(i);
        end

    end
    SNRm(index_matrix > SNR_index )=0; % Set all values above cloud index to -1
                        % Create final matrix where clouds are represended by zeros and all data above
                             represented by -1
end

end

end
```