DIODE-LASER-BASED HIGH SPECTRAL RESOLUTION LIDAR

by

Luke Stewart Colberg

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Optics and Photonics

MONTANA STATE UNIVERSITY
Bozeman, Montana

December 2021

©COPYRIGHT

by

Luke Stewart Colberg

2021

All Rights Reserved

ACKNOWLEDGEMENTS

I want to thank my advisor, Dr. Kevin Repasky, for the opportunity to work on this project and for mentoring me as I pursue a career in research. I would also like to thank Owen Cruikshank for answering countless questions about the MPD instruments over the last two years and for help aligning and maintaining the instrument. Thank you to Dr. Robert Stillwell from the National Center for Atmospheric Research for teaching me to align the MPD, showing me how to build the potassium cell, and helping with data processing. Thanks to Dr. Joseph Shaw for letting me use the radiosonde station and for maintaining the ECE weather station. I would also like to thank the National Science Foundation who provided funding for this research through NSF Grant #1917851.

TABLE OF CONTENTS

1. INTRODUCTION 1

The Atmosphere 1
Atmospheric Layers 1
Aerosols 4
Overview of Lidar for Atmospheric Research 6
Motivation for a Ground-Based Thermodynamic Profiling Lidar 8
Thesis Organization 10

2. THEORY 11

Principles of Lidar 11
The Lidar Equation 13
Atmospheric Scattering 16
Overview 16
Molecular Scattering 18
Rayleigh Scattering 18
Brillouin Scattering 20
The Tenti-S6 Model 23
Aerosol Scattering 26
Other Scattering 27
Total Atmospheric Scattering 27
HSRL Theory 28
Problem Statement and Solution 28
Principles of HSRL 30
Measurement Theory 31
HSRL Optical Filter 34
DIAL Theory 35
Overview 35
The DIAL Equation 35

3. INSTRUMENT DESCRIPTION 38

Overview 38
Laser Transmitter 40
Receiving Optics 43
Control System 49
Additional Instrumentation 50
Eye Safety 51
Calibration 52

TABLE OF CONTENTS – CONTINUED

4. AEROSOL OPTICAL PROPERTY RETRIEVAL 58

Overview 58
Data Processing 58
Pre-Processing 58
Atmospheric Modeling 62
Aerosol Optical Property Retrieval 63
Masking 64
Cloud Mask 64
Signal-to-Noise Ratio Mask 65
Smoothing 65
Data Examples 65
April 1-April 3, 2021 66
June 6-June 8, 2021 69
July 17-July 19, 2021 72
Future Work 75

5. PBLH RETRIEVAL 77

Introduction 77
Methods 78
Haar Wavelet Transform 78
PBLH Retrieval Algorithm 82
Radiosonde Retrievals 88
Data and Radiosonde Comparisons 90
Future Work 100

6. CONCLUSION 101

REFERENCES CITED 103

APPENDICES 112

APPENDIX A : Acronyms 113
APPENDIX B : PBLH Comparisons 115
APPENDIX C : MATLAB Code 147
Aerosol Optical Property and PBLH Retrieval Code 148
Main Function 148
LoadNetCDF 150
NonLinearEffects 151
LoadWeatherFile 152

TABLE OF CONTENTS – CONTINUED

LoadSunriseSunsetData 153
LoadRadiosondes....154
BackgroundSubtraction....154
TrimData....154
OverlapCorrection 154
TimeGrid....155
TrimTimes....156
TemperatureAndPressure....156
BackscatterRetrieval....157
FindGradient 157
FindVariance 158
FindClouds 158
SNRMasking....158
ApplyMasks....159
DataRemoval 159
FindCloudHeight....160
NormalizeData 160
BackscatterSmoothing 160
HaarWaveletTransform 161
BoundaryLayerMainFunction 161
FindFreeAtmosphere....162
FreeAtmosphereDiscontinuities....162
FindCappingInversion....163
CappingInversionDiscontinuities 164
FindConvectionPoint 165
DetermineCloudType....165
FindResidualLayers 166
FindMixedLayers....166
MixedLayerDiscontinuities 168
BoundaryLayerSmoothing....171
RadiosondeProcessing....171
VAPSaveFile....172
Calibration Code 174
Main Function 174
ReadScanFiles....174
DetectorCorrection 178
RayleighBrillouinSpectrum....178
crbs6 179
w0_func 180
CalibrationConstantsTables 181

TABLE OF CONTENTS – CONTINUED

Radiosonde Code 183
Main Function 183
LoadSonde 183
tsvread 184
SondeCheck 184
RemoveSondeData 185
FindHeight 185
FindPotentialTemperature 186
FindAbsoluteHumidity 186
SmoothSondeData 186
FindSondeGradients 187
CreateBoundaryLayerPlots 187
HeffterMethodRetrieval 188
WriteSondeFile 188
APPENDIX D : Permissions for Copyrighted Material 189

LIST OF FIGURES

Figure Page

1.1 Planetary boundary layer regimes and diurnal cycle. (Stull, 1988) Copyright © 1988, Kluwer Academic Publishers. 2

1.2 Top left to top right: pollen, quartz particles, and iron-oxide particles. Bottom left to bottom right: asbestos particles, volcanic ash. (Not on the same scale) (Hinds, 1999) Copyright © 1999, John Wiley and Sons. 4

2.1 Schematic of the main components of a typical pulsed lidar system. 12

2.2 Lidar schematic with components that contribute to terms from lidar equation. 17

2.3 Doppler broadening in molecular scattering. 20

2.4 Cabannes, Stokes, and anti-Stokes scattering lines. 21

2.5 Rayleigh and Brillouin scattering. 22

2.6 Rayleigh and Rayleigh-Brillouin scattering. 23

2.7 Influence of temperature on the Rayleigh-Brillouin spectrum. 25

2.8 Influence of pressure on the Rayleigh-Brillouin spectrum. 26

2.9 Total atmospheric scattering frequency spectrum. 28

2.10 Top: Combined channel transmission and combined atmospheric scattering spectrum. Bottom: Molecular channel transmission and combined atmospheric scattering spectrum. 31

3.1 Schematic diagram of the combined HSRL and O2-DIAL instrument. 39

3.2 Left: An image of the prototype MPD instrument at MSU excluding the computer. Top right: Transmitting optics and the first stage receiving optics. Bottom right: Second stage receiving optics. (Photos courtesy of Owen Cruikshank) 40

3.3 Potassium cell electrical schematic diagram. 46

LIST OF FIGURES – CONTINUED

Figure Page

3.4 Top: Etalon transmission. Center: Potassium cell transmission. Bottom: Transmission through the etalon and potassium cell. 47

3.5 Top: Combined channel transmission. Bottom: Molecular channel transmission. 48

3.6 Combined and molecular receiver scan photon counts. 53

3.7 Top: Combined channel transmission and return signal spectra. Bottom: Molecular channel transmission and return signal spectra. 55

3.8 C_{MM} vs temperature and pressure. 57

3.9 C_{MC} vs temperature and pressure. 57

4.1 APD correction factor. 59

4.2 Data with the APD correction factor vs original data. 59

4.3 Overlap correction factor. 61

4.4 April 1-April 3, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio. 66

4.5 April 3, 2021, 7:05 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio. 67

4.6 June 6-June 8, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio. 69

4.7 June 7, 2021, 9:30 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio. 70

4.8 July 17-July 19, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio. 72

4.9 July 19th, 2021, 9:00 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio. 73

LIST OF FIGURES – CONTINUED

Figure Page

5.1 Haar wavelet, dilation a=1. 79

5.2 HWT for various dilation values. 80

5.3 Range-dependent dilation function. 81

5.4 Top: Cloud height retrieval. Center: Free atmosphere retrieval. Bottom: Capping inversion retrieval. 84

5.5 Top: Residual layer retrieval. Center: Convection point retrieval. Bottom: PBLH limits and retrieval. 87

5.6 Heffter retrieval demonstration. 89

5.7 March 13-March 19, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval. 91

5.8 March 28-April 3, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval. 92

5.9 May 11-May 12, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval. 93

5.10 June 1-June 4, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval. 94

5.11 PBLH values found by the HWT-lidar method and by the Heffter-radiosonde method. 96

5.12 Radiosonde comparison, March 13, 2021-20:30 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=861 m; PBLH-Lidar=674 m. 98

LIST OF FIGURES – CONTINUED

Figure Page

5.13 Radiosonde comparison, June 2, 2021-16:15 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2552 m; PBLH-Lidar=771 m. 99

B.1 PBLH values found by the HWT-lidar method and by the Heffter-radiosonde method. 116

B.2 Radiosonde comparison, March 13, 2021-20:30 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=861 m; PBLH-Lidar=674 m. 117

B.3 Radiosonde comparison, March 17, 2021-21:42 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=1830 m. 118

B.4 Radiosonde comparison, March 18, 2021-22:44 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3796 m; PBLH-Lidar=2585 m. 119

B.5 Radiosonde comparison, March 28, 2021-22:54 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3917 m; PBLH-Lidar=3746 m. 120

B.6 Radiosonde comparison, March 29, 2021-00:41 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3409 m. 121

LIST OF FIGURES – CONTINUED

Figure Page

B.7 Radiosonde comparison, March 31, 2021-15:35 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2828 m; PBLH-Lidar=599 m. 122

B.8 Radiosonde comparison, March 31, 2021-18:26 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1940 m; PBLH-Lidar=824 m. 123

B.9 Radiosonde comparison, March 31, 2021-20:42 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3294 m; PBLH-Lidar=2997 m. 124

B.10 Radiosonde comparison, April 1, 2021-1:38 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3534 m; PBLH-Lidar=2992 m. 125

B.11 Radiosonde comparison, April 1, 2021-16:43 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=733 m; PBLH-Lidar=412 m. 126

B.12 Radiosonde comparison, April 1, 2021-21:40 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=2285 m. 127

LIST OF FIGURES – CONTINUED

Figure Page

B.13Radiosonde comparison, April 2, 2021-1:28 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3222 m; PBLH-Lidar=1836 m. 128

B.14Radiosonde comparison, April 2, 2021-17:33 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=815 m; PBLH-Lidar=450 m. 129

B.15Radiosonde comparison, April 2, 2021-21:05 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3109 m. 130

B.16Radiosonde comparison, April 3, 2021-1:14 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3222 m. 131

B.17Radiosonde comparison, May 11, 2021-17:38 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=959 m. 132

B.18Radiosonde comparison, May 11, 2021-20:51 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3697 m; PBLH-Lidar=2870 m. 133

LIST OF FIGURES – CONTINUED

Figure Page

B.19Radiosonde comparison, May 12, 2021-1:07 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3779 m; PBLH-Lidar=3581 m. 134

B.20Radiosonde comparison, June 1, 2021-17:09 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2643 m; PBLH-Lidar=584 m. 135

B.21Radiosonde comparison, June 1, 2021-19:03 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2513 m; PBLH-Lidar=1896 m. 136

B.22Radiosonde comparison, June 1, 2021-20:43 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2769 m; PBLH-Lidar=2045 m. 137

B.23Radiosonde comparison, June 1, 2021-23:59 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3041 m; PBLH-Lidar=3282 m. 138

B.24Radiosonde comparison, June 2, 2021-16:15 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2552 m; PBLH-Lidar=771 m. 139

LIST OF FIGURES – CONTINUED

Figure Page

B.25Radiosonde comparison, June 2, 2021-18:49 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1662 m; PBLH-Lidar=1409 m. 140

B.26Radiosonde comparison, June 2, 2021-21:32 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1942 m; PBLH-Lidar=1708 m. 141

B.27Radiosonde comparison, June 3, 2021-17:45 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=809 m. 142

B.28Radiosonde comparison, June 3, 2021-20:46 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=1146 m. 143

B.29Radiosonde comparison, June 3, 2021-22:52 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=2345 m. 144

B.30Radiosonde comparison, June 4, 2021-16:29 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=886 m; PBLH-Lidar=659 m. 145

LIST OF FIGURES – CONTINUED

Figure Page

B.31 Radiosonde comparison, June 4, 2021-19:23 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=850 m; PBLH-Lidar=1146 m. 146

ABSTRACT

This thesis describes the design, construction, and testing of a high spectral resolution lidar (HSRL) as a part of a combined HSRL and differential absorption lidar (DIAL) system. The combined HSRL and DIAL instrument is constructed using the MicroPulse DIAL (MPD) architecture and uses distributed Bragg reflector lasers. The MPD architecture is unique because it is eye-safe and cost-effective; therefore, it is ideal for creating a network of ground-based lidars. This instrument is designed for thermodynamic profiling of the lower troposphere. A network of these instruments would be helpful for wide-scale atmospheric monitoring for weather forecasting and climate science. The purpose of the HSRL is to retrieve the optical properties of aerosols in the lower troposphere. The HSRL uses the DIAL offline laser, which has a wavelength of 770.1085 nm, and a potassium vapor cell as the spectral filter.

The data retrieved from the HSRL provides the aerosol backscatter coefficient and the backscatter ratio up to an altitude of 7 km during nighttime operation and 5 km during daytime operation. The time resolution for these measurements is 5 minutes, and the range resolution is 150 m. These aerosol optical properties are valuable for aerosol studies and climate modeling; aerosols introduce the most significant degree of uncertainty in modeling the heat flux of the atmosphere. Additionally, these aerosol optical properties can be used to find the planetary boundary layer height (PBLH). The planetary boundary layer controls the exchange of heat, water vapor, aerosols, and momentum between the surface and the atmosphere. It has been demonstrated that the PBLH strongly affects turbulent mixing, convective transport, and cloud entrainment, which makes the PBLH an important parameter for weather forecasting and climate modeling. Despite its significance in atmospheric science, there is no standard method for defining the PBLH. A retrieval method for finding the daytime PBLH using HSRL data is proposed, and data comparisons to radiosonde PBLH retrievals are provided. The algorithm shows a good agreement with the radiosonde retrievals for conditions with a well-behaved boundary layer.

1

INTRODUCTION

The Atmosphere

Atmospheric Layers

The earth's atmosphere is composed of several distinct layers that span from the surface to outer space. The troposphere is the lowest layer, extending from sea level to approximately 12 km. This layer is the densest part of the atmosphere and contains most of the atmospheric water vapor and aerosols. Additionally, most weather phenomena occur in this layer. The troposphere can further be separated into the planetary boundary layer (PBL) and the free troposphere. The PBL is the lowest part of the troposphere; this layer extends from the earth's surface to a few hundred meters at night and several kilometers during the day. The PBL is defined by Stull [1] as "that part of the troposphere that is directly influenced by the presence of the earth's surface, and responds to forcings within an hour or less." The earth's surface transfers heat, humidity, and aerosols to the PBL, which creates a buoyant layer that is more humid and has more aerosols than the free troposphere. The PBL varies with a diurnal cycle due to the diurnal variation of surface temperature. The boundary between the PBL and the free atmosphere is typically marked by a gradient in potential temperature, wind speeds, aerosol content, and water vapor content. The planetary boundary layer height (PBLH) is the height of this inversion layer.

The PBL can be divided into three regimes: the stable layer (SL), the mixed layer (ML), and the residual layer (RL). The SL forms in the evening from the radiative cooling of the earth's surface. Atmospheric greenhouse gases absorb some of the infrared (IR) radiation from the earth's surface. This absorption creates a temperature inversion where the SL is warmer than the surface. This temperature inversion is generally a few degrees in the

middle latitudes but can be tens of degrees in the arctic. Generally, this layer extends to a few hundred meters in depth. The ML begins forming in the morning when solar radiation heats the earth's surface. The warm surface transfers heat to the PBL primarily through convection. This convective mixing causes the uplifting of water vapor and aerosols as the layer grows. The ML grows as more heat is transferred from the surface and reaches a maximum height in the mid-afternoon. The ML has a wide range of depths, from approximately 150 m to 4 km. In the evening, a new SL forms beneath the ML of the previous day. However, aerosols, water vapor, and heat remain at heights up to the PBLH of the past day. Because this layer is no longer affected by surface forcings, it is no longer the PBL. This layer is known as an RL, and it does not form in all conditions. The capping inversion (CI) is the inversion that characterizes the transition between either the RL or the ML and the free troposphere. The PBLH is the height of the inversion that marks the top of either the SL or the ML. Figure 1.1 shows the diurnal cycle of the boundary layer regimes.

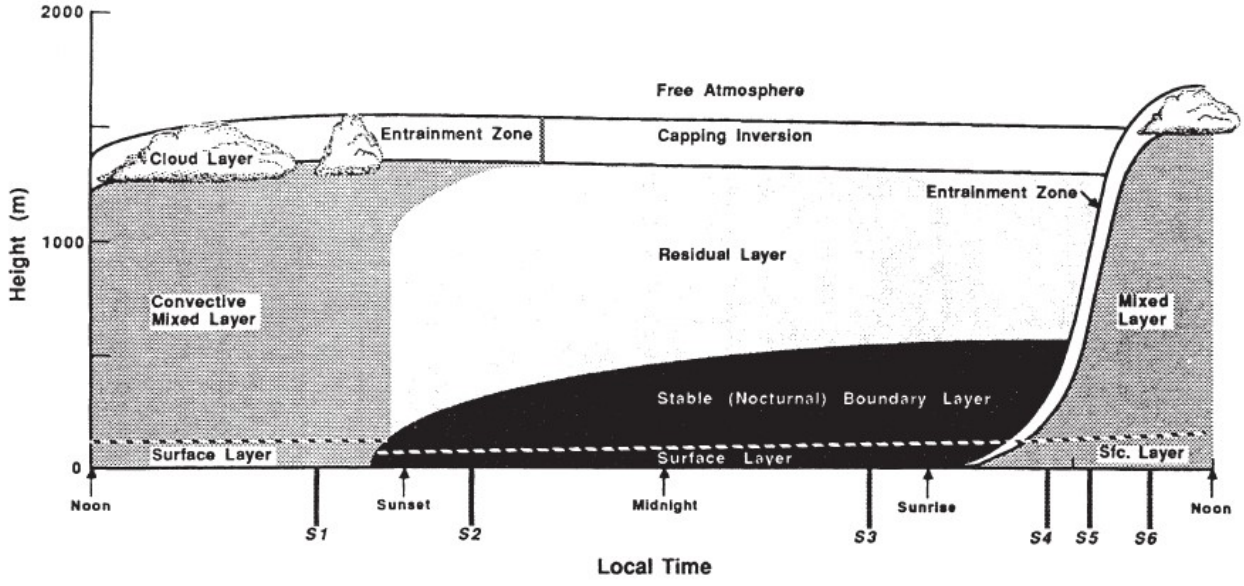


Figure 1.1: Planetary boundary layer regimes and diurnal cycle. (Stull, 1988) Copyright © 1988, Kluwer Academic Publishers.

Although it is a small fraction of the earth's atmosphere, the PBL is of utmost importance because, except for air travel, humans spend their entire lives within it. The atmospheric parameters in the PBL, such as temperature, humidity, aerosol content, and the PBLH, directly affect climate and are useful for weather forecasting. Additionally, understanding the thermodynamics of this lower layer is critical for understanding the heat flux of the earth and its effect on climate change.

Throughout the troposphere, the temperature decreases in a primarily linear fashion. The tropopause is a thin buffer zone at the top of the troposphere. The temperature in this zone remains constant as altitude increases. The stratosphere begins just above the tropopause and extends to about 50 km. This layer has very low humidity compared to the troposphere. The stratosphere also contains most of the atmosphere's ozone which absorbs harmful ultraviolet (UV) radiation. The temperature increases in the stratosphere with altitude because of this UV radiation absorption. The mesosphere, the thermosphere, and the exosphere are above the stratosphere. The mesosphere extends from 50 to 80 km and contains noctilucent clouds, the highest in the earth's atmosphere. The thermosphere extends from 80 to about 700 km. In this layer, charged particles from the sun collide with atmospheric molecules. Some of the energy released through these collisions is the visible aurora. The exosphere is the highest layer of the earth's atmosphere, from 700 to 10000 km. In this layer, the atmosphere merges with the solar wind. Molecules are very sparse in this region, and most satellites orbit the earth in this layer. Although all layers are essential, the troposphere and the PBL are particularly important because most weather occurs in the troposphere, and the majority of tropospheric weather occurs in the PBL. The instrument described in this thesis probes the aerosol optical properties of the PBL and the lower region of the free troposphere.

Aerosols

The atmosphere contains a mixture of gases, some well-mixed and some spatially varying. Suspended solid or liquid particles within that gaseous mixture are called aerosols [2]. Aerosol is a general term for any suspended particle in a gaseous medium. Common atmospheric aerosols include dust, sea salt, smoke, and clouds. Figure 1.2 shows a variety of aerosol particles. Particle sizes can range from a few nanometers to hundreds of micrometers.

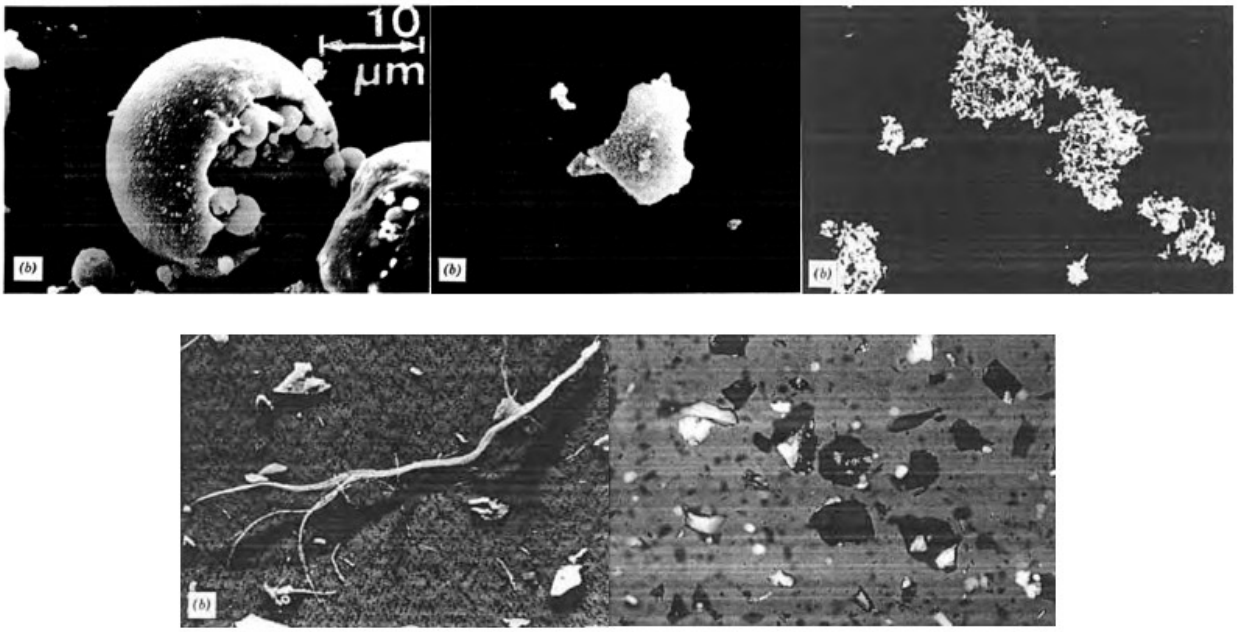


Figure 1.2: Top left to top right: pollen, quartz particles, and iron-oxide particles. Bottom left to bottom right: asbestos particles, volcanic ash. (Not on the same scale) (Hinds, 1999) Copyright © 1999, John Wiley and Sons.

Radiative forcing is the change in energy flux in the atmosphere. If the net radiative forcing is positive, the result is a warming Earth. Aerosols affect the radiative forcing of the earth's atmosphere both directly and indirectly. They introduce the most significant uncertainty in climate modeling [3]. The radiative forcing from anthropogenic aerosols has a net cooling effect, with a net radiative forcing of -0.9 (-1.9 to -0.1) $\frac{W}{m^2}$. The wide variety of aerosols and their optical properties cause this high degree of uncertainty. For example, sulfur

aerosols created by the interaction of sulfur gases from volcanic eruptions and atmospheric water and oxygen have a net cooling effect. These sulfur-based aerosol particles reduce the net radiative forcing of the atmosphere, especially when they are located in the stratosphere. In 1991, Mount Pinatubo erupted in the Philippines and spewed enormous quantities of sulfur gas into the stratosphere; this cooled the earth's atmosphere by approximately 0.5 K for the next 18 months [4]. On the other hand, black carbon aerosols absorb sunlight and IR radiation, creating a greenhouse effect that warms the earth [5]. These are examples of the direct aerosol effect on the atmosphere, where aerosol particles directly reflect or absorb radiation.

Clouds cause the indirect aerosol effect on the radiative forcing of the atmosphere. Clouds form because naturally occurring aerosols act as condensation nuclei for atmospheric water vapor. Similar to aerosols, clouds usually have a net negative effect on the radiative forcing of the earth. Clouds reflect sunlight and prevent it from reaching the earth's surface, resulting in cooling. This net radiative forcing is highly dependent on the type of cloud and the type of surface that the cloud is above. For example, low-altitude water clouds over a rainforest have a strong cooling effect because the cloud has a higher albedo than the underlying rainforest. However, an ice cloud over arctic sea ice will have a warming effect because it has a lower albedo than the underlying sea ice and absorbs IR radiation from the radiative cooling of the earth [6].

The Aerosol Robotic Network (AERONET) is the largest and most successful network for measuring aerosols and their properties. AERONET currently uses a network of ground-based solar radiometers (CIMEL Electronique 318A), which are passive remote sensing instruments that measure solar radiation at several wavelength bands. AERONET primarily monitors the aerosol optical depth (AOD); however, this AOD is a path-averaged measurement, so there is no vertical resolution associated with the measurements. Satellite-based remote sensing from the moderate resolution imaging spectroradiometer (MODIS)

located on the Terra and Aqua satellites is also used for aerosol studies. MODIS measures the wavelength and amplitude of light emitted or reflected by aerosols in the earth's atmosphere in addition to the AOD. These instruments scan the planet's surface with the satellite's movement, but once again, these are path-averaged measurements without vertical resolution. Active remote sensing instruments like lidar are becoming more prevalent because of their ability to produce vertically resolved data.

Overview of Lidar for Atmospheric Research

Lidar is an acronym for light detection and ranging and is a form of active remote sensing similar to radar. The difference is that the radiation is in the form of laser light rather than a radio wave. Almost since the invention of the laser in 1960, lidar has been used to study the atmosphere. Fiocco and Smullins recorded the first use of lasers to probe the atmosphere [7] when they used a ruby "laser radar" to detect "echoes" from the atmosphere in 1963. Ligda [8] used a ruby laser "lidar" to probe meteorological phenomena in the lower troposphere at about the same time. Since then, lidar has contributed significantly to our knowledge of the atmosphere. Lidar is particularly useful because of its ability to produce temporally and spatially varying data, so it can measure atmospheric parameters that are highly variable. Lidar has monitored trace gases, water vapor concentrations, temperature, clouds, winds, turbulence, and the PBLH. This section will describe some of the current lidar technologies to demonstrate the diversity of lidar in atmospheric research.

The first lidars used in atmospheric research were elastic backscatter lidars (EBLs), the simplest for atmospheric measurements. Generally speaking, an EBL is a lidar that emits a single wavelength and uses a single detector to measure the return signal. These EBLs detect the total backscattered signal from the atmosphere with vertical and temporal resolution. An EBL detects light from elastic scattering, which is light that molecules or aerosols scatter at the same wavelength of the outgoing laser beam [9]. EBLs have measured aerosol optical

properties [10, 11] and have monitored the PBLH [12, 13, 14].

The high spectral resolution lidar (HSRL) is a specific type of EBL in which the receiving optics split the return signal into two channels based on the return signal frequency. HSRLs measure aerosol optical properties with fewer assumptions than standard EBLs [15]. HSRLs have monitored aerosol optical properties [16, 17] and the PBLH [18, 19]. Chapter 2 will describe this technique in detail.

Dual-polarization lidar is a specific type of EBL that divides the return signal into two signals based on the polarization states of the received photons. Cloud phase can be determined using this information about polarization [20].

Differential absorption lidars (DIALs) are two-wavelength lidar systems that take advantage of an absorption band of a molecular species in the atmosphere. The molecular species of interest strongly absorbs the online laser wavelength, and it absorbs the offline laser wavelength to a much lesser extent. DIALs have measured trace gases in the atmosphere such as SO2, NO2, O3, hydrazine [21], and water vapor [22]. Recent improvements in laser technology have shown the potential for DIALs to be developed for atmospheric temperature profiling [23]. Chapter 2 will briefly describe the DIAL technique for atmospheric temperature retrievals.

Raman lidars take advantage of the inelastic scattering effect called Raman scattering. Raman scattering is the coupling of vibrational or rotational energy states in the molecules and the scattered photons. This scattering effect causes the wavelength of the scattered photons to be different from the wavelength of the transmitted laser. Raman lidars have monitored atmospheric water vapor concentrations using the vibrational Raman technique [24] and temperature using the rotational Raman technique [25].

Doppler lidar uses the Doppler shift caused by the movement of aerosol particles in the atmosphere to retrieve air velocity profiles. This technology has measured wind speed [26], turbulence, and the PBLH [27].

Fluorescence lidar is a technique used for monitoring minor species in the atmosphere. Fluorescence lidar excites chemical species with a resonantly tuned probe laser. The atom or molecule quickly decays to the lowest excitation energy level, which is semi-stable. Then, the semi-stable level decays to the original ground state, emitting a photon of a lower frequency than the laser frequency. Fluorescence lidar has monitored metallic ions in the thermosphere [28].

Motivation for a Ground-Based Thermodynamic Profiling Lidar

Ground-based water vapor and temperature profiling in the lower troposphere is essential for improving weather predictions and advancing climate science. However, measurements from traditional methods suffer from observational gaps. The standard process for measuring atmospheric temperature and humidity profiles uses a radiosonde attached to a weather balloon. This method only provides a single range-resolved profile when a weather balloon is launched and fails to capture temporally varying processes. Furthermore, radiosonde stations typically launch two radiosondes per day, retrieving a profile only once every 12 hours. Researchers need continuous data with a high temporal and vertical resolution to advance atmospheric science and improve weather forecasting models. The United States Department of Energy has identified closing this observational gap as a high priority research area [29]. Optical remote sensing with lidars is a potential method of providing continuous observation of tropospheric thermodynamics with the required vertical and temporal resolution.

Currently, only Raman lidars have demonstrated complete thermodynamic profiling of the atmosphere [30, 31]. However, spontaneous Raman scattering is a highly inefficient

process, so these lidar systems require extremely high-powered lasers to operate. These high-powered lasers are expensive and unsafe for the human eye, making Raman lidars a poor candidate for use in a network of instruments for continuous data collection. In contrast to Raman lidar, DIAL uses more efficient elastic scattering. Therefore, they are much less expensive and can be made with eye-safe lasers, making them a good candidate for a network of unattended instruments operating autonomously. However, to date, no DIAL has demonstrated complete thermodynamic profiling.

Researchers are currently developing a combined diode-laser-based (DLB) system at the National Center for Atmospheric Research (NCAR) and Montana State University (MSU). The proposed instrument includes an O2-DIAL for temperature profiling, a water vapor DIAL (WV-DIAL) for humidity profiling, and an HSRL for aerosol optical property profiling [23]. This instrument would provide complete thermodynamic profiling of the lower troposphere using the MicroPulse DIAL (MPD) architecture. WV-DIALs [32, 33, 34] and HSRLs [35] have been demonstrated using the MPD architecture. However, atmospheric temperature profiling using the O2-DIAL has yet to be demonstrated with high accuracy. The current prototype MPD instrument at MSU contains an O2-DIAL and HSRL. This instrument is an experimental prototype for validating the temperature profiling, but this research is incomplete [36].

An HSRL in this system is necessary for temperature profiling because the O_2 absorption measurement requires the aerosol backscatter coefficient. Previous work using DIALs for temperature profiling showed that a lack of the aerosol backscatter coefficient and water vapor number density led to a high degree of error [37, 38]. Additionally, the HSRL is valuable as a stand-alone instrument because it can monitor atmospheric aerosol optical properties and the PBLH.

Thesis Organization

This thesis focuses on the design and demonstration of the HSRL instrument of the prototype MPD at MSU. Chapter 2 will introduce background theory and describe the HSRL technique. Chapter 3 will describe the instrument in detail, including the calibration process. Chapter 4 will describe a retrieval algorithm for the aerosol optical properties and present data from the HSRL. Chapter 5 will propose an algorithm for retrieving the PBLH from the aerosol optical properties and display data with radiosonde comparisons. Chapter 6 will conclude the thesis with final thoughts and potential future work. The appendices contain a list of acronyms used throughout the thesis, additional radiosonde comparison data for the PBLH retrieval algorithm, the MATLAB code, and permissions for copyrighted figures.

THEORY

Principles of Lidar

Lidar is an acronym for light detection and ranging. It is a form of active remote sensing where a coherent light source is transmitted, and a detector receives the light scattered by a target. In pulsed lidar, short pulses of collimated laser light are the transmitted signal. The target scatters the light, and the receiver, usually a telescope, collects the scattered signal. The range of the scattering target is determined by measuring the time between the transmitted laser pulse and the return signal. For this reason, lidar is a valuable tool for surveying, forestry, and oceanography. It also has potential for applications in machine vision and self-driving vehicles. These lidars are surface-scattering lidars because they detect the distance to an opaque target. Lidar is also a valuable tool for retrieving range-resolved profiles about atmospheric parameters such as aerosol optical properties, water vapor concentrations, and temperature profiles. When using lidar to measure the atmosphere, gaseous molecules or aerosols are the targets, and only a tiny fraction of the scattered signal is collected. Because there is no hard target, lidars for atmospheric science are volume-scattering lidars. Volume-scattering lidars measure parameters that describe the optical path. A parcel of air that contains molecules and aerosols can be thought of as a scattering volume.

Figure 2.1 is a schematic diagram of a typical pulsed lidar system. A pulsed lidar consists of three main components, (1) a transmitter that sends collimated pulses of laser light at a target, (2) a receiver that collects the return signal into a detector, and (3) a computer that controls the system and measures the time from the transmitted pulse to the detected return signal. There are many types of pulsed lidars, but all of them have some form of these three main components.

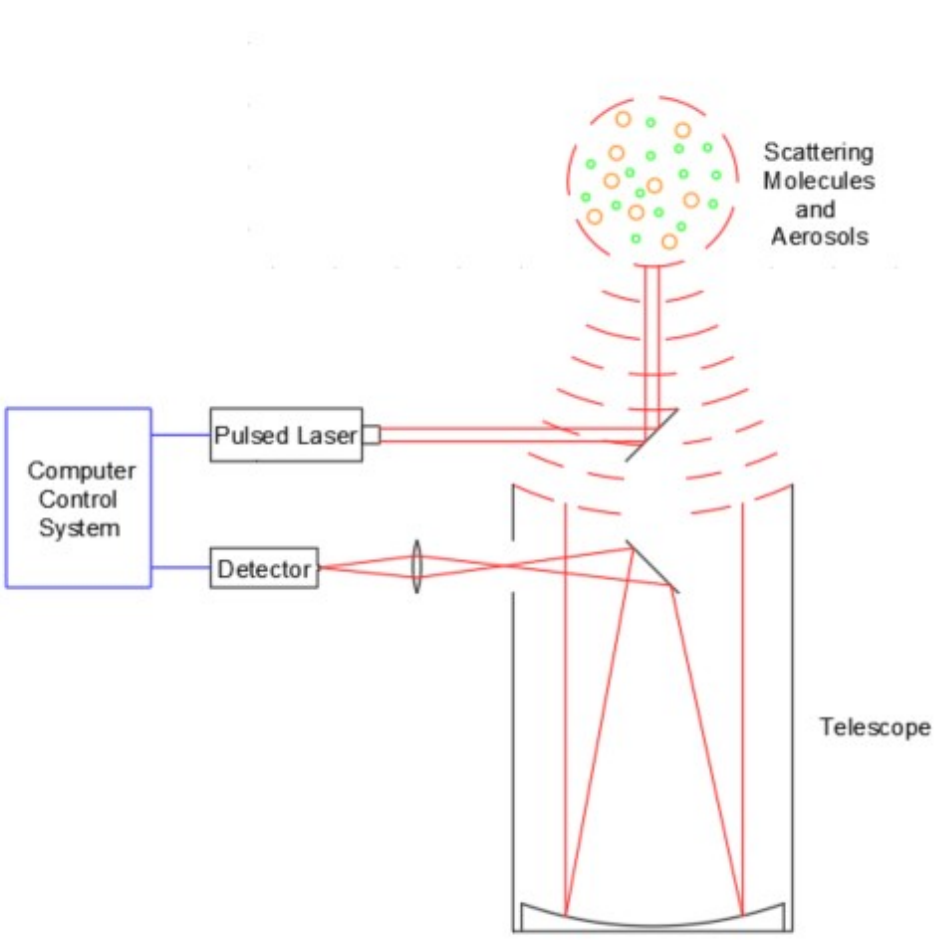


Figure 2.1: Schematic of the main components of a typical pulsed lidar system.

The range, r, is the distance between the detector and the target, c is the speed of light, and t is the time between pulse transmission and return signal detection. The range can be determined by considering that the light pulse travels to the target and back at the speed of light, so the range is half of the speed of light times the flight time.

$$r = \frac{c \cdot t}{2} \tag{2.1}$$

The Lidar Equation

The lidar equation describes the return signal in terms of the transmitter, the receiver, the target, and the optical path. The lidar equation comes in many forms and notations. The form used in this thesis is similar to that used by Bösenberg [38].

$$P_R(\nu, r) = P_T \cdot \frac{c \cdot \Delta t}{2} \cdot \frac{A}{r^2} \cdot \eta(\nu, r) \cdot \beta(\nu, r) \cdot O(\nu, r) \cdot T^2(\nu, r) + P_B \tag{2.2}$$

$P_R(\nu, r)$ is the optical power collected at frequency ν and range r and has units of W.

P_T is the transmitted optical power and has units of W.

$\frac{c \cdot \Delta t}{2}$ is the range resolution and has units of m. c is the speed of light in $\frac{m}{s}$ and Δt is the pulse width duration of the laser in s. This range resolution is the uncertainty in the one-way range measurement between the detector and the scattering volume.

$\frac{A}{r^2}$ is the projected solid angle of the receiver from the perspective of the target and has units of sr. A solid angle is a three-dimensional analog to an angle; an angle subtends an arc on the unit circle, and a solid angle subtends a surface area on the unit sphere. In the small-angle approximation, the projected solid angle is equal to the solid angle. A is the entrance pupil area of the receiver in m^2, and r is the range from the scattering volume in m.

$\eta\,(\nu, r)$ is the combined efficiency of the lidar system as a function of frequency and range.

$\beta(\nu, r)$ is the backscatter coefficient in units of $m^{-1} \cdot sr^{-1}$. This coefficient is the volume density of the backscatter cross sections per unit solid angle. The backscatter cross section is the area of a perfect scatterer that would produce the observed scattering in the backward direction. The volume density of these cross sections represents the probability that a molecule or aerosol scatters incident light in the backward direction per unit solid angle per unit length of the optical path. Multiplying the backscatter coefficient, $\beta(\nu, r)$,

by the projected solid angle of the telescope as seen by a scattering particle, $\frac{A}{r^2}$, gives the probability of collecting a photon in the telescope per unit length of the optical path.

$O(\nu, r)$ is the overlap as a function of frequency and range. This overlap is the portion of the outgoing laser beam visible in the telescope's field of view.

$T(\nu, r)$ is the one-way transmission of the atmosphere. $T(\nu, r) = exp[-\int_0^r \alpha(\nu, r)dr]$ where α is the atmospheric extinction coefficient in units of m^{-1}. The atmospheric extinction coefficient is the term that describes how losses affect light due to both scattering and absorption. It is the probability that a photon will be absorbed or scattered from its optical path per unit length of the optical path.

P_B is the optical power collected from background radiance and has units of W.

Equation 2.2 is similar to standard lidar equations [9, 39, 40], but relies on several assumptions. These assumptions from Bösenberg [38] are:

1. The scattering is incoherent, meaning that the scattered photons have random phase components in the direction of interest. Backward scattering in a lidar application is incoherent.

2. Only single scattering contributes significantly to the received signal. This assumption is valid for standard atmospheric conditions, but this assumption is not valid in clouds or extremely dense aerosol layers.

3. The time constant of scattering is short compared to the pulse length. For the 1-μs pulse width used by the MPD instruments, this assumption is valid.

4. The laser radiation is monochromatic. This assumption is valid for the distributed Bragg reflector lasers because their emission frequency spectra are sufficiently narrow compared to the spectrum of the received signal.

5. The frequency does not change significantly during the scattering process. This assumption is valid for aerosol and molecular scattering, as the frequency shift is small (10^9) compared to the frequency of the laser light (10^{14}).

Equation 2.2 can be converted from power to photon counts. The received power, the transmitted power, and the power from background radiance in terms of photon counts are:

$$P_R(\nu, r) = \frac{N_R(\nu, r) \cdot h \cdot \nu}{\Delta t} \tag{2.3}$$

$$P_T = \frac{N_T \cdot h \cdot \nu}{\Delta t} \tag{2.4}$$

$$P_B = \frac{B \cdot h \cdot \nu}{\Delta t} \tag{2.5}$$

In these equations, h is Planck's constant, ν is the photon frequency, $N_R(\nu, r)$ is the photon counts received by the detector as a function of frequency and range, N_T is the number of photons transmitted by the laser pulse, and B is the number of photons from background radiance. Inserting Equations 2.3, 2.4, and 2.5 into Equation 2.2 and simplifying yields the lidar equation in terms of photon counts.

$$N_R(\nu, r) = N_T \cdot \frac{c \cdot \Delta t}{2} \cdot \frac{A}{r^2} \cdot \eta(\nu, r) \cdot \beta(\nu, r) \cdot O(\nu, r) \cdot T^2(\nu, r) + B \tag{2.6}$$

Let $N(\nu, r)$ be defined as the background-subtracted photon counts.

$$N(\nu, r) = N_R(\nu, r) - B \tag{2.7}$$

The background-subtracted lidar equation then becomes:

$$N(\nu, r) = N_T \cdot \frac{c \cdot \Delta t}{2} \cdot \frac{A}{r^2} \cdot \eta(\nu, r) \cdot \beta(\nu, r) \cdot O(\nu, r) \cdot T^2(\nu, r) \tag{2.8}$$

A visual representation of the background-subtracted lidar equation is shown in Figure 2.2.

Atmospheric Scattering

Overview

Atmospheric scattering can be classified broadly into two types: scattering by gaseous molecules and scattering by aerosols. These molecules are small and the aerosols large compared to the wavelength of light. Atmospheric scattering by molecules can be divided into elastic scattering and inelastic scattering. There is no frequency shift of the scattered photon for elastic scattering, and there is a frequency shift for inelastic scattering. Rayleigh scattering theory, which describes the scattering effects of spheres that are small compared to the wavelength of light, can be used to model elastic scattering by molecules. Mie scattering theory is needed to describe the elastic scattering effects of aerosols, as Mie scattering deals with spheres large as well as those small compared to the wavelength of light. Inelastic scattering can be divided into Brillouin scattering, which involves scattering by acoustic phonons, and Raman scattering, which involves vibrational and rotational energy levels within a molecule. Regardless of the type of particle, all scattering operates according to the same principles. As defined by Bohren and Huffman [41]: "If an obstacle, which could

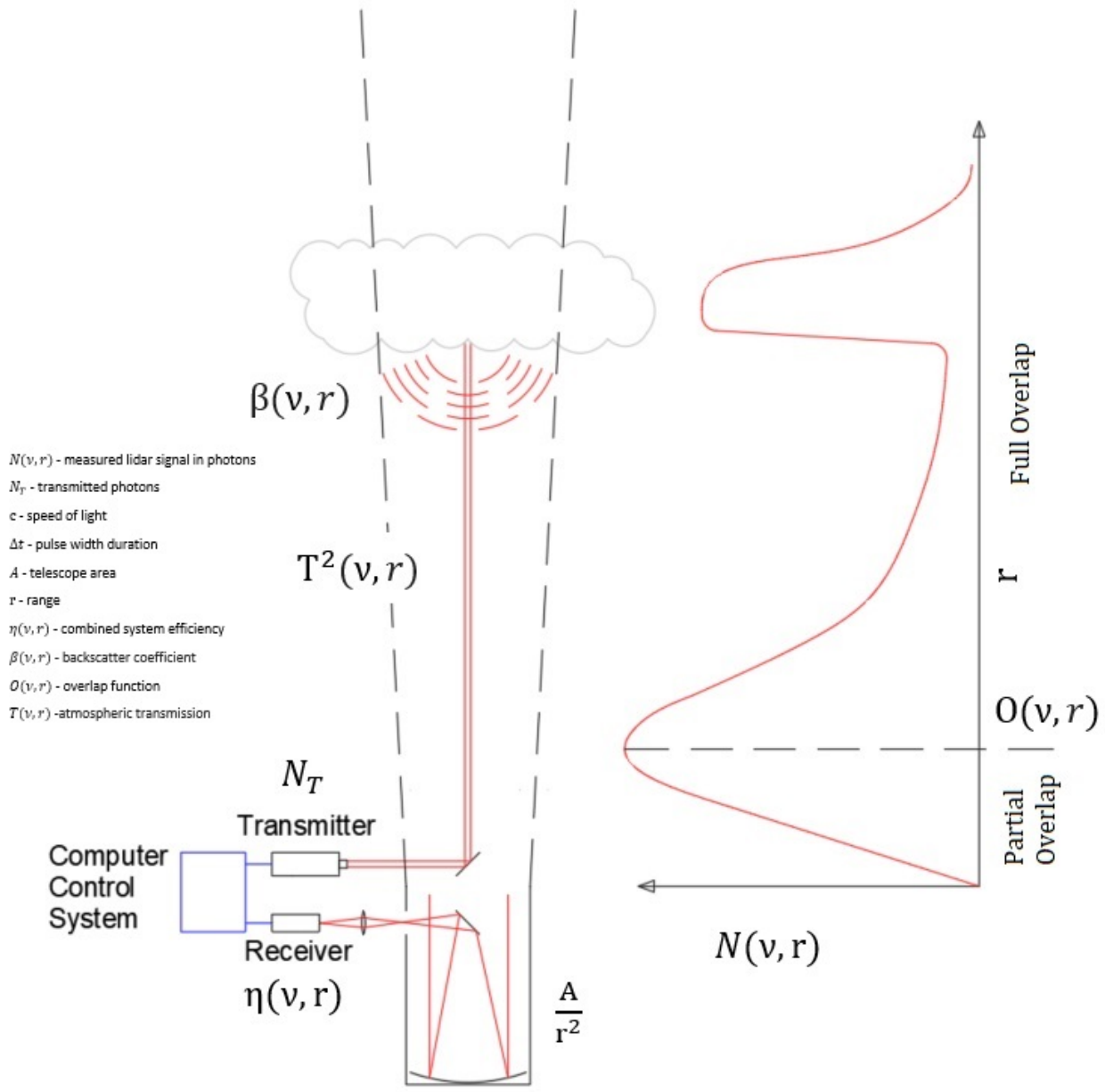


Figure 2.2: Lidar schematic with components that contribute to terms from lidar equation.

be a single electron, an atom or molecule, a solid or liquid particle, is illuminated by an electromagnetic wave, electric charges in the obstacle are set into oscillatory motion by the electric field of the incident wave. Accelerated electric charges radiate electromagnetic energy in all directions; it is this secondary radiation that is called the radiation scattered by the obstacle."

Molecular Scattering

Rayleigh Scattering Light scattered by gaseous molecules with a radius less than $\frac{1}{10}$ of the wavelength of light can be described by Rayleigh scattering theory [42]. Rayleigh scattering theory describes a scattering mechanism where a spherical particle scatters light with intensity, I, inversely proportional to the wavelength to the fourth power ($I \propto \frac{1}{\lambda^4}$) and proportional to one plus the square of the cosine of the scattering angle, θ ($I \propto (1+cos^2(\theta))$). The equation for the intensity of Rayleigh scattering, where I_0 is the incident intensity and α_p is the molecular polarizability, is given by equation 2.9. Polarizability is a measure of the tendency of a molecule to form an electric dipole moment in the presence of an electromagnetic field. It should not be confused with the extinction coefficient in the lidar equation.

$$I = I_0 \cdot \frac{8 \cdot \pi^4 \cdot \alpha_p^2}{\lambda^4 \cdot r^2} \cdot (1 + cos^2(\theta)) \tag{2.9}$$

For a lidar with a co-located transmitter and receiver with negligible multiple scattering, θ is equal to π. Assuming a temporally fixed mixture of molecular species in the atmosphere, the polarizability, α_p, is constant. For laser transmitters with a sufficiently narrow bandwidth, it can be assumed that the wavelength, λ, is constant. According to these assumptions, the intensity of scattered light through Rayleigh scattering varies linearly with the incident intensity and is inversely proportional to the square of the range ($I \propto \frac{I_0}{r^2}$).

In the reference frame of the scattering molecule, the scattered photon experiences no

frequency shift. However, the thermal motion of the gaseous molecules causes the scattering reference frame to move relative to a stationary observer. From the observer's reference frame, there will be an observed frequency shift caused by the relative motion of the molecules to the observer's reference frame. This frequency shift is the Doppler shift and is responsible for the change in pitch of a siren as an ambulance rushes past. It is also responsible for the redshift in the observed spectral lines of stars. The Doppler shift for a gaseous mixture moving randomly due to thermal motion causes a broadening of the frequency spectrum of the return signal scattered by molecules. The velocity of these molecules follows a Maxwell-Boltzmann distribution dependent on the molecular mass and temperature. Air molecules have an average velocity of about 300 $\frac{m}{s}$ at standard atmospheric conditions. This motion causes a frequency shift with a Gaussian distribution that has a full width at half maximum (FWHM) of approximately 2 GHz. Where k_B is the Boltzmann constant, m is the molecular mass of air, c is the speed of light, and ν_0 is the central frequency, the FWHM of the frequency shift spectrum is given by equation 2.10. This equation assumes an ideal gas and is not valid for gases at high pressures. For atmospheric pressures, this equation gives a good approximation.

$$\nu_{FWHM} = \sqrt{\frac{32 \cdot k_B \cdot T \cdot ln(2)}{m \cdot c^2}} \cdot \nu_0 \tag{2.10}$$

Due to the almost constant molecular mass in the atmosphere and the narrow spectrum of the transmitted laser, the shape of the Doppler broadening depends entirely on the temperature of the scattering molecules. As the temperature increases, the average velocity of the gas molecules also increases, and the FWHM of the frequency shift spectrum widens. Figure 2.3 demonstrates the Doppler broadening of light scattered by molecules at 300 K and 200 K.

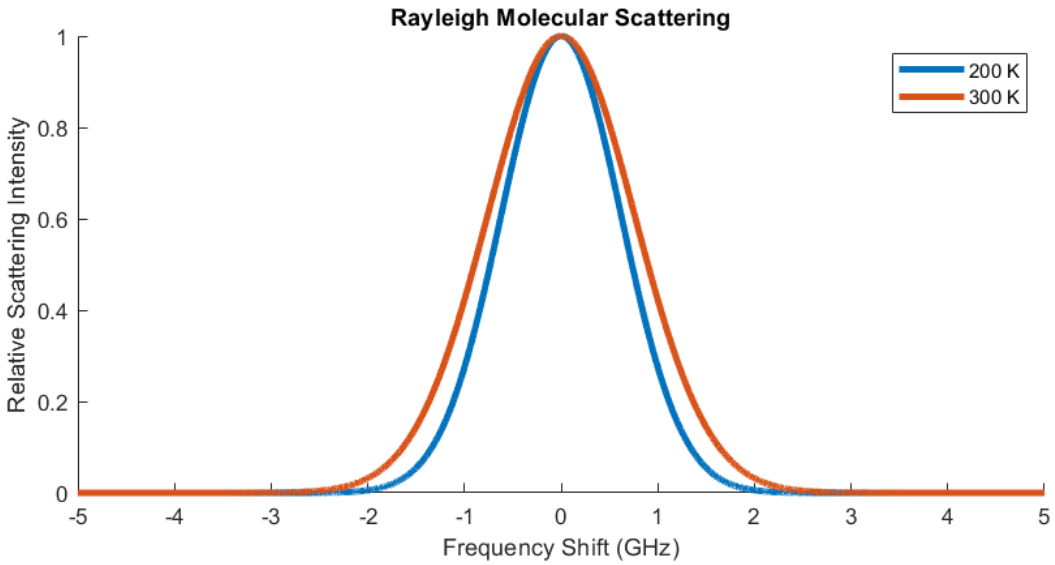


Figure 2.3: Doppler broadening in molecular scattering.

Brillouin Scattering Sound waves propagate through the atmosphere as low-frequency fluctuations in pressure and density. Brillouin scattering is the inelastic scattering of light by these sound waves, where the mechanical energy in the acoustic phonon couples with the electromagnetic energy in the photon, causing a frequency shift [43]. In Stokes Brillouin scattering, a phonon is created, and the scattered photon has less energy and, therefore, a lower frequency than the incident photon. In anti-Stokes Brillouin scattering, an existing phonon is annihilated, and the scattered photon has a higher frequency than the incident photon. The frequency shift introduced by Brillouin scattering depends on the speed of sound, v_s, the wavelength, λ, and the scattering angle, θ. Where n is the number of phonons that couple with the scattered photon, the frequency shift is:

$$\Delta\nu = \frac{2 \cdot n \cdot v_s}{\lambda} \cdot sin(\frac{\theta}{2}) \tag{2.11}$$

Figure 2.4 demonstrates the Brillouin spectrum, ignoring Doppler broadening from the thermal motion of the scattering molecules. The central peak is known as the Cabannes line and represents elastic scattering. The two lines surrounding the central peak of the Cabannes line are a Brillouin doublet for n=1, which is the excitation or annihilation of a single phonon in the scattering event. Higher orders ($n > 1$) are rare in spontaneous Brillouin scattering, so they usually are not considered for atmospheric models. The line corresponding to an increase in frequency is the anti-Stokes line, and the line corresponding to a decrease in frequency is the Stokes line, as shown in Figure 2.4. The Doppler broadened frequency spectrum is shown for scale.

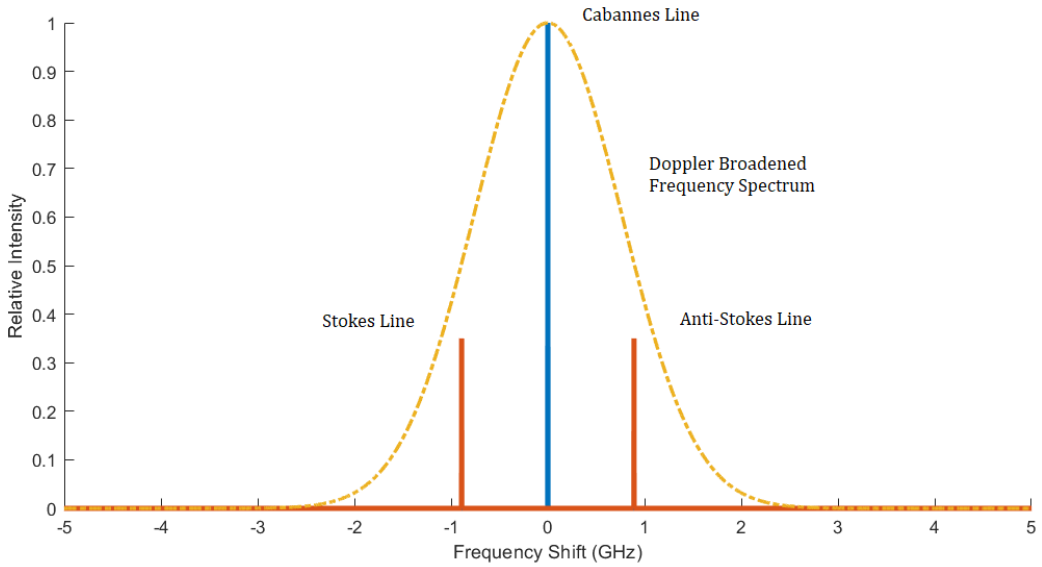


Figure 2.4: Cabannes, Stokes, and anti-Stokes scattering lines.

Unless the gaseous particles are at zero kelvin, there will always be thermal broadening. The combined Rayleigh-Brillouin Spectrum can be visualized as a convolution of the Gaussian profile caused by thermal Doppler broadening and the delta functions shown in Figure 2.4. In this figure, the relative intensities of the Stokes and Anti-Stokes lines are realistic values but were arbitrarily chosen for the visualization. Figure 2.5 shows a Gaussian

profile convolved with the delta functions demonstrating the frequency spectrum of molecular scattering from elastic Rayleigh scattering and inelastic Brillouin scattering.

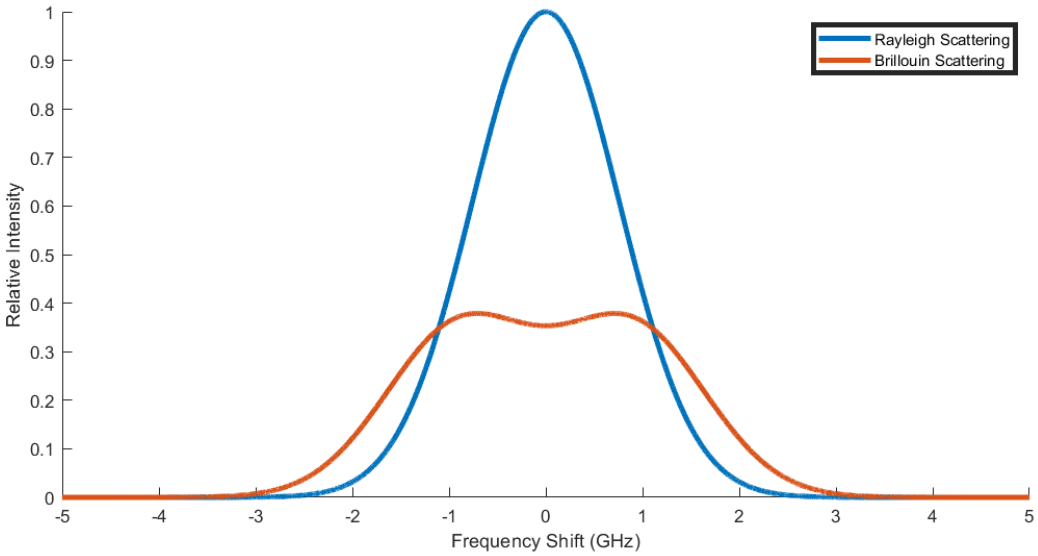


Figure 2.5: Rayleigh and Brillouin scattering.

The frequency spectrum of light scattered by molecules in the atmosphere is affected by both Rayleigh and Brillouin scattering, and it is commonly known as the Rayleigh-Brillouin spectrum. This spectrum is simply the addition of the Brillouin spectrum and the Rayleigh spectrum shown in Figure 2.5. Figure 2.6 displays the superposition of the Rayleigh and Brillouin spectra. The addition of Brillouin scattering causes a significant change in the frequency spectrum of the backscattered return signal from atmospheric molecules. As shown in Figure 2.6, when Rayleigh scattering dominates, the Rayleigh-Brillouin spectrum looks like a broadened Gaussian profile from Rayleigh scattering at the same molecular temperature.

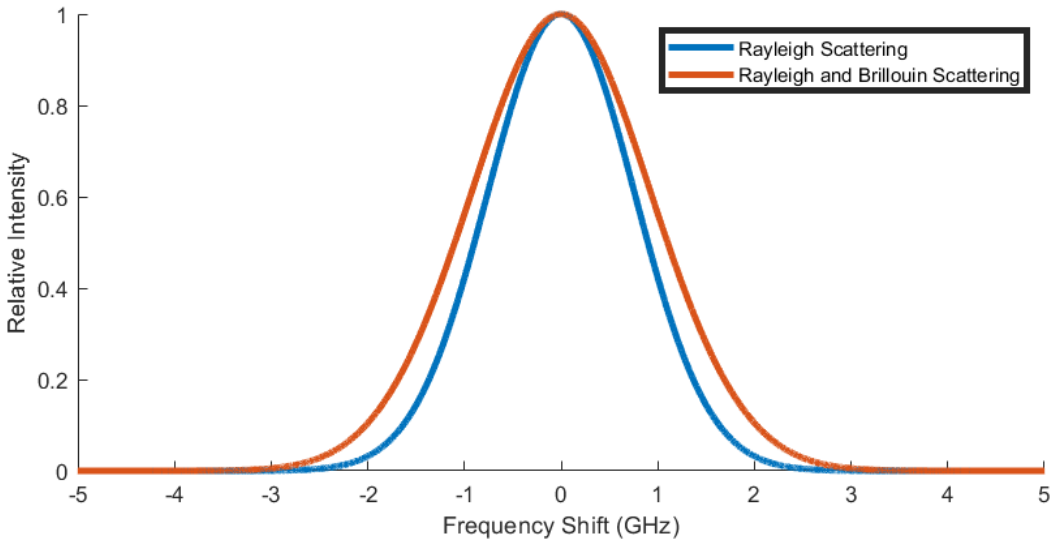


Figure 2.6: Rayleigh and Rayleigh-Brillouin scattering.

The Tenti-S6 Model The Rayleigh-Brillouin scattering spectrum differs significantly from the Rayleigh scattering spectrum, so the effects of Brillouin scattering must be accounted for. The most accurate models of the Rayleigh-Brillouin spectrum are the Tenti-S6 [44, 45] and the Pan-S7 models [46, 47, 48]. The Tenti-S6 model was used to simulate the spectra of molecular scattering in this thesis. The Tenti-S6 model considers six eigenvectors. As described by Wang [49], the eigenvectors are related to six gas parameters:

1. The fraction of molecules in different internal states,
2. The momentum,
3. The translational energy,
4. The translational heat flux,
5. The internal energy,
6. The internal heat flux.

A MATLAB code written by John Smith at the University of Colorado generated the frequency spectra for different conditions [50]. The numerical model requires the molecular mass, molecular temperature, pressure, shear viscosity, thermal conductivity, internal specific heat capacity, and bulk viscosity of the gas. The molecular mass of air is known to be 28.97 $\frac{g}{mol}$. The temperature and pressure are left as variables. The shear viscosity and the thermal conductivity of air were determined using the Sutherland equations, which are empirical equations based on the kinetic theory by Sutherland (1893) [51]. These equations give less than 2% error for temperatures between 170-1900 K in air. The Sutherland model, where μ is the shear viscosity, and κ is the thermal conductivity, is as follows:

$$\frac{\mu}{\mu_0} = (\frac{T}{T_0})^{\frac{3}{2}} \cdot \frac{T_0 + S_\mu}{T + S_\mu} \tag{2.12}$$

$$\frac{\kappa}{\kappa_0} = (\frac{T}{T_0})^{\frac{3}{2}} \cdot \frac{T_0 + S_\kappa}{T + S_\kappa} \tag{2.13}$$

Table 2.1: Sutherland Equation Constants.

$T_0(K)$	μ_0 $(\frac{kg}{(m \cdot s)})$	$\kappa_0(\frac{W}{(K \cdot m)})$	$S_\mu(K)$	$S_\kappa(K)$
273	$1.716 \cdot 10^{-5}$	0.0241	111	194

The internal specific heat capacity of air is known to be 1. There is no empirical model for bulk viscosity across the required temperature range. At temperatures close to the standard atmosphere, it is approximately 0.71 times the shear viscosity, so the bulk viscosity is modeled using this relationship.

This model was used to generate frequency spectra for various molecular temperatures and pressures. Frequency return spectra for intermediate values are found through interpolation. Figure 2.7 demonstrates the influence of temperature on the Rayleigh-Brillouin spectrum at 100 kPa and Figure 2.8 shows the influence of pressure on the Rayleigh-Brillouin spectrum at 300 K. The Brillouin doublet is stronger for higher pressures and lower temperatures, and the Doppler broadening is wider at higher temperatures.

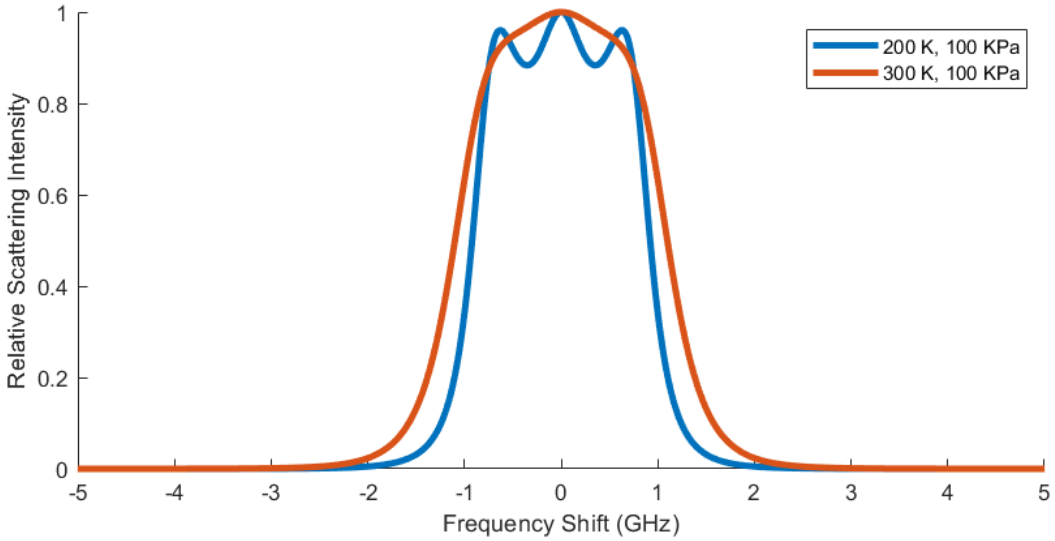


Figure 2.7: Influence of temperature on the Rayleigh-Brillouin spectrum.

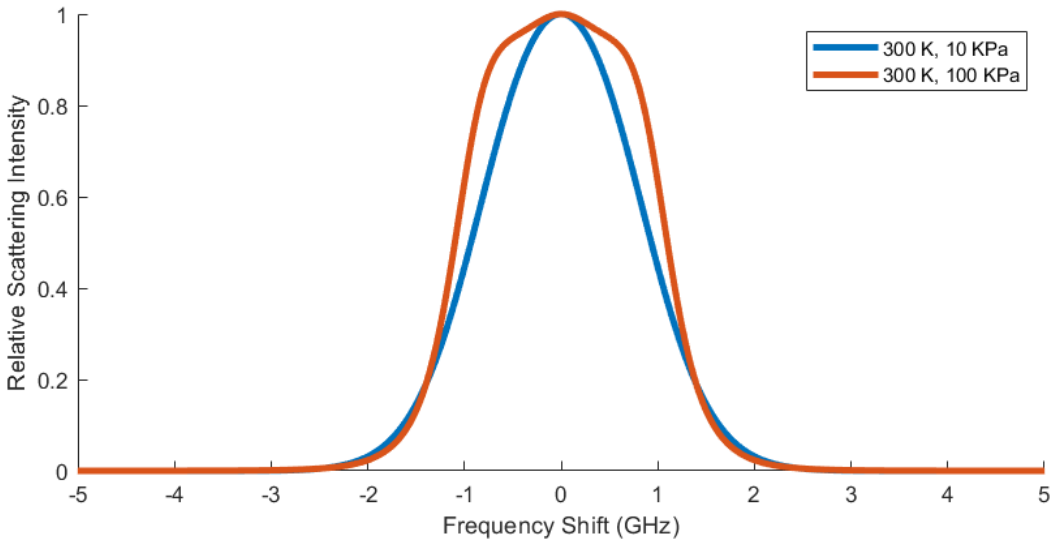


Figure 2.8: Influence of pressure on the Rayleigh-Brillouin spectrum.

Aerosol Scattering

Particles that are large compared to the wavelength of light scatter photons through a scattering mechanism described by Mie scattering theory. Mie scattering describes the electromagnetic scattering of a plane wave by a homogeneous sphere that is either small or large compared to the wavelength of the scattered photon [41]. Mie scattering theory includes Rayleigh scattering as well as scattering by aerosols. When possible, Rayleigh scattering theory is preferred due to the complicated formulation of Mie scattering theory. Scattering by large particles can only be solved analytically and is highly dependent on the size of the particle. For this reason, it is impossible to determine quantitative information about the aerosols in the atmosphere from the lidar return signal without knowing the specific aerosol geometry. However, the optical properties of these aerosols are still valuable.

One crucial difference between the scattering of light by molecules and the scattering of light by aerosols is that the scattering of light by aerosols introduces a much narrower frequency shift. Atmospheric aerosols move with the wind at speeds of about 10 $\frac{m}{s}$ and with

turbulence at speeds of about 1 $\frac{m}{s}$. The FWHM for the aerosol scattering frequency spectrum is on the order of MHz. The Doppler broadening introduced by these lower speeds causes a much narrower frequency shift distribution in the return signal. In addition, aerosols will not introduce inelastic scattering mechanisms such as Brillouin scattering because sound waves do not propagate through these suspended particles.

Other Scattering

Raman scattering is an inelastic scattering mechanism in which vibrational-rotational energy from molecules couples with the scattered radiation. The frequency of the scattered photons is higher or lower than the incident photons from the reference frame of the scattering molecule [52]. The frequency shift introduced by Raman scattering can be hundreds of THz and can cause a return signal that could be hundreds of nanometers different from the transmitted laser. However, spontaneous Raman scattering occurs less than once in a thousand scattering events, so its effects can be neglected.

Photons scattered by several atmospheric molecules or aerosols experience multiple scattering. Multiple scattering is negligible except in clouds or extremely dense aerosol layers, where multiple scattering is quite common. Multiple scattering can be ignored for standard atmospheric scattering if the receiver field of view is narrow.

Total Atmospheric Scattering

The total atmospheric scattering model is the superposition of the molecular Rayleigh-Brillouin spectrum and the narrow frequency spectrum of aerosol scattering. Figure 2.9 shows the model for combined atmospheric scattering. The Doppler broadening introduced by aerosol motion is three orders of magnitude more narrow than the Doppler broadening introduced by molecular thermal motion, and it appears like a delta function at the center of the frequency spectrum.

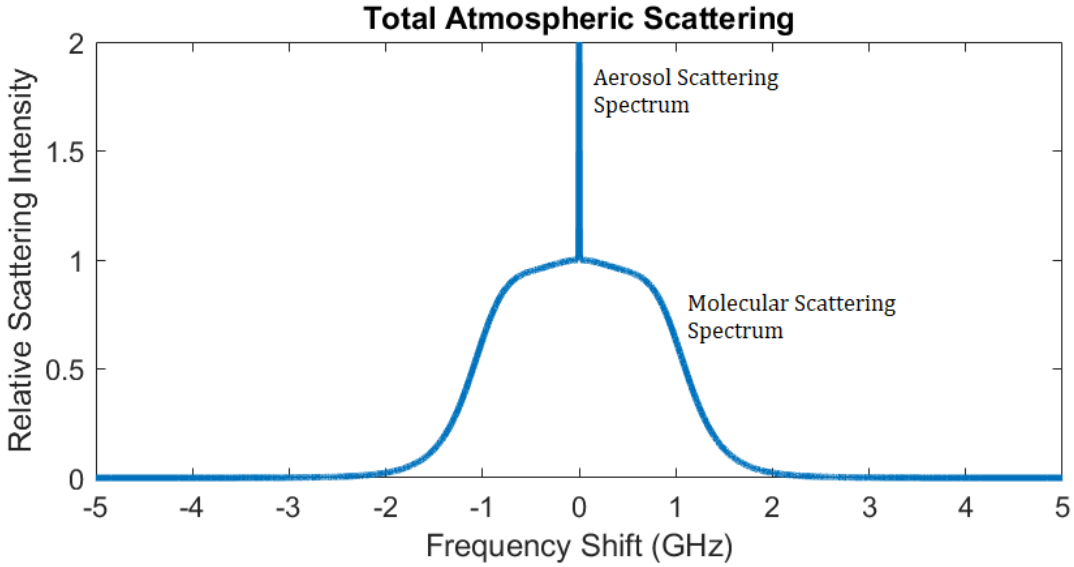


Figure 2.9: Total atmospheric scattering frequency spectrum.

HSRL Theory

Problem Statement and Solution

The problem statement that the HSRL addresses will be described, and the solution will be derived by a method similar to the method used by Eloranta [15]. The HSRL technique seeks to obtain the aerosol backscatter coefficient from the backscattered lidar signal. Standard lidar profiles are composed of several parameters describing the lidar instrument and the optical path. Begin with the background-subtracted lidar equation in terms of photon counts, Equation 2.8.

$$N(\nu, r) = N_T \cdot \frac{c \cdot \Delta t}{2} \cdot \frac{A}{r^2} \cdot \eta(\nu, r) \cdot \beta(\nu, r) \cdot O(\nu, r) \cdot T^2(\nu, r) \tag{2.8}$$

The transmission is the exponential integral function containing the extinction coefficient α.

$$T(\nu, r) = exp[-\int_0^r \alpha(\nu, r) dr] \tag{2.14}$$

The lidar equation relates the measured photon counts to the unknown backscatter coefficient, $\beta(\nu, r)$, and the unknown atmospheric extinction coefficient, $\alpha(\nu, r)$, assuming that all other parameters can be measured. The backscatter coefficient can be divided into two components: the backscatter coefficient due to molecular scattering and the backscatter coefficient due to aerosol scatting. The atmospheric extinction coefficient can be divided into four components: the atmospheric extinction coefficient due to molecular scattering, the atmospheric extinction coefficient due to molecular absorption, the atmospheric extinction coefficient due to aerosol scattering, and the atmospheric extinction coefficient due to aerosol absorption. The backscatter and extinction coefficients become:

$$\beta(\nu, r) = \beta_{mol}(\nu, r) + \beta_{aer}(\nu, r) \tag{2.15}$$

$$\alpha(\nu, r) = \alpha_{mol}(\nu, r) + \alpha_{aer}(\nu, r) \tag{2.16}$$

$$\alpha_{mol}(\nu, r) = \alpha_{mol,sca}(\nu, r) + \alpha_{mol,abs}\ (\nu, r) \tag{2.17}$$

$$\alpha_{aer}(\nu, r) = \ \alpha_{aer,sca}(\nu, r) + \alpha_{aer,abs}(\nu, r) \tag{2.18}$$

These six coefficients are impossible to derive from a single lidar profile. The molecular backscatter coefficient, $\beta_{mol}(\nu, r)$, and the molecular extinction coefficient from scattering, $\alpha_{mol,sca}(\nu, r)$, are derived from Rayleigh scattering theory. For frequencies that are not significantly absorbed by any molecular species, the molecular extinction coefficient from absorption, $\alpha_{mol,abs}\ (\nu, r)$, is assumed to be 0. There are still three unknowns that cannot be obtained using a single lidar profile. There are two possible methods for extracting β_{aer}. The first method assumes a functional dependency of $\alpha_{aer}(\nu, r)$ and $\beta_{aer}(\nu, r)$. It is known

as the Fernald [53] or Klett [54] inversion method and can only be used if the exact type of aerosol is known. This assumption is difficult to make in an atmosphere containing a variety of aerosols. The second method involves separating $\beta_{aer}(\nu, r)$ and $\beta_{mol}(\nu, r)$ in the lidar receiving optics by obtaining two return signals from a single transmitted laser pulse. The efficiencies of the scattering by molecules and the scattering by aerosols can be measured for each receiver channel and, therefore, $\beta_{aer}(\nu, r)$ can be derived. The HSRL technique uses the second method.

Principles of HSRL

HSRL utilizes the difference in the frequency spectra of the return signal from molecular scattering and the return signal from combined atmospheric scattering. The combined scattering spectrum is the superposition of the molecular scattering spectrum and the aerosol scattering spectrum. The HSRL separates the signals by splitting the return signal from a single transmitted laser pulse into two receiver channels: the combined receiver channel and the molecular receiver channel. The combined receiver channel allows the entire return signal frequency spectrum to pass through with minimal attenuation. The molecular receiver channel includes an optical filter that blocks photons with frequencies at the center of the combined return frequency spectrum. This part of the spectrum contains all of the return signal frequency from aerosols and the center of the molecular scattering spectrum. Figure 2.10 demonstrates the combined channel transmission and molecular channel transmission as a function of frequency compared to the combined atmospheric scattering spectrum.

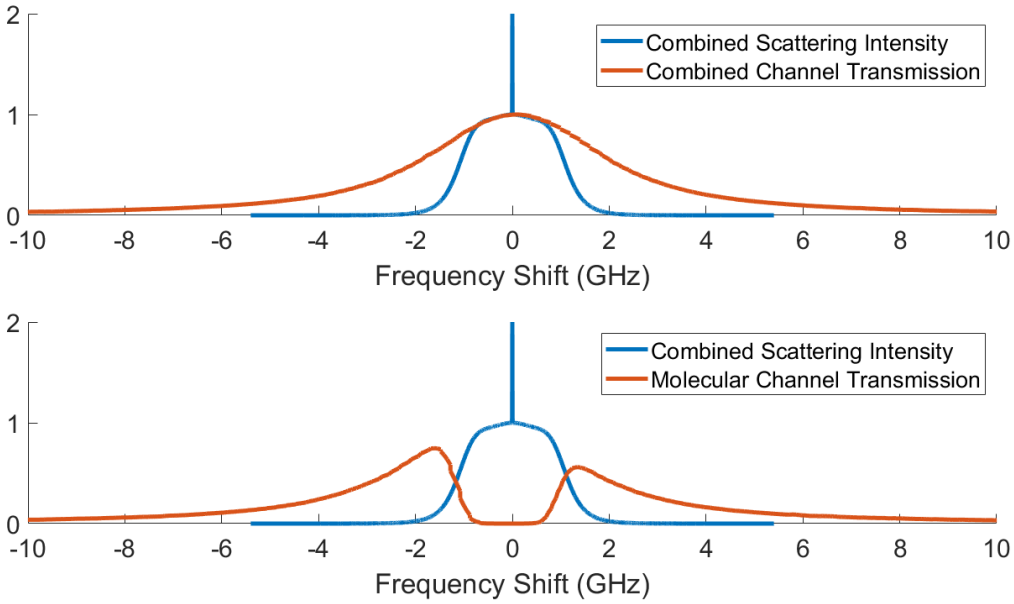


Figure 2.10: Top: Combined channel transmission and combined atmospheric scattering spectrum. Bottom: Molecular channel transmission and combined atmospheric scattering spectrum.

Measurement Theory

The HSRL measurement depends on a theoretical molecular backscatter coefficient that is derived from Rayleigh scattering theory [55]. The Rayleigh molecular extinction coefficient, α_{mol}, is the product of the molecular density as a function of range, $n(r)$, and the Rayleigh scattering cross section as a function of frequency, $\sigma(\nu)$. The extinction coefficient, α_{mol}, can be thought of as the volume density of scattering cross sections caused by molecules.

$$\alpha_{mol}(\nu, r) = n(r) \cdot \sigma(\nu) \tag{2.19}$$

For a standard atmosphere at a temperature of 288.15 K, a pressure of 101.325 kPa, and light with a wavelength of 770.1085 nm, $\alpha_{s,mol}$ is $2.929 \cdot 10^{-6}$ m^{-1}. The subscript s represents the standard atmosphere. The molecular backscatter coefficient is strictly proportional to the molecular extinction coefficient by the molecular lidar ratio, S_{mol}. For Rayleigh scattering theory, the molecular lidar ratio is equal to $\frac{8 \cdot \pi}{3} sr^{-1}$. Recall that the extinction coefficient is the volume density of scattering cross sections, and the backscatter coefficient is the volume density of scattering cross sections per unit solid angle.

$$\alpha_{mol}(\nu, r) = S_{mol} \cdot \beta_{mol}(\nu, r) \tag{2.20}$$

$$\alpha_{mol}(\nu, r) = \frac{8 \cdot \pi}{3} sr^{-1} \cdot \beta_{mol}(\nu, r) \tag{2.21}$$

The molecular backscatter coefficient at a standard atmosphere, $\beta_{s,mol}$, is $3.496 \cdot 10^{-7}$ $m^{-1} \cdot sr^{-1}$ for a wavelength of 770.1085 nm. The molecular backscatter coefficient is proportional to the molar number density, and the molar number density is proportional to pressure divided by temperature. Therefore, the molecular backscatter coefficient can be scaled by the pressure divided by the temperature.

$$\beta_{mol}(r) = \beta_{s,mol} \cdot (\frac{P(r)}{P_s}) \cdot (\frac{T_s}{T(r)}) \tag{2.22}$$

With the standard atmospheric values, this equation becomes:

$$\beta_{mol}(r) = 9.94266 \cdot 10^{-7} \cdot (\frac{P(r)}{T(r)}) \ m^{-1} \cdot sr^{-1} \tag{2.23}$$

Where P is the pressure in kPa, and T is the temperature in K.

There are two remaining unknowns in the lidar equation: α_{aer} and β_{aer}. The aerosol backscatter coefficient can be found using the two return signals with different receiver efficiencies for aerosol and molecular scattering.

To compensate for these different efficiencies through the combined and molecular channels, the calibration constants $C_{MM}(\nu)$, $C_{AM}(\nu)$, $C_{MC}(\nu)$, and $C_{AC}(\nu)$ are introduced by the method described by Stillwell et al. [21]. M stands for molecular scattering and A for aerosol scattering in the first subscript. M stands for the molecular receiver channel and C for the combined receiver channel in the second subscript. These calibration constants are the frequency-dependent transmission efficiencies of the return signals from molecules and aerosols through the molecular and combined channels. The background-subtracted lidar equation for each channel becomes:

$$N_C(\nu,r) = N_T \cdot \eta_{comb}(\nu) \cdot (\frac{A}{r^2}) \cdot O_{comb}(r) \cdot (\frac{c \cdot t}{2}) \cdot (C_{MC}(\nu) \cdot \beta_{mol}(\nu,r) + C_{AC}(\nu) \cdot \beta_{aer}(\nu,r)) \cdot exp[-2 \cdot \int_0^r \alpha(\nu,r) \cdot dr] \tag{2.24}$$

$$N_M(\nu,r) = N_T \cdot \eta_{mol}(\nu) \cdot (\frac{A}{r^2}) \cdot O_{mol}(r) \cdot (\frac{c \cdot t}{2}) \cdot (C_{MM}(\nu) \cdot \beta_{mol}(\nu,r) + C_{AM}(\nu) \cdot \beta_{aer}(\nu,r)) \cdot exp[-2 \cdot \int_0^r \alpha(\nu,r) \cdot dr] \tag{2.25}$$

Where $\eta_C(\nu,r)$ and $\eta_M(\nu,r)$ are all of the other terms in the lidar equation excluding the frequency dependent efficiencies of the optical filters. Equations 2.24 and 2.25 can be converted to matrix form.

$$\begin{bmatrix} N_C(\nu,r) \\ N_M(\nu,r) \end{bmatrix} = \begin{bmatrix} \eta_C(\nu,r) & 0 \\ 0 & \eta_M(\nu,r) \end{bmatrix} \begin{bmatrix} C_{MC}(\nu) & C_{AC}(\nu) \\ C_{MM}(\nu) & C_{AM}(\nu) \end{bmatrix} \begin{bmatrix} \beta_{mol}(\nu,r) \\ \beta_{aer}(\nu,r) \end{bmatrix} \tag{2.26}$$

The calibration constants and the combined efficiencies can be measured through calibration. The calibration process is described in Chapter 3. Dropping the range and frequency dependent notation and solving for β_{aer} yields:

$$\beta_{aer} = \beta_{mol} \cdot [\frac{N_M \cdot \eta_C \cdot C_{MC} - N_C \cdot \eta_M \cdot C_{MM}}{N_C \cdot \eta_M \cdot C_{AM} - N_M \cdot \eta_C \cdot C_{AC}}] \tag{2.27}$$

The aerosol backscatter coefficient is now expressed in terms of measured parameters and the modeled molecular backscatter coefficient. Another important aerosol optical property from the HSRL is the backscatter ratio, BR. It is defined as the ratio of the total atmospheric backscatter coefficient, β, which is the sum of β_{mol} and β_{aer} to the molecular backscatter coefficient, β_{mol}.

$$BR = \frac{\beta_{mol} + \beta_{aer}}{\beta_{mol}} \tag{2.28}$$

The backscatter ratio is the ratio of the signal received to the signal received from molecular scattering. It is a critical parameter for using the DIAL technique for tropospheric temperature retrievals.

HSRL Optical Filter

Several optical filtering methods have been demonstrated to effectively prevent the center of the total atmospheric scattering spectrum from being transmitted through the molecular channel. Scanning Fabry-Perot interferometers [56] and fixed Fabry-Perot interferometers [16, 17] have been used effectively as the HSRL optical filter. Atomic

absorption filters have also been used as the HSRL optical filter. Barium [57], iodine [58], rubidium [35], and potassium [23] absorption filters have all been demonstrated to be effective. These atomic absorption filters are attractive as they are not sensitive to mechanical vibrations or changes in ambient temperature like Fabry-Perot cavities.

DIAL Theory

Overview

Differential absorption lidar (DIAL) is a lidar technique that allows precise measurements of trace gas concentrations. DIAL makes use of a single absorption band of a molecular species of interest. It transmits two laser wavelengths: the online wavelength, which is strongly absorbed by the molecular species of interest, and the offline wavelength, which is absorbed to a lesser extent. Initial success has been made for temperature profiling in the lower troposphere using an O2-DIAL. The topic of this thesis is the development of an HSRL channel of a prototype MPD instrument that retrieves aerosol optical properties. However, it is still worth introducing atmospheric temperature profiling by the O2-DIAL so that the need for the HSRL as part of the MPD instrument can be better understood. This section will briefly present the principles of DIAL, especially the importance of the aerosol optical properties for accuracy in temperature retrievals.

The DIAL Equation

The work of Theopold and Bösenberg demonstrates that accurate temperature profiling with the DIAL technique requires high-quality measurements of the backscatter ratio, O_2 absorption spectra, and water vapor number densities [37, 38, 59]. Tropospheric temperature profiling using the DIAL technique is different from most DIAL applications because instead of detecting the number density of a trace gas, temperature profiling uses a well-mixed gas to measure the temperature-dependent absorption coefficient. The temperature is then

extracted using a perturbative solution method. Diatomic oxygen is well mixed in the atmosphere, and if the water vapor mixing ratio is known, then the number density of O_2 can be determined.

The DIAL equation from Stillwell et al. [23] is:

$$N^{\lambda}_{R_x}(\nu, r) = N^{\lambda}_{T_x} \cdot \Delta r \cdot \frac{A}{r^2} \cdot \eta^{\lambda}_{T_x} \cdot O^{\lambda}(r) \cdot \int_{-\infty}^{\infty} [l^{\lambda}(\nu) \cdot T^{\lambda}(\nu, r) \cdot \beta^{\lambda}(\nu, r)] \cdot T^{\lambda}(\nu, r) \cdot \eta^{\lambda}_{R_x}(\nu) d\nu - B^{\lambda}(r) \tag{2.29}$$

Where $N^{\lambda}_{R_x}$ is the number of photons received as a function of frequency and range, $N^{\lambda}_{T_x}$ is the number of photons transmitted, Δr is the receiver range resolution, A is the area of the receiver aperture, $\eta^{\lambda}_{T_x}$ and $\eta^{\lambda}_{R_x}$ are the transmitter and receiver optical efficiencies, l^{λ} is the normalized laser spectrum, T^{λ} is the atmospheric transmission, β^{λ} is the backscatter coefficient, and $B^{\lambda}(r)$ is the number of background photon counts received.

If the transmitted laser has a narrow frequency spectrum, then l^{λ} can be assumed to be a delta function ($l^{\lambda}(\nu) = \delta(\nu - \nu_{\lambda})$) where ν_{λ} is the emission frequency. The backscatter coefficient can be written according to Equation 2.15.

$$\beta(\nu, r) = \beta_{mol}(\nu, r) + \beta_{aer}(\nu, r) \tag{2.15}$$

Additionally, the previously defined backscatter ratio is:

$$BR = \frac{\beta_{mol} + \beta_{aer}}{\beta_{mol}} \tag{2.28}$$

Defining the normalized photon counts as

$$N^{\lambda}(r) = \frac{N^{\lambda}_{R_x}(\nu, r) - B^{\lambda}(r)}{N^{\lambda}_{T_x}} \tag{2.30}$$

and splitting the molecular backscatter coefficient, $\beta^{\lambda}_{mol}(\nu, r)$, into a spectral shape, $\tilde{\beta}_{mol}(\nu, r)$, and a range dependent scaling factor, $\beta^{\lambda}_{mol}(r)$

$$\beta^{\lambda}_{mol}(\nu, r) = \tilde{\beta}_{mol}(\nu, r) \cdot \beta^{\lambda}_{mol}(r) \tag{2.31}$$

allows equation 2.29 to be rewritten as:

$$N^{\lambda}(r) = \Delta r \cdot \frac{A}{r^2} \cdot \eta^{\lambda}_{T_x} \cdot \eta^{\lambda}_{R_x}(\nu_{\lambda}) \cdot O^{\lambda}(r) \cdot T^{\lambda^2}(\nu_{\lambda}, r) \cdot \beta^{\lambda}{}_{mol}(r) \cdot \{[\mathrm{BR}^{\lambda}(\nu, r) - 1] + \int_{-\infty}^{\infty} \tilde{\beta}^{\lambda}_{mol}(\nu - \nu_{\lambda}, r) \cdot \frac{T^{\lambda}(\nu)}{T^{\lambda}(\nu_{\lambda}, r)} \cdot \frac{\eta_{R_x}(\nu)}{\eta_{R_x}(\nu_{\lambda})} d\nu\} \tag{2.32}$$

One of the requirements to invert this equation using a perturbative solution method is the backscatter ratio. Previous work by Theopold and Bösenberg attempted to estimate the backscatter ratio by using a Fernald or a Klett inversion method, but the error was too high for accurate temperature retrievals. The HSRL provides measurements of the backscatter ratio that allow the DIAL equation to be solved without significant errors. In summary, the HSRL is not only a valuable tool for measuring the optical properties of atmospheric aerosols, but it is also necessary for accuracy in tropospheric temperature profiling using the DIAL technique.

INSTRUMENT DESCRIPTION

Overview

The combined O2-DIAL and HSRL instrument utilizes a diode-laser-based (DLB) MicroPulse DIAL (MPD) architecture similar to the instrument described in Spuler et al. [34, 60] with a few modifications. The following section will reiterate the design. A 24" x 48" optical breadboard contains the laser transmitter and the first stage of the receiving optics. A 12" x 24" optical breadboard contains the second stage of the receiving optics and the potassium cell electrical components. This instrument is designed to create maximum optical power using diode lasers while maintaining an eye-safe optical output. The instrument is shown schematically in Figure 3.1. Images of the lidar are displayed in Figure 3.2.

Researchers at NCAR designed the majority of this instrument. My contribution to the instrumentation included assembling the potassium cell and receiving optics, troubleshooting, aligning, maintaining the lidar system, performing the receiver calibration, and designing and machining the mounting bracket for the wide field of view telescope.

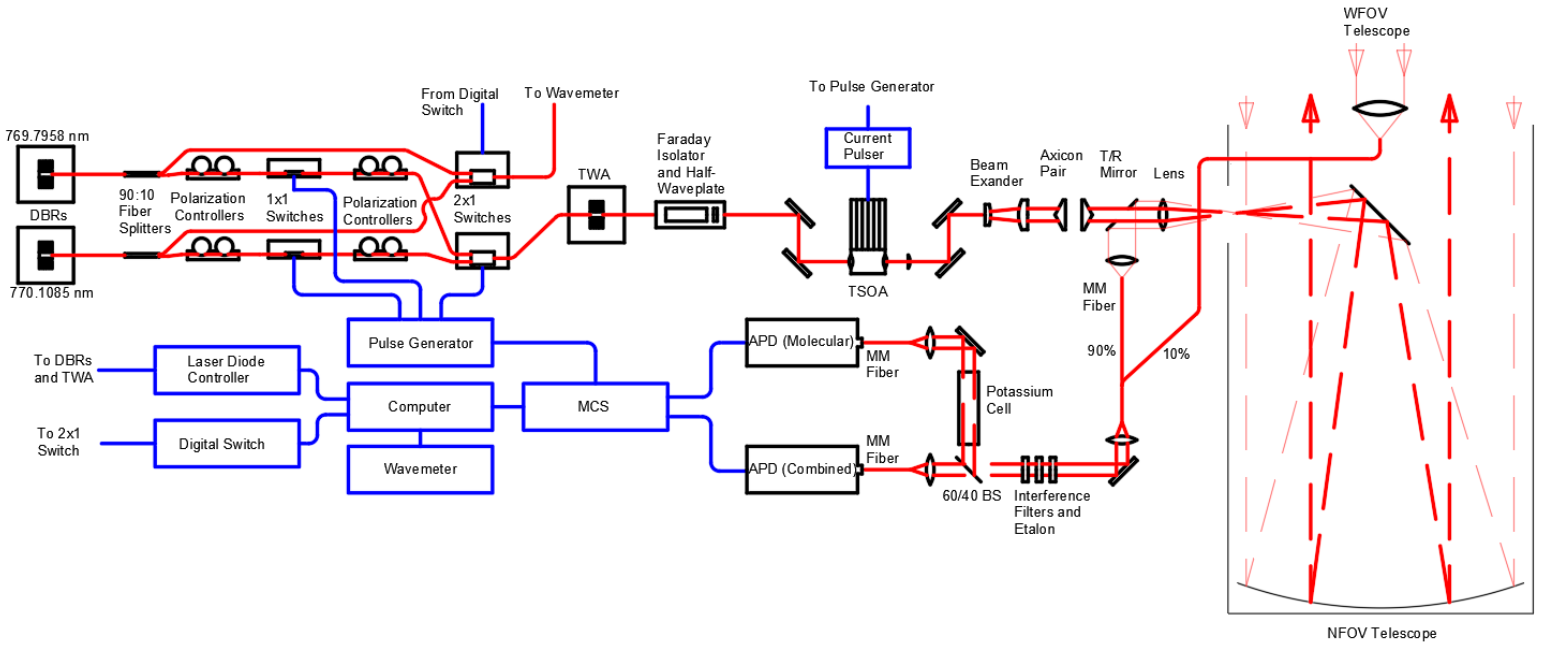


Figure 3.1: Schematic diagram of the combined HSRL and O2-DIAL instrument.

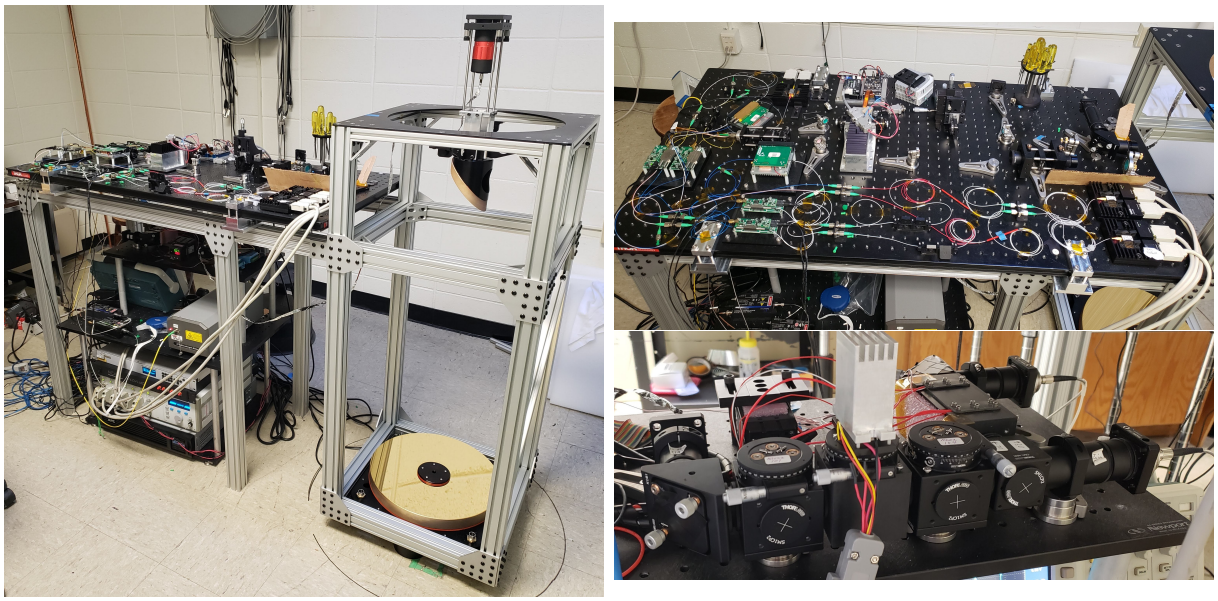


Figure 3.2: Left: An image of the prototype MPD instrument at MSU excluding the computer. Top right: Transmitting optics and the first stage receiving optics. Bottom right: Second stage receiving optics. (Photos courtesy of Owen Cruikshank)

Laser Transmitter

The laser transmitter uses two distributed Bragg reflector (DBR) lasers (Photodigm; PH770BDR015BF-ISO). These DBR lasers incorporate an intrinsic Bragg grating mirror which forces the laser to operate with a single longitudinal mode. The result is an extremely narrow emission bandwidth. One of these lasers transmits the online DIAL wavelength of 769.7958 nm, and the other transmits the offline DIAL wavelength of 770.1085 nm. The HSRL uses the offline DIAL laser. The two lasers run in continuous wave (CW) mode, and the lidar is pulsed using fiber-coupled optical switches. The fiber-coupled laser output power is over 15 mW, and the measured FWHM of the frequency bandwidth is less than 1 MHz. Manual fiber polarization controllers adjust the polarization states of the transmitted laser light. The fibers are wrapped around the spools of the polarization controllers, and the spools are manually rotated. This rotation produces stress-induced birefringence and a

change in the polarization state of the light transmitted in the fiber.

The instrument utilizes a series of fiber splitters and fiber-coupled electro-optic switches to pulse the CW laser light before amplifying it. The first device in the series is a 1 x 2, 90:10, single-mode fiber coupler (Thorlabs; TN785R2A1), and there is one after each laser. The fibers are positioned closely together in this device so the evanescent fields of the two fibers couple with each other and transfer energy like a pair of coupled pendulums. The 90% fiber output from this splitter is sent forward to a 1 x 1 switch. These switches (Agiltron; NPSW-117113323) allow transmission with the correct applied voltage and completely attenuate the light if no voltage is applied. They have a crosstalk suppression of 20 dB and an insertion loss of less than 1 dB. After the 1 x 1 switch, the laser light from both lasers is directed to a 2 x 1 switch. These switches (Agiltron; NPSW-127113323) also have an insertion loss of less than 1 dB and a crosstalk suppression of 20 dB. The 1 x 1 and 2 x 1 switches are pulsed at 100 Hz, transmitting one laser wavelength past the switch at a time and suppressing the other with 40 dB. The 10% fiber is directed into a different 2 x 1 switch. The output of this 2 x 1 switch is directed to a wavelength meter for monitoring the laser wavelengths. A data acquisition (DAQ) module (National Instruments; USB-6000) acting as a digital switch connected to the computer controls the 2 x 1 switch for the wavelength locking control loop. The wavelength locking control loop is controlled separately from the rest of the lidar system.

The output of the first 2 x 1 switch is directed to a traveling wave amplifier (TWA). This TWA is a semiconductor optical amplifier (SOA) module (Superlum; SOA-332) that amplifies the laser pulses from the DBRs to make up for losses through the switches. The TWA increases the optical power for saturating the tapered semiconductor optical amplifier (TSOA). The output of the TWA is directed into a fiber collimator to convert the optical fiber output of the TWA to a free-space beam. The free-space beam is then directed into a Faraday isolator to prevent unwanted feedback. This Faraday isolator uses the magneto-optical effect, where a magnetic field induces a polarization rotation independent of the direction of

propagation. The direction-independent waveplate is sandwiched between crossed polarizers, causing light to transmit in the forward direction but not in the reverse direction. A half-waveplate controls the polarization of the transmitted light for optimizing the TSOA seeding. Saturation of the TSOA occurs at around 10 mW of input power.

The TSOA is a semiconductor optical amplifier that is tapered so that the beam area expands as it passes through the gain material. This allows for greater amplification because the power density is decreased as the beam is expanded outwards. The TSOA is sensitive to retro-reflection, and passing light through the TSOA in reverse causes a high power density which can damage the semiconductor. The light is focused into the TSOA using a pair of mirrors in a z-fold configuration and an aspheric lens (f=4.6 mm, NA=0.55). An aspheric lens is needed to focus light onto the 3 μm x 3 μm square input aperture of the TSOA. The TSOA (Eagleyard; EYP-TPA-0780-03000-4006-CMT04-0000) is pulsed at a rate of 7 kHz with a drive current of 10 A. The pulse length for the TSOA is 1 μs, which is the limiting pulse width for range resolution. This 1 μs pulse length gives a range resolution, Δr, of 150 m. The pulse width could be reduced to obtain a better range resolution at the cost of a lower return signal. The output of the TSOA is a diverging, astigmatic beam. This beam is collimated with a pair of lenses, one aspheric (f=4.5 mm, NA=0.53), and one cylindrical (f=50 mm).

The instrument utilizes a shared telescope design for improved optomechanical stability and cost-effectiveness. The collimated output of the TSOA is expanded and then directed into an axicon pair using a pair of mirrors in a z-fold configuration. An axicon is a conical prism, and the output from a pair of axicons is a ring-shaped power distribution. The beam's maximum power is on the outside of the ring. The axicon pair is used so that the Newtonian telescope's secondary mirror does not obstruct most of the beam's power. The ring-shaped beam passes through a hole cut in the center of an elliptical mirror. This mirror is the Transmit/Receive (T/R) mirror. The hole in the mirror maps onto the center

of the diameter of the Newtonian telescope's primary mirror. This center portion of the mirror expands the transmitted laser pulse. The outer half of the primary mirror's diameter collects the received signal, as this part of the mirror reflects light onto the reflective surface of the T/R mirror. After the T/R mirror, a lens that forms a shared focal system with the telescope's primary mirror focuses the output beam. The lens (f=60 mm) is plano-convex to minimize aberrations, and the telescope is an f/3 system with a primary mirror diameter of 400 mm. The beam is then directed through a window in the ceiling of the laboratory and into the atmosphere. The outgoing pulse has an outer diameter of approximately 200 mm and an inner diameter of about 100 mm.

Receiving Optics

The receiver uses two telescopes to collect the return signal. The primary telescope is a Newtonian telescope, and the transmitting and receiving optics share this telescope. This telescope has an overlap function that reaches complete overlap at about 1.4 km and is called the narrow field of view (NFOV) telescope. This telescope receives signals from 0.75-5 km well but cannot collect light effectively from the lowest range bins. For this reason, an additional telescope collects light from the lowest altitudes. An achromatic fiber collimator (Thorlabs; C80FC-A) is mounted above the secondary mirror to increase the counts at low ranges. This telescope reaches complete overlap at around 150 m and is called the wide field of view (WFOV) telescope. This WFOV also significantly increases photon counts from background radiance because it has a wider cone of acceptance.

The light gathered by the NFOV telescope is collimated by the spherical lens, which is shared with the transmitting optics. The reflective surface of the T/R mirror maps onto the outer diameter of the telescope's primary mirror, so the received signal incident on the outer half of the telescope's primary mirror is focused into a fiber collimator. The received signals from the NFOV and WFOV telescopes are combined using a 1 x 2, 90:10, multimode fiber

coupler (Thorlabs; TM105R2F1A) with the NFOV signal entering the 90% fiber input and the WFOV signal entering the 10% fiber. The light is then directed into the second stage of the receiving optics.

The second stage of the receiving optics begins with a fiber collimator, which converts the light into a collimated free-space beam. The beam is then directed through a series of optical filters that remove the signal from background radiance. The filter series consists of an interference filter, then an etalon, and then another interference filter. The interference filters have an FWHM bandpass of 1.2 nm with a centerline at a wavelength of 769.96 nm (Alluxa; 769.96-1 OD7). These filters operate using an optical coating with thin layers that produce a transmission function that constructively interferes for a selected frequency band and destructively interferes for all other frequencies. They are mounted on kinematic rotating mounts (Thorlabs; B4CRP) for fine transmission spectrum adjustments.

The etalon (Light Machinery, OP-11951) is a Fabry-Perot cavity that only allows transmission for wavelengths that can be produced by dividing the optical path length through the cavity by an integer. It has a free spectral range of 158.14 GHz and a finesse of 40. These parameters correspond to maximum transmission peaks at wavelengths 0.31 nm apart and an FWHM of approximately 7.75 pm in the range around 770 nm. This etalon is mounted on a kinematic mount like the transmission filters with an attached thermoelectric controller (TEC). The kinematic mount allows for manual adjustment of the transmission spectra by the Fabry-Perot cavity by changing the angle of the optical path through the cavity. The TEC allows for precise adjustments by controlling the temperature of the Fabry-Perot cavity. The cavity expands when heated and contracts when cooled, so changing the temperature changes the cavity length and, therefore, the wavelengths that resonate in the cavity. The transmission filters and the etalon reduce the background to less than 10 counts per bin in nighttime operation and less than 30 counts per bin in daytime operation for clear sky conditions. After the filters, the signal is directed to a 60:40 beamsplitter where

40% of the light enters a fiber collimator. This fiber is then sent to the combined channel detector. This detector measures the online and offline DIAL signals; the combined channel HSRL signal is the offline DIAL signal. The remaining 60% of the light is reflected into the potassium absorption cell.

The potassium absorption cell (Thorlabs; GC25075-K) is the optical filter of the HSRL. This borosilicate glass cell contains potassium at 378 K (105°C) at a low pressure, which causes the potassium to evaporate. A potassium vapor cell is used because the KD1 spectral line of potassium is at 770.1085 nm, near an oxygen absorption line at 769.7958 nm. The proximity of these absorption lines allows the offline DIAL laser to be used for the HSRL technique. The cell absorbs light at 770.1085 nm with an FWHM of approximately 0.4 pm. The potassium cell electrical schematic diagram is displayed in Figure 3.3. Using heaters (Omega; KHLVA-103/10), a heater controller (Omega; CN243-R1-F2), a relay (IDEC; RV8H-L-AD24), a 24 V power supply (Mouser; 2902992), and a resistance temperature detector (RTD) (Minco; S665PDZ12AC), a control loop maintains the potassium cell at the desired temperature. After passing through the potassium cell, the light is coupled into an optical fiber and directed to the molecular detector. This detector measures the molecular return signal of the HSRL. The online laser signal in the molecular channel is used as a reference for measuring the difference in overlap functions between channels. The transmission through the optical filters is shown in Figure 3.4 and the transmission curves for the combined and the molecular receiver channels are shown in Figure 3.5.

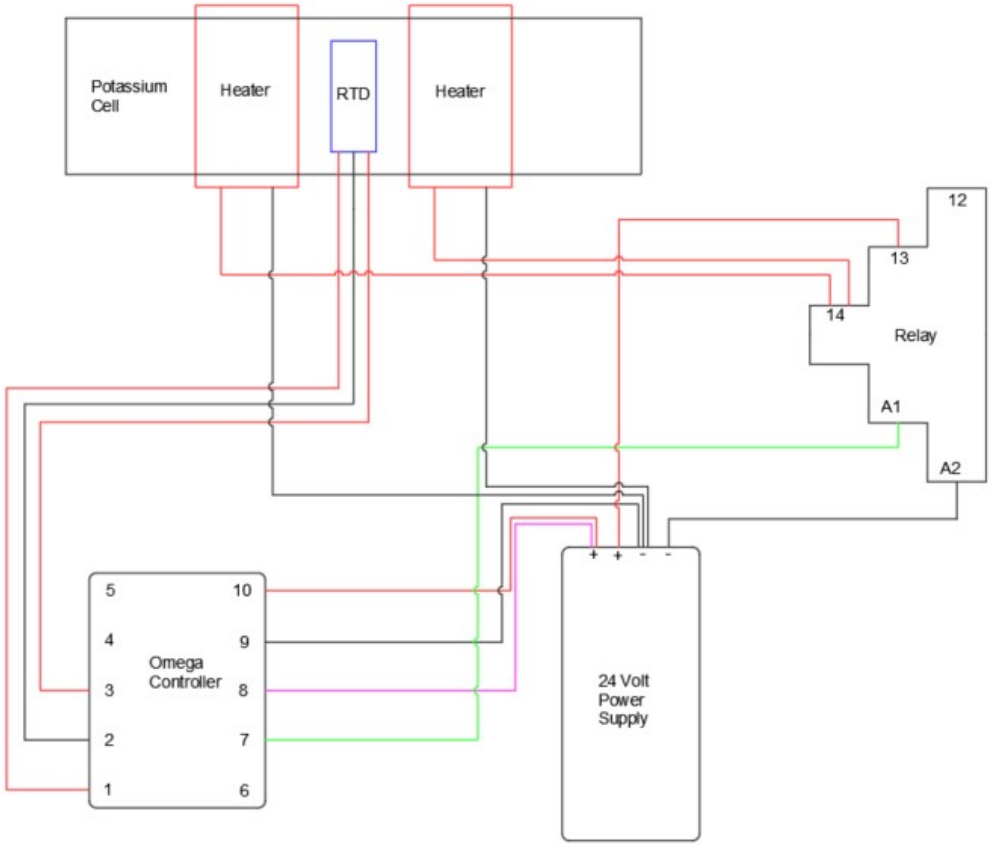


Figure 3.3: Potassium cell electrical schematic diagram.

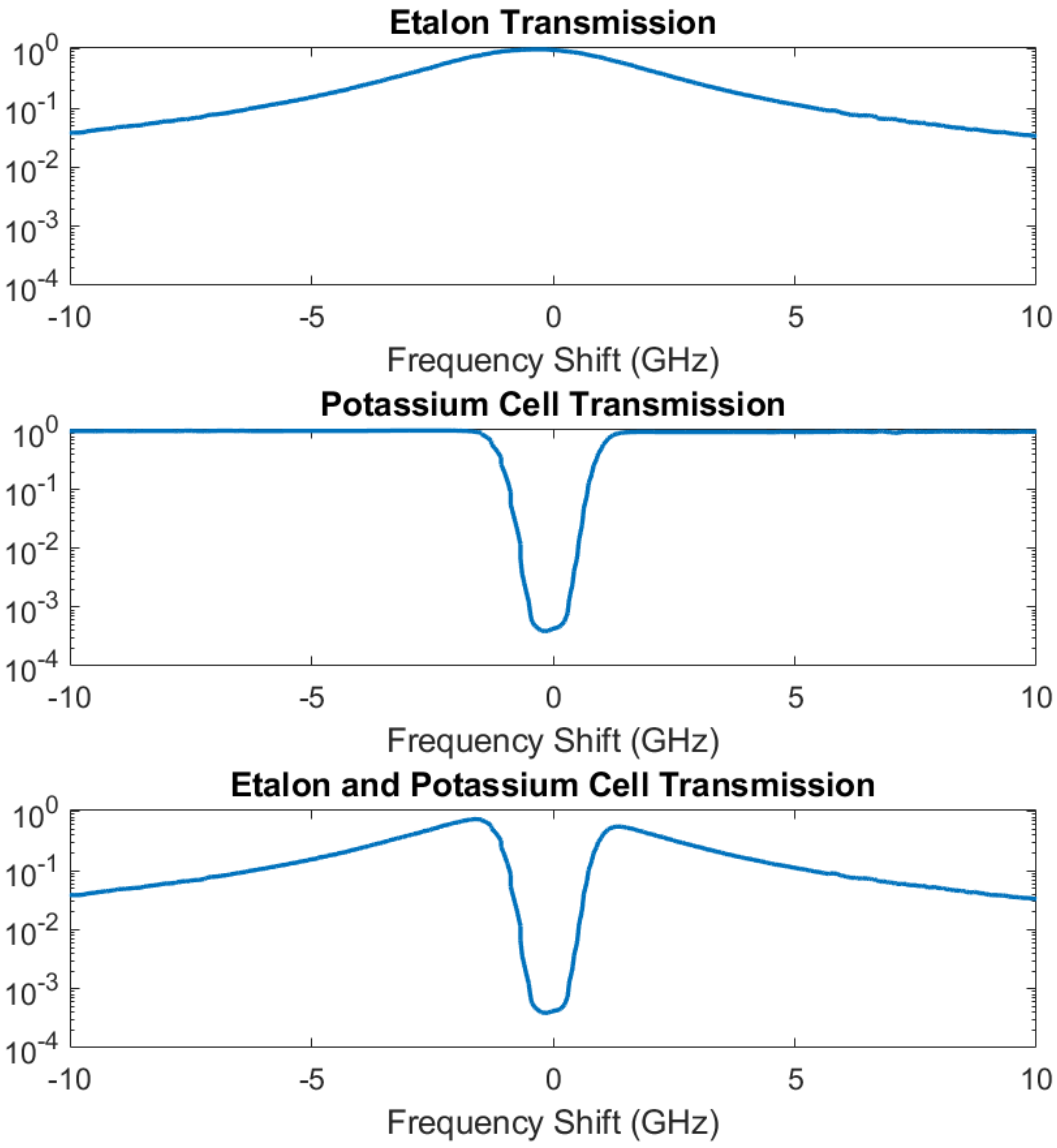


Figure 3.4: Top: Etalon transmission. Center: Potassium cell transmission. Bottom: Transmission through the etalon and potassium cell.

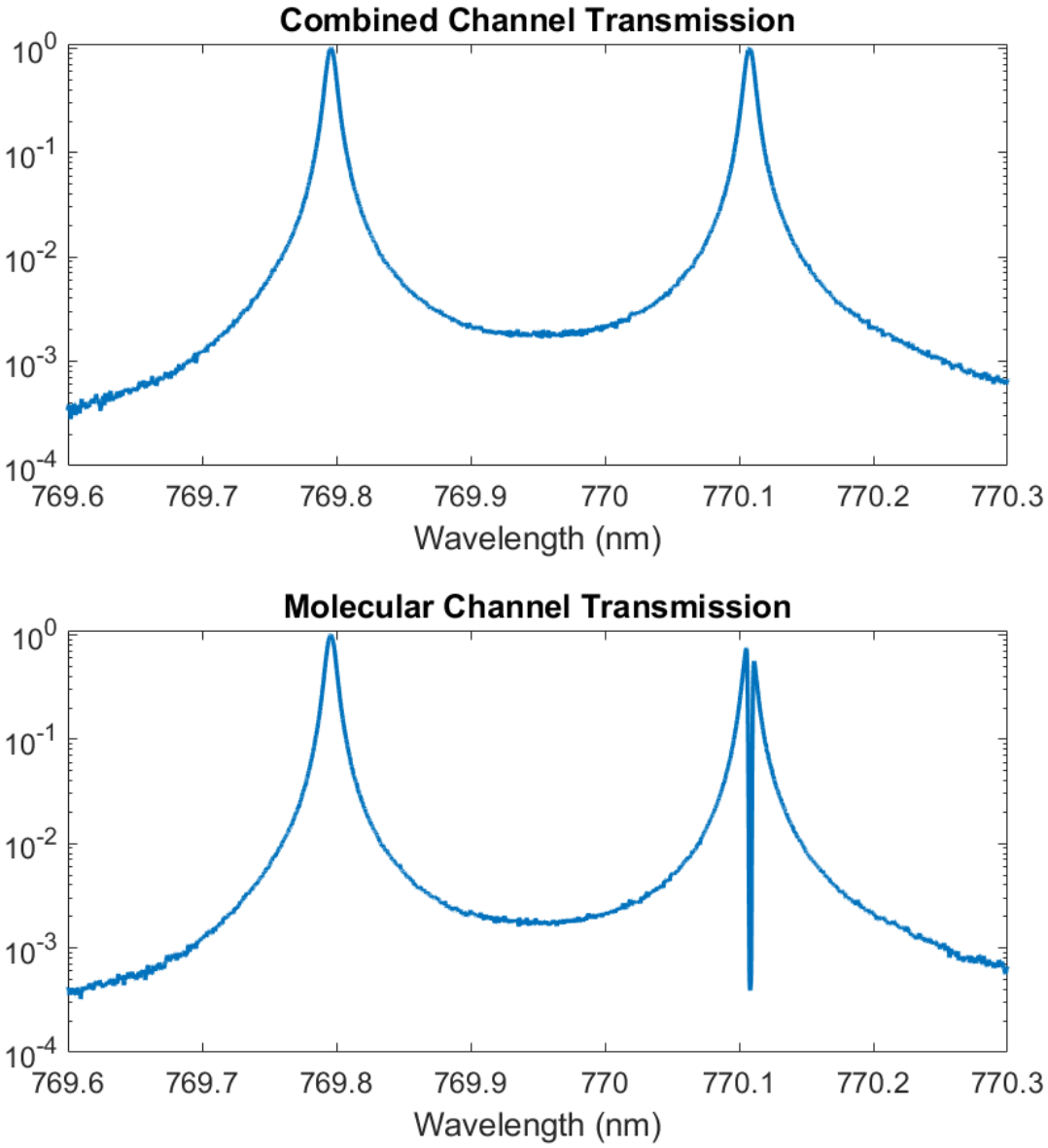


Figure 3.5: Top: Combined channel transmission. Bottom: Molecular channel transmission.

Avalanche photodiode (APD) modules (PerkinElmer; SPCM-AGR-13-FC) operating in Geiger, or single-photon counting, mode are used as the detectors. These detectors are similar to PIN junctions with an applied reverse bias voltage and an additional avalanche region. An incident photon is absorbed in the depletion region and creates an electron-hole pair. The applied electric field accelerates the electron and the hole. The electron enters the avalanche region, collides with other particles, and creates additional electron-hole pairs through impact ionization. The electric field accelerates these additional electrons and holes, and the electrons collide with more particles and create even more electron-hole pairs. This cascade of electron-hole generation is the avalanche effect. The gain across the avalanche region can be greater than 1000. These detectors count the number of avalanche events, and therefore the number of photons received for a time interval specified by the trigger pulse from the pulse generator. When photon count rates are high, the APDs fail to count every photon received as the semiconductor cannot build enough electrical energy for the avalanche effect to occur. Due to this, a correction factor is implemented for high photon counts. This correction factor will be discussed in Chapter 4.

Control System

A laser diode controller (Thorlabs; LDC8010) adjusts the currents for the lasers and the TWA. TECs (Thorlabs; TED8020) control the temperature of the lasers, the TWA, and the etalon. The lasers maintain a constant wavelength because of the precise laser cavity temperature control. By controlling the temperature of the laser cavity, the cavity length and, therefore, the wavelength that resonates can be adjusted through thermal expansion and contraction. A laser diode controller and TEC module (Wavelength Electronics; LDTC0520) control the TSOA current and temperature.

The pulse generator (Quantum Composer; 9530 Pulse Generator) triggers and provides synchronization for all switches and the TSOA. It also triggers the multi-channel scaler

(MCS) to record data, controlling the system's timing. A custom-made MCS designed for the MPD instruments records the number of photons collected by the APDs and the positions of the switches. The pulse generator triggers the MCS at the same rate as the TSOA. It records data from the APDs for a time interval of 250 ns. This bin interval of 250 ns corresponds to a bin height of 37.5 m. The photons collected during each bin come from spatial bins of 37.5 m with uncertainty due to a range resolution of 150 m. The MCS also demuxes the switches and integrates the data for 2-second intervals before sending the data to the computer. The wavemeter (Bristol; 671 Series) monitors the laser wavelength to a precision of .2 pm.

A LabVIEW program running on the computer controls the entire system. This program writes the MCS and the wavemeter information into a binary file and converts the binary file to a NetCDF file every 10 minutes. The LabVIEW program also runs a control loop with the wavemeter and the TECs to maintain constant operating wavelengths.

Additional Instrumentation

In addition to the lidar instrumentation, retrieval of the aerosol optical properties relies on surface temperature and pressure measurements. The Montana State University Electrical and Computer Engineering weather station operated by Dr. Joseph A. Shaw provides these measurements. Thermocouples are a secondary temperature monitoring method. A thermocouple module (Advantech; ADAM-6018) and type T thermocouples monitor the temperature of the potassium cell, the room in which the lidar operates, and the TSOA.

Eye Safety

For continuous and autonomous operation, the MPD must not threaten the community. For this reason, eye safety at the exit aperture of the telescope is a critical design feature. Furthermore, these instruments must comply with regulations from the Federal Aviation Administration to be practical. Since this lidar operates in the near-IR, the blink reflex does not protect the viewer, so the beam must be safe for continuous exposure. The maximum permissible exposure (MPE) for a single pulse between 1 ns and 18 μs for a wavelength range of 0.7-1.05 μm is:

$$MPE = 0.5 \cdot C_A \cdot 10^{-6} \frac{J}{cm^2} \tag{3.1}$$

Where C_A is a correction factor for the specific wavelength.

$$C_A = 10^{2 \cdot (\lambda - .7 \mu m)} \tag{3.2}$$

A pulsed laser with n pulses in a time interval has a correction factor applied to the MPE. There is no blink aversion protecting the viewer because the human eye cannot detect near-IR light, so the time interval is 10 seconds. At 7 kHz, this corresponds to 70,000 pulses.

$$MPE_{repeating\ pulses} = MPE \cdot n^{\frac{-1}{4}} \tag{3.3}$$

These calculations set the MPE in energy density of this lidar to be 42.43 $\frac{nJ}{cm^2}$, which corresponds to an average power density of 297 $\frac{\mu W}{cm^2}$. The measured power output of the lidar is 40 mW, and the pulse rate is 7 kHz, so the pulse energy is 7.142 $\frac{\mu J}{pulse}$. The output beam is ring-shaped with an inner diameter of 10 cm and an outer diameter of 20 cm. The beam has an area of about 235.6 cm^2. This area corresponds to an average energy density of 30.31 $\frac{nJ}{cm^2}$, which is less than the MPE with a factor of safety of 1.4.

Calibration

For HSRL measurements, the precise transmission spectra through both the combined and the molecular receiver channels must be measured. The calibration coefficients, ($C_{MM}(\nu)$, $C_{AM}(\nu)$, $C_{MC}(\nu)$, and $C_{AC}(\nu)$), and the different efficiencies between the receiver channels, ($\eta_C(\nu)$ and $\eta_M(\nu)$), must be known to find the aerosol backscatter coefficient (Eq. 2.22)). In order to measure these values, a calibration scan is performed.

The TSOA and one of the DBR lasers are turned off at the beginning of a calibration scan. The other DBR laser and the TWA continue to run in CW mode. The semiconductor material in the TSOA absorbs most of the laser light, allowing only a tiny fraction to pass through. The transmitted beam is blocked by a solid target placed between the last lens in the transmitting optics and the telescope shutter. The diffuse reflection from this target is scattered back into the receiver at the transmitted laser wavelength. The temperature of the DBR laser cavity is slowly changed to vary the wavelength of the transmitted light. The calibration program scans a temperature range several times and records the wavelength and photon counts at each step. The counts for a sample calibration scan are shown in Figure 3.6.

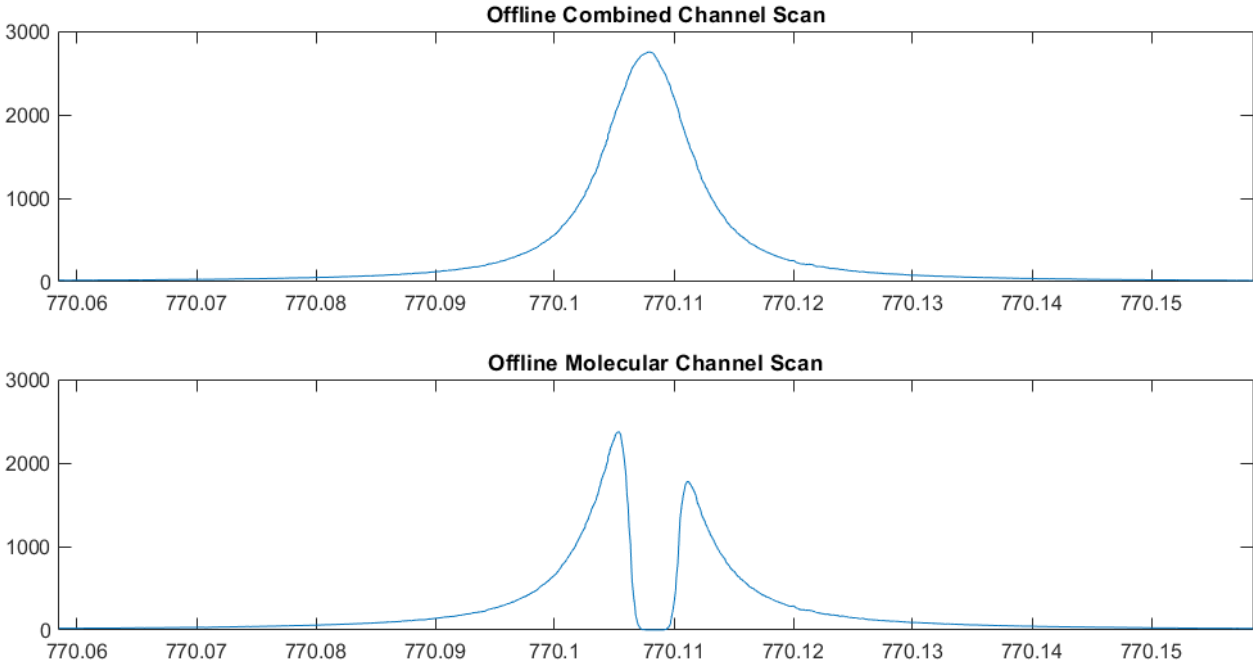


Figure 3.6: Combined and molecular receiver scan photon counts.

The efficiencies, $\eta_C(\nu)$ and $\eta_M(\nu)$, are normalized such that the sum of the efficiencies is equal to one.

$$\eta_C(\nu) \ + \eta_M(\nu) = 1 \tag{3.4}$$

The ratio of the efficiencies is equal to the ratio of the photon counts recorded by each detector, and this ratio can be found by dividing the two curves, excluding the region of potassium cell absorption.

$$\frac{\eta_C(\nu)}{\eta_M(\nu)} = \frac{N_C(\nu)}{N_M(\nu)} \tag{3.5}$$

Dividing Equation 3.4 by $\eta_M(\nu)$ and combining it with Equation 3.5 yields:

$$\left(\frac{N_C\,(\nu)}{N_M\,(\nu)} - 1\right)^{-1} = \eta_M(\nu) \tag{3.6}$$

Once the efficiencies are found, the transmission spectrum for each receiver channel can be determined. The combined channel is normalized to its maximum transmission by dividing the counts by the maximum counts recorded. The molecular channel transmission curve is normalized by dividing the maximum recorded counts in the molecular channel by the maximum recorded counts in the combined channel multiplied by the molecular efficiency and divided by the combined efficiency. The normalized transmission of the combined and molecular receivers compared to the molecular and aerosol scattering spectra is shown in Figure 3.7. Standard atmospheric conditions (288.15 K, 101.325 kPa) were used to generate the molecular scattering spectra in these plots. The peak of the transmission curve does not match the center wavelength because the etalon was optimized for the online DIAL wavelength.

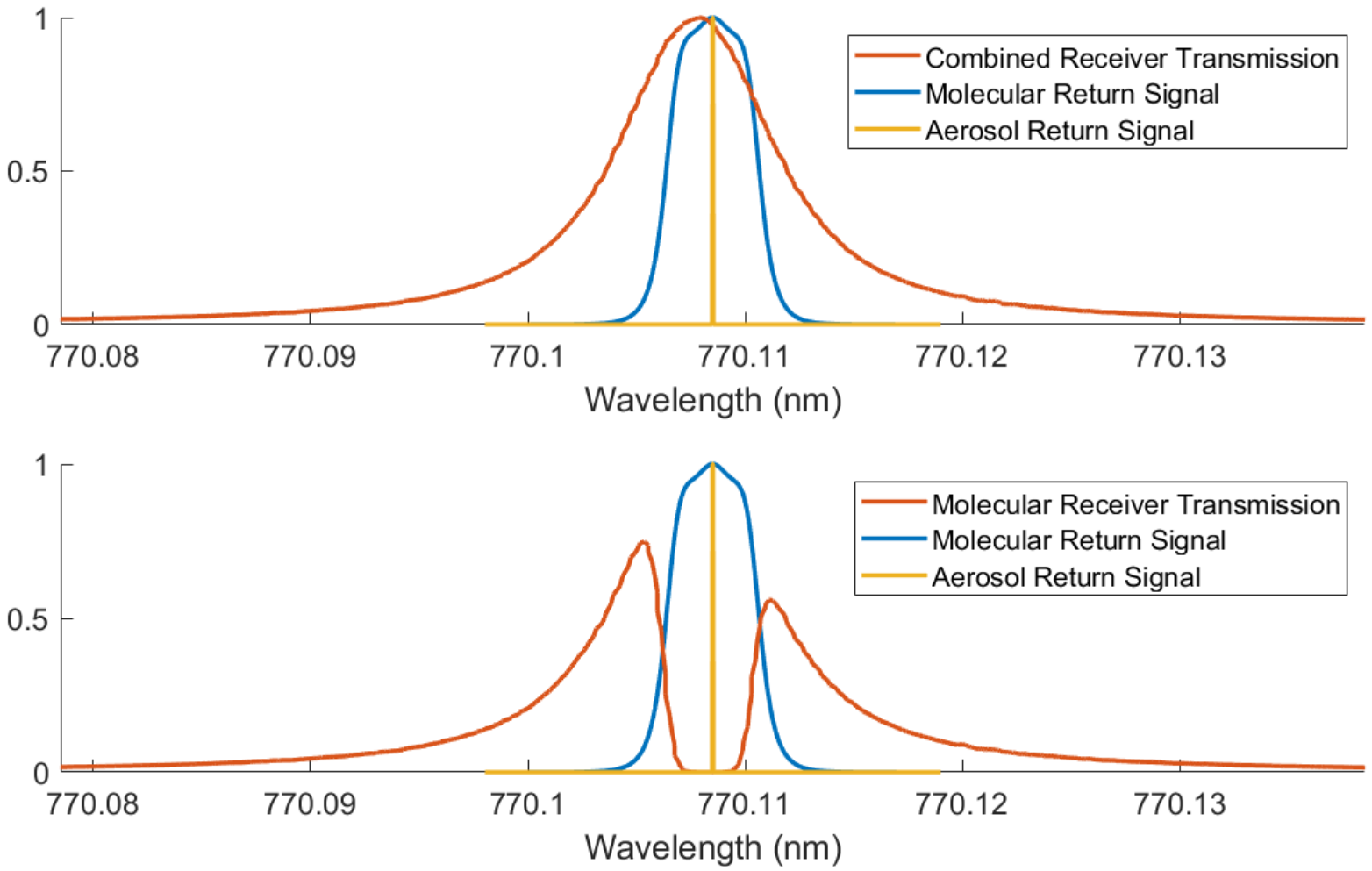


Figure 3.7: Top: Combined channel transmission and return signal spectra. Bottom: Molecular channel transmission and return signal spectra.

The calibration coefficients can be determined once the transmission through the receiver channels is calculated for every wavelength. The calibration coefficients associated with aerosol scattering, $C_{AM}(\nu)$ and $C_{AC}(\nu)$, are equal to the transmission through the receiver channels at 770.1085 nm. The calibration coefficients associated with molecular scattering depend on the spectrum of the scattered light, which depends on the temperature and pressure of the scattering molecules. A two-dimensional table is generated for C_{MM} and C_{MC} so that these coefficients can be determined for a given temperature and pressure. These calibration coefficients are determined for temperatures from 170 to 320 K and pressures from 10 to 100 kPa with step sizes of 1 K and 1 kPa. C_{MM} and C_{MC} are interpolated from this table for each bin in the backscatter retrieval. Finally, the calibration coefficients are normalized to the value of C_{AC} so that $C_{AC} = 1$. This calibration scan should be performed regularly so that changes to the receiver efficiencies can be detected.

The calibration coefficient associated with aerosol scattering through the potassium cell is approximately $C_{AM} = 0.004$. The calibration coefficient associated with molecular scattering through the potassium cell ranges from $C_{MM}(T, P) = 0.05 - 0.14$, and the calibration coefficient associated with the molecular scattering through the combined channel of the receiver ranges from $C_{MC}(T, P) = 0.91 - 0.95$ depending on the temperature and pressure of the scattering molecules. Because of the high dependency of C_{MM} on atmospheric conditions, a poor estimation of the atmospheric temperature and pressure will lead to high errors in the retrieved aerosol optical properties. C_{MM} and C_{MC} vs the temperature and pressure of the scattering molecules are displayed in Figures 3.8 and 3.9.

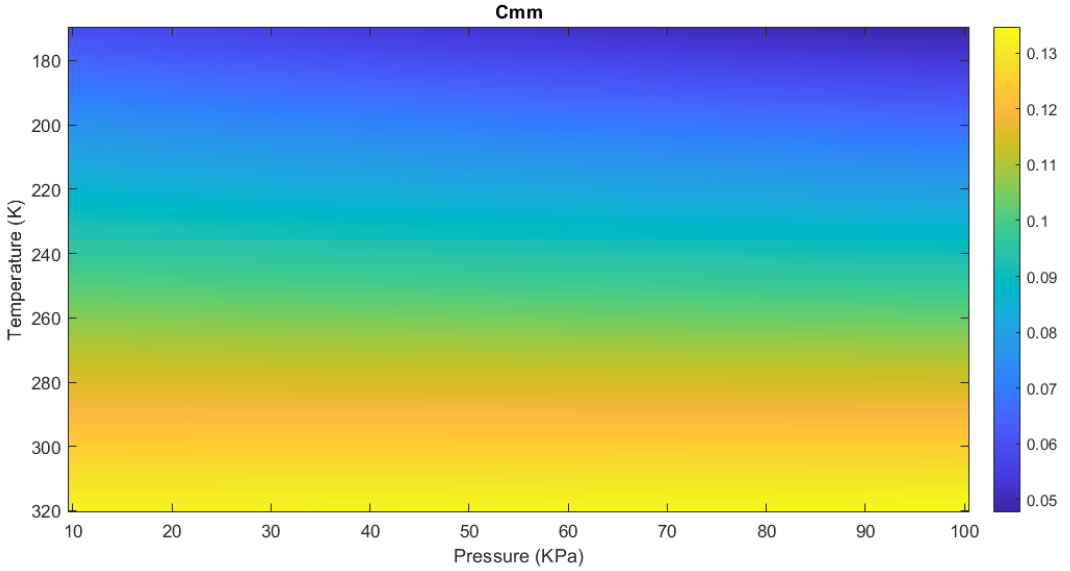


Figure 3.8: C_{MM} vs temperature and pressure.

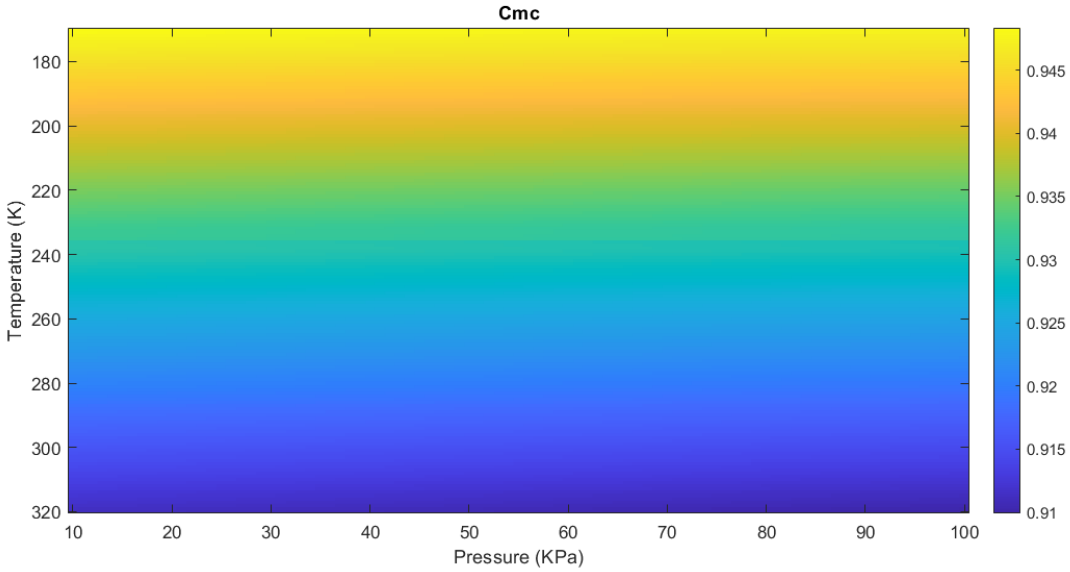


Figure 3.9: C_{MC} vs temperature and pressure.

AEROSOL OPTICAL PROPERTY RETRIEVAL

Overview

The following section describes how to obtain the aerosol backscatter coefficient and the backscatter ratio from the NetCDF files generated by the MPD. My contributions to the aerosol optical property retrieval were writing the MATLAB code, which extracts the aerosol optical properties from the raw lidar profiles, and developing a more accurate method of modeling the molecular backscatter coefficient.

Data Processing

Pre-Processing

The NetCDF files generated by the LabVIEW program contain lidar profiles integrated for 2 seconds for each laser and each channel. The first step in processing these lidar profiles is to manipulate the profiles with several correction factors. The first correction factor deals with nonlinear effects in the APDs. For high photon count rates, the applied reverse bias voltage cannot generate enough electrical charge for the avalanche effect to occur for each electron-hole pair generated, and this causes the APD to fail to count each incident photon. The dead time of the detector is the time required for the detector to build up enough electric charge for the avalanche effect to occur. A correction factor based on the count rate of each bin is applied to correct for these nonlinear effects. The correction factor is:

$$CF = \frac{1}{1 - (t_d \cdot CR)} \tag{4.1}$$

The parameter t_d is the dead time, and CR is the count rate. A typical dead time of 60 ns was used for determining the correction factor. The correction factor as a function of the count rate is shown in Figure 4.1. The detectors operate in the linear region with

correction factors up to about 5% except for cloudy data. Figure 4.2 demonstrates the difference between a typical lidar profile and the same lidar profile with the correction factor applied. In this lidar profile, the peak at 7000 m is a thin cloud.

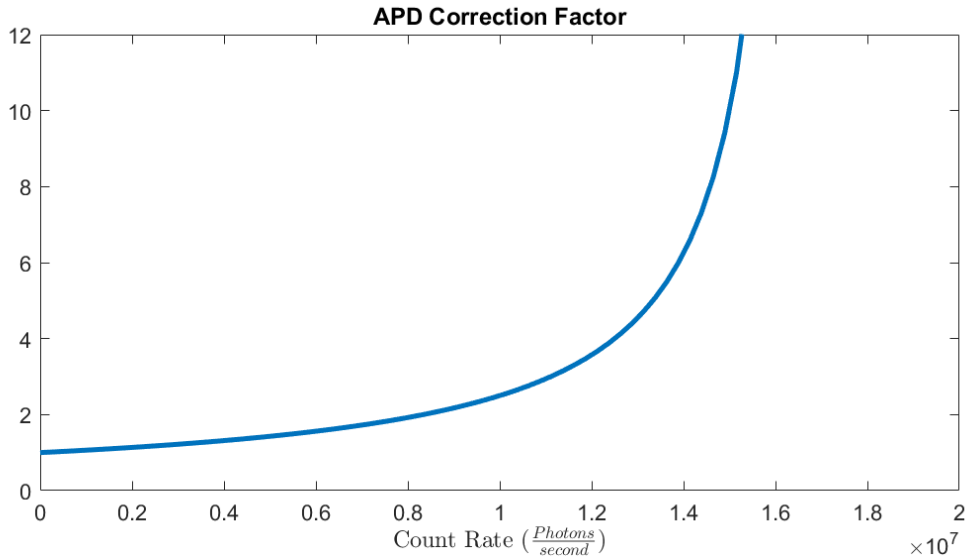


Figure 4.1: APD correction factor.

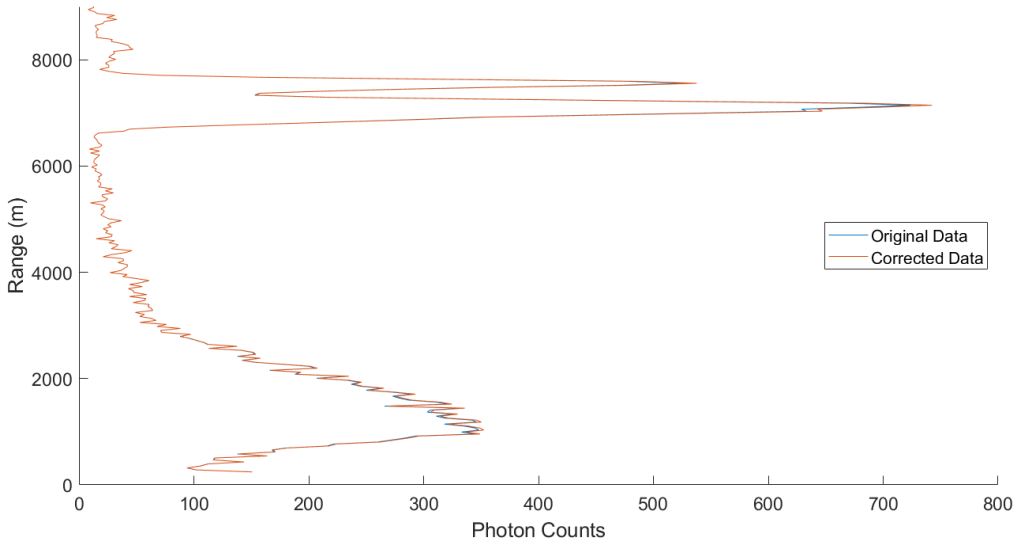


Figure 4.2: Data with the APD correction factor vs original data.

The next step in processing the data is to subtract the background counts from the lidar profiles. This removes the photons from background radiance. The background counts are the average photon counts for the last 3 km, 18 to 21 km. At this range, the lidar signal is entirely attenuated, and the signal is nonexistent. The signal does not change from about 12 to 21 km, which means that the high altitude counts are a reasonable estimation of counts caused by background radiance. If there were counts from the lidar at this range, the bins at 12 km would have more photons than the bins at 21 km because counts decrease with range squared. This background count value is subtracted from the entire lidar profile, leaving only the lidar return signal. After removing the background, the lidar profile is trimmed to a maximum range of 7.5 km to reduce computing time. The signal-to-noise ratios (SNRs) from altitudes higher than this are too low for valuable data. Additionally, because the laser pulse corrupts low-range data, all data below 250 m is deleted from the data set.

Once the background-subtracted counts are trimmed, the difference in the overlap functions for the molecular and combined receiver channels is corrected. The HSRL retrieval does not require overlap or range corrections like other lidar methods, but it requires a correction for the difference between the overlap functions of the combined and molecular receiver channels. This correction factor can be thought of as a range-dependent efficiency correction between the two channels. The online DIAL laser signal is used for determining the overlap correction as this wavelength experiences no attenuation through the potassium cell. The lidar profiles for the online wavelength in the molecular and combined channels should have the same shape with different magnitudes. Measured profiles of the online laser through the combined and molecular receiver channels do not have the same shape, caused by a difference in overlap functions between the channels.

The overlap correction factor, $O(r)$, is calculated by dividing the online molecular counts by the online combined counts then averaging this ratio for a day as a function of range. The overlap correction factor is normalized to its value beyond full overlap (1.5 km). The offline

molecular counts are then multiplied by this difference in the overlap functions. Figure 4.3 shows the difference in overlap functions for one day. The overlap correction factor can be reduced by aligning the lidar more precisely and is nearly equal to 1 for the entire range, but a small, consistent error due to differences in overlap functions from 250 m to 1.5 km has persisted throughout retrievals. Equations 4.2 and 4.3 describe the overlap function and the corrected molecular counts mathematically.

$$O(r) = \frac{N^{online}_{combined}(r)}{N^{online}_{molecular}(r)} \tag{4.2}$$

$$N^{offline}_{molecular,corrected}(r) = O(r) \cdot N^{offline}_{molecular} \tag{4.3}$$

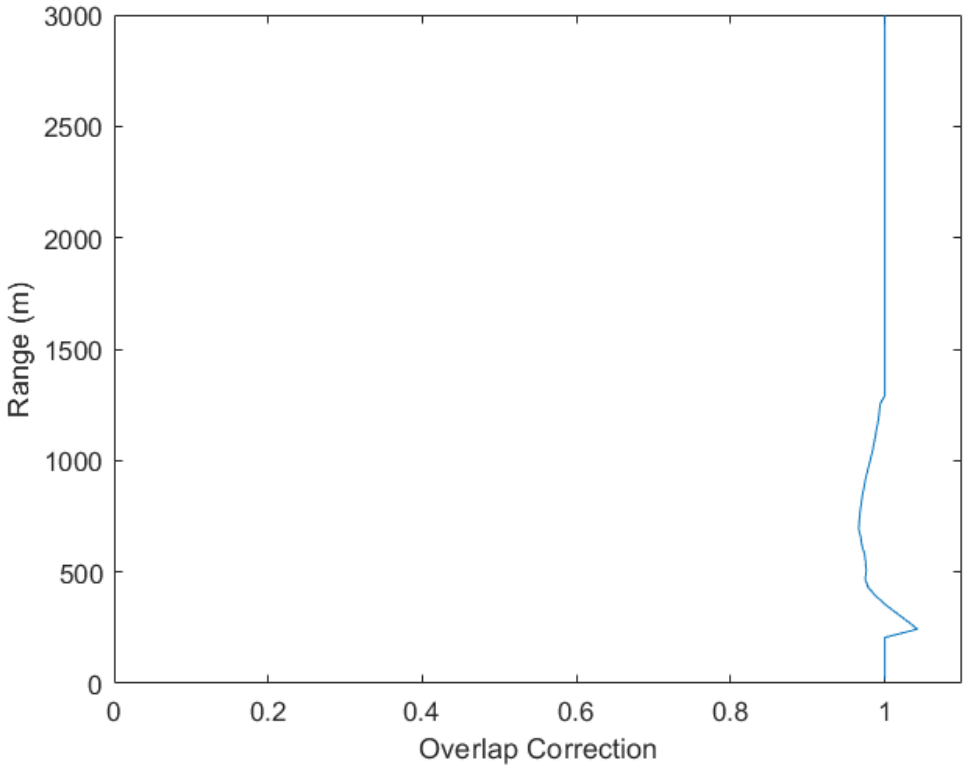


Figure 4.3: Overlap correction factor.

Once the overlap correction is applied, the counts are integrated for the desired time interval. An interval of 5 minutes was used for the retrievals in this thesis. This interval was chosen because it provides enough counts for the maximum range of the lidar to be greater than the range of a typical capping inversion and provides a high enough time resolution to capture the variability of the PBLH. All 150 profiles from 5 minutes are summed for each receiver channel to generate an integrated lidar profile. The interval could be reduced to provide better temporal resolution at the cost of lower counts and, therefore, a lower maximum range.

Atmospheric Modeling

Accurate models of the temperature and pressure of the atmosphere are necessary for high-quality retrievals of the aerosol optical properties. At higher altitudes, a poor temperature model leads to a high degree of error in the retrieved aerosol backscatter coefficient. Using the dry adiabatic lapse rate of 9.8 $\frac{K}{km}$ resulted in underestimating the temperature at high altitudes, and a standard moist adiabatic lapse rate of 5 $\frac{K}{km}$ led to overestimation. For this reason, the lapse rate was modeled using a linear fit from radiosonde data. From 43 radiosondes launched between August 2020 and June 2021, the mean lapse rate is 7 $\frac{K}{km}$. The measured surface temperature provides the surface condition for the atmospheric temperature model. The temperature was assumed to decrease linearly with the modeled lapse rate of 7 $\frac{K}{km}$. Where T_s is the measured surface temperature, γ is the lapse rate in $\frac{K}{m}$, and r is the range in m, the modeled atmospheric temperature is given by Equations 4.4 and 4.5.

$$T(r) = T_s - \gamma \cdot r \tag{4.4}$$

$$T(r) = T_s - 0.007\frac{K}{m} \cdot r \tag{4.5}$$

The pressure is determined by using the exponential atmosphere model, which assumes an ideal gas. Air containing water vapor does not behave as an ideal gas, but the exponential atmosphere is the best approximation without knowing the number density of water vapor. The surface pressure is the boundary condition of the pressure model. Where P_s is the surface pressure, m_{air} is the molar mass of air in $\frac{kg}{mol}$, g is the gravitational constant in $\frac{m}{s^2}$, and R is the universal gas constant in $\frac{J}{mol \cdot K}$, pressure is given by Equations 4.6 and 4.7.

$$P(r) = P_s \cdot (\frac{T_s}{T(r)})^{\frac{m_{air} \cdot g}{R \cdot \gamma}} \tag{4.6}$$

$$P(r) = \ P_s \cdot (\frac{T_s}{T(r)})^{-4.8783} \tag{4.7}$$

Aerosol Optical Property Retrieval

The aerosol backscatter coefficient and the backscatter ratio are found using the background-subtracted lidar counts, the modeled atmospheric temperature, the modeled atmospheric pressure, the measured receiver efficiencies, and the measured calibration coefficients. Because C_{MM} and C_{MC} are dependent on the temperature and pressure of the scattering molecules, different values for these calibration coefficients are found for each bin. A 2-dimensional table with values for the calibration coefficients for a range of temperatures and pressures is created from calibration data. The calibration constants in the aerosol optical property retrievals are interpolated from this table.

The molecular backscatter coefficient is generated using the modeled temperature, modeled pressure, and the theoretical molecular backscatter coefficient for a standard atmosphere described by Equation 4.8.

$$\beta_{mol}(r) = 9.94266 \cdot 10^{-7} \cdot (\frac{P(r)}{T(r)}) \ [m^{-1} \cdot sr^{-1}] \tag{4.8}$$

The aerosol backscatter coefficient is then found using Equation 4.9, where η_C, η_M, C_{AC}, and C_{AM} are constant values, and β_{mol}, N_M, N_C, C_{MC}, and C_{MM} are matrices with a unique value for each bin.

$$\beta_{aer} = \beta_{mol} \cdot [\frac{N_M \cdot \eta_C \cdot C_{MC} - N_C \cdot \eta_M \cdot C_{MM}}{N_C \cdot \eta_M \cdot C_{AM} - N_M \cdot \eta_C \cdot C_{AC}}] \tag{4.9}$$

The backscatter ratio is then found using Equation 4.10.

$$BR = \frac{\beta_{mol} + \beta_{aer}}{\beta_{mol}} \tag{4.10}$$

These data sets often have issues because of clouds in the data and low SNRs. Certain regions of the data are removed using masks to deal with these issues. After the data is masked, the data is smoothed in time. The aerosol optical properties must be found before smoothing, as smoothing in time reduces the temporal resolution of the data.

Masking

Cloud Mask The cloud masking technique utilizes a Sobel operator for 2-dimensional edge detection and a variance method where the variance of a 5x5 cell is determined, and that value is assigned to the center bin in the 5x5 cell. This method is similar to that used by Binietoglu et al. [61]. These operations were applied to the offline combined counts as the molecular channel would filter out most of the aerosol scattering of clouds. The edges of clouds have a high gradient as counts sharply rise on the edge of a cloud. The variance of a 5x5 cell is high throughout the cloud as the lidar signal is large and rapidly varying within the cloud. Once these features are extracted from the lidar data, any cells with values higher than a certain threshold for the Sobel operator and the variance method are marked by the cloud mask and removed from the data set. For the aerosol optical property retrieval, a Sobel threshold of 10^7 was used, and a variance threshold of 10^9 was used. Data above the

clouds is also removed because the assumption of negligible multiple scattering from these bins is no longer valid.

Signal-to-Noise Ratio Mask An SNR mask is used to avoid excessive noise in the lidar retrievals. The noise associated with a photon-counting lidar is shot noise governed by Poisson statistics, where the noise is the square root of the background-subtracted photon counts N. The minimum SNR is 3; all bins with lower SNRs are masked and removed from the data set. This threshold was chosen through trial and error; bins with SNRs below 3 tend to be too noisy to be valuable.

$$SNR = \sqrt{N} \leq 3 \tag{4.11}$$

Smoothing

Noisy data is challenging to use in the PBLH retrieval algorithm. For accurate PBLH retrievals, it is better to smooth the data in time and sacrifice temporal resolution than to apply the PBLH retrieval method to noisy data. This smoothing also makes the data easier to read. The masked aerosol optical properties are smoothed with a Gaussian window with a width of 1 hour. The data is not smoothed in space, as the spatial resolution is critical for determining the PBLH.

Data Examples

The data collection period for this thesis is from March 1 to July 31, 2021. During this period, the combined O2-DIAL and HSRL operated continuously except for short periods when the instrument was calibrated or realigned and from June 10-June 14, when the computer experienced software issues. Three periods of aerosol optical property retrievals are shown: April 1-April 3, June 6-June 8, and July 17-July 19.

April 1-April 3, 2021

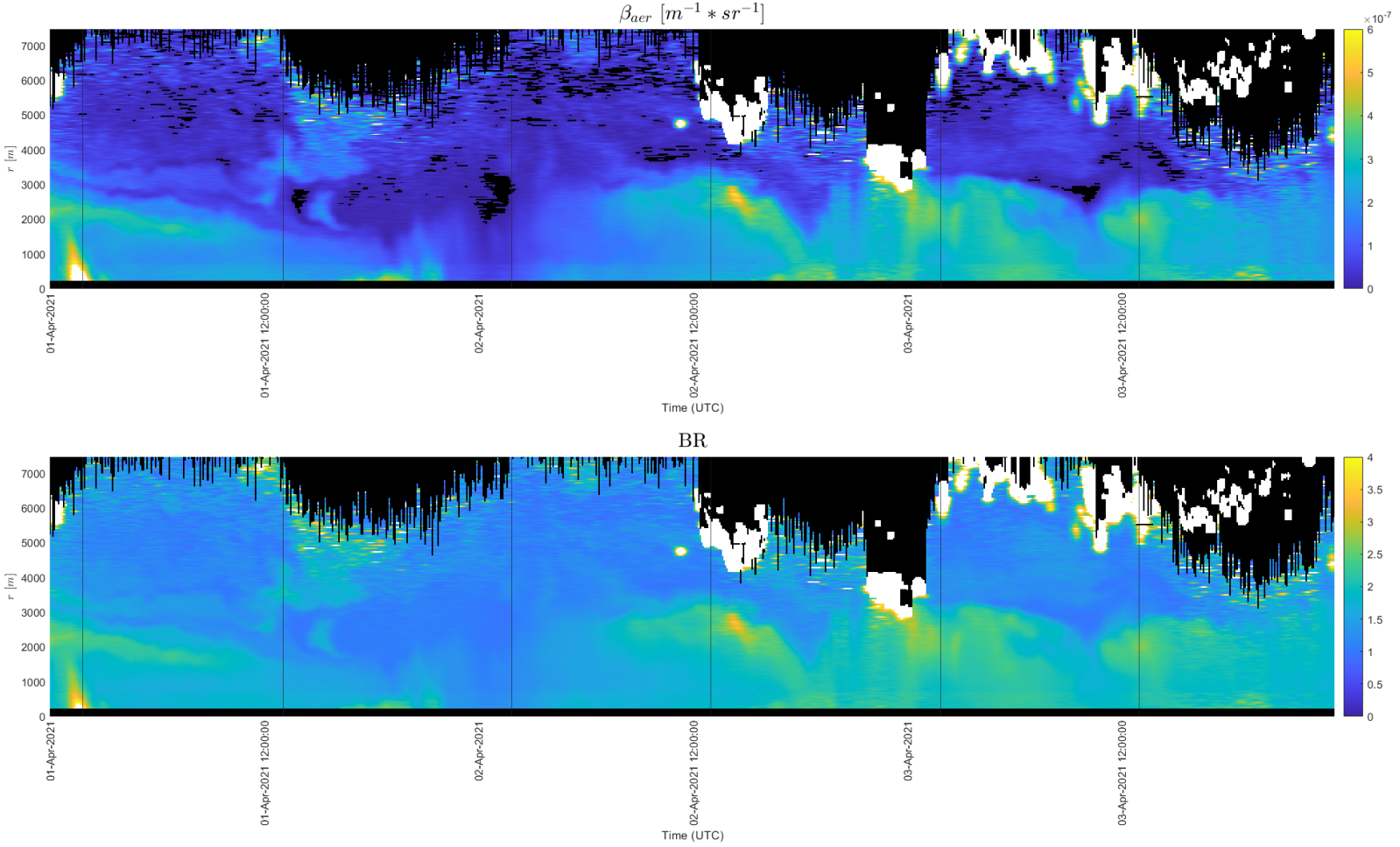


Figure 4.4: April 1-April 3, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio.

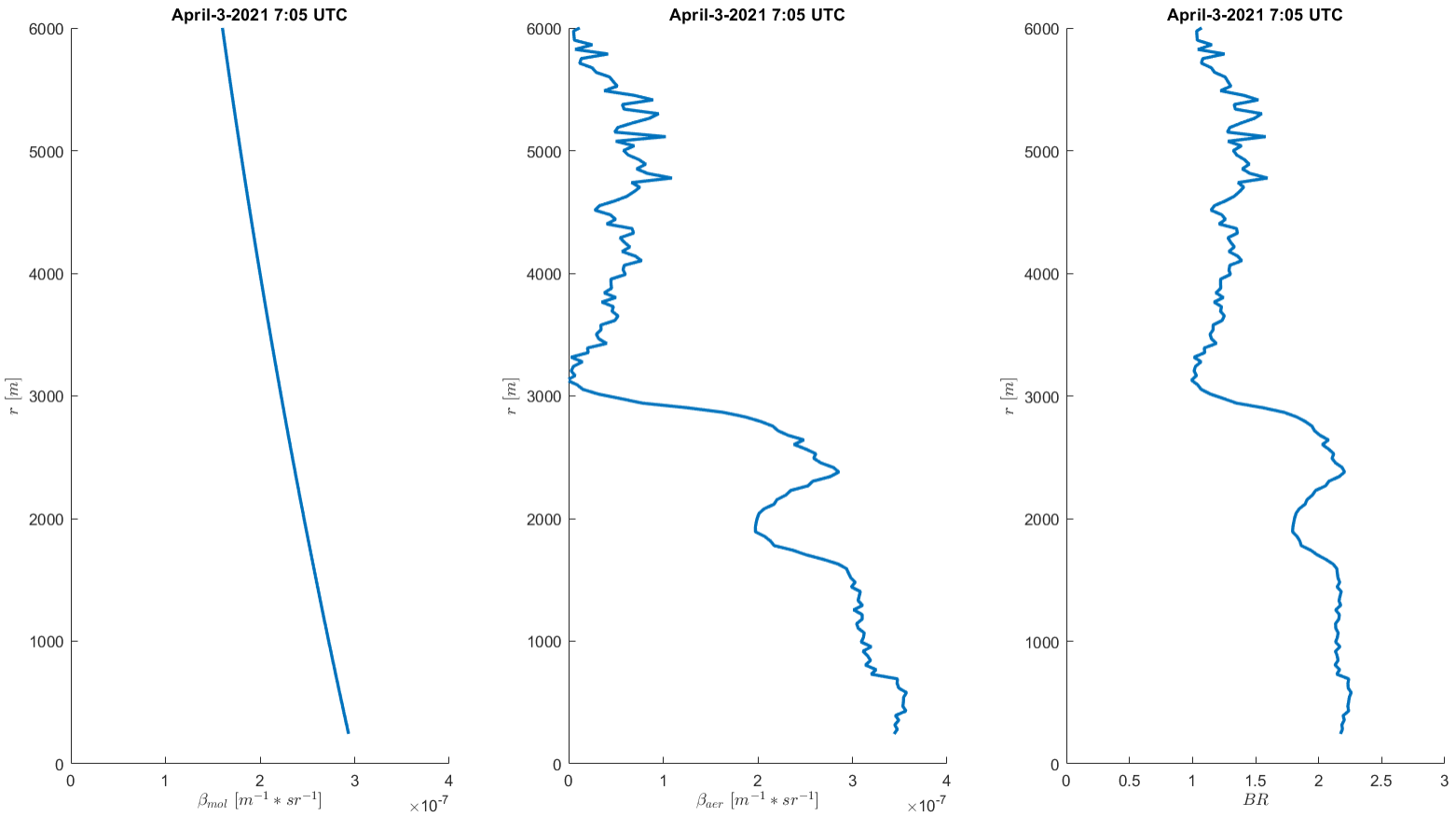


Figure 4.5: April 3, 2021, 7:05 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio.

The data collected in early April 2021 is the best clear sky data from the collection period. During this period, the lidar was well aligned, the TSOA was producing 40 mW of average power, and the skies were clear with low aerosol concentrations and few clouds. The vertical lines in Figure 4.4 represent the sunrise and sunset, the white regions represent data that the cloud mask has removed, and the black regions represent data that the SNR mask has removed. The black spots in the β_{aer} plot below the SNR mask represent regions where the measured β_{aer} is negative, which is impossible. In reality, there are very few aerosols in this region, and the actual aerosol backscatter coefficient is close to zero. Slight errors in the estimation of the β_{mol} due to a difference in the modeled atmospheric temperature and actual temperature lead to this error. The measured backscatter ratio for these regions is less than 1, which is also impossible. Any value of the backscatter ratio less than 1 should be set to 1 in the DIAL equation for the temperature retrievals. During this period, the instrument collected data with a maximum range of 7 km at night and 5 km during daytime operation. Figure 4.5 represents a vertical profile of the atmospheric optical properties at a single point in time on April 3. The steep gradient at 3000 m is the steep aerosol gradient associated with the capping inversion. Below the capping inversion, the air is full of aerosols and has backscatter ratios between 2 and 3. There are few aerosols above the capping inversion in the free troposphere, and the backscatter ratio is close to 1.

June 6-June 8, 2021

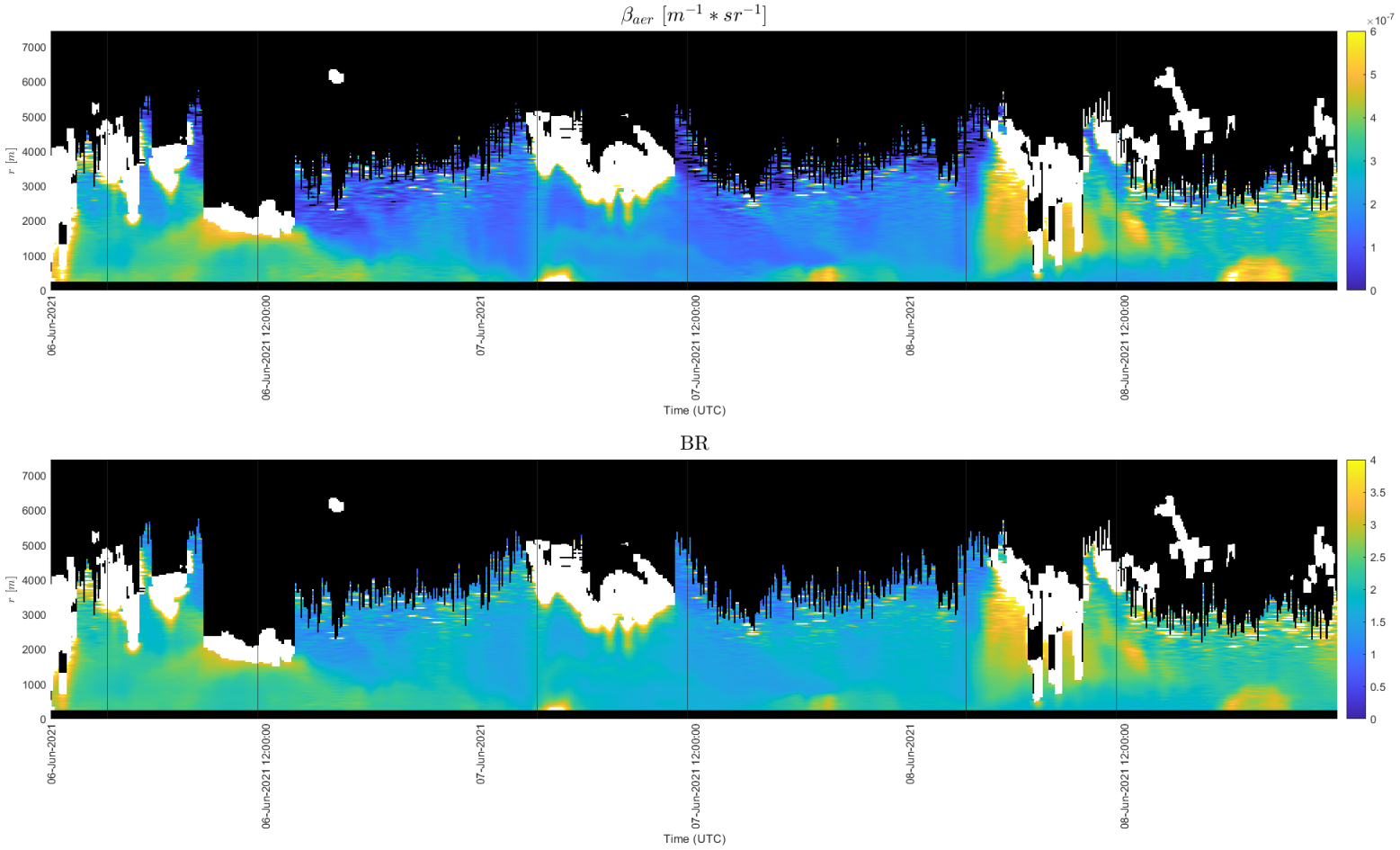


Figure 4.6: June 6-June 8, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio.

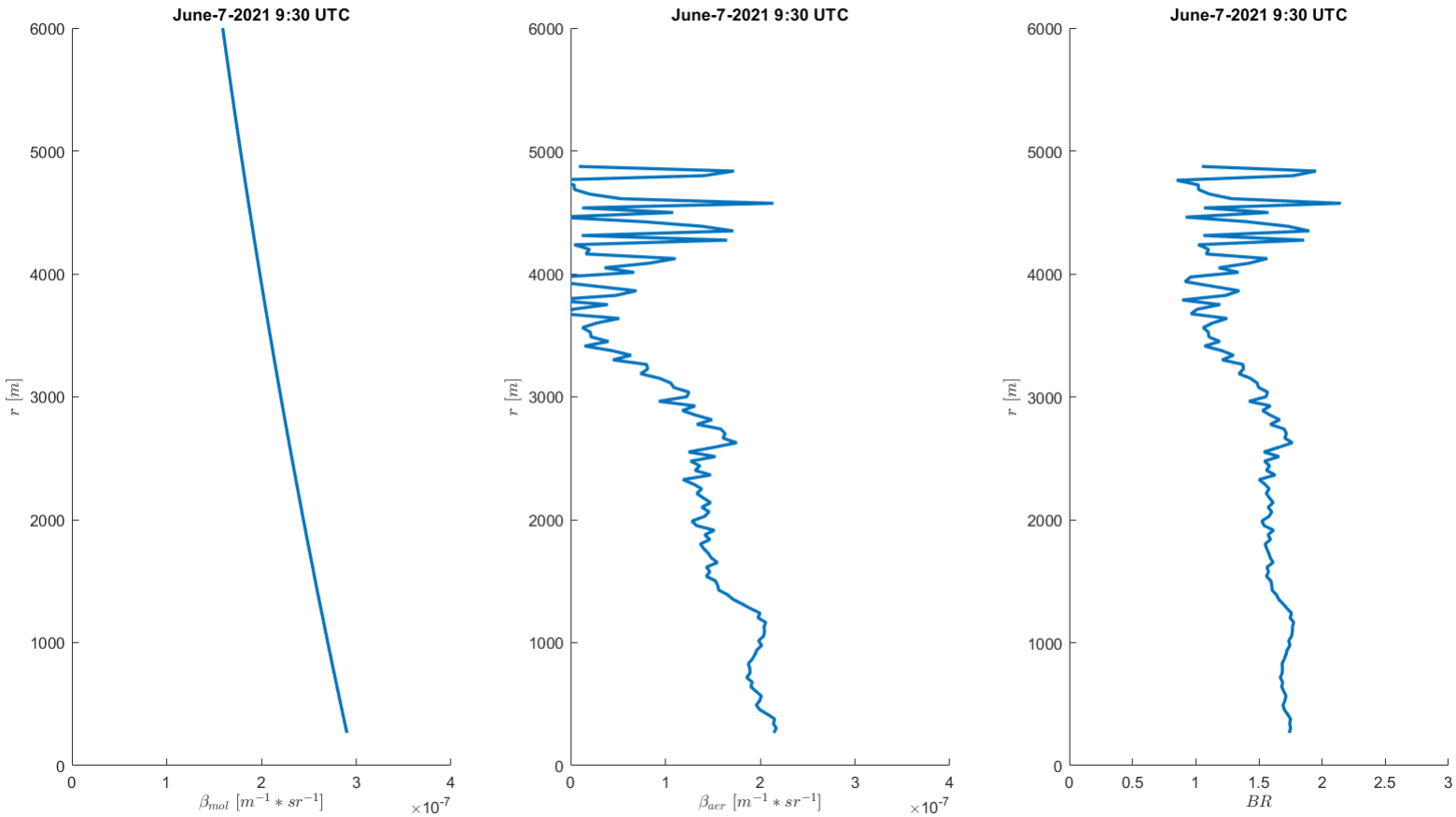


Figure 4.7: June 7, 2021, 9:30 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio.

The data collected in early June 2021 is lower quality than the data collected in early April 2021. During this period, the SOA was performing poorly, and the TSOA was not perfectly aligned. The power output from the TSOA was about 30 mW. This data is a good representation of the quality of data produced when the instrument is poorly aligned or the optical power is lower than the maximum. Like in Figure 4.4, the vertical lines in Figure 4.6 represent sunrises and sunsets, the white regions represent sections of the data removed by the cloud mask, and the black regions represent data removed by the SNR mask. Figure 4.7 displays the atmospheric optical properties for a single point in time. The noise above 3500 m in the aerosol backscatter coefficient and the backscatter ratio is typical of poor-quality data. The maximum range of the lidar during early June was about 5 km at night and 3.5 km in the daytime. Data like this should be avoided, and frequent realignment and monitoring are needed to prevent these results.

July 17-July 19, 2021

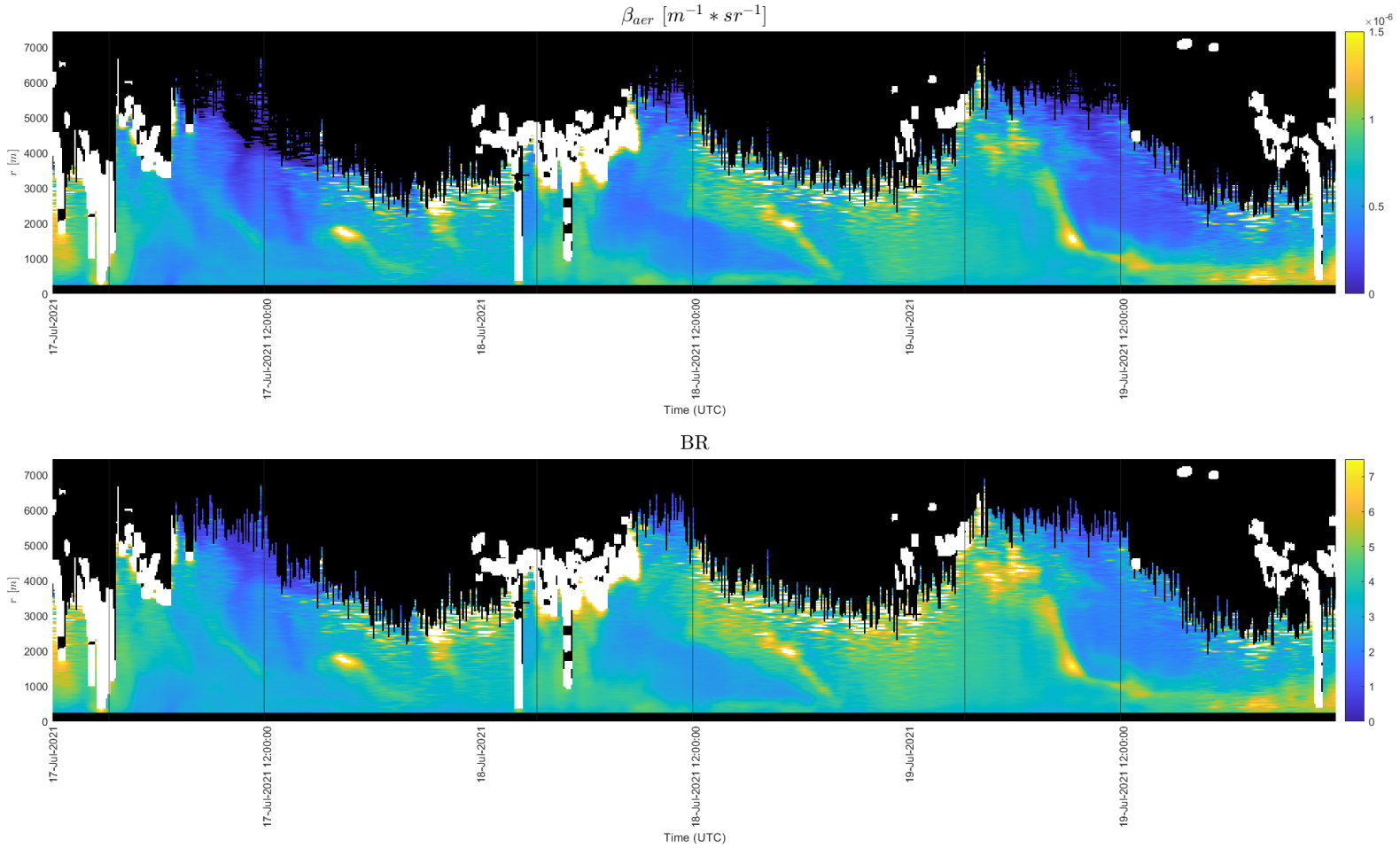


Figure 4.8: July 17-July 19, 2021; Top: Aerosol backscatter coefficient. Bottom: Backscatter ratio.

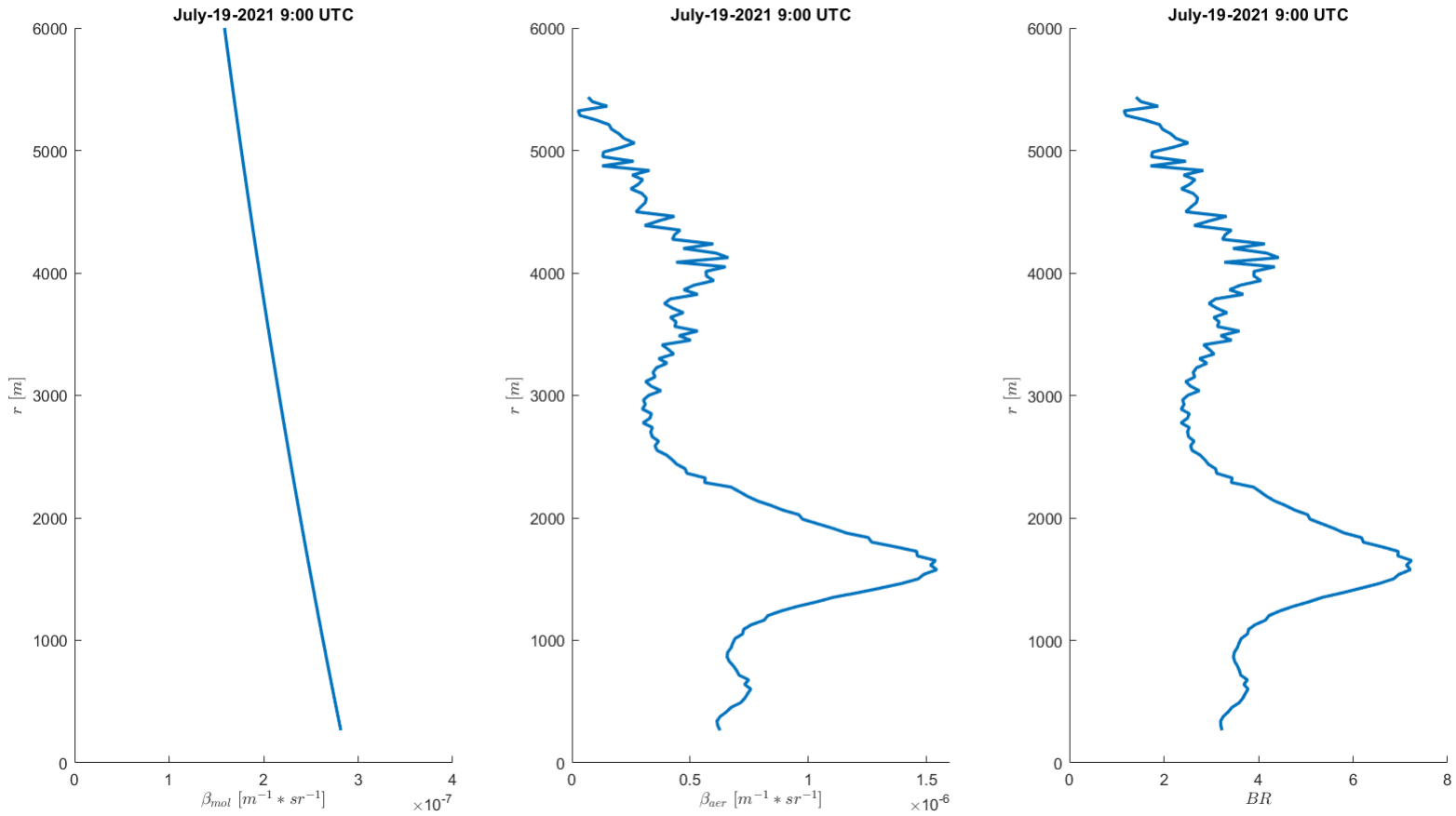


Figure 4.9: July 19th, 2021, 9:00 UTC; Left: Molecular backscatter coefficient. Center: Aerosol backscatter coefficient. Right: Backscatter ratio.

The data from mid-July 2021 is characterized by dense forest fire smoke. The axes of the color bars in Figure 4.8 are scaled to show much higher aerosol backscatter coefficients and backscatter ratios than Figures 4.4 and 4.6. Like in Figures 4.4 and 4.6, the vertical lines in Figure 4.8 are at sunrises and sunsets, the white regions represent data removed by the cloud mask, and the black regions represent data removed by the SNR mask. During this period, the lidar was well aligned, but the SOA was performing poorly. The output from the TSOA was about 35 mW. The lower maximum range is due to a high degree of attenuation through the smoke. Additionally, the thick smoke causes a high degree of multiple scattering, so the assumption of negligible multiple scattering might be invalid. The maximum range for the lidar in this period is about 5 km at night and 3.5 km during the daytime. Figure 4.9 displays the atmospheric optical properties for a single point in time on July 19. The aerosol concentrations are much higher than the previously displayed vertical profiles, and the scales have been changed accordingly. The aerosol backscatter coefficient and the backscatter ratio show two different residual layers. During periods with high concentrations of forest fire smoke, multiple residual layers form instead of the typical diurnal cycle with only one residual layer. Another issue with data containing high smoke concentrations is the tendency for forest fire smoke to cause a strong nocturnal inversion. The black carbon aerosols in smoke absorb IR radiation from surface radiative cooling, which creates a strong greenhouse effect, and in turn, creates a strong nocturnal inversion [62]. This inversion causes the atmospheric temperature at high altitudes to be consistently above the temperature modeled by a constant lapse rate model. This disparity between the predicted and actual temperatures leads to unrealistic negative measured aerosol backscatter coefficients at high altitudes.

Future Work

There are several potential directions for future work in the development of the HSRL and the MPD instruments. The next addition to the prototype MPD instrument at Montana State University will be the addition of the WV-DIAL for humidity measurements. Several of the assumptions in modeling the atmosphere, including the β_{mol}, relied on the assumption of an ideal gas. Moist air is not an ideal gas, and accounting for the humidity of air would increase the accuracy of modeling the atmospheric temperature, pressure, and molecular backscatter coefficient, which would, in turn, increase the accuracy of the aerosol optical property measurement by the HSRL. The humidity measurement is also necessary for accuracy in the atmospheric temperature retrieval using the DIAL technique.

Once the O2-DIAL tropospheric temperature measurement is validated, an iterative solution method could improve the accuracy of the MPD products. The modeled atmosphere could be used to find an initial backscatter ratio using the HSRL retrievals. This backscatter ratio would then be used in the DIAL equation for temperature retrievals, which would generate a measured temperature profile. This measured temperature profile could then be used to model the atmosphere, and more accurate measurements of the aerosol optical properties could be obtained. After a few iterations of this loop, the aerosol optical properties, and the temperature profiles should converge to more accurate values than the initial measurement. This iterative solution method has not yet been implemented because the temperature retrieval method by the O2-DIAL has not yet been fully validated.

Several issues limit the performance of the combined O2-DIAL and HSRL that we are actively working to resolve. The HSRL as a stand-alone system is highly sensitive to the modeling of the atmosphere. If the assumed lapse rate differs significantly from the actual lapse rate, then the atmospheric model is inaccurate, causing errors in the HSRL retrievals. These incorrect temperatures are an issue, especially in the case of nighttime temperature

inversions, because they cause values of the β_{aer} to be less than 0 for high altitudes. The iterative solution with the atmospheric temperature retrievals would help resolve this issue for the MPD products. This method would only improve an HSRL as part of an MPD system, further demonstrating the utility of the full thermodynamic profiling MPD. Another issue with the prototype MPD at Montana State is the tendency for the instrument to drift out of alignment over time. For long-term autonomous data collection, the optomechanical stability of the instruments will need to be improved, or else they will require frequent realignment.

PBLH RETRIEVAL

Introduction

Despite its importance to weather forecasting and climate science, traditional meteorological techniques fail to capture the diurnal cycle of the PBLH. Additionally, there is no widely agreed-upon standard for defining the PBLH, and different methods produce different PBLH values for the same atmospheric conditions. The large variability of the PBLH and its diurnal cycle requires continuous monitoring similar to the thermodynamic parameters in the lower troposphere. Profiles of atmospheric parameters, including the potential temperature [63, 64, 65] and wind speed [64, 65] can be used to determine the PBLH by radiosonde. Ground-based remote sensing provides a promising method for continuous monitoring of the PBLH because the entrainment zone (EZ) or capping inversion (CI) that marks the top of the PBL can be identified in lidar profiles. The underlying assumption is that the inversion layer that defines the PBLH is co-located with a steep aerosol gradient. This assumption is valid because the boundary layer has a much higher aerosol concentration than the free troposphere.

Methods of retrieving the PBLH with an aerosol lidar use the aerosol concentrations to locate inversions between buoyant layers in the atmosphere. Aerosol lidars have been thoroughly demonstrated in their capabilities for PBLH monitoring [12, 18, 66, 67, 68, 69]. Several methods exist for determining the PBLH from lidar profiles such as the gradient method [68, 70], variance methods [67, 70], curve fitting methods [66, 71], and Haar Wavelet Transform (HWT) or wavelet covariance transform methods [12, 18, 66, 69, 72, 73, 74]. The gradient method and the HWT method assume that the aerosol concentration in the PBL is much higher than in the free troposphere, and there is a sharp decrease in aerosol concentration at the PBLH. The variance method uses the temporal variation in lidar signal at the top of the PBL caused by convection between air without aerosols and air with high

concentrations of aerosols in the EZ. The curve fitting method fits an idealized boundary layer profile to match the lidar signal, and the PBLH is set to be the midway point in the transition region of the fit lidar signal.

Methods

The aerosol backscatter coefficient from the HSRL is used to find the PBLH for various conditions. The HSRL collected this data between March and July 2021. The HWT method extracts the features that mark the PBLH from the aerosol backscatter coefficient profiles. Unlike most other retrieval algorithms, the method presented in this thesis uses the HWT of the aerosol backscatter coefficient. Most other lidar PBLH retrieval methods use the HWT of the range-corrected backscattered lidar signal. This data was compared to PBLH values from radiosonde retrievals using the Heffter (1980) method [63]. The nocturnal stable boundary layer was frequently below the minimum range of the lidar, so it is excluded from the retrieval.

Haar Wavelet Transform

The HWT method detects an abrupt change in the lidar signal and can be thought of as a smoothed gradient that detects regions rather than points of change. The Haar wavelet, h, and the Haar wavelet transform, H are described in Equations 5.1 and 5.2 as according to Brooks [72]. In these equations, z is the range of the lidar signal, $f(z)$ is the lidar signal, r is the range of the center of the wavelet, and a is the dilation of the wavelet. Figure 5.1 displays the Haar wavelet.

$$h(\frac{z-r}{a}) = \begin{cases} +1: \ r - \frac{a}{2} \leq z \leq r \\ -1: \ r \leq z \leq r + \frac{a}{2} \\ \ \ 0: \ elsewhere \end{cases} \tag{5.1}$$

$$H(a,r) = \frac{1}{a} \cdot \int_{r-\frac{a}{2}}^{r+\frac{a}{2}} f(z) \cdot h(\frac{z-r}{a}) dz \tag{5.2}$$

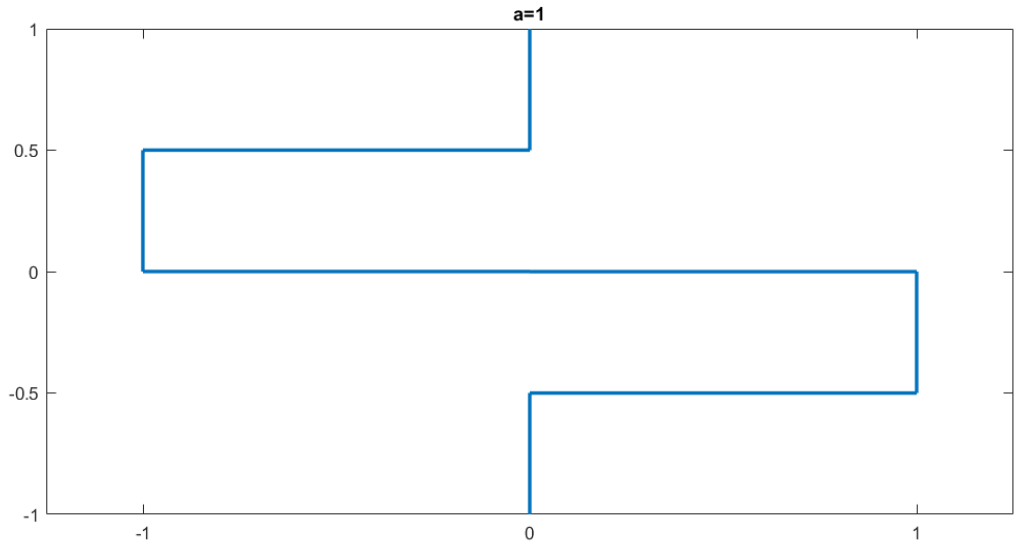


Figure 5.1: Haar wavelet, dilation a=1.

In all HWT methods of retrieving the PBLH, the correct choice of dilation is a critical parameter for success. Choosing the right dilation is challenging for any HWT retrieval method, and it is difficult to find one that works for all conditions. If the dilation is too narrow, the retrieval method frequently detects spurious layers leading to unreliable retrievals. A dilation that is too large, on the other hand, sacrifices spatial resolution and accuracy. Additionally, the HWT has a minimum range equal to the lidar's minimum range plus one-half of the dilation, so large dilations cannot detect the PBLH at low altitudes.

Brooks demonstrated that the ideal choice of dilation is equal to the depth of the transition zone between the boundary layer and the free troposphere [72]. Unfortunately, this transition zone depth varies greatly and is usually not known. Figure 5.2 demonstrates the difference in HWT values for several different choices of dilations. The normalized aerosol backscatter coefficient is used to find the PBLH. This profile is normalized to the mean value for the aerosol backscatter coefficient from the minimum range of the lidar to 400 m. The data is normalized so that the same retrieval method can be used whether the air is relatively free of aerosols or full of forest fire smoke.

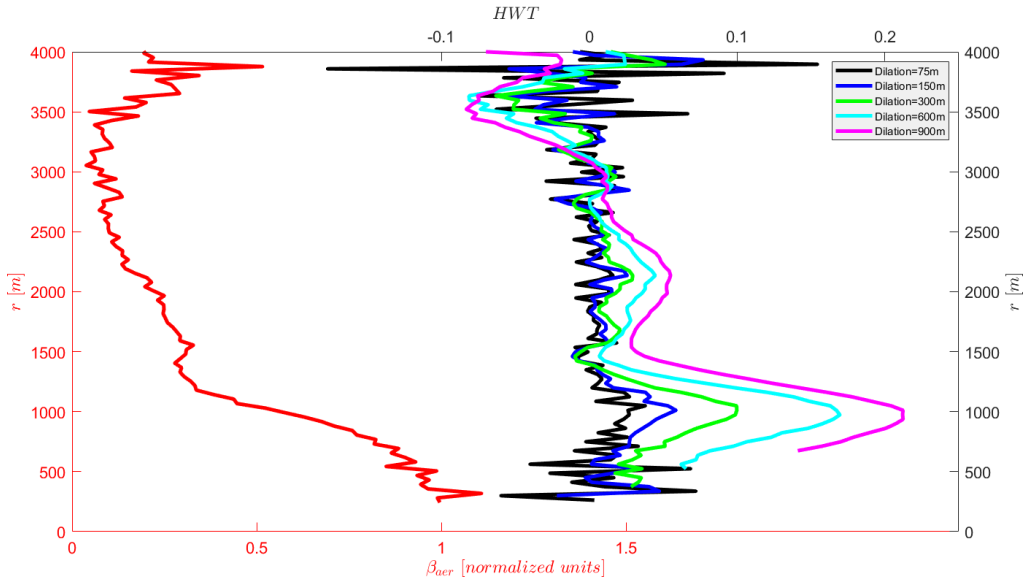


Figure 5.2: HWT for various dilation values.

The mixing at the EZ and the transition zone depth increase as the PBLH is lifted through convective processes. In a method similar to the one used by Baars [73], larger dilations were used for higher range values. The dilation used is given by Equation 5.3. The dilation increases linearly with the range with a constant of proportionality of 0.44. The dilation is rounded to the nearest multiple of 75 because the dilation is limited to integer multiples of twice the range bin. Figure 5.3 displays the dilation graphically.

$$a(r) = \begin{cases} 0.44 \cdot r; \ r < 2600m \\ \\ 1200m; \ r \geq 2600m \end{cases} \tag{5.3}$$

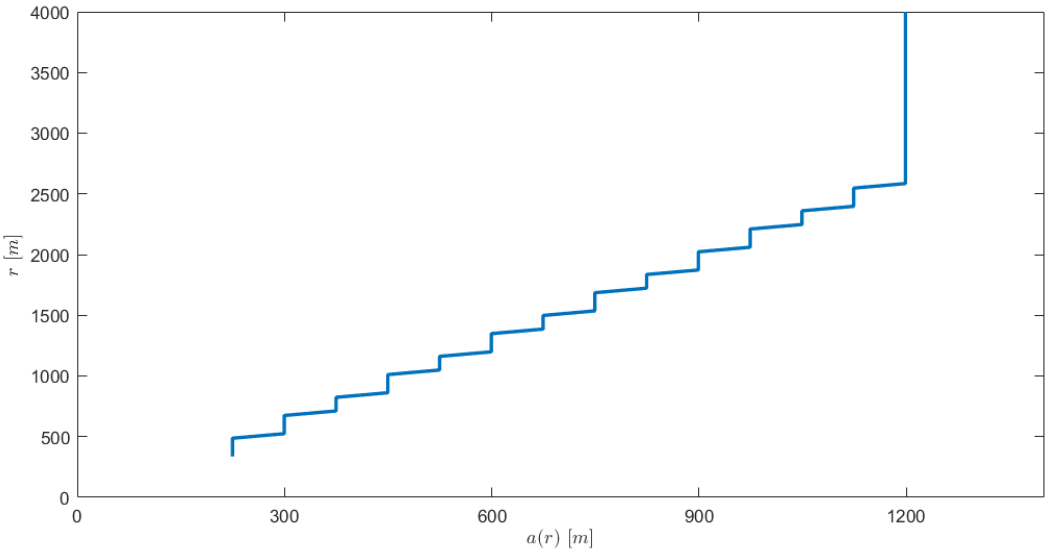


Figure 5.3: Range-dependent dilation function.

PBLH Retrieval Algorithm

The most significant issue with the HWT transform method is that the maximum value of the HWT is frequently not the top of the PBL but rather the top of a residual layer (RL). A method proposed by Dang et al. detects the bottom of RLs, which have negative HWT values to set limits to where the algorithm searches for the PBLH [66]. The PBLH retrieval algorithm used for this thesis uses a similar method. The retrieval algorithm presented here uses the HWT of the normalized β_{aer} to extract aerosol features from the atmosphere and set limits for the next layer. The retrieval algorithm finds PBLH and the height of the CI.

The first aerosol structures that the retrieval algorithm finds are cloud layers. For each time step, the retrieval algorithm finds the lowest bin removed by the cloud mask.

The next aerosol structure that the algorithm finds is the free atmosphere. The purpose of the free atmosphere layer is to find a layer above the CI to set as a maximum height for all further layers. The free atmosphere layer is defined as the first point that the β_{aer} profile is less than $5 \cdot 10^{-8}\ m^{-1} \cdot sr^{-1}$. Unmasked data is used to find the free atmosphere because the SNR mask often removes data at these ranges. After the free atmosphere is found, it is smoothed by a Gaussian filter with a width of 1 hour. If the free atmosphere is beneath the cloud height, these clouds are considered decoupled from the CI and are not considered for the rest of the retrieval algorithm.

The next layer to find is the CI. The CI is either the top of the PBL or the RL if one is present. It is important to note the difference between the CI and the PBLH. The CI is the inversion between the last aerosol dense buoyant layer and the low aerosol density free atmosphere. The retrieval algorithm presented does not distinguish between the CI and the EZ and will refer to the inversion layer between the free troposphere and the underlying layer, whether that layer is the RL or the PBL, as the CI. The algorithm defines the CI to be the last significant aerosol gradient, and therefore the last significant HWT value in the normalized β_{aer} profile. The retrieval algorithm sets a lower limit at 800 m and an upper

limit at the location of the smoothed free atmosphere layer. If the free atmosphere retrieval fails, then the upper limit is set to 6 km. The CI is set at the last point where the HWT of the normalized β_{aer} is greater than 0.05. If this point does not exist, then the CI is set to the location of the maximum value of the HWT of the normalized β_{aer} profile. Once this height is located, the CI layer is smoothed with a Gaussian filter with a width of 1 hour. The layers found up to this point are displayed in Figure 5.4 for 24 hours in March 2021. The left vertical black line is at sunrise, and the right vertical black line is at sunset.

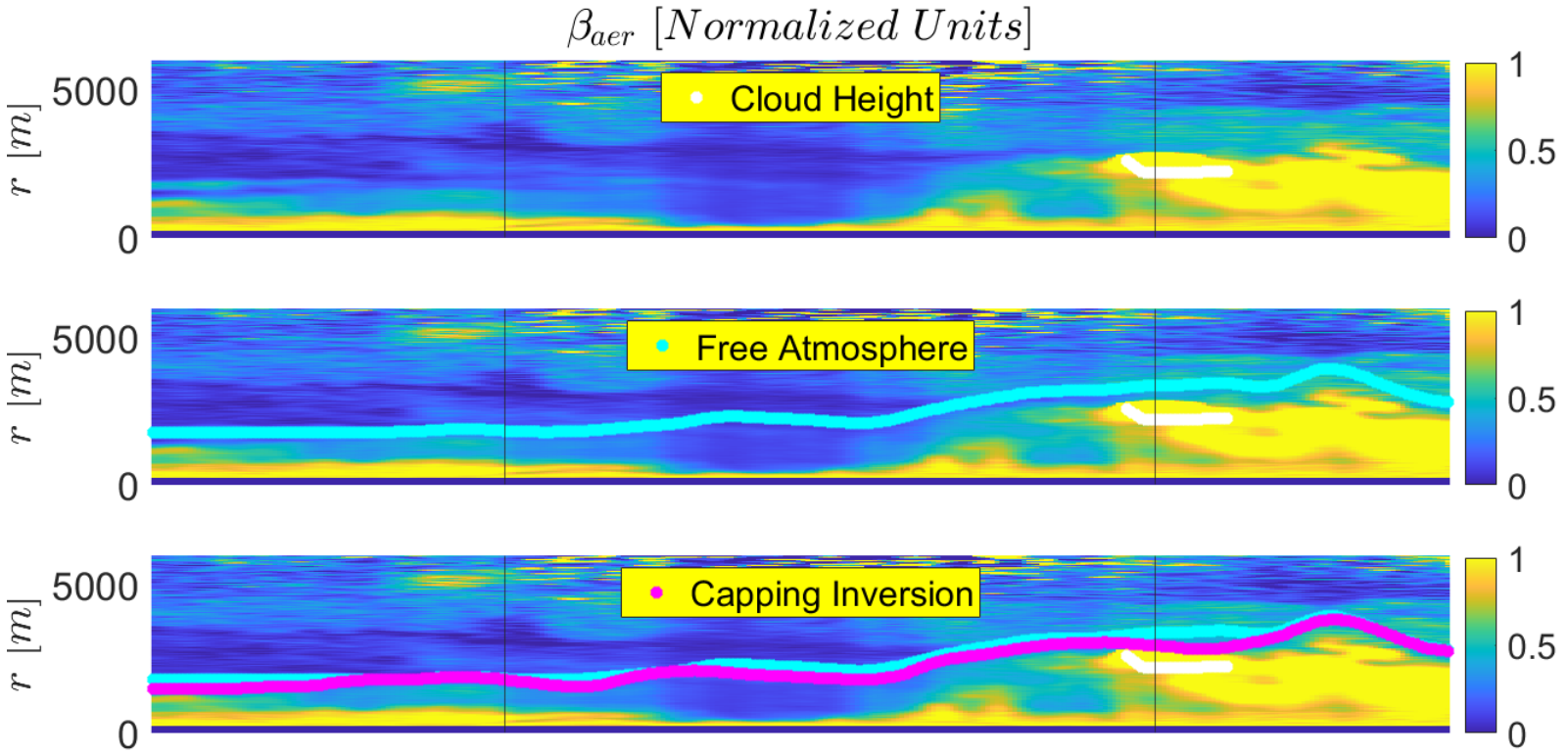


Figure 5.4: Top: Cloud height retrieval. Center: Free atmosphere retrieval. Bottom: Capping inversion retrieval.

The next part of the algorithm involves setting upper and lower limits on the altitudes where the retrieval algorithm searches for the PBLH. The morning CI is particularly troublesome for HWT PBLH retrieval methods because the transition between the RL and the free troposphere frequently has a stronger gradient than the transition layer between the PBL and the RL. These strong gradients at the top of the RL cause the algorithm to mistake the CI for the PBLH. This error is common for the first few hours after sunrise. In the afternoon, the most common error is that the boundary layer transition zone is too large, and the retrieval method picks up spurious layers. The retrieval algorithm splits the day into two parts to prevent these errors. The CI tends to slowly decrease in height from sunset until it is lifted by the convective processes of the rising PBLH the next day. Before this, the CI is set at the top of the previous day's RL, and afterward, the CI is usually at the PBLH. This point is called the convection point, and its height is called the convection height. The convection point and height are not from other literature and were defined for this retrieval algorithm. The retrieval algorithm does not attempt to find the PBLH during the night because the nocturnal SL inversion is frequently below the minimum range of the lidar.

The HWT algorithm searches for the bottom of residual layers to set the upper limit of the morning PBLH retrieval. The bottom of an RL is any point where the HWT profile is negative, indicating an increase in aerosol concentration with altitude associated with a lofted aerosol layer decoupled from the surface forcings of the earth. An additional upper limit is placed along a straight line between a height of 500 m at sunrise and the convection point. For the afternoon, the upper limit is set to the capping inversion. The lower limit for the morning is set to 300 m and 500 m for the afternoon. For the afternoon PBLH retrieval, an additional lower limit is set to the first range bin with a normalized backscatter coefficient that has changed by 40% from the low altitude value. In other words, the lower limit is set to the first point where the normalized backscatter coefficient is greater than 1.4 or less than

0.6. If this bin is below 500 m, the lower limit is set to 500 m. If clouds exist below the CI, and the algorithm fails to retrieve the PBLH, the PBLH is set to the cloud top. The top of the cloud is determined to be the maximum value of the HWT above the cloud base.

In the morning PBLH retrieval, the PBLH is set to the first point after the peak HWT value where the HWT is $\frac{2}{3}$ of the value of the peak. If there is no peak in the HWT, then the PBLH is probably below the minimum range of the lidar, and the retrieval sets the PBLH to a not a number (NaN) value. In the afternoon PBLH retrieval, the PBLH is set to the first point after the peak where the HWT is less than 0.04. If there is no location where this is the case, then the PBLH is set to the first point where the HWT is $\frac{3}{4}$ of the peak value. Afterward, discontinuous points are removed, and new points near the smoothed PBLH are found. Figure 5.5 displays the RL, the convection point, and the PBLH retrieval with upper and lower limits for the same 24 hour period as shown in Figure 5.4.

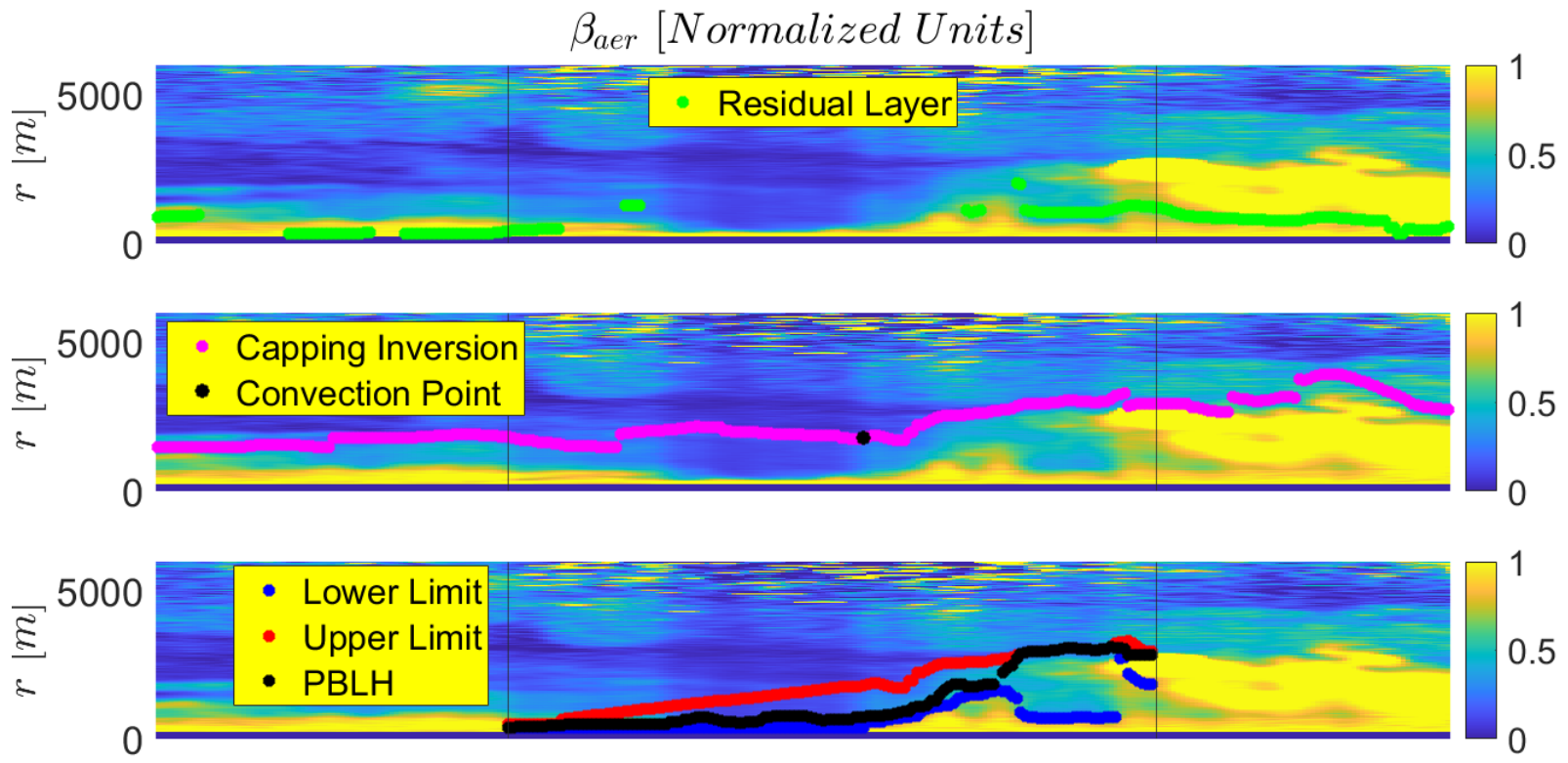


Figure 5.5: Top: Residual layer retrieval. Center: Convection point retrieval. Bottom: PBLH limits and retrieval.

Radiosonde Retrievals

Data collected from radiosondes are used to validate the HWT PBLH retrieval algorithm. Between March and June 2021, 36 weather balloons were launched from the roof of Cobleigh Hall, equipped with Vaisala RS92-SGP radiosondes. These balloons had an average climb rate of approximately 5 $\frac{m}{s}$ and they measured atmospheric pressure, temperature, and relative humidity. Of the three radiosonde methods outlined in the Atmospheric Radiation Measurement (ARM) PBL value-added product (VAP) report [65], the Heffter method [63] requires only the potential temperature and altitude from a radiosonde profile. The other two methods require wind speed measurements which require expensive GPS-equipped radiosondes. The potential temperature, θ, is the temperature that a parcel of dry air would have if taken to sea-level pressure in an isentropic process. Where T is the temperature of the parcel, P is pressure, P_0 is the sea-level pressure, and k is the isentropic expansion factor, the potential temperature is given by:

$$\vartheta = T \cdot (\frac{P}{P_0})^{(1-\frac{1}{k})} \tag{5.4}$$

The Heffter method utilizes criteria related to the strength of the potential temperature lapse rate and the difference in potential temperature between the top and bottom of an inversion layer. The Heffter method was implemented as it was in the ARM PBL VAP report [65]. The data is first smoothed to 15 $mbar$ (1.5 kPa) to reduce the detection of spurious layers. Next, inversion layers with a lapse rate greater than 0.005 $\frac{K}{m}$ are identified. The PBLH is the top of the lowest inversion layer with a potential temperature difference greater than 2 K between the top and bottom of the layer. If no layer qualifies, then the inversion with the highest potential temperature difference between the top and bottom of the layer is found, and the PBLH is set to the altitude where the lapse rate reaches a maximum within that layer. An example of a PBLH retrieval using the Heffter Method is shown in Figure

5.6. An inversion layer is located between 3300 and 3800 m, and the Heffter method places the PBLH at 3800 m.

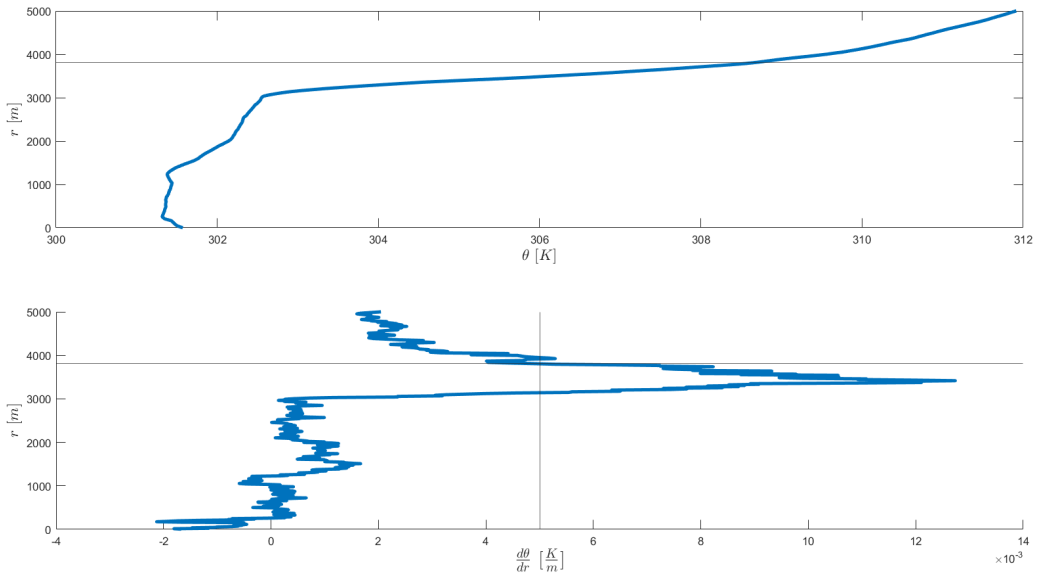


Figure 5.6: Heffter retrieval demonstration.

The PBLH retrieval by the Heffter method is not exact. Data from the ARM Southern Great Plains (SGP) site with Heffter, Liu-Liang, and Bulk Richardson PBLH retrievals demonstrate that the results these methods produce vary widely in conditions without an obvious boundary layer [65]. Like the HWT PBLH retrieval algorithm, the Heffter radiosonde method frequently detects the CI rather than the PBLH underneath it because the PBLH inversion does not meet the 2 K potential temperature difference criteria. When the PBLH determined from lidar data is different from the PBLH as retrieved by the Heffter method, this does not necessarily indicate an incorrect result from the lidar data, and the potential temperature and aerosol optical profiles must be carefully inspected to determine the PBLH.

Data and Radiosonde Comparisons

In the following figures, the PBLH retrieval is shown for four periods in 2021: March 13-March 19, March 28-April 3, May 11-May 12, and June 1-June 4. The normalized β_{aer} and the HWT of the normalized β_{aer} are shown as these parameters were used for the PBLH retrieval. The PBLH, the CI, and clouds are also shown in the plots. These periods are used because radiosondes were launched from the roof of Cobleigh Hall during these periods. Radiosonde launches are shown as cyan vertical lines on the plots. Notice the high degree of variability and diurnal variation of the PBLH. Tracking the PBLH requires continuous observations, and using twice-daily radiosonde profiles would miss this variability. The current algorithm works well for the morning and daytime PBLH but frequently fails to track the PBLH in the evening transition period. The PBLH should decrease in altitude as the earth's surface cools, but the algorithm frequently tracks a lofted residual layer instead.

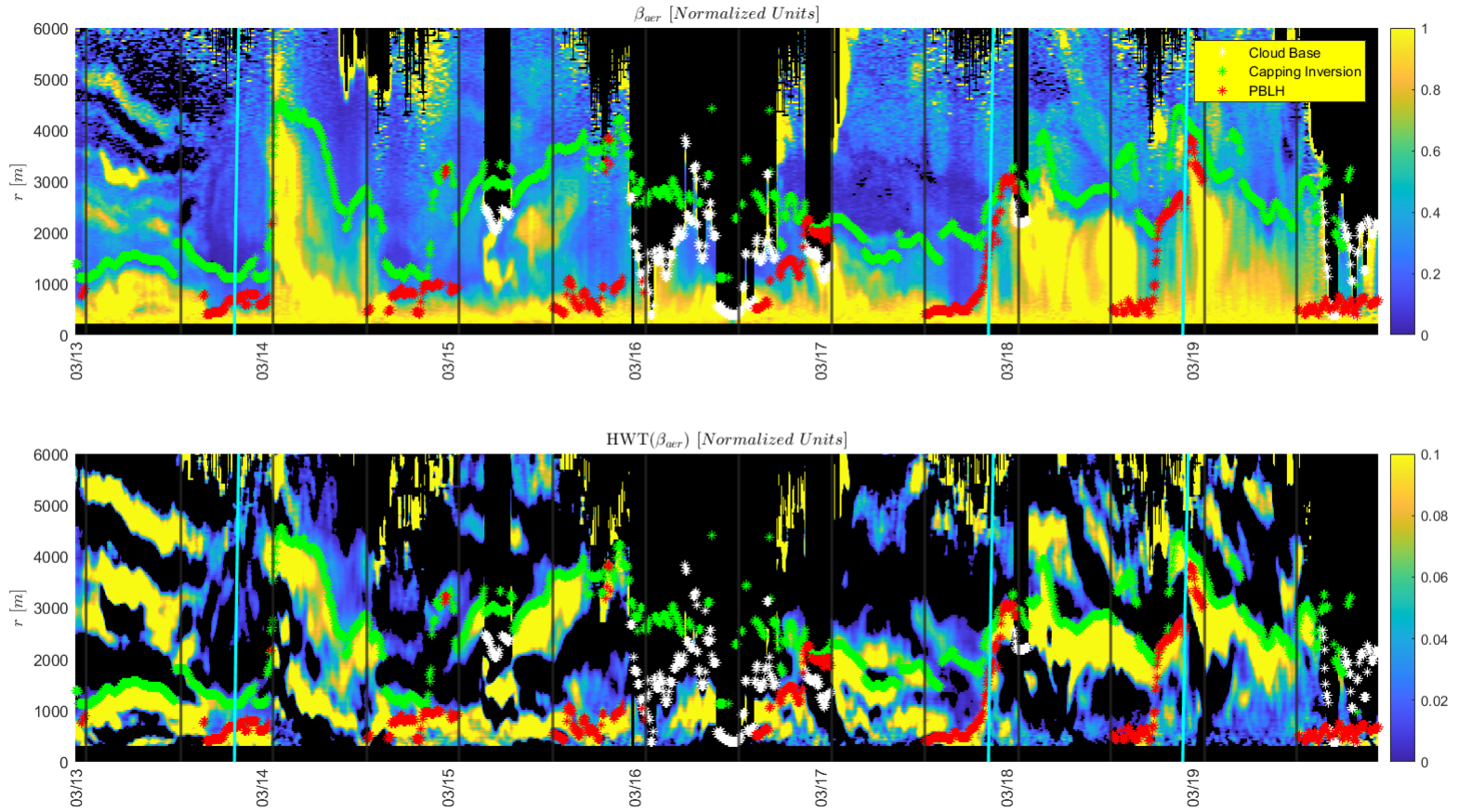


Figure 5.7: March 13-March 19, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval.

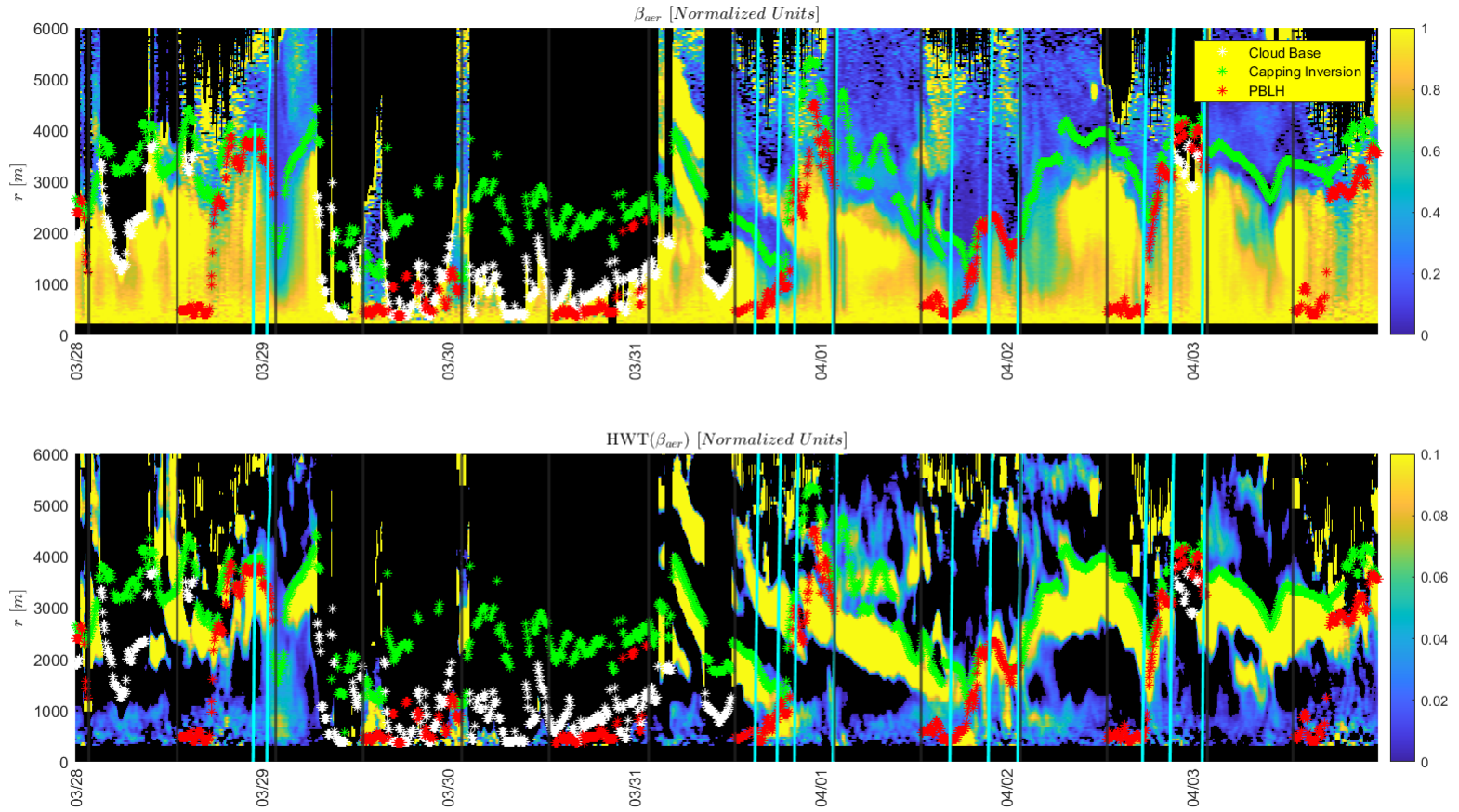


Figure 5.8: March 28-April 3, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval.

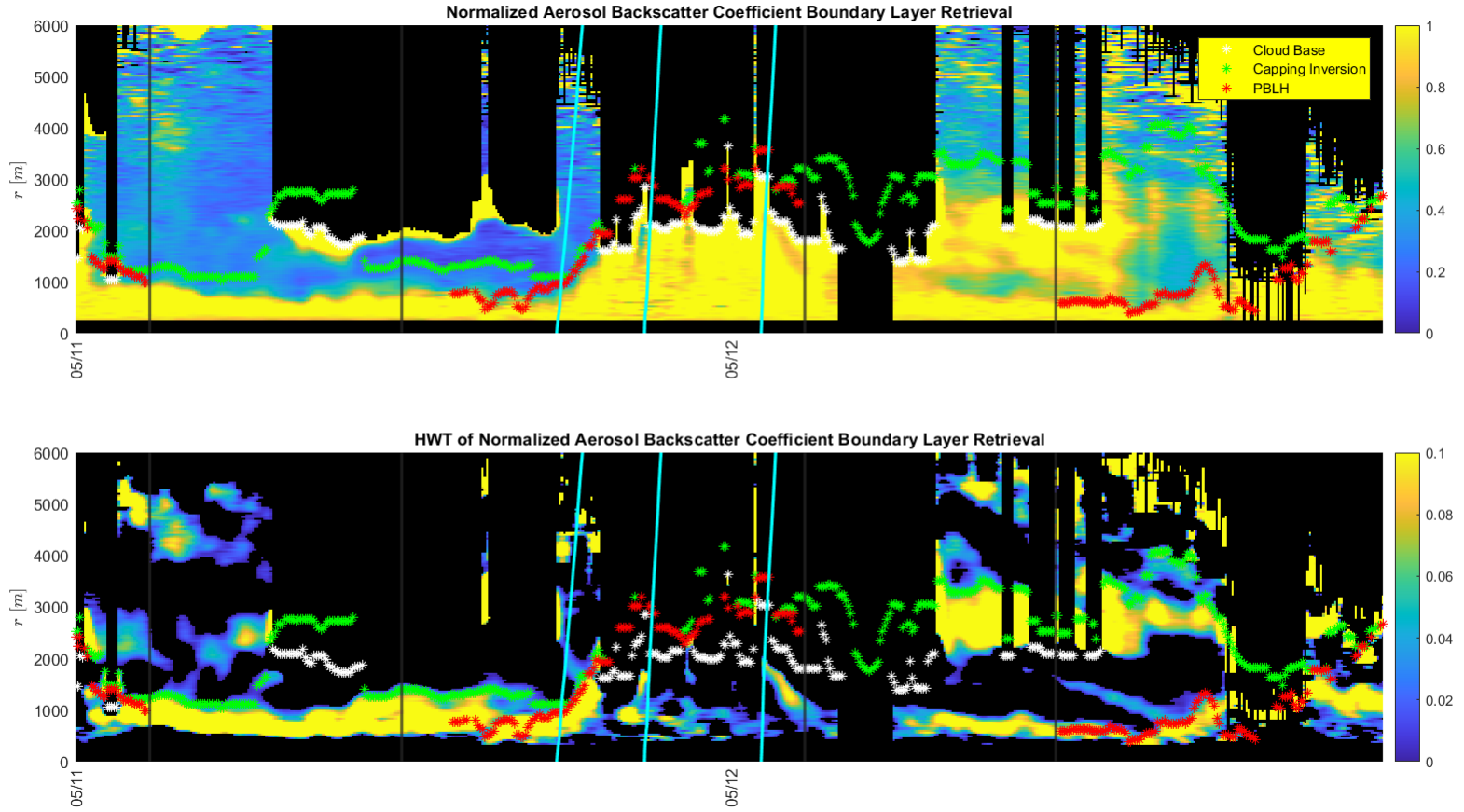


Figure 5.9: May 11-May 12, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval.

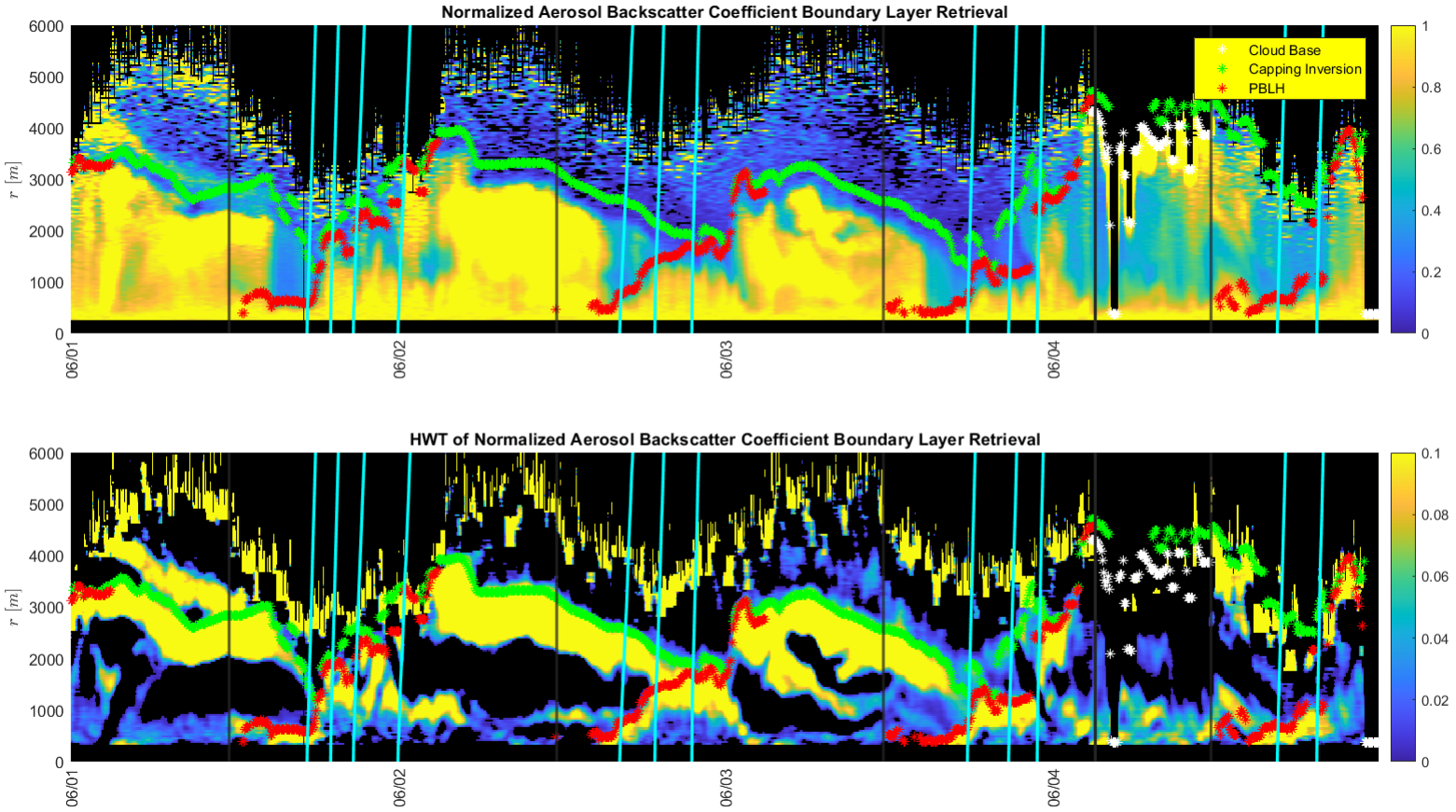


Figure 5.10: June 1-June 4, 2021; Top: Normalized aerosol backscatter coefficient and PBLH retrieval. Bottom: HWT of normalized aerosol backscatter coefficient and PBLH retrieval.

During these periods, a total of 30 radiosondes with acceptable data were launched from the roof of Cobleigh Hall. Of these radiosondes, 9 of the retrieved profiles had no determinate boundary layer because there were no potential temperature inversion layers that satisfied the 0.005 $[\frac{K}{m}]$ requirement of the Heffter retrieval method. Of the remaining 21 radiosondes, 6 of the profiles found different layers than the lidar retrieval algorithm. This error occurred because the Heffter retrieval method set the PBLH to the height of the CI instead of the actual PBLH. The algorithm identified the CI because the potential temperature difference at the PBLH inversion was less than the 2 K required for the Heffter method. In the 15 radiosondes where the Heffter retrieval method and the lidar retrieval method identified the PBLH at the same inversion layer, the lidar method identified the PBLH to be lower than the Heffter method by an average difference of 295 m. This difference is not necessarily an issue because the PBLH is dependent on the retrieval algorithm used. Even among potential temperature methods applied to a single radiosonde profile, there is rarely agreement on the PBLH. For example, Sivaraman et al. showed that for data collected from the ARM SGP site, the median difference between the Heffter method and Liu-Liang method was 185 m and the median difference between the Heffter method and the Bulk Richardson method was 124 m. The Heffter method was, on average, higher in altitude than both the Liu-Liang and Bulk-Richardson methods [65]. Additionally, there may have been some bias introduced by the winds in Bozeman, Montana. The winds tend to blow from the southwest; there are foothills and mountains to the northeast of the launch site, which would cause an increase in the measured PBLH by the radiosonde. Figure 5.11 demonstrates the difference between the PBLH as retrieved by the HWT algorithm and the Heffter method.

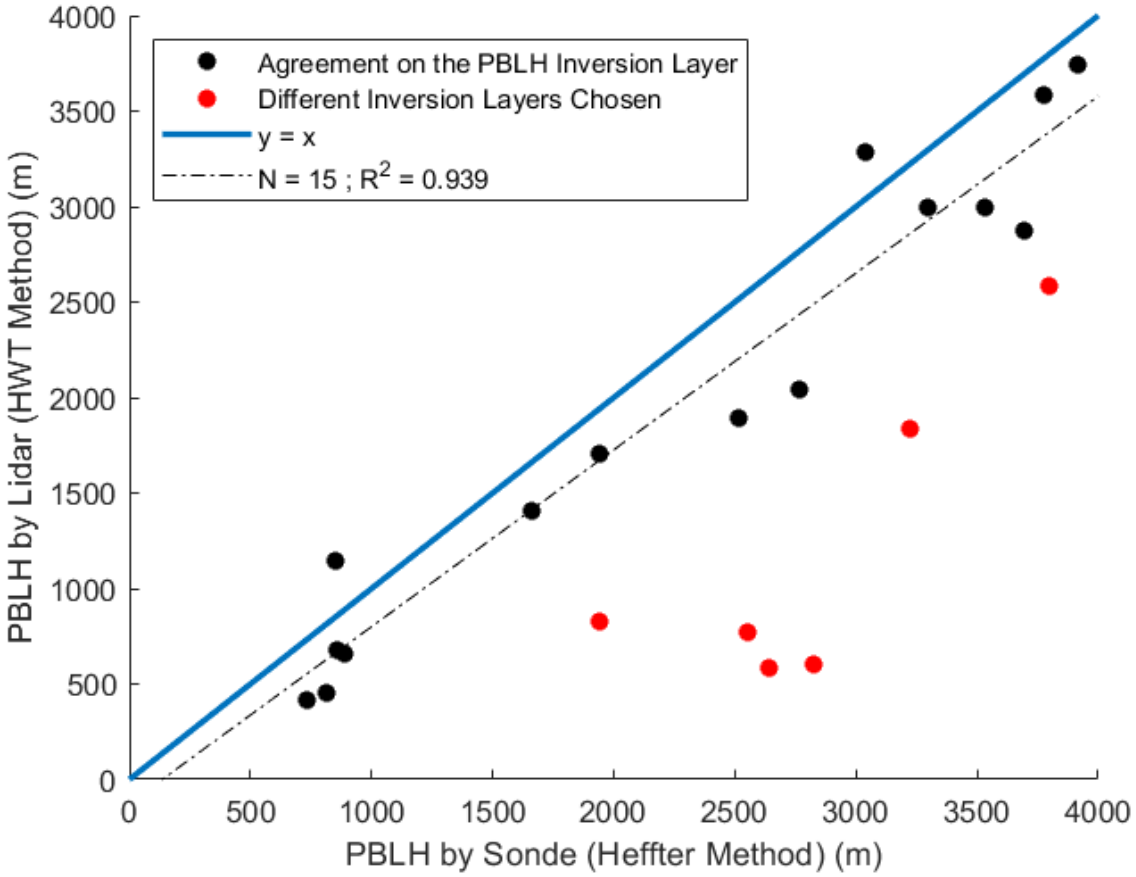


Figure 5.11: PBLH values found by the HWT-lidar method and by the Heffter-radiosonde method.

Even when the lidar retrieval method and the Heffter method agree on which inversion layer is located at the PBLH, the HWT of the aerosol optical properties tends to lead the potential temperature lapse rate, leading to a consistently lower value for the PBLH. An example of this effect is shown in Figure 5.12 where the HWT peak occurs about 100 m below the peak in the potential temperature lapse rate. An example of the lidar method and the Heffter method choosing different inversion layers is shown in Figure 5.13. In this case, the PBLH inversion from 500 to 1000 m did not meet the 2 K potential temperature difference criteria, and the Heffter method identified the capping inversion from 2000 to 2500 m instead of the PBLH.

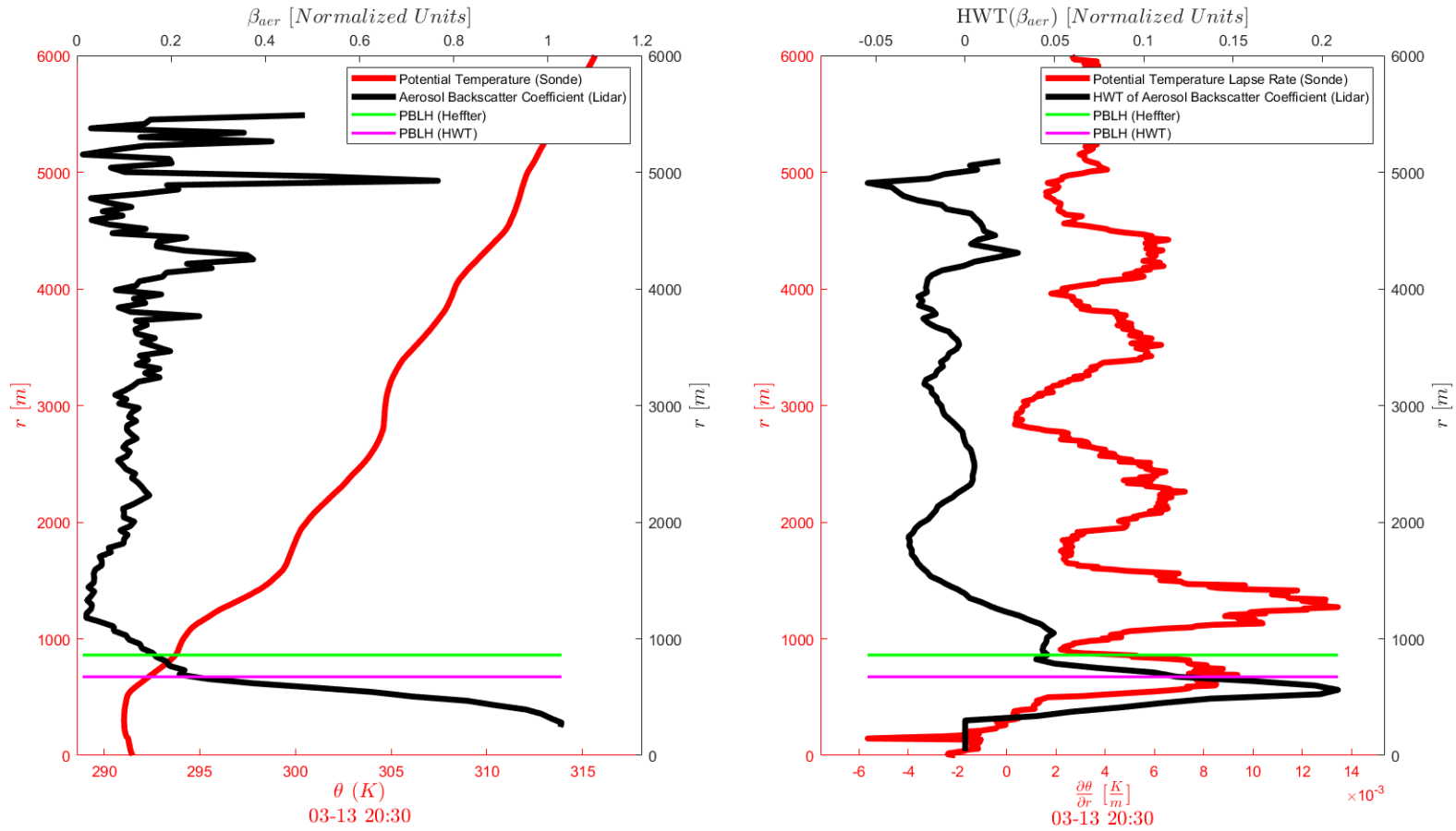


Figure 5.12: Radiosonde comparison, March 13, 2021-20:30 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=861 m; PBLH-Lidar=674 m.

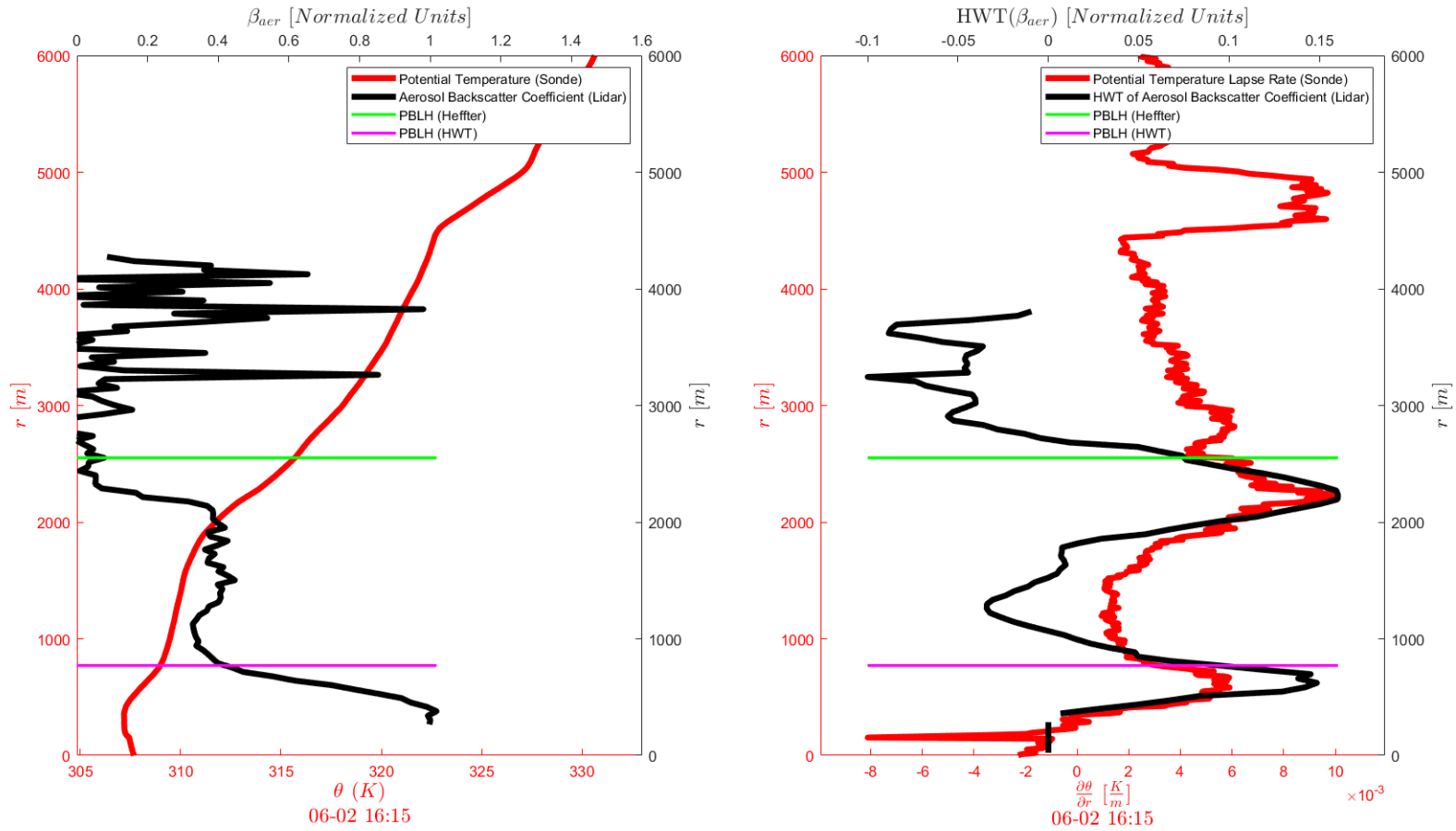


Figure 5.13: Radiosonde comparison, June 2, 2021-16:15 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2552 m; PBLH-Lidar=771 m.

Future Work

Further work is needed for the HSRL PBLH retrieval algorithm to be used in weather forecasting or climate research. The PBLH retrieval algorithm frequently fails to track the PBLH during the evening transition period and instead tends to track a residual layer. The algorithm should be improved so that the PBLH is consistently tracked throughout the entire day. Additionally, the retrieval method should be compared to other PBLH retrieval methods like the Liu-Liang and the bulk Richardson number radiosonde methods.

Another potential route for future research is to include the products of the O2-DIAL and the WV-DIAL in the PBLH retrieval algorithm. The inversion layer that marks the top of the PBLH can be identified by the strong lapse rate in the potential temperature profile and a strong negative gradient in the absolute humidity profile. If the MPD could measure the potential temperature and the absolute humidity, then PBLH retrieval algorithms utilizing these products could be implemented. These methods could potentially increase accuracy in the PBLH retrieval.

CONCLUSION

The HSRL is one of three lidars in the proposed MPD instrument, and it measures the aerosol optical properties of the atmosphere. The proposed instrument also contains an O2-DIAL for tropospheric temperature measurements and a WV-DIAL for tropospheric humidity measurements. The HSRL is useful for monitoring aerosol optical properties, and these aerosol optilcal properties can be used for qualitative aerosol observations. Additionally, the aerosol optical properties are necessary for accuracy in the temperature retrievals by the O2-DIAL. For this reason, the O2-DIAL is only valuable when used in conjunction with the HSRL. The aerosol optical properties can also be used to find the PBLH, a significant meteorological parameter. The HSRL is a valuable instrument as a stand-alone system because it can locate the PBLH.

This lidar provided the aerosol backscatter coefficient and the backscatter ratio for the lower troposphere to an altitude of 5 km in the daytime and 7 km at night for clear conditions. Additionally, the aerosol optical properties can be used to find the PBLH. An algorithm based on the Haar wavelet transform was created and used to track the PBLH. This algorithm was compared to the Heffter Radiosonde method and retrieved a PBLH value on average 295 m below the Heffter method for conditions with a well-behaved boundary layer. The PBLH increases the usefulness of the MPD instruments, as the PBLH is an essential parameter for weather forecasting and climate research.

My contribution to the instrumentation of the HSRL and the MPDs is in assembling, aligning, and troubleshooting the lidar. Previous researchers designed the hardware and software systems. My contribution to the aerosol optical property retrievals is writing the code which extracted the aerosol optical properties from the raw lidar profiles. I also found a better method of modeling the atmosphere, which previously used a less accurate empirical equation for modeling the β_{mol}. My most significant contribution to this project is the PBLH retrieval algorithm using the HSRL data products.

REFERENCES CITED

[1] R. B. Stull. *An Introduction to Boundary Layer Meteorology*. Kluwer Academic, 1988. doi: 10.1007/978-94-009-3027-8.

[2] W. C. Hinds. *Aerosol Technology: Properties, Behavior, and Measurement of Airborne Particles*. Wiley, 1999.

[3] O. Edenhofer, R. Pichs-Madruga, Y. Sokona, E. Farahani, K. Kadner, A. Adler, K. Seyboth, P. Eickemeier, B. Kriemann, J. Savolainen, S. Schlömer, C. von Stechow, T. Zwickel, and J.C. Minx. Ipcc 2014: Climate change 2014: Mitigation of climate change. contribution of working group iii to the fifth assessment report of the intergovernmental panel on climate change. Technical report, Intergovernmental Panel on Climate Change, 2014.

[4] B.J. Soden, R.T. Wetherald, G.L. Stenchikov, and A. Robock. Global cooling after the eruption of mount pinatubo: A test of climate feedback by water vapor. *Science*, 296 (5568):727–730, April 2002. doi: 10.1038/1991275a0.

[5] C. Wang. A modeling study on the climate impacts of black carbon aerosols. *Journal of Geophysical Research: Atmospheres*, 109(D3), 2004. doi: https://doi.org/10.1029/2003JD004084. URL `https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2003JD004084`.

[6] Nathan Seldomridge. Dual-polarization cloud lidar design and characterization. Master's thesis, Montana State University, 2005.

[7] G. Fiocco and L.D. Smullin. Detection of scattering layers in the upper atmosphere (60–140 km) by optical radar. *Nature*, 199:1275, 1963. doi: 10.1038/1991275a0.

[8] M.G.H. Ligda. The laser in meteorology. *Discovery*, 26:30–35, 1965.

[9] V.A. Kovalev and W.E. Eichinger. *Elastic Lidar: Theory, Practice, and Analysis Methods*. Wiley, 2004. doi: 10.1002/0471643173.

[10] A. Ansmann, M. Riebesell, U. Wandinger, C. Weitkamp, E. Voss, W. Lahmann, and W. Michaelis. Combined raman elastic-backscatter lidar for vertical profiling of moisture, aerosol extinction, backscatter, and lidar ratio. *Applied Physics B*, 55(1): 18–28, July 1992. doi: 10.1007/BF00348608.

[11] B. Rosati, E. Herrmann, S. Bucci, F. Fierli, F. Cairo, M. Gysel, R. Tillmann, J. Größ, G. P. Gobbi, L. Di Liberto, G. Di Donfrancesco, A. Wiedensohler, E. Weingartner, A. Virtanen, T. F. Mentel, and U Baltensperger. Studying the vertical aerosol extinction coefficient by comparing in situ airborne data and elastic backscatter lidar. *Atmospheric Chemistry and Physics*, 16:4539–4554, 2016. doi: 10.5194/acp-16-4539-2016.

[12] S. Pal, A. Behrendt, and V. Wulfmeyer. Elastic-backscatter-lidar-based characterization of the convective boundary layer and investigation of related statistics. *Annales*

Geophysicae, 28(3):825–847, 2010. doi: 10.5194/angeo-28-825-2010. URL https://angeo.copernicus.org/articles/28/825/2010/.

[13] Gregori de Arruda Moreira, Juan Luis Guerrero-Rascado, Juan Antonio Bravo-Aranda, José Antonio Benavent-Oltra, Pablo Ortiz-Amezcua, Roberto Róman, Andrés Esteban Bedoya-Velásquez, Eduardo Landulfo, and Lucas Alados-Arboledas. Study of the planetary boundary layer by microwave radiometer, elastic lidar and Doppler lidar estimations in Southern Iberian Peninsula. *Atmospheric Research*, 213:185–195, November 2018. doi: 10.1016/j.atmosres.2018.06.007.

[14] V. Matthias and J. Bösenberg. Aerosol climatology for the planetary boundary layer derived from regular lidar measurements. *Atmospheric Research*, 63(3):221–245, January 2002. doi: 10.1016/S0169-8095(02)00043-1.

[15] Edwin E. Eloranta. *High Spectral Resolution Lidar*, pages 143–163. Springer New York, New York, NY, 2005. ISBN 978-0-387-25101-1. doi: 10.1007/0-387-25101-4_5. URL https://doi.org/10.1007/0-387-25101-4_5.

[16] 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. *Appl. Opt.*, 22(23):3716–3724, Dec 1983. doi: 10.1364/AO.22.003716. URL http://ao.osa.org/abstract.cfm?URI=ao-22-23-3716.

[17] Christian J. Grund and Edwin W. Eloranta. University of Wisconsin High Spectral Resolution Lidar. *Optical Engineering*, 30(1):6 – 12, 1991. doi: 10.1117/12.55766. URL https://doi.org/10.1117/12.55766.

[18] Jennifer D. Hegarty, Jasper Lewis, Erica L. McGrath-Spangler, John Henderson, Amy Jo Scarino, Philip DeCola, Richard Ferrare, Micheal Hicks, Rebecca D. Adams-Selin, and Ellsworth J. Welton. Analysis of the planetary boundary layer height during discover-aq baltimore–washington, d.c., with lidar and high-resolution wrf modeling. *Journal of Applied Meteorology and Climatology*, 57(11):2679 – 2696, 2018. doi: 10.1175/JAMC-D-18-0014.1. URL https://journals.ametsoc.org/view/journals/apme/57/11/jamc-d-18-0014.1.xml.

[19] Jeffrey S. Reid, Ralph E. Kuehn, Robert E. Holz, Edwin W. Eloranta, Kathleen C. Kaku, Shi Kuang, Michael J. Newchurch, Anne M. Thompson, Charle R. Trepte, Jianglong Zhang, Samuel A. Atwood, Jenny L. Hand, Brent N. Holben, Patrick Minnis, and Derek J. Posselt. Ground-based High Spectral Resolution Lidar observation of aerosol vertical distribution in the summertime Southeast United States. *JOURNAL OF GEOPHYSICAL RESEARCH-ATMOSPHERES*, 122(5):2970–3004, Mar 2017. ISSN 2169-897X. doi: {10.1002/2016JD025798}.

[20] D. Seldomridge, J.A. Shaw, and K.S. Repasky. Dual-polarization lidar using a liquid crystal variable retarder. *Optical Engineering*, 45(10), Oct 2006. doi: 10.1117/1.2358636.

[21] D.K. Killinger. *Optical and Laser Remote Sensing*. Springer, 1983.

[22] E. V. Browell, T. D. Wilkerson, and T. J. Mcilrath. Water vapor differential absorption lidar development and evaluation. *Appl. Opt.*, 18(20):3474–3483, Oct 1979. doi: 10.1364/AO.18.003474. URL http://ao.osa.org/abstract.cfm?URI=ao-18-20-3474.

[23] 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. *Opt. Express*, 28(1): 71–93, Jan 2020. doi: 10.1364/OE.379804. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-28-1-71.

[24] David D. Turner and J.E. M. Goldsmith. Twenty-four-hour raman lidar water vapor measurements during the atmospheric radiation measurement program's 1996 and 1997 water vapor intensive observation periods. *Journal of Atmospheric and Oceanic Technology, 16(8):1062-1076*, Aug 1999. doi: 10.1175/1520-0426(1999)016⟨1062:TFHRLW⟩2.0.CO;2. URL https://www.osti.gov/biblio/15010671.

[25] E. Hammann, A. Behrendt, F. Le Mounier, and V. Wulfmeyer. Temperature profiling of the atmospheric boundary layer with rotational raman lidar during the hd(cp)^2 observational prototype experiment. *Atmospheric Chemistry and Physics*, 15(5):2867–2881, 2015. doi: 10.5194/acp-15-2867-2015. URL https://acp.copernicus.org/articles/15/2867/2015/.

[26] M.L. Chanin, A. Garnier, A. Hauchecorne, and J. Porteneuve. A doppler lidar for measuring winds in the middle atmosphere. *Geophysical Research Letters*, 16(11):1273–1276, 1989. doi: 10.1029/GL016i011p01273.

[27] A. Ansmann, J. Fruntke, and R. Engelmann. Updraft and downdraft characterization with doppler lidar: cloud-free versus cumuli-topped mixed layer. *Atmospheric Chemistry and Physics*, 10(16):7845–7858, 2010. doi: 10.5194/acp-10-7845-2010. URL https://acp.copernicus.org/articles/10/7845/2010/.

[28] X. Chu and G.C. Papen. *Resonance Fluorescence Lidar for Measurements of the Middle and Upper Atmospherer*, pages 197–450. CRC Press, 2005. ISBN 9780429135743. doi: 10.1201/9781420030754-8.

[29] Volker Wulfmeyer, R. Michael Hardesty, David D. Turner, Andreas Behrendt, Maria P. Cadeddu, Paolo Di Girolamo, Peter Schlüssel, Joël Van Baelen, and Florian 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. *Reviews of Geophysics*, 53(3):819–895, 2015. doi: https://doi.org/10.1002/2014RG000476. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2014RG000476.

[30] J. E. M. Goldsmith, Forest H. Blair, Scott E. Bisson, and David D. Turner. Turn-key raman lidar for profiling atmospheric water vapor, clouds, and aerosols. *Appl. Opt.*, 37(21):4979–4990, Jul 1998. doi: 10.1364/AO.37.004979. URL http://ao.osa.org/abstract.cfm?URI=ao-37-21-4979.

[31] Jens Reichardt, Ulla Wandinger, Volker Klein, Ina Mattis, Bernhard Hilber, and Robert Begbie. Ramses: German meteorological service autonomous raman lidar for water vapor, temperature, aerosol, and cloud measurements. *Appl. Opt.*, 51(34):8111–8131, Dec 2012. doi: 10.1364/AO.51.008111. URL http://ao.osa.org/abstract.cfm?URI=ao-51-34-8111.

[32] 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. *Journal of Atmospheric and Oceanic Tehcnology*, 28(2):131–147, Feb 2011. doi: 10.1175/2010JTECHA1452.1.

[33] Kevin S. Repasky, Drew Moen, Scott Spuler, Amin R. Nehrir, and John L. Carlsten. Progress towards an autonomous field deployable diode-laser-based differential absorption lidar (dial) for profiling water vapor in the lower troposphere. *Remote Sensing*, 5(12):6241–6259, 2013. ISSN 2072-4292. doi: 10.3390/rs5126241. URL https://www.mdpi.com/2072-4292/5/12/6241.

[34] S. M. 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. *Atmospheric Measurement Techniques*, 8(3):1073–1087, 2015. doi: 10.5194/amt-8-1073-2015. URL https://amt.copernicus.org/articles/8/1073/2015/.

[35] M. Hayman and S. Spuler. Demonstration of a diode-laser-based high spectral resolution lidar (hsrl) for quantitative profiling of clouds and aerosols. *Opt. Express*, 25(24):A1096–A1110, Nov 2017. doi: 10.1364/OE.25.0A1096. URL http://www.opticsexpress.org/abstract.cfm?URI=oe-25-24-A1096.

[36] O. Cruikshank, L. Colberg, K. Repasky, R. Stillwell, M. Hayman, and S. Spuler. Modeling the micro-pulse dial (mpd) performance at low ranges, and improving temperature profiling retrieval algorithms. American Meterological Society 100th Annual Meeting, January 13, 2021. URL https://ams.confex.com/ams/101ANNUAL/meetingapp.cgi/Paper/381873.

[37] 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, Apr 1993. doi: 10.1175/1520-0426(1993)010⟨0165:DALMOA⟩2.0.CO;2.

[38] J. Bösenberg. Ground-based differential absorption lidar for water-vapor and temperature profiling: methodology. *Appl. Opt.*, 37(18):3845–3860, Jun 1998. doi: 10.1364/AO.37.003845. URL http://ao.osa.org/abstract.cfm?URI=ao-37-18-3845.

[39] T. Fujii and T. Fukuchi. *Laser Remote Sensing*. CRC Press, 2005. ISBN 9780429135743. doi: 10.1201/9781420030754.

[40] P. Dong and Q. Chen. *LIDAR Remote Sensing and Applications*. CRC Press, 2018. ISBN 9781138747241.

[41] C. Bohren and D. Huffman. *Absorption and Scattering of Light by Small Particles*. Wiley, Mar 1998. ISBN 978-0-471-29340-8.

[42] Andrew T. Young. Rayleigh scattering. *Appl. Opt.*, 20(4):533–535, Feb 1981. doi: 10.1364/AO.20.000533. URL `http://ao.osa.org/abstract.cfm?URI=ao-20-4-533`.

[43] Alain Polian. Brillouin scattering at high pressure: an overview. *Journal of Raman Spectroscopy*, 34(7-8):633–637, 2003. doi: https://doi.org/10.1002/jrs.1031. URL `https://analyticalsciencejournals.onlinelibrary.wiley.com/doi/abs/10.1002/jrs.1031`.

[44] C. D. Boley, Rashmi C. Desai, and G. Tenti. Kinetic models and brillouin scattering in a molecular gas. *Canadian Journal of Physics*, 50(18):2158–2173, 1972. doi: 10.1139/p72-286.

[45] G. Tenti, C. D. Boley, and Rashmi C. Desai. On the kinetic model description of rayleigh–brillouin scattering from molecular gases. *Canadian Journal of Physics*, 52(4):285–290, 1974. doi: 10.1139/p74-041.

[46] Xingguo Pan, Mikhail N. Shneider, and Richard B. Miles. Coherent rayleigh-brillouin scattering. *Phys. Rev. Lett.*, 89:183001, Oct 2002. doi: 10.1103/PhysRevLett.89.183001.

[47] Xingguo Pan, Mikhail N. Shneider, and Richard B. Miles. Coherent rayleigh-brillouin scattering in molecular gases. *Phys. Rev. A*, 69:033814, Mar 2004. doi: 10.1103/PhysRevA.69.033814.

[48] Xingguo Pan, Mikhail N. Shneider, and Richard B. Miles. Power spectrum of coherent rayleigh-brillouin scattering in carbon dioxide. *Phys. Rev. A*, 71:045801, Apr 2005. doi: 10.1103/PhysRevA.71.045801.

[49] Y. Wang, Z. Gu, K. Liang, and W. Ubachs. Rayleigh-brillouin light scattering spectroscopy of air; experiment, predictive model and dimensionless scaling. *Molecular Physics*, 119(1-2):e1804635, Aug 2020. ISSN 1362-3028. doi: 10.1080/00268976.2020.1804635. URL `http://dx.doi.org/10.1080/00268976.2020.1804635`.

[50] J. Smith. Coherent spontaneous rayleigh-brillouin scattering spectra. MATLAB Central File Exchange, 2021. URL `https://www.mathworks.com/matlabcentral/fileexchange/29108-coherent-spontaneous-rayleigh-brillouin-scattering-spectra`.

[51] F. White. *Viscous Fluid Flow*. McGraw Hill, 2006. ISBN 007-124493-X.

[52] D.A. Long. *The Raman Effect: A Unified Treatment of the Theory of Raman Scattering by Molecules*. Wiley, 2002. ISBN 978-0471490289. doi: 10.1002/0470845767.

[53] F.G. Fernald, B.M. Herman, and J.A. Reagan. Determination of aerosol height distributions by lidar. *Journal of Applied Meteorology and Climatology*, 11(3):482 – 489, 1972. doi: 10.1175/1520-0450(1972)011⟨0482:DOAHDB⟩2.0.CO;2.

[54] James D. Klett. Stable analytical inversion solution for processing lidar returns. *Appl. Opt.*, 20(2):211–220, Jan 1981. doi: 10.1364/AO.20.000211. URL `http://www.osapublishing.org/ao/abstract.cfm?URI=ao-20-2-211`.

[55] A. Bucholtz. Rayleigh-scattering calculations for the terrestrial atmosphere. *Applied Optics*, 34(15):2765–2773, May 1995. doi: 10.1364/AO.34.002765.

[56] G. Fiocco, G. Benedetti-Michaelangeli, K. Maishberger, and E. Madonna. Rayleigh-scattering calculations for the terrestrial atmosphere. *Nature Physical Science*, 229(3): 78–79, 1971. doi: 10.1038/physci229078a0.

[57] H. Shimizu, S. A. Lee, and C. Y. She. High spectral resolution lidar system with atomic blocking filters for measuring atmospheric parameters. *Appl. Opt.*, 22(9):1373–1381, May 1983. doi: 10.1364/AO.22.001373.

[58] P. Piironen and E. W. Eloranta. Demonstration of a high-spectral-resolution lidar based on an iodine absorption filter. *Opt. Lett.*, 19(3):234–236, Feb 1994. doi: 10.1364/OL.19.000234.

[59] J. Bösenberg. *Differential-Absorption Lidar for Water Vaporand Temperature Profiling*, pages 213–228. Springer New York, New York, NY, 2005. ISBN 978-0-387-25101-1. doi: 10.1007/0-387-25101-4_5.

[60] S. M. Spuler, M. Hayman, R. A. Stillwell, J. Carnes, T. Bernatsky, and K. S. Repasky. Micropulse dial (mpd) – a diode-laser-based lidar architecture for quantitative atmospheric profiling. *Atmospheric Measurement Techniques*, 14(6):4593–4616, 2021. doi: 10.5194/amt-14-4593-2021. URL `https://amt.copernicus.org/articles/14/4593/2021/`.

[61] Ioannis Binietoglou, Giuseppe D´Amico, Holger Baars, Livio Belegante, and Eleni Marinou. A methodology for cloud masking uncalibrated lidar signals. *EPJ Web Conf.*, 176:05048, 2018. doi: 10.1051/epjconf/201817605048.

[62] Y. Liu, S. Goodrick, and W. Heilman. Wildland fire emissions, carbon, and climate: Wildfire–climate interactions. *Forest Ecology and Management*, 317:80–96, 2014. ISSN 0378-1127. doi: https://doi.org/10.1016/j.foreco.2013.02.020. Wildland fire emissions, carbon, and climate: Science overview and knowledge needs.

[63] J.L. Heffter. Transport layer depth calculations. Second Joint Conference on Applications of Air Pollution Meteorology, New Orleans, LA, 1980.

[64] S. Liu and X. . Observed diurnal cycle climatology of planetary boundary layer height. *Journal of Climate*, 23(21):5790 – 5809, 2010. doi: 10.1175/2010JCLI3552.1.

[65] C. Sivaraman, S. McFarlane, E. Chapman, M. Jensen, T. Toto, S. Liu, and M. Fischer. Planetary boundary layer (pbl) height value added product (vap): Radiosonde retrievals. Technical Report 1, Office of Science, U.S. Department of Energy, Aug 2013.

[66] R. Dang, Y. Yang, H. Li, X. Hu, Z. Wang, Z. Huang, T. Zhou, and T. Zhang. Atmosphere boundary layer height (ablh) determination under multiple-layer conditions using micro-pulse lidar. *Remote Sensing*, 11(3), 2019. ISSN 2072-4292. doi: 10.3390/rs11030263.

[67] A.K. Piironen and E.W. Eloranta. Convective boundary layer mean depths and cloud geometrical properties obtained from volume imaging lidar data. *Journal of Geophysical Research: Atmospheres*, 100(D12):25569–25576, 1995. doi: https://doi.org/10.1029/94JD02604.

[68] C. Flamant, J.-P. Chaboureau, D. J. Parker, C. M. Taylor, J.-P. Cammas, O. Bock, F. Timouk, and J. Pelon. Airborne observations of the impact of a convective system on the planetary boundary layer thermodynamics and aerosol distribution in the inter-tropical discontinuity region of the west african monsoon. *Quarterly Journal of the Royal Meteorological Society*, 133(626):1175–1189, 2007. doi: https://doi.org/10.1002/qj.97.

[69] A. J. Scarino, M. D. Obland, J. D. Fast, S. P. Burton, R. A. Ferrare, C. A. Hostetler, L. K. Berg, B. Lefer, C. Haman, J. W. Hair, R. R. Rogers, C. Butler, A. L. Cook, and D. B. Harper. Comparison of mixed layer heights from airborne high spectral resolution lidar, ground-based measurements, and the wrf-chem model during calnex and cares. *Atmospheric Chemistry and Physics*, 14(11):5547–5560, 2014. doi: 10.5194/acp-14-5547-2014.

[70] A. Lammert and J. Bösenberg. Determination of the convective boundary-layer height with laser remote sensing. *Boundary-Layer Meteorology*, 119(1):159–170, Apr 2006. doi: 10.1007/s10546-005-9020-x.

[71] D.G. Steyn, M. Baldi, and R.M. Hoff. *A New Technique To Derive Mixed Layer Depth And Entrainment Zone Thickness From Lidar Profiles*, pages 613–620. Springer US, Boston, MA, 2000. doi: 10.1007/978-1-4615-4153-0_63. URL `https://doi.org/10.1007/978-1-4615-4153-0_63`.

[72] Ian M. Brooks. Finding boundary layer top: Application of a wavelet covariance transform to lidar backscatter profiles. *Journal of Atmospheric and Oceanic Technology*, 20(8):1092 – 1105, 2003. doi: 10.1175/1520-0426(2003)020⟨1092:FBLTAO⟩2.0.CO;2.

[73] H. Baars, A. Ansmann, R. Engelmann, and D. Althausen. Continuous monitoring of the boundary-layer top with lidar. *Atmospheric Chemistry and Physics*, 8(23):7281–7296, 2008. doi: 10.5194/acp-8-7281-2008.

[74] S.A. Cohn and W.M Angevine. Boundary layer height and entrainment zone thickness measured by lidars and wind-profiling radars. *Journal of Applied Meteorology*, 39(8): 1233 – 1247, 2000. doi: 10.1175/1520-0450(2000)039⟨1233:BLHAEZ⟩2.0.CO;2.

APPENDICES

APPENDIX A

ACRONYMS

Acronym	Description
AERONET	Aerosol Robotic Network
AOD	Aerosol Optical Depth
APD	Avalanche Photodiode
ARM	Atmospheric Radiation Measurement
BR	Backscatter Ratio
CI	Capping Inversion
CW	Continuous Wave
DAQ	Data Acquisition
DBR	Distributed Bragg Reflector
DIAL	Differential Absorption Lidar
DLB	Diode-Laser-Based
EBL	Elastic Backscatter Lidar
EZ	Entrainment Zone
FWHM	Full Width at Half Maximum
HSRL	High Spectral Resolution Lidar
HWT	Haar Wavelet Transform
IR	Infrared
MCS	Multichannel Scaler
ML	Mixed Layer
MODIS	Moderate Resolution Imaging Spectroradiometer
MPD	Micropulse DIAL
MPE	Maximum Permissible Exposure
MSU	Montana State University
NaN	Not a Number
NCAR	National Center for Atmospheric Research
NetCDF	Network Common Data Format
NFOV	Narrow Field Of View
O2-DIAL	Oxygen Differential Absorption Lidar
PBL	Planetary Boundary Layer
PBLH	Planetary Boundary Layer Height
RL	Residual Layer
RTD	Resistance Temperature Detector
SGP	Southern Great Plains
SL	Stable Layer
SNR	Signal-to-Noise Ratio
SOA	Semiconductor Optical Amplifier
TEC	Thermoelectric Controller
TSOA	Tapered Semiconductor Optical Amplifier
T/R	Transmit/Receive
TWA	Travelling Wave Amplifier
UTC	Coordinated Universal Time
UV	Ultraviolet
VAP	Value-Added Product
WFOV	Wide Field Of View
WV-DIAL	Water Vapor Differential Absorption Lidar

APPENDIX B

PBLH COMPARISONS

From March 13-June 4, 2021, 30 radiosondes with acceptable data were launched from the roof of Cobleigh Hall. Using the potential temperature profiles and the Heffter PBLH retrieval method, the PBLH was derived from these radiosonde launches. For the 15 radiosondes where the lidar method and the Heffter method agreed on which inversion was the PBLH, the PBLH as determined by lidar was lower in altitude than the Heffter method by a mean difference of 295 m. The comparison of PBLH values retrieved by the Heffter-radiosonde method and the HWT-lidar algorithm is displayed in Figure B.1. The PBLH, as determined by the Heffter method, is compared to the PBLH as determined from the lidar profiles. The profiles for all 30 launches are displayed on the following pages. NaN values mean that the PBLH retrieval method failed to find an inversion layer that met its criteria.

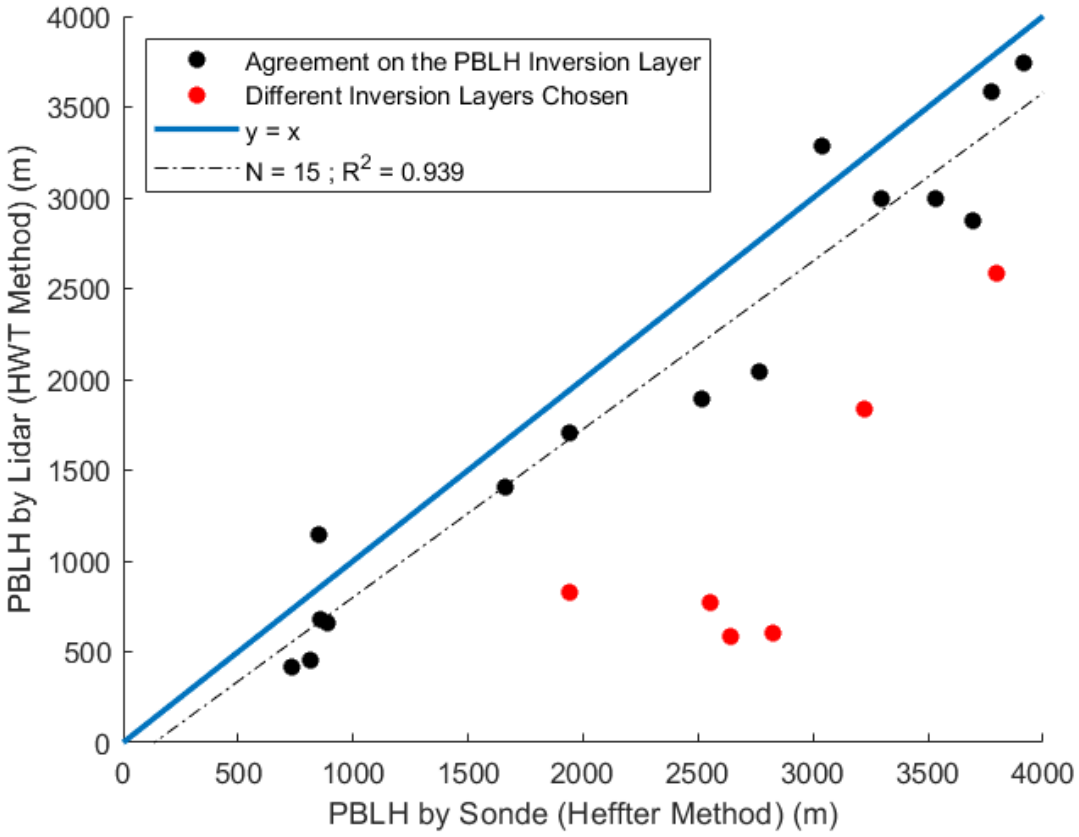


Figure B.1: PBLH values found by the HWT-lidar method and by the Heffter-radiosonde method.

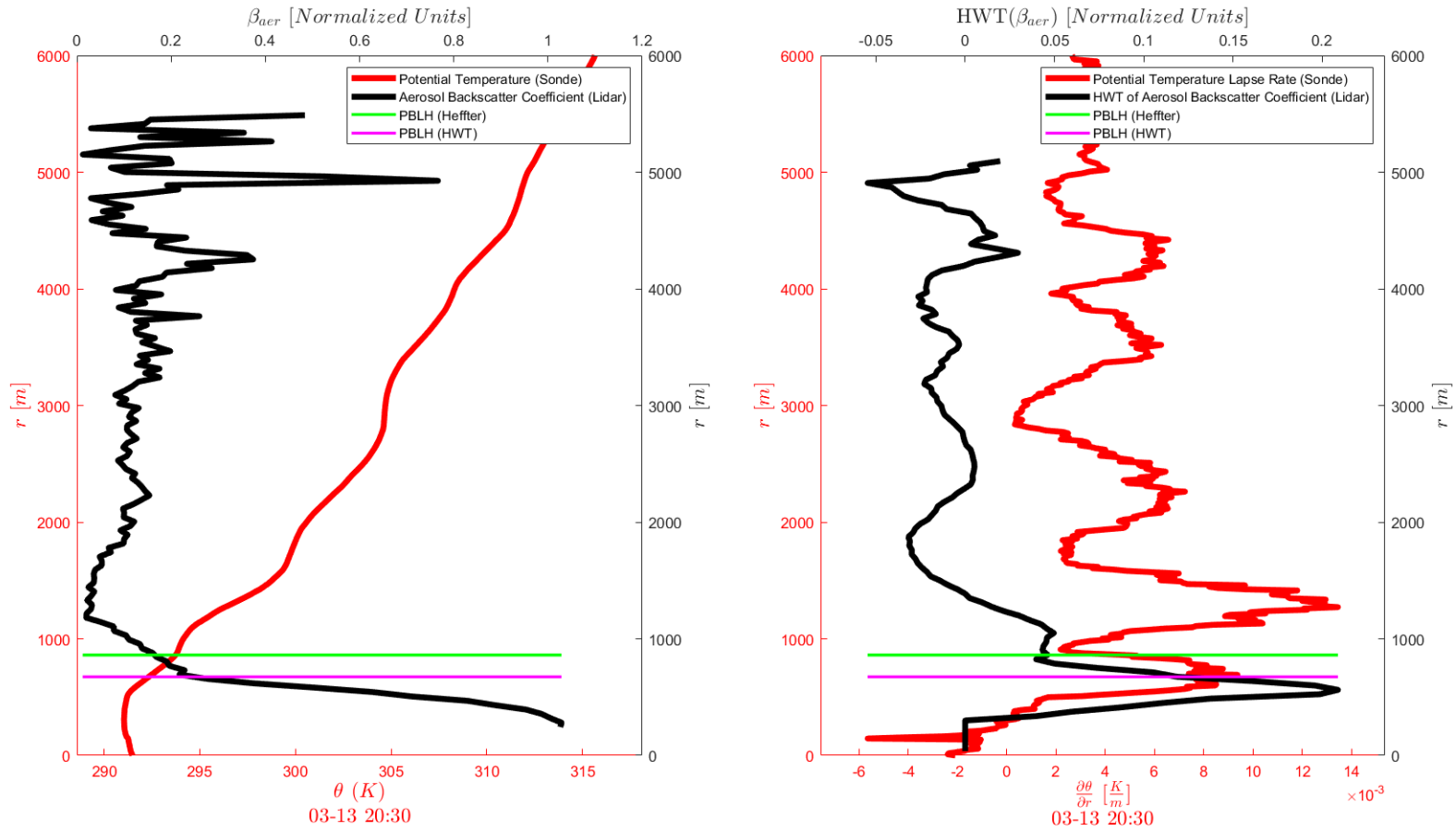


Figure B.2: Radiosonde comparison, March 13, 2021-20:30 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=861 m; PBLH-Lidar=674 m.

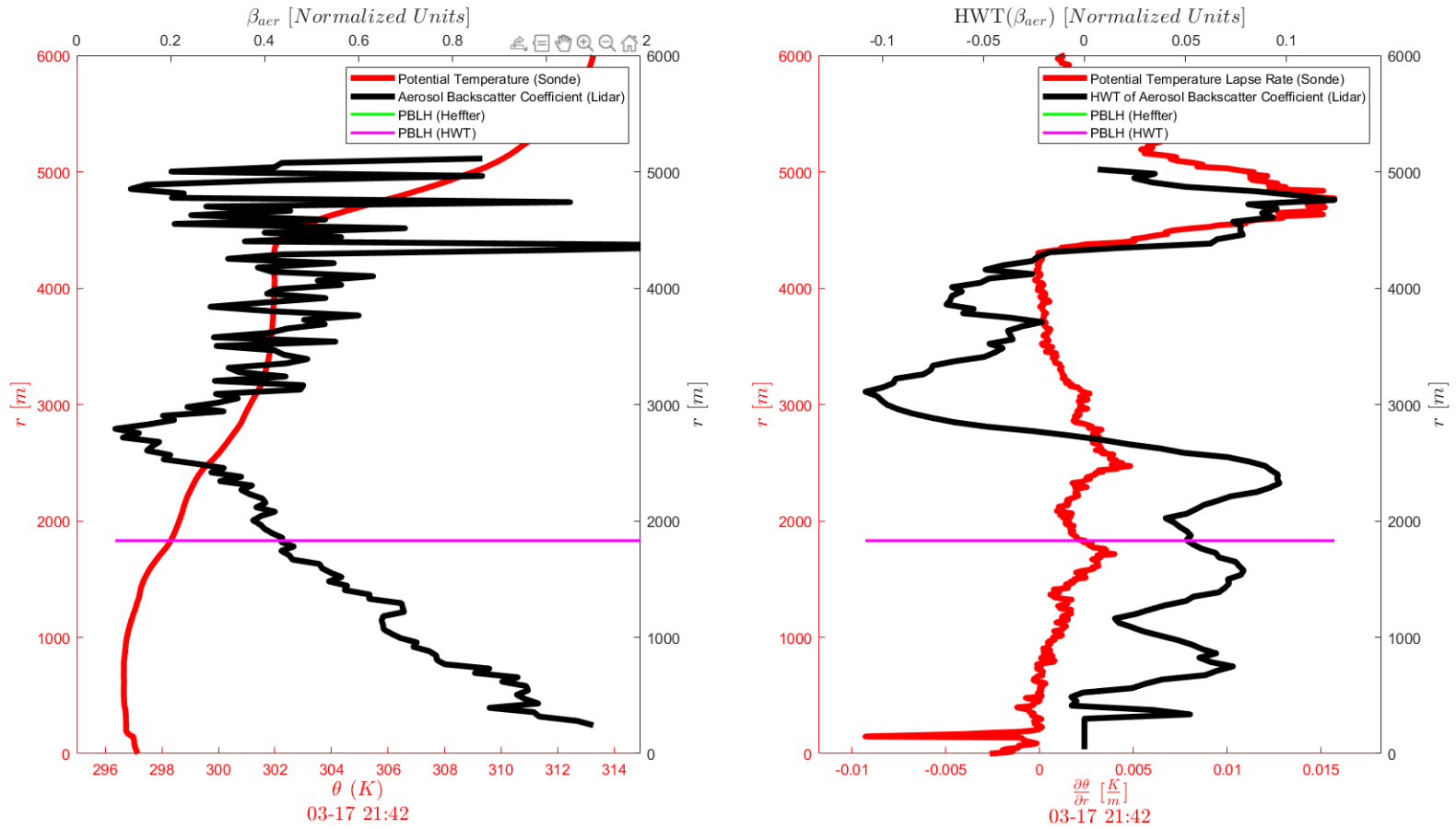


Figure B.3: Radiosonde comparison, March 17, 2021-21:42 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=1830 m.

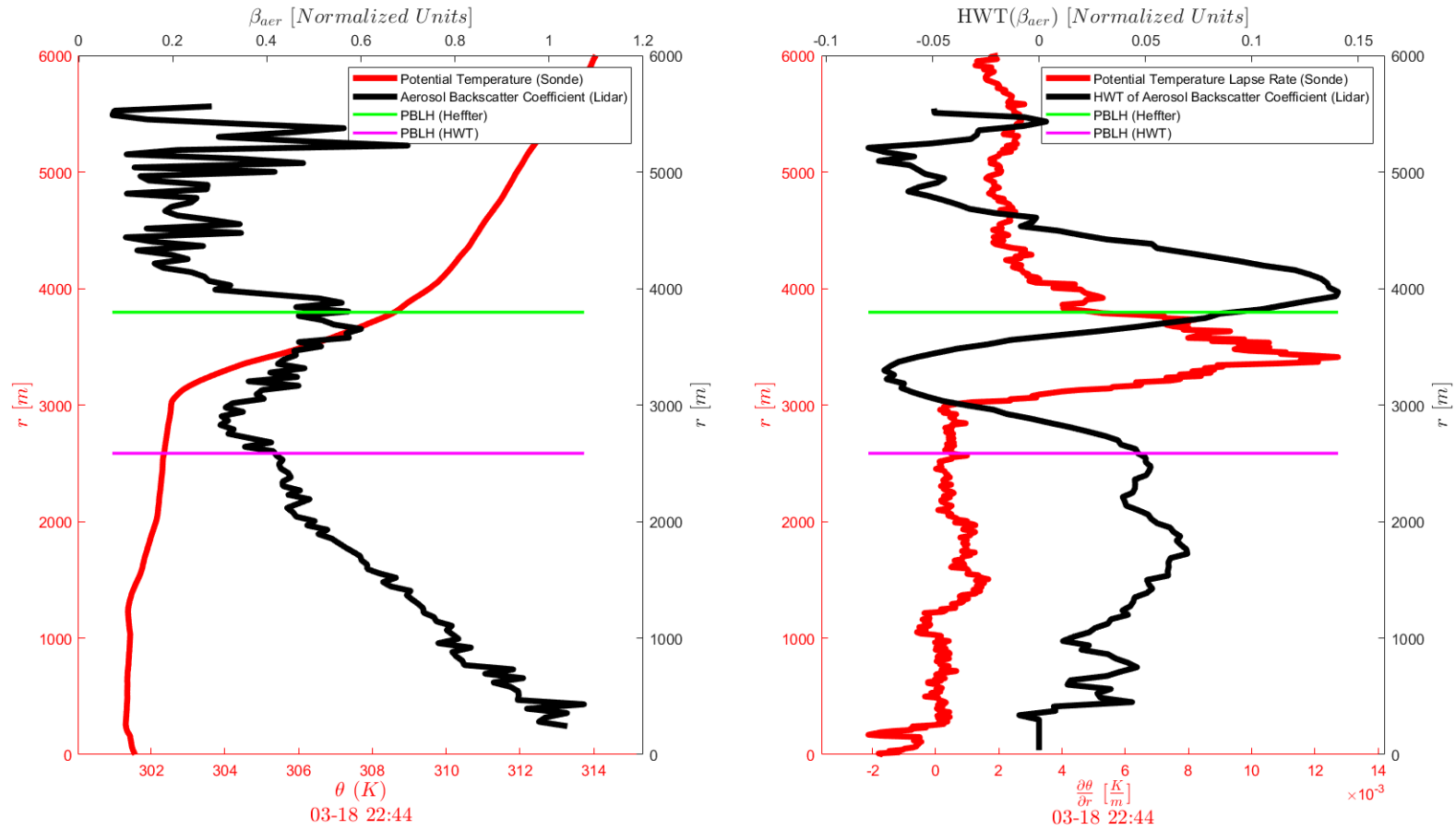


Figure B.4: Radiosonde comparison, March 18, 2021-22:44 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3796 m; PBLH-Lidar=2585 m.

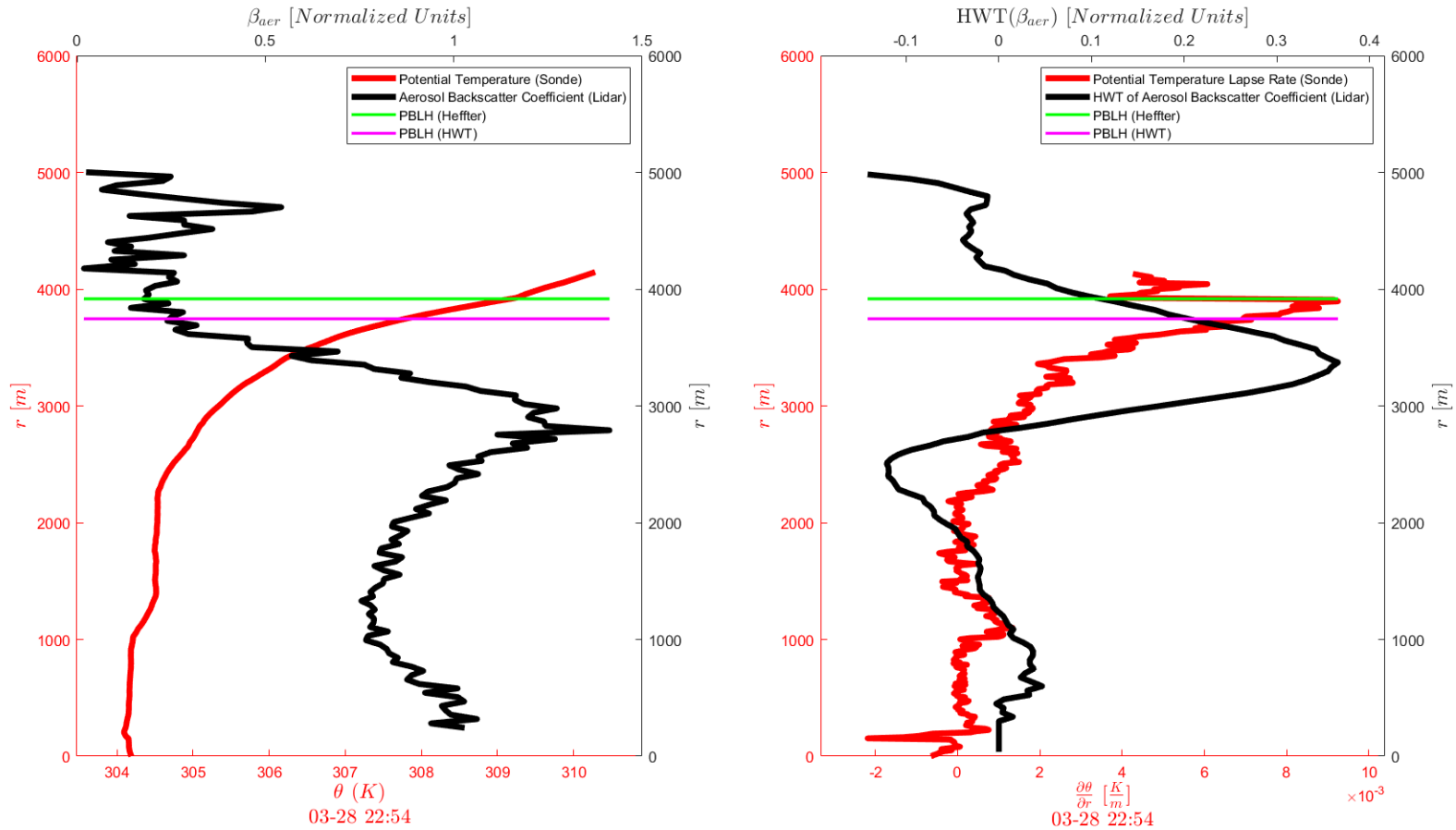


Figure B.5: Radiosonde comparison, March 28, 2021-22:54 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3917 m; PBLH-Lidar=3746 m.

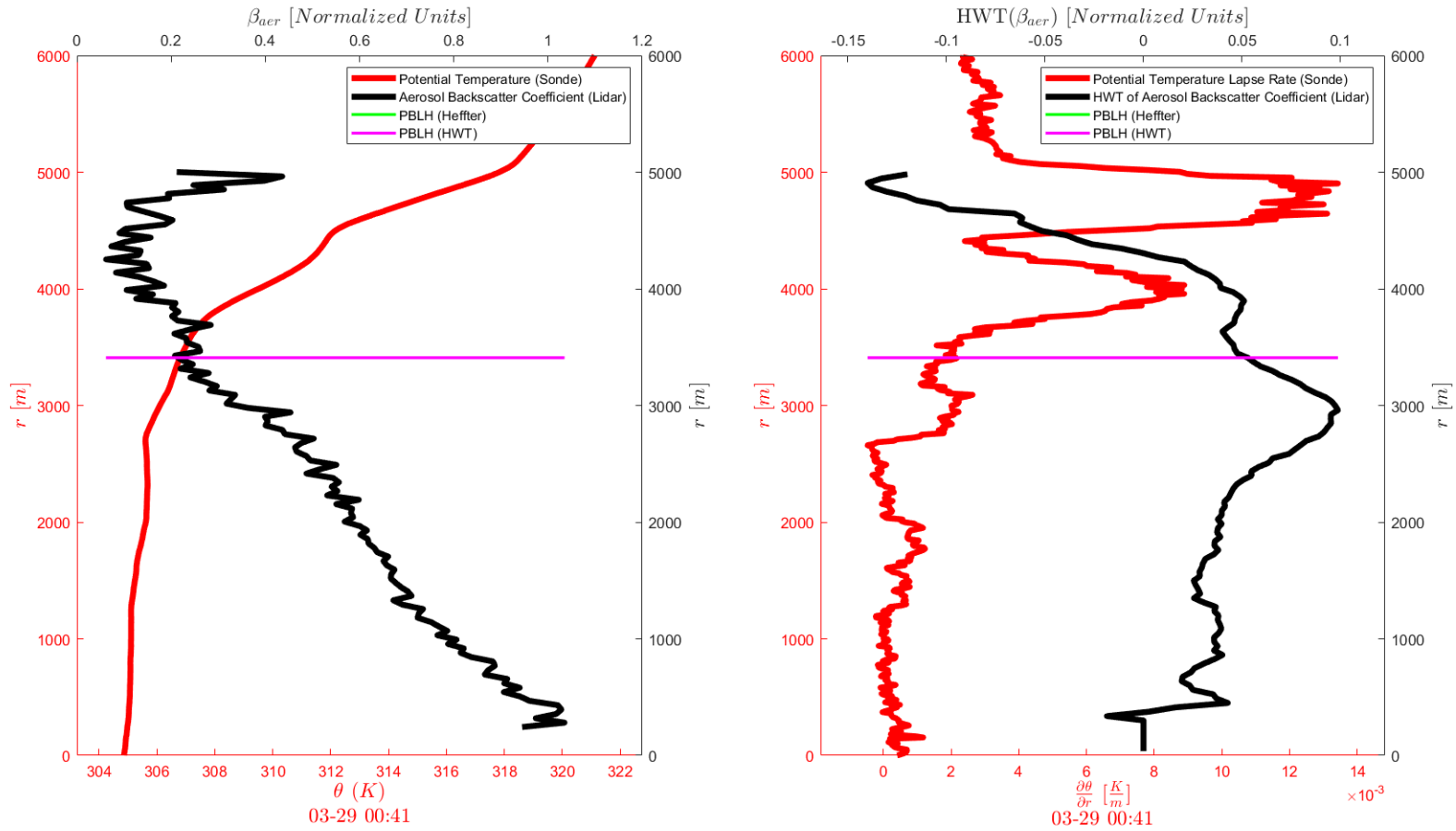


Figure B.6: Radiosonde comparison, March 29, 2021-00:41 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3409 m.

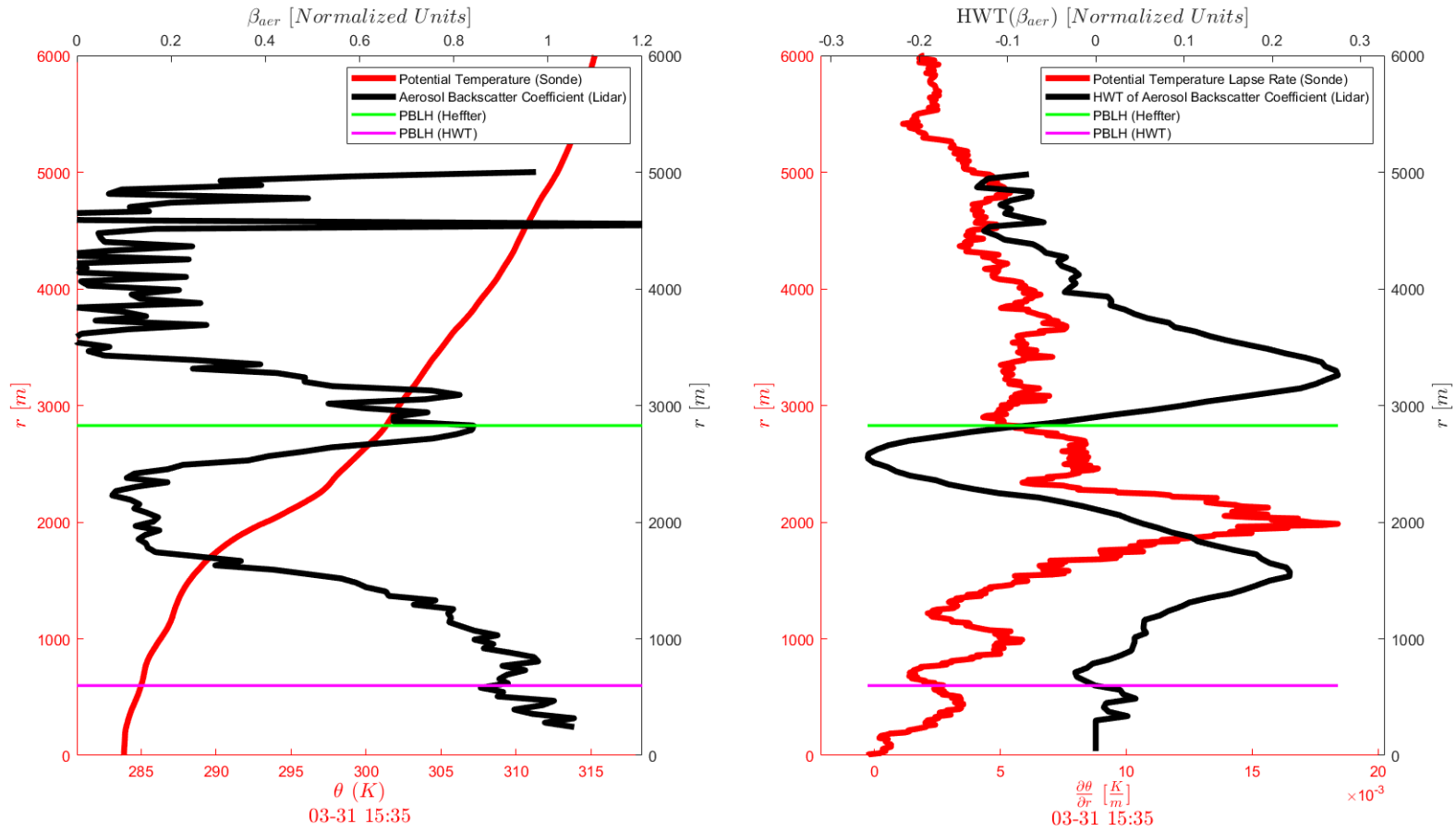


Figure B.7: Radiosonde comparison, March 31, 2021-15:35 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2828 m; PBLH-Lidar=599 m.

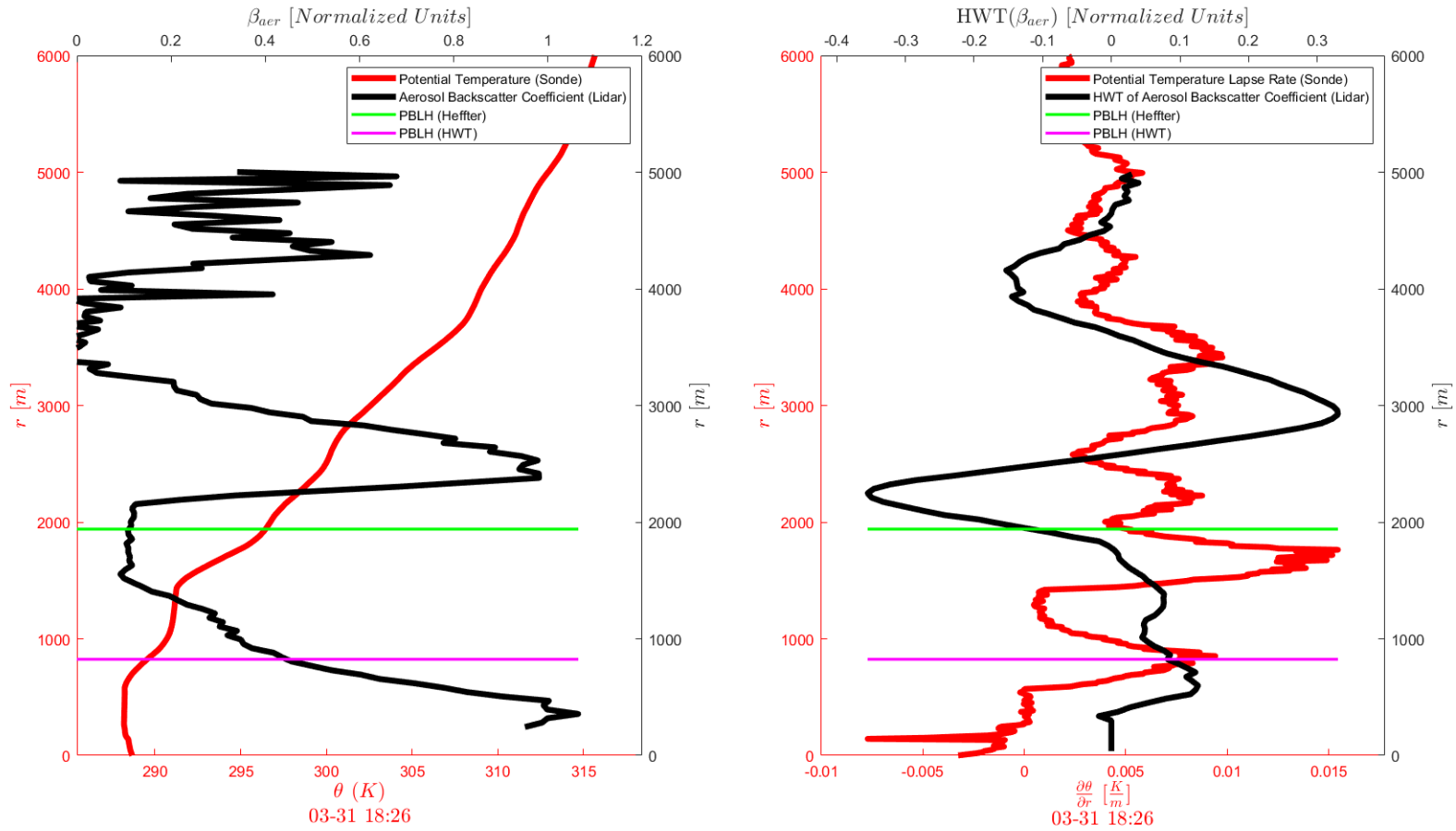


Figure B.8: Radiosonde comparison, March 31, 2021-18:26 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1940 m; PBLH-Lidar=824 m.

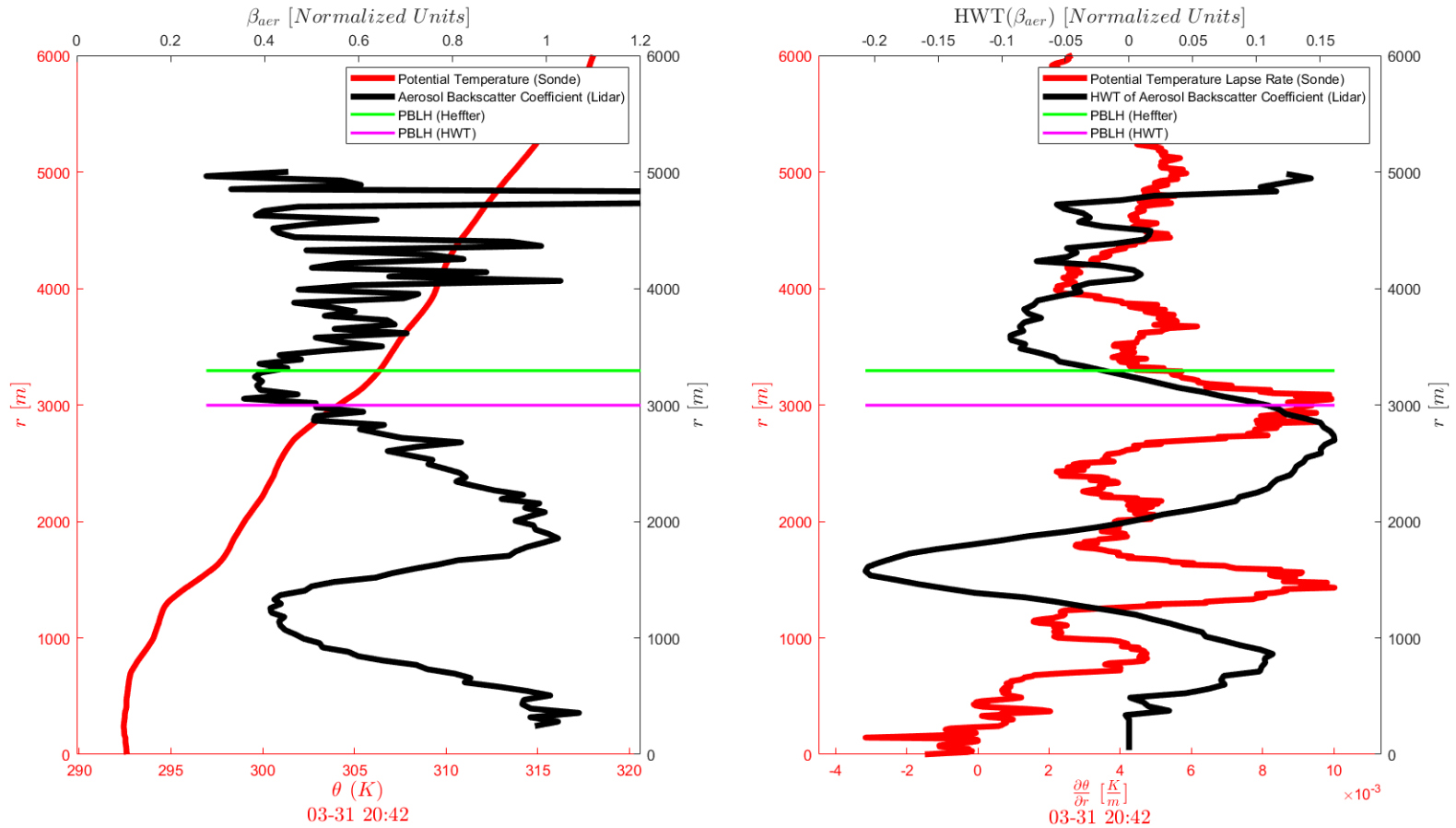


Figure B.9: Radiosonde comparison, March 31, 2021-20:42 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3294 m; PBLH-Lidar=2997 m.

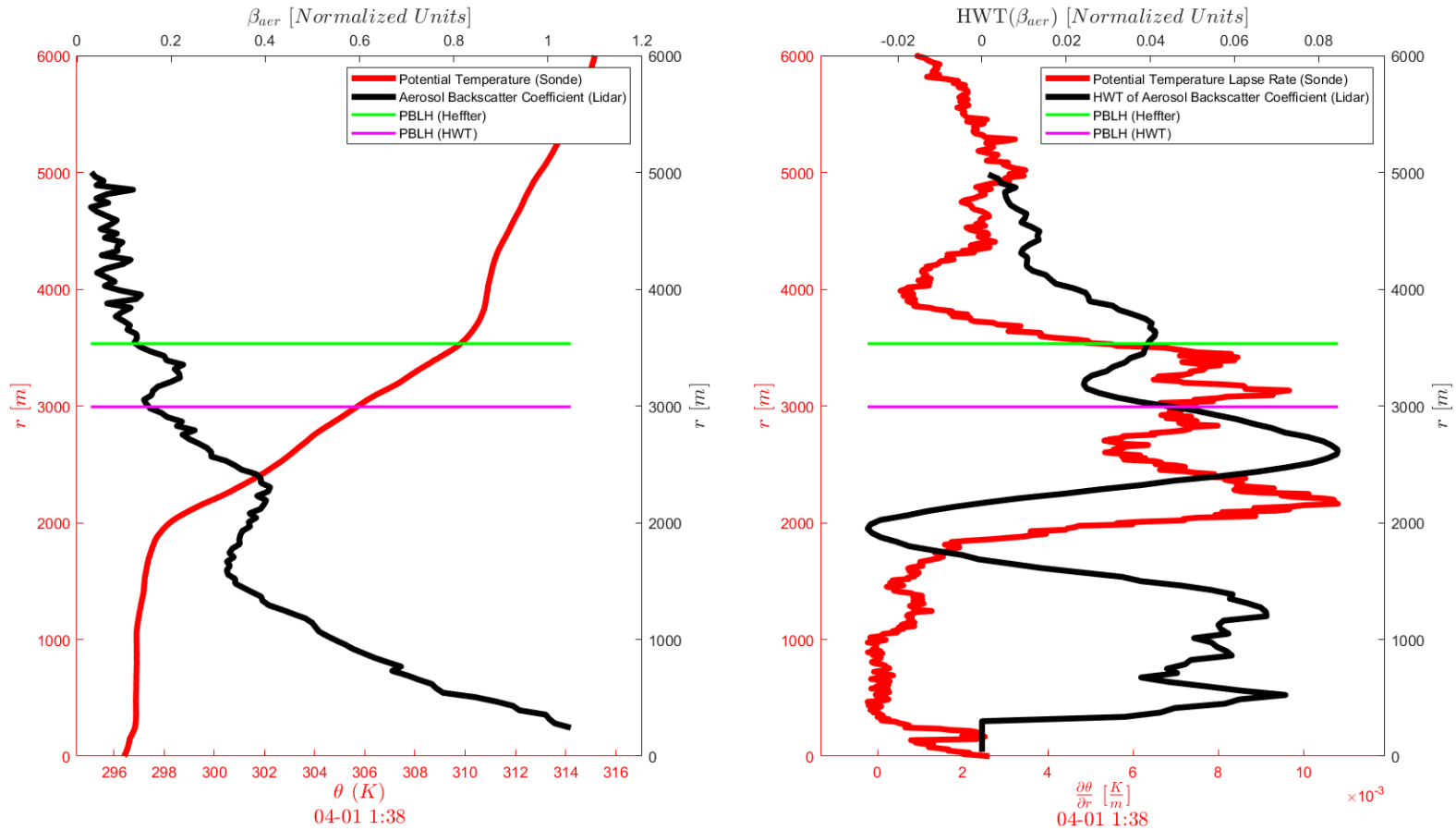


Figure B.10: Radiosonde comparison, April 1, 2021-1:38 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3534 m; PBLH-Lidar=2992 m.

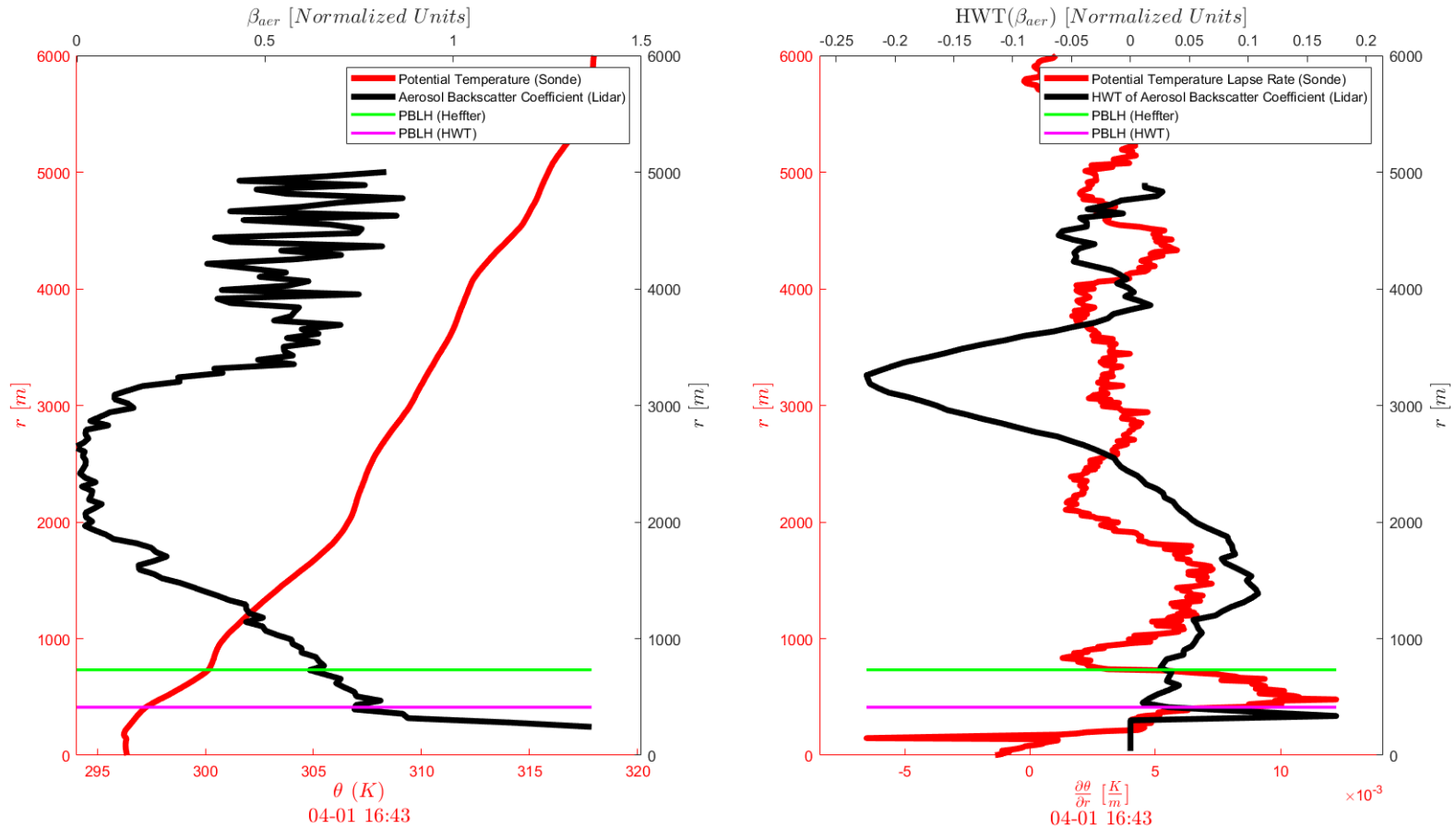


Figure B.11: Radiosonde comparison, April 1, 2021-16:43 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=733 m; PBLH-Lidar=412 m.

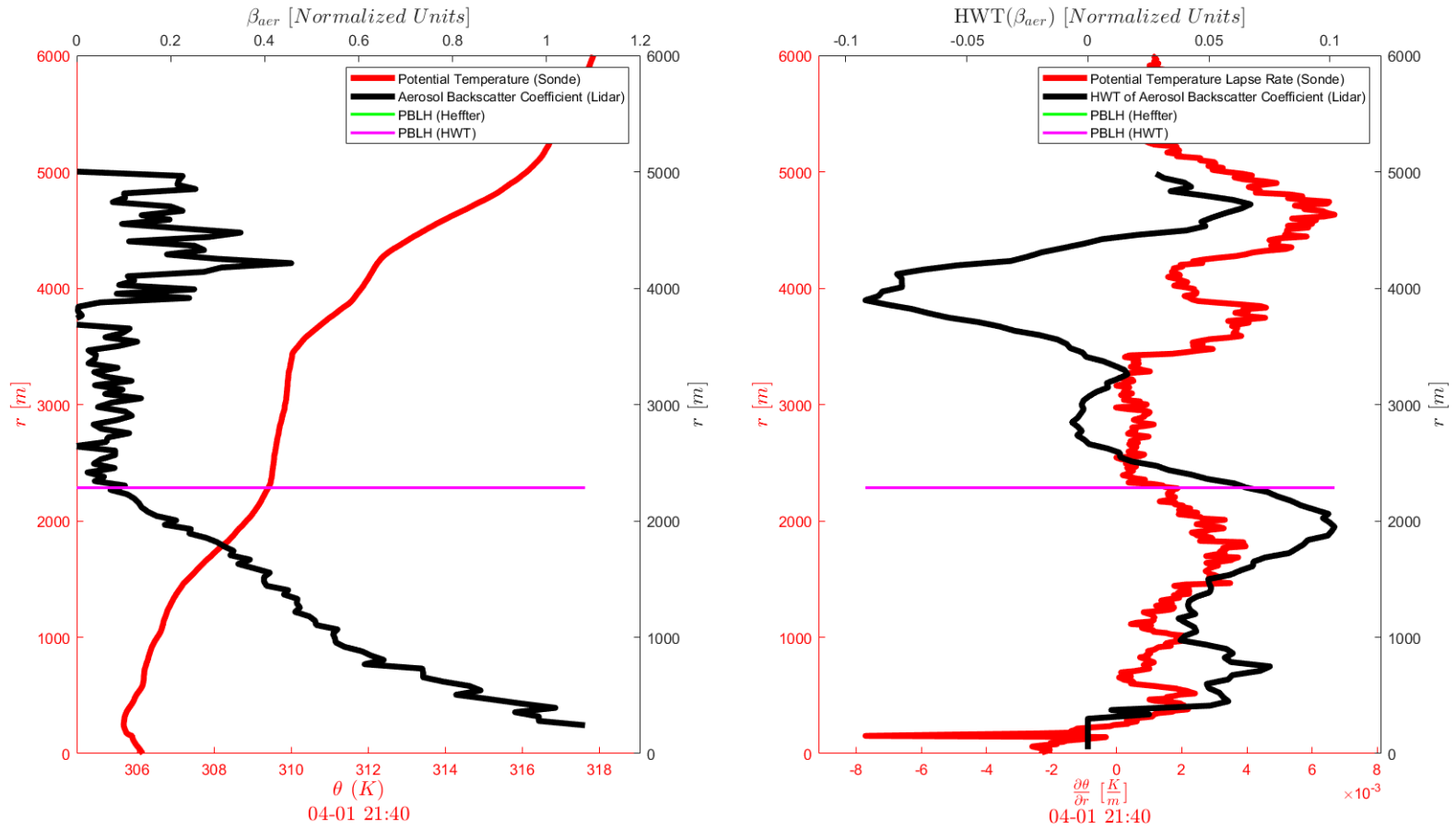


Figure B.12: Radiosonde comparison, April 1, 2021-21:40 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=2285 m.

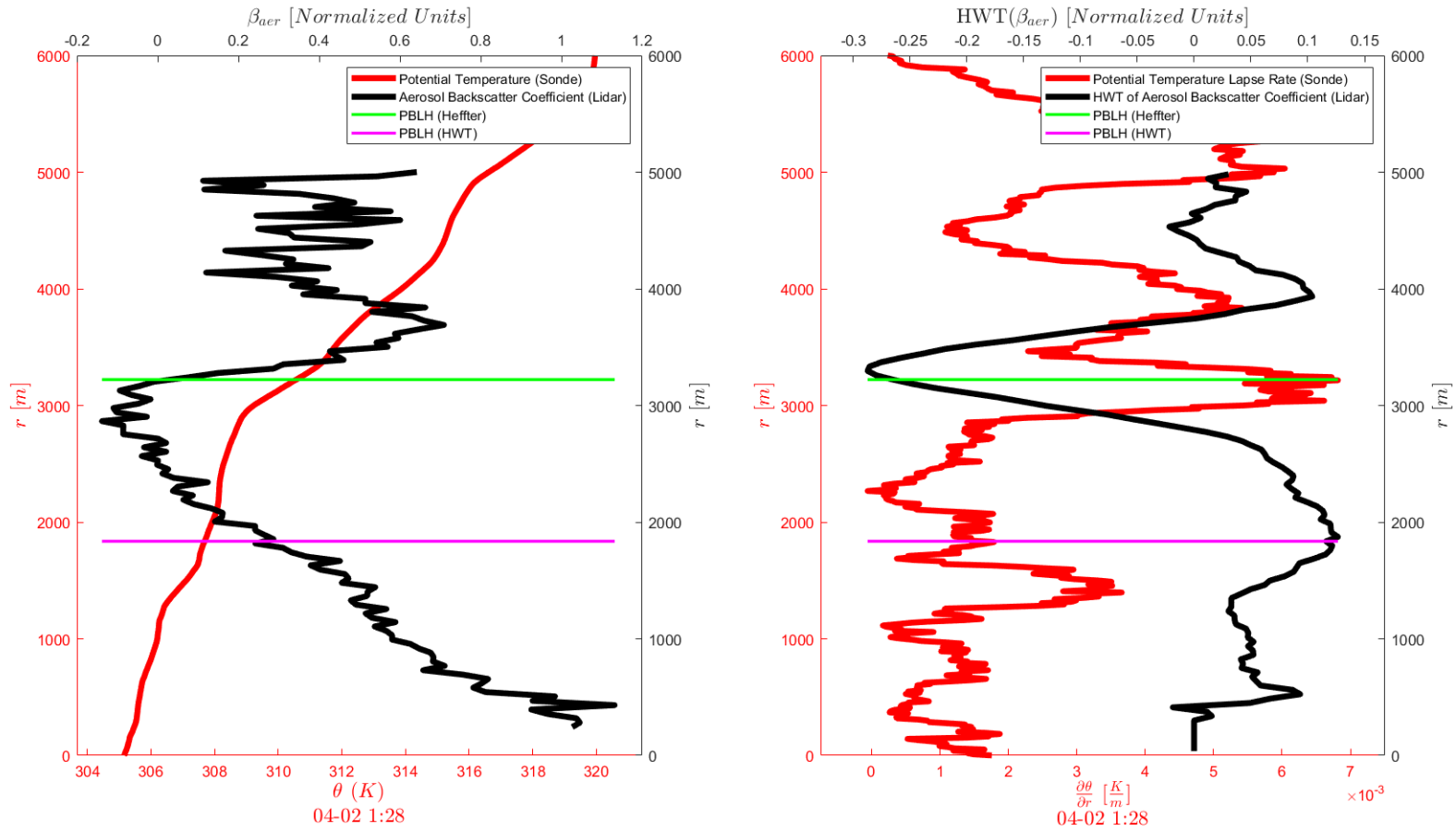


Figure B.13: Radiosonde comparison, April 2, 2021-1:28 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3222 m; PBLH-Lidar=1836 m.

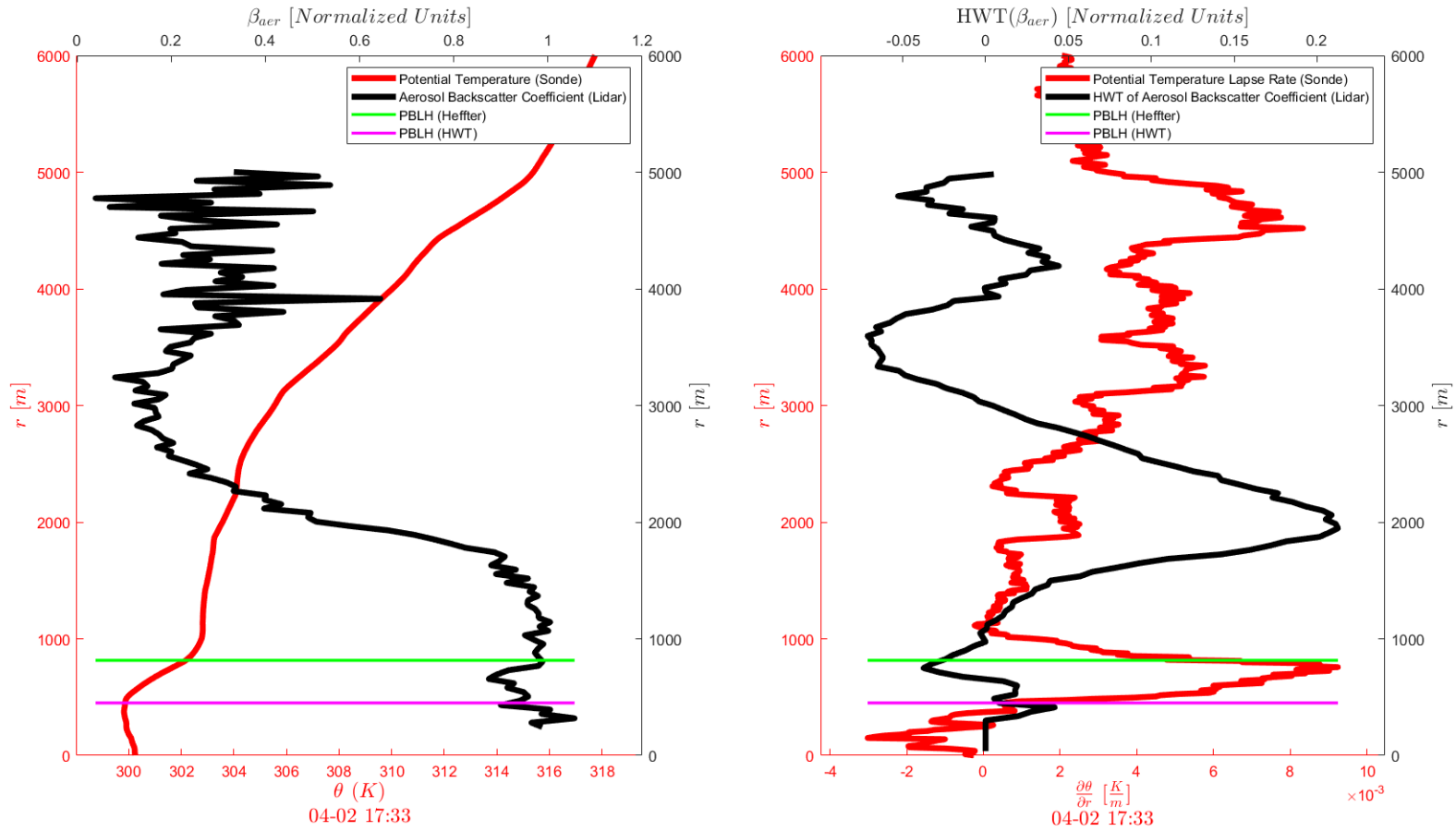


Figure B.14: Radiosonde comparison, April 2, 2021-17:33 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=815 m; PBLH-Lidar=450 m.

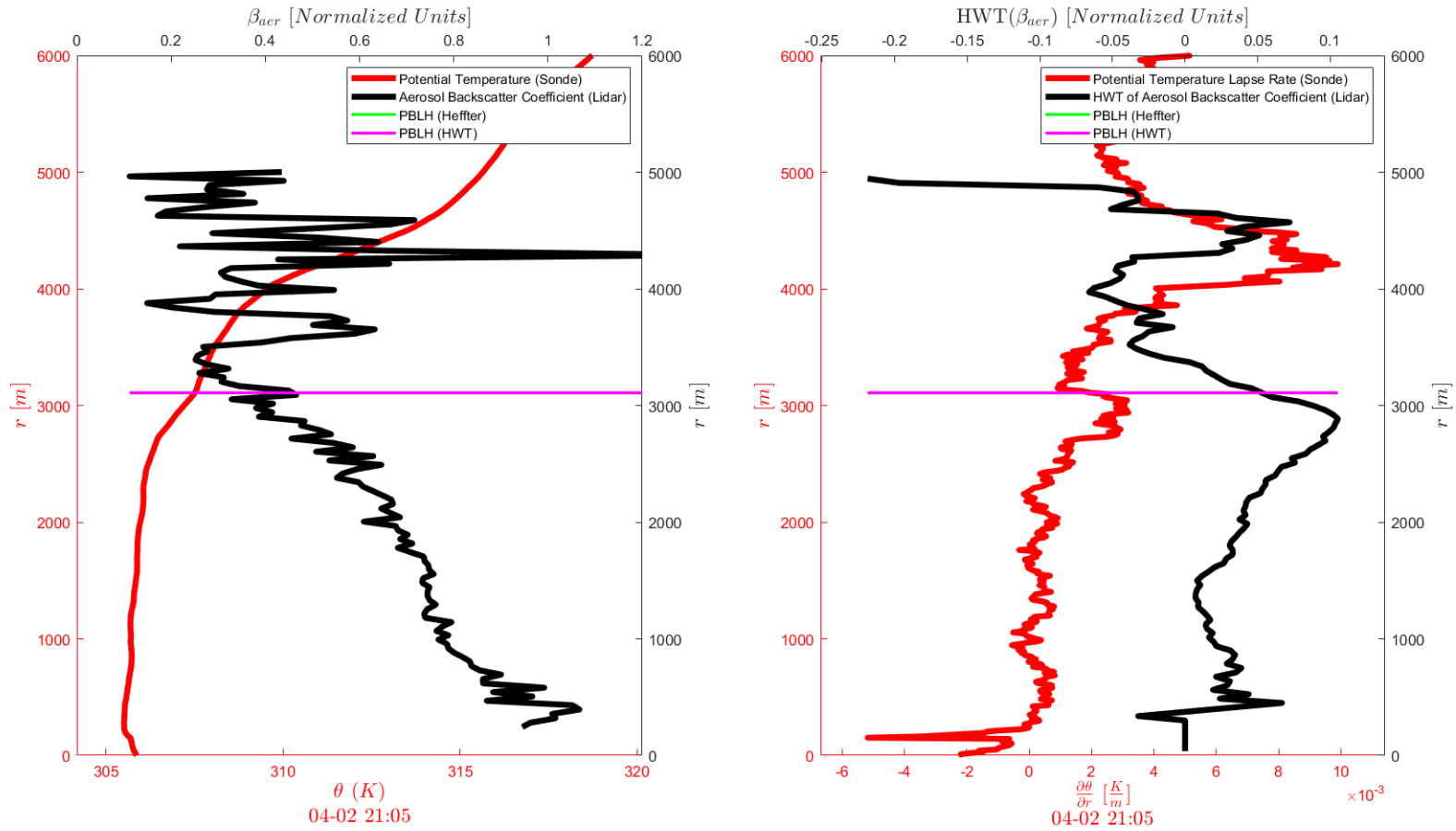


Figure B.15: Radiosonde comparison, April 2, 2021-21:05 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3109 m.

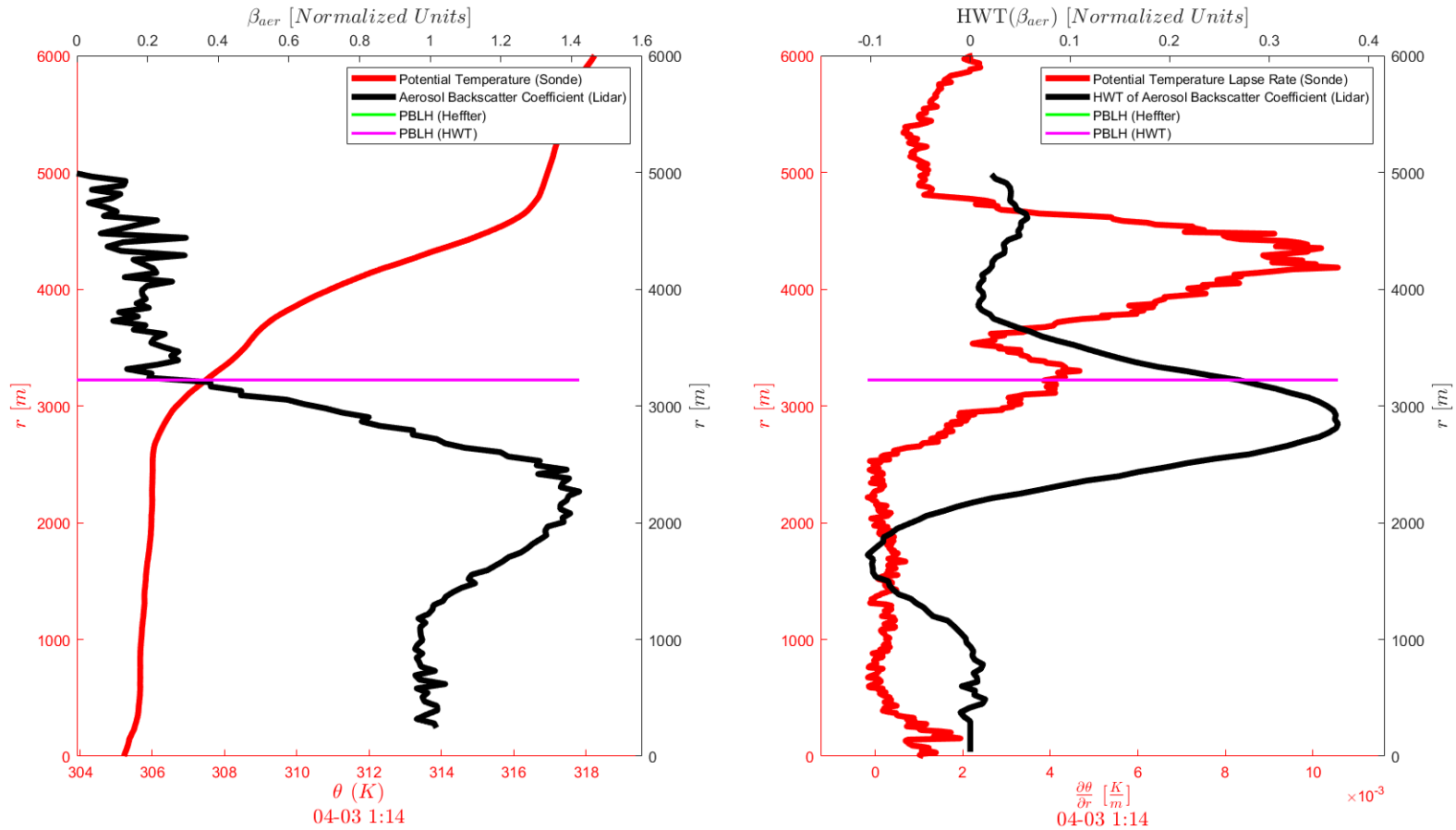


Figure B.16: Radiosonde comparison, April 3, 2021-1:14 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=3222 m.

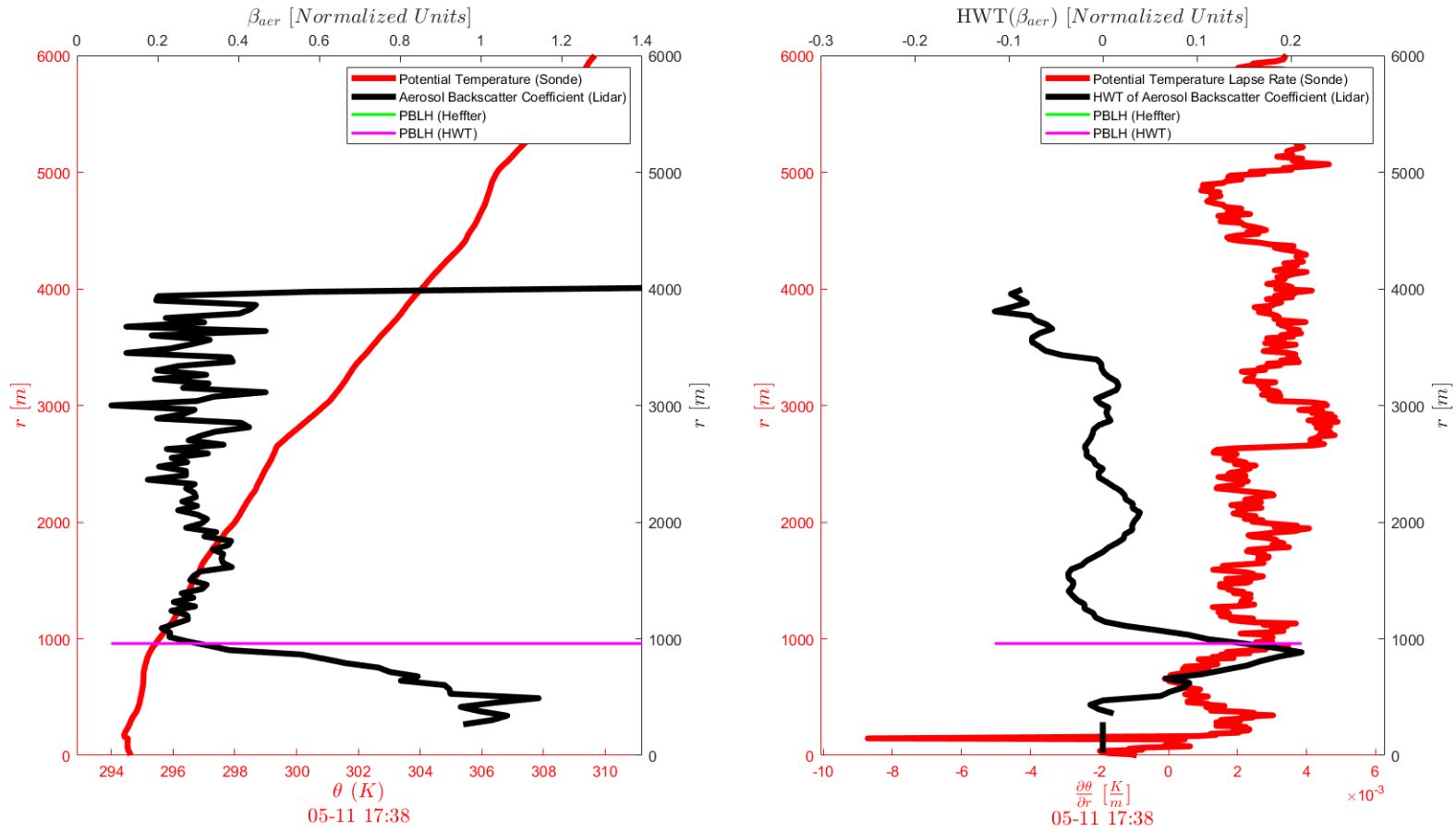


Figure B.17: Radiosonde comparison, May 11, 2021-17:38 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=959 m.

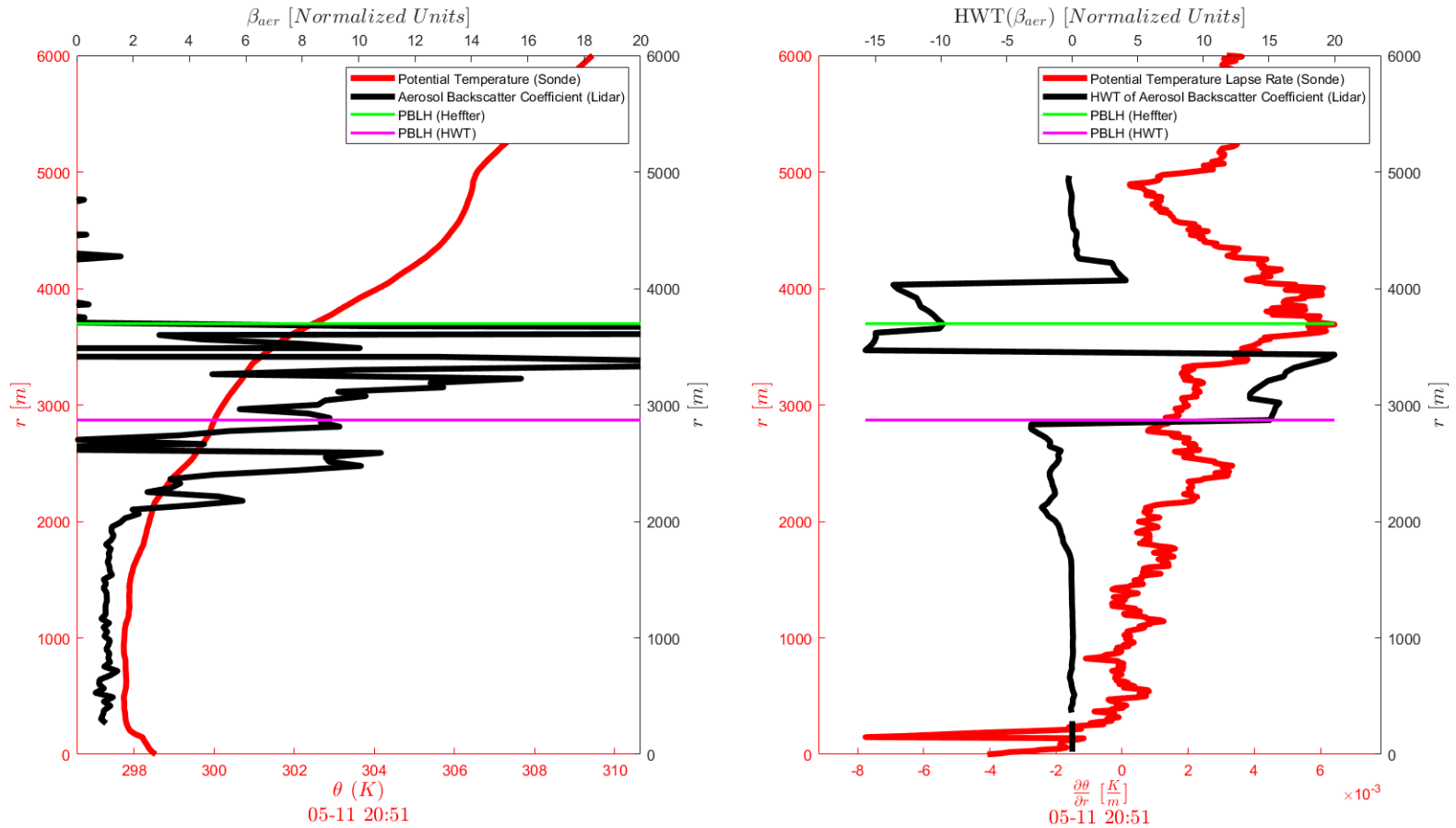


Figure B.18: Radiosonde comparison, May 11, 2021-20:51 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3697 m; PBLH-Lidar=2870 m.

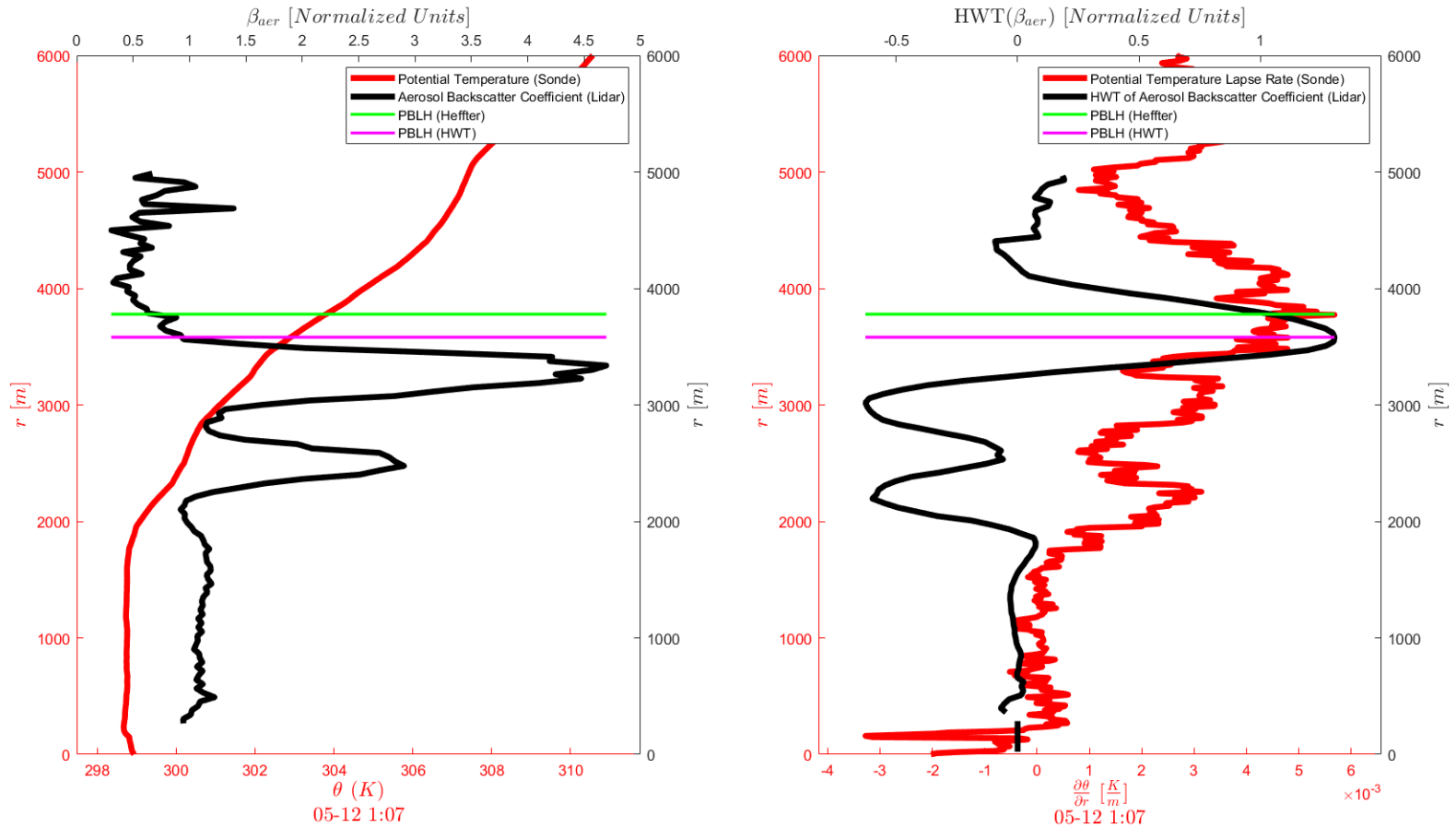


Figure B.19: Radiosonde comparison, May 12, 2021-1:07 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3779 m; PBLH-Lidar=3581 m.

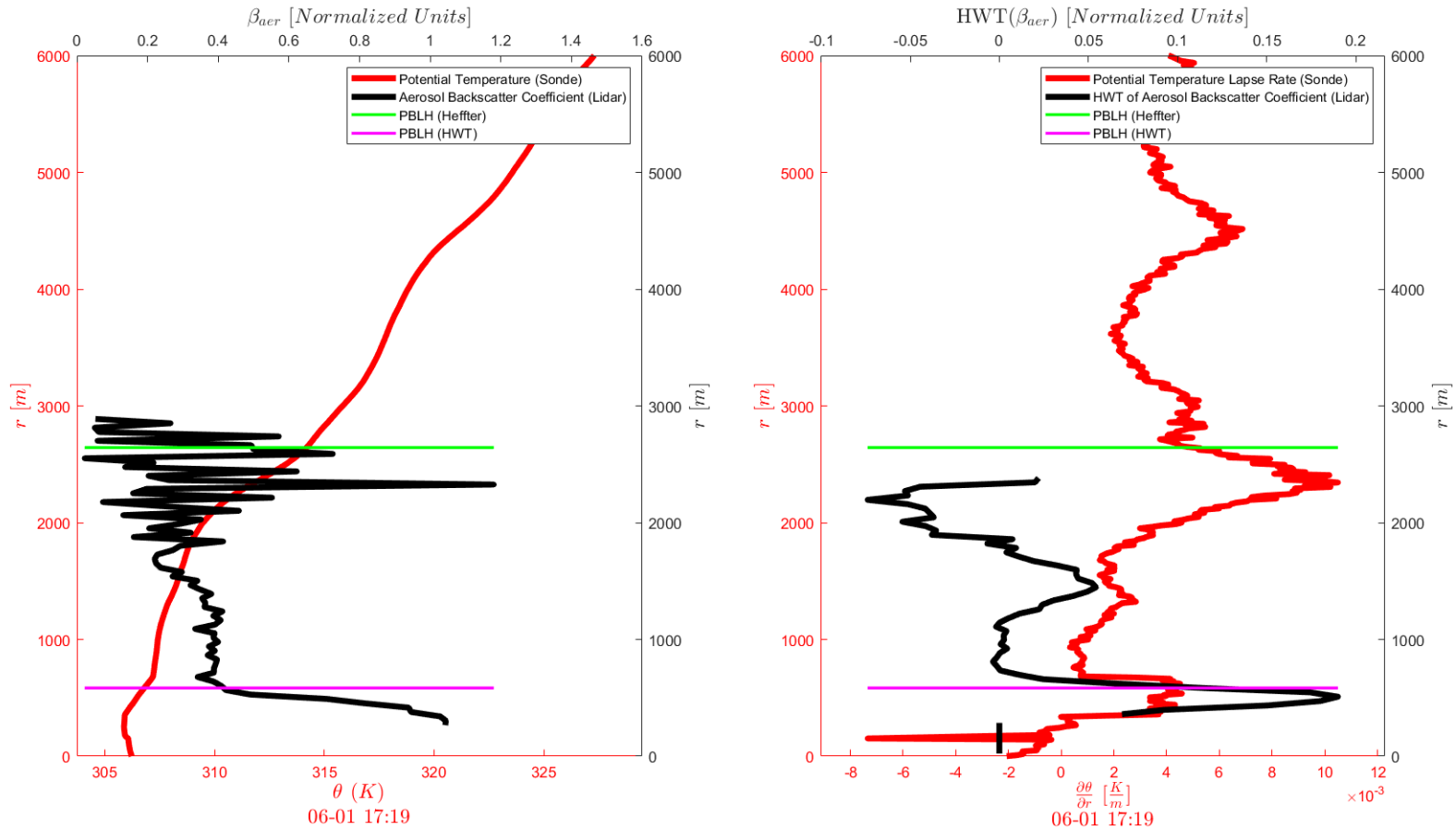


Figure B.20: Radiosonde comparison, June 1, 2021-17:09 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2643 m; PBLH-Lidar=584 m.

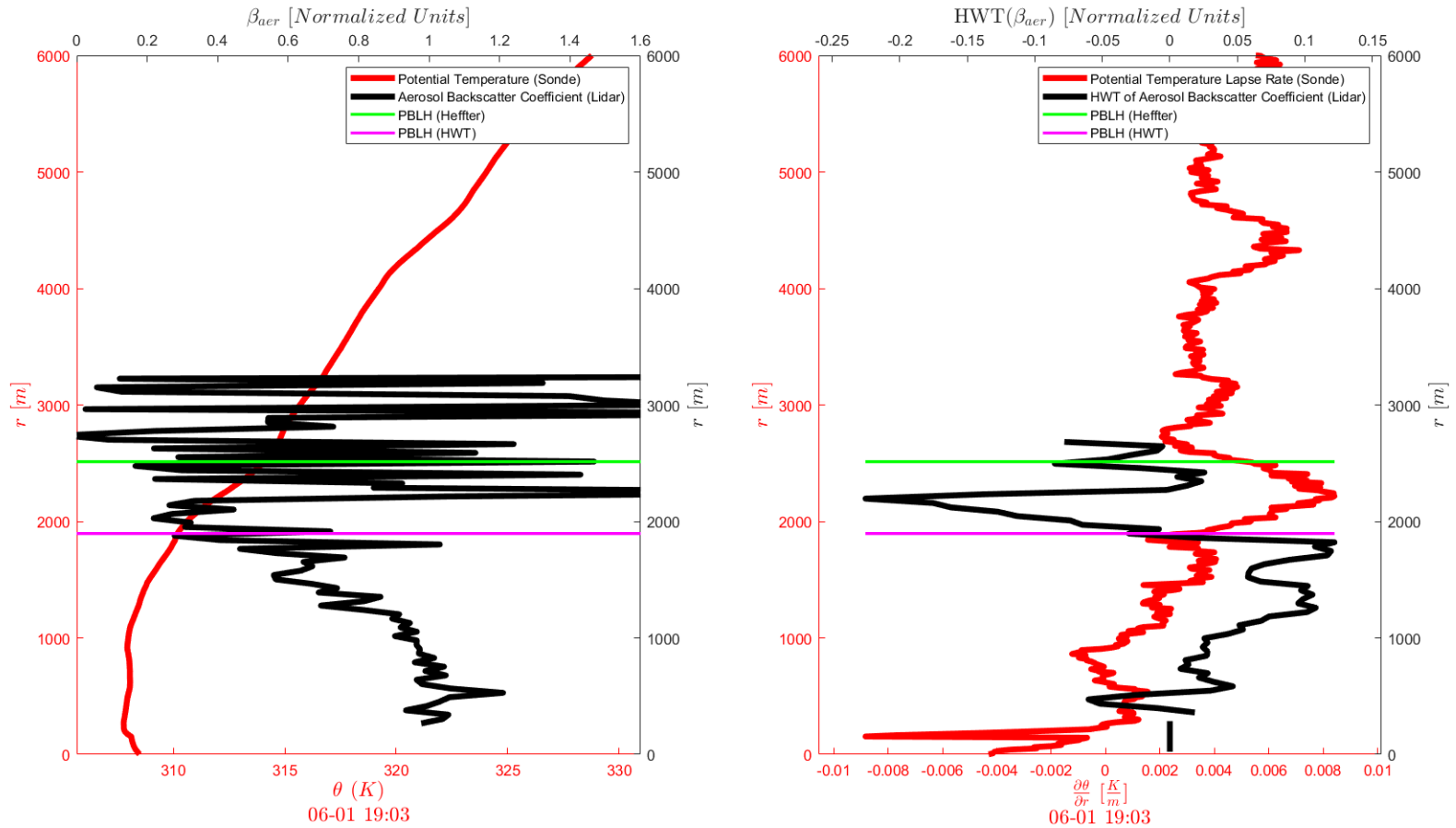


Figure B.21: Radiosonde comparison, June 1, 2021-19:03 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2513 m; PBLH-Lidar=1896 m.

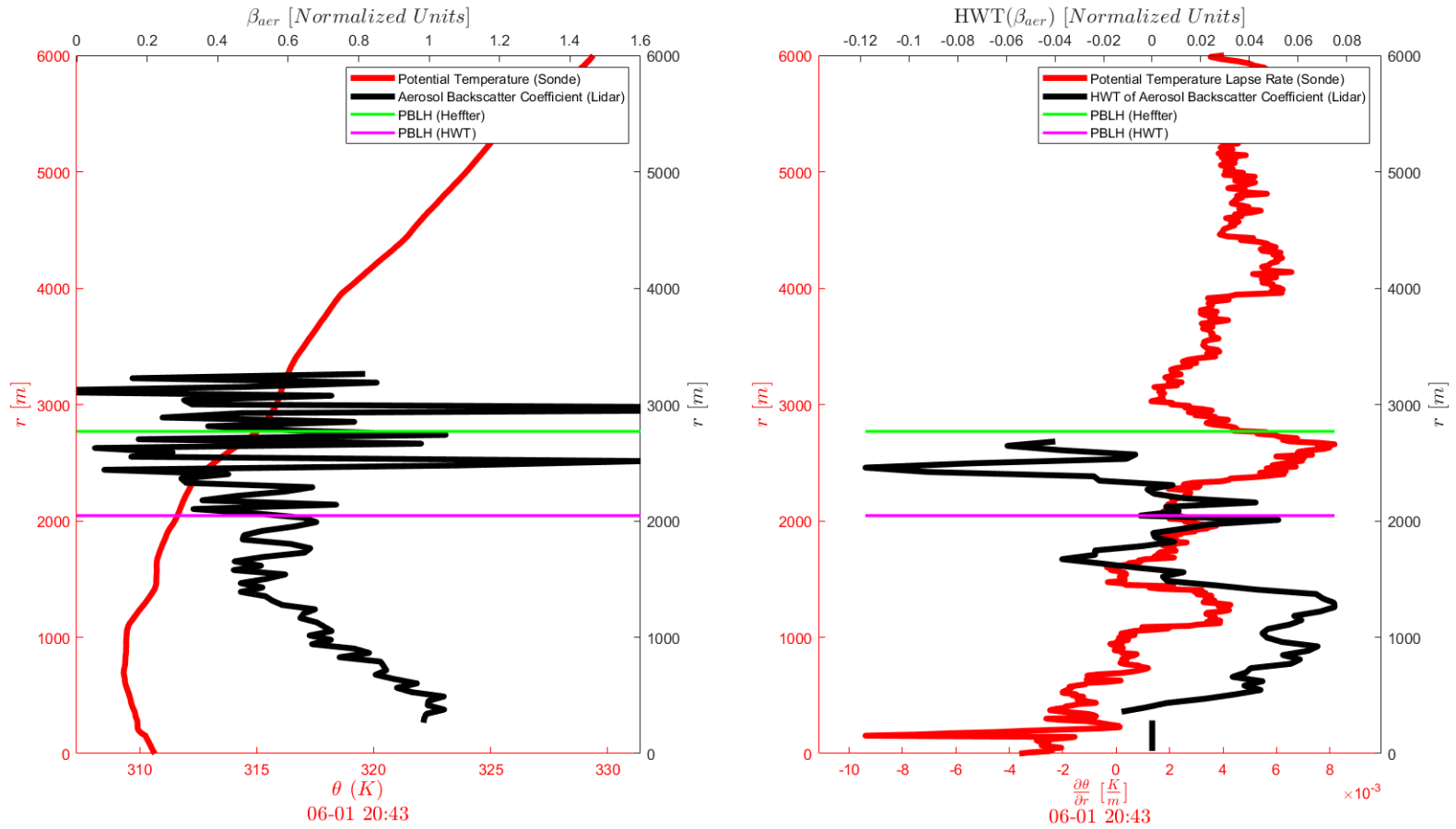


Figure B.22: Radiosonde comparison, June 1, 2021-20:43 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2769 m; PBLH-Lidar=2045 m.

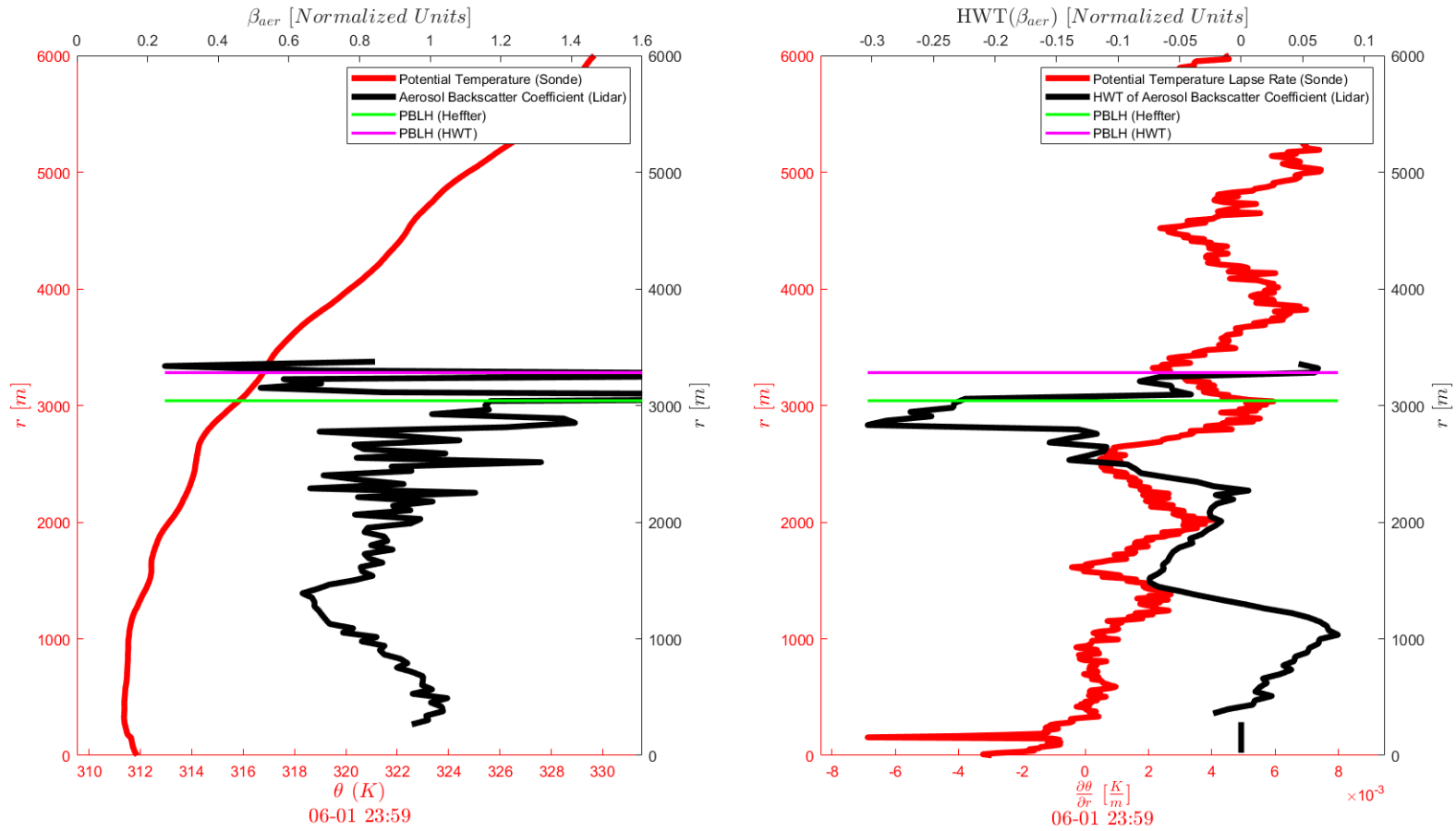


Figure B.23: Radiosonde comparison, June 1, 2021-23:59 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=3041 m; PBLH-Lidar=3282 m.

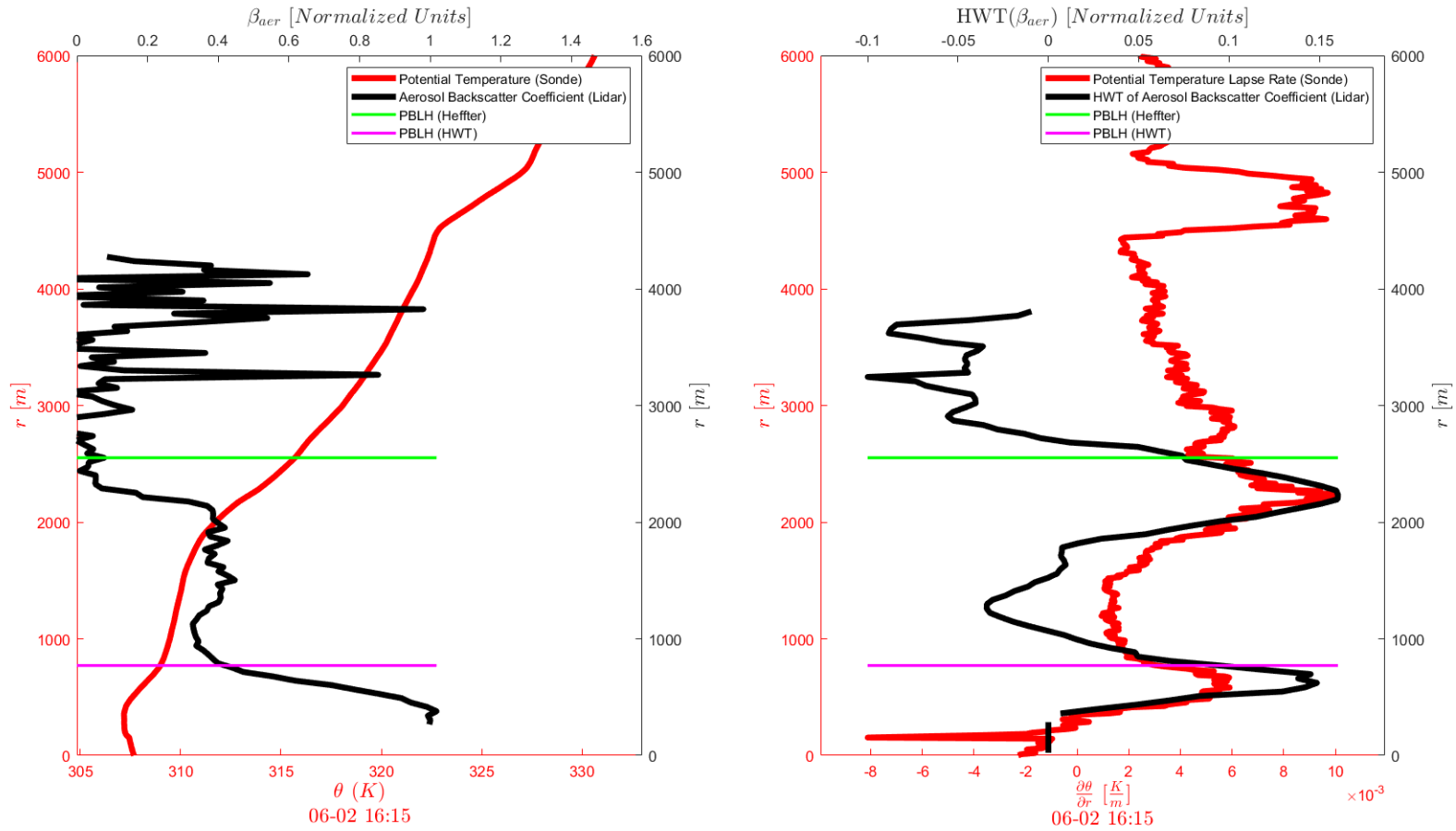


Figure B.24: Radiosonde comparison, June 2, 2021-16:15 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=2552 m; PBLH-Lidar=771 m.

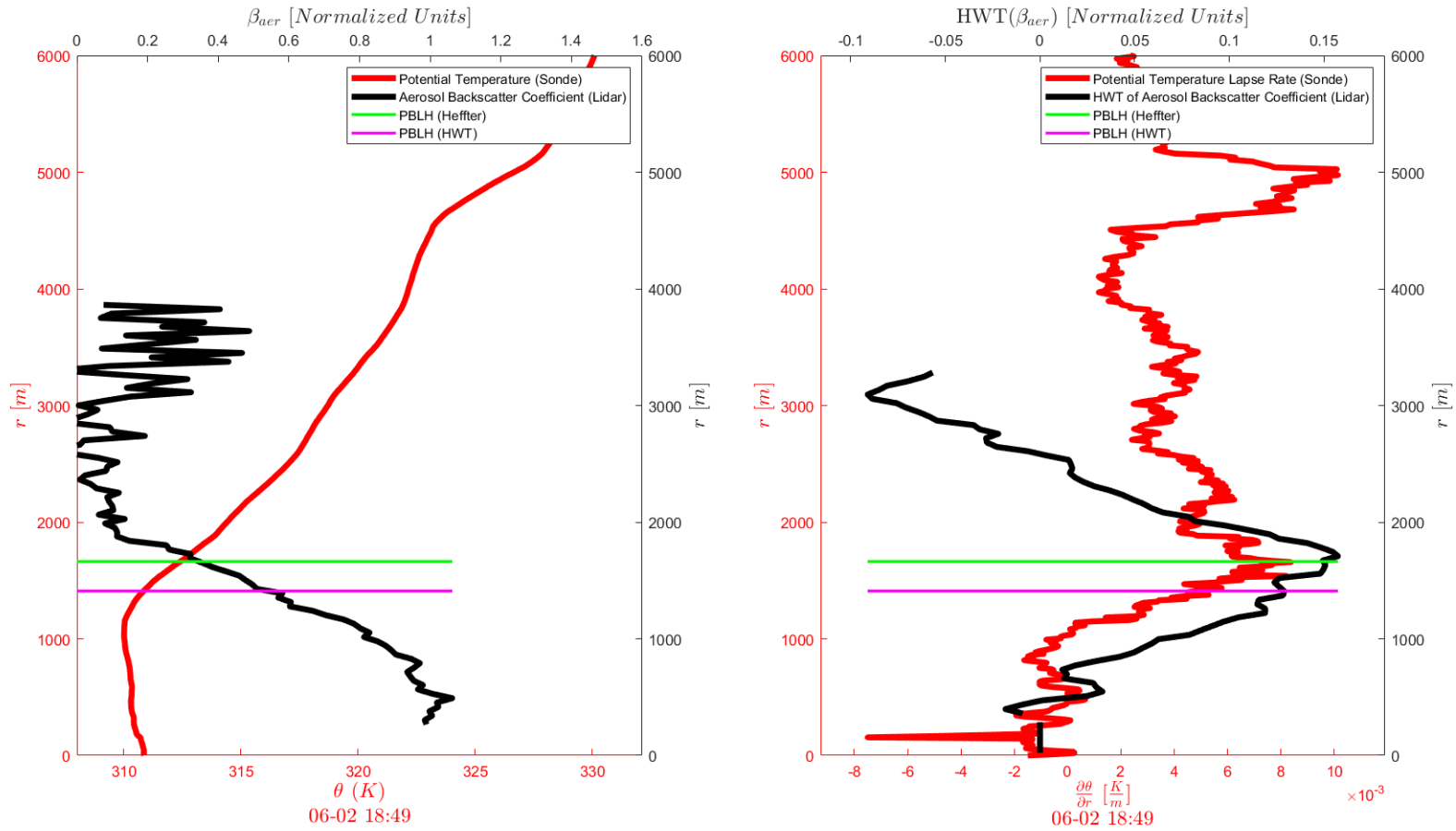


Figure B.25: Radiosonde comparison, June 2, 2021-18:49 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1662 m; PBLH-Lidar=1409 m.

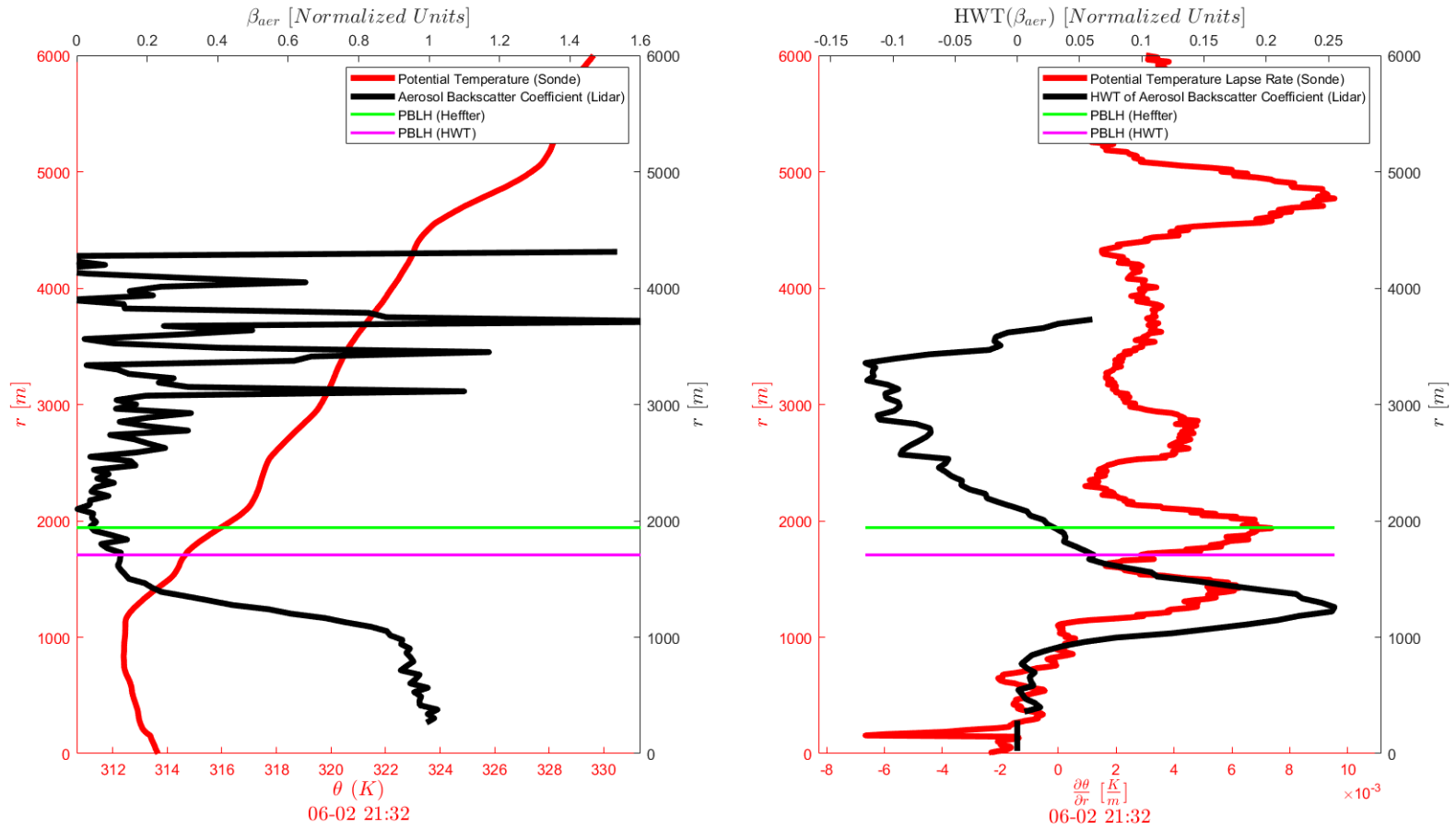


Figure B.26: Radiosonde comparison, June 2, 2021-21:32 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=1942 m; PBLH-Lidar=1708 m.

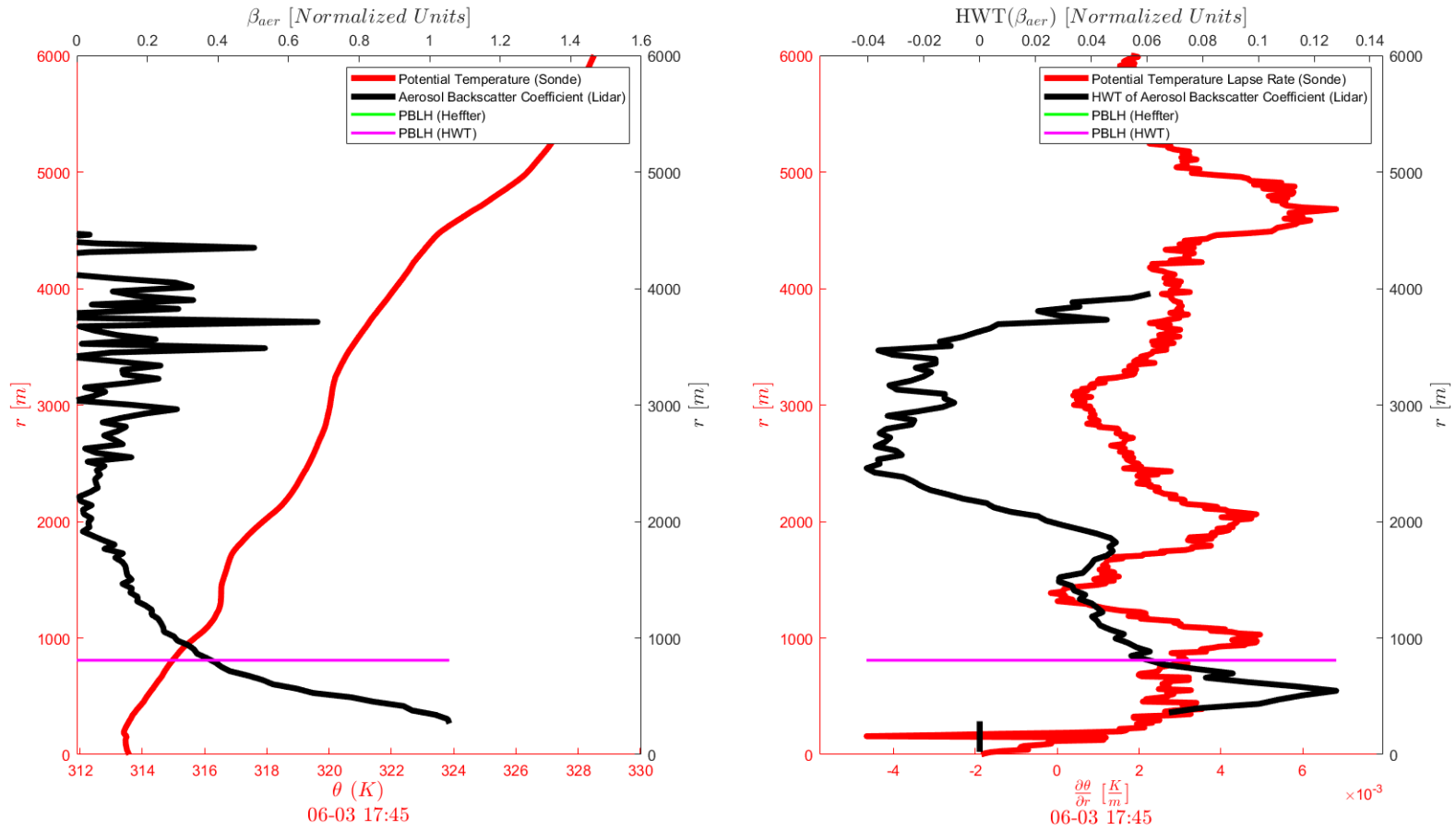


Figure B.27: Radiosonde comparison, June 3, 2021-17:45 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=809 m.

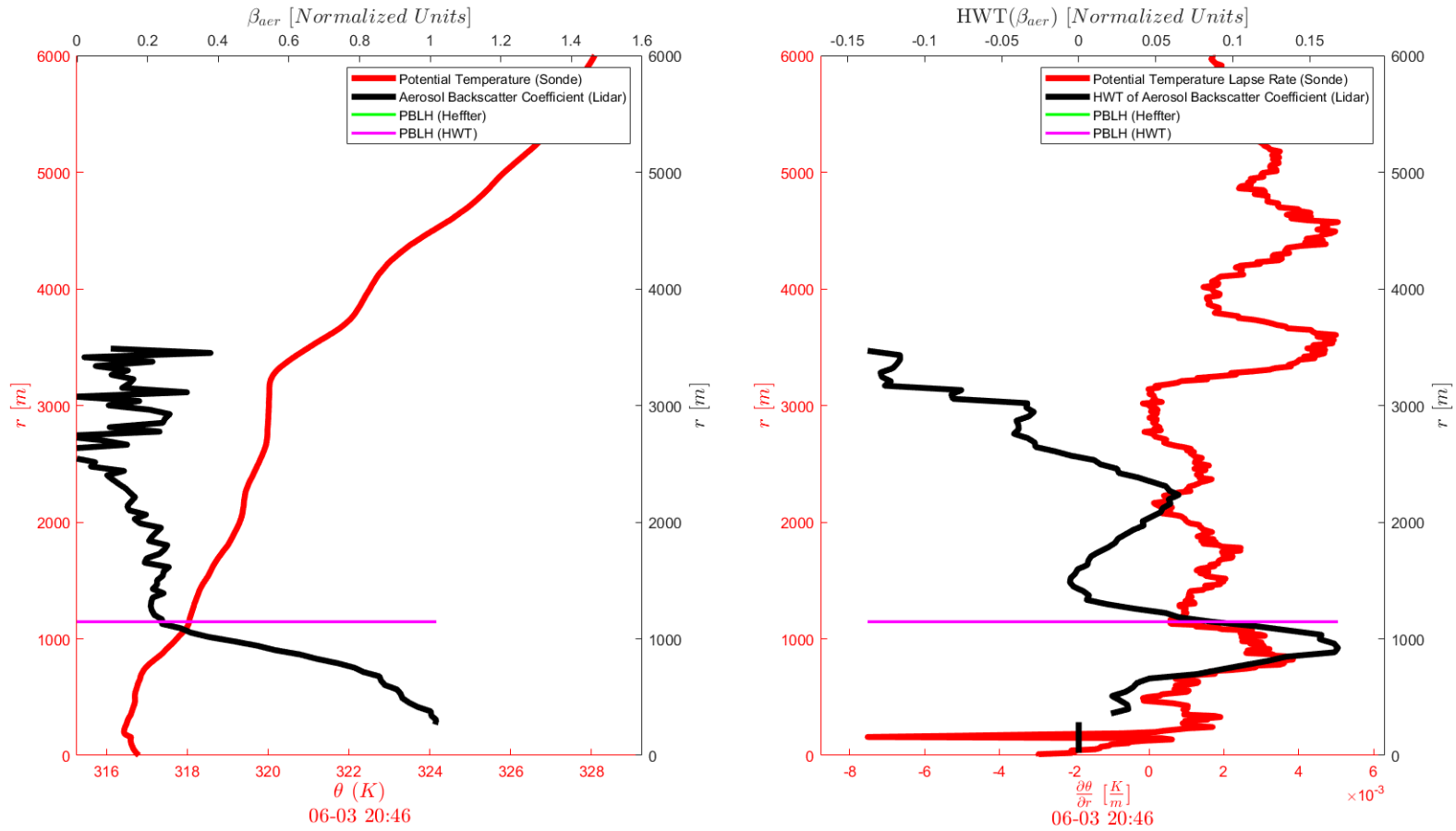


Figure B.28: Radiosonde comparison, June 3, 2021-20:46 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=1146 m.

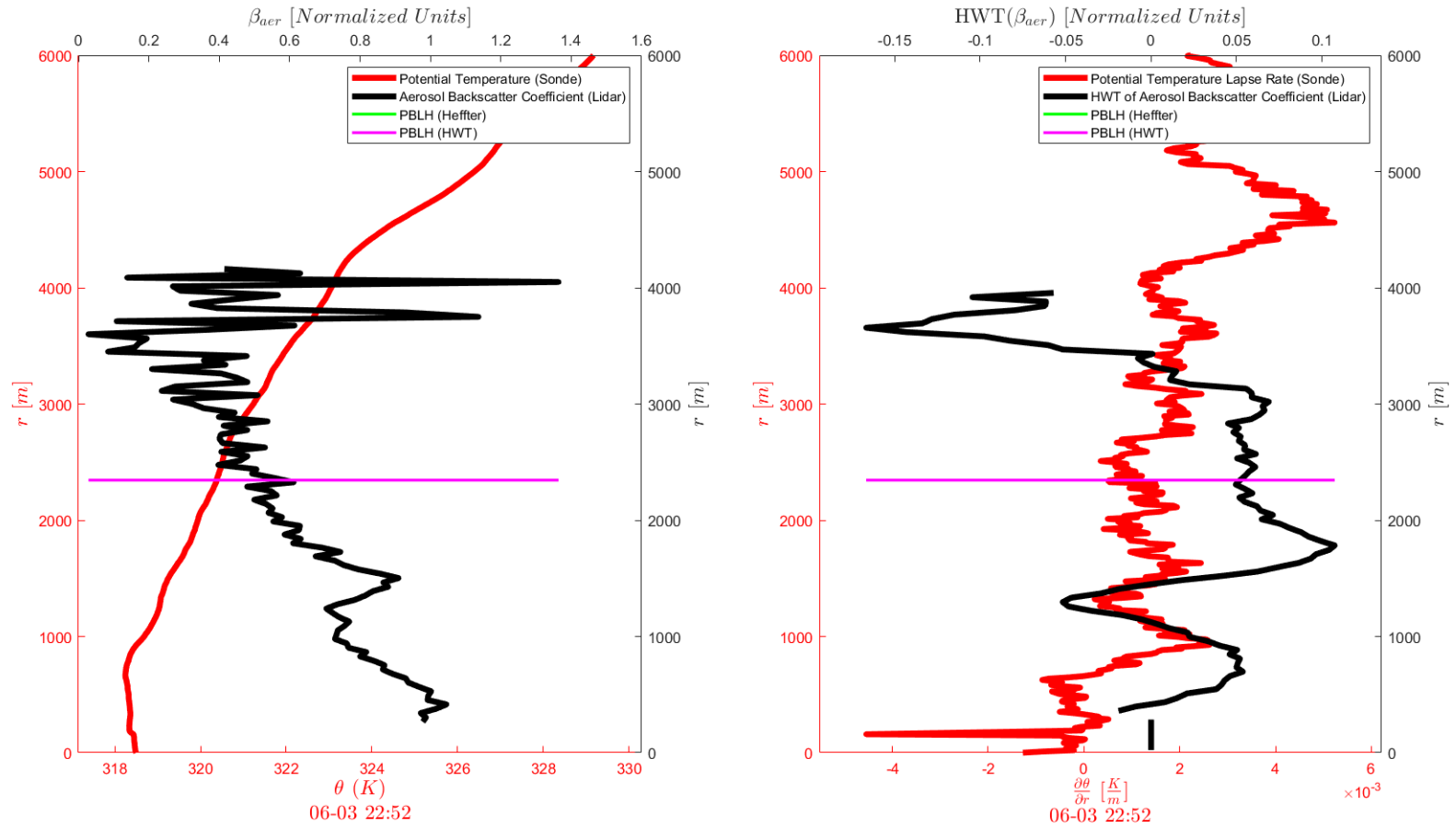


Figure B.29: Radiosonde comparison, June 3, 2021-22:52 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=NaN; PBLH-Lidar=2345 m.

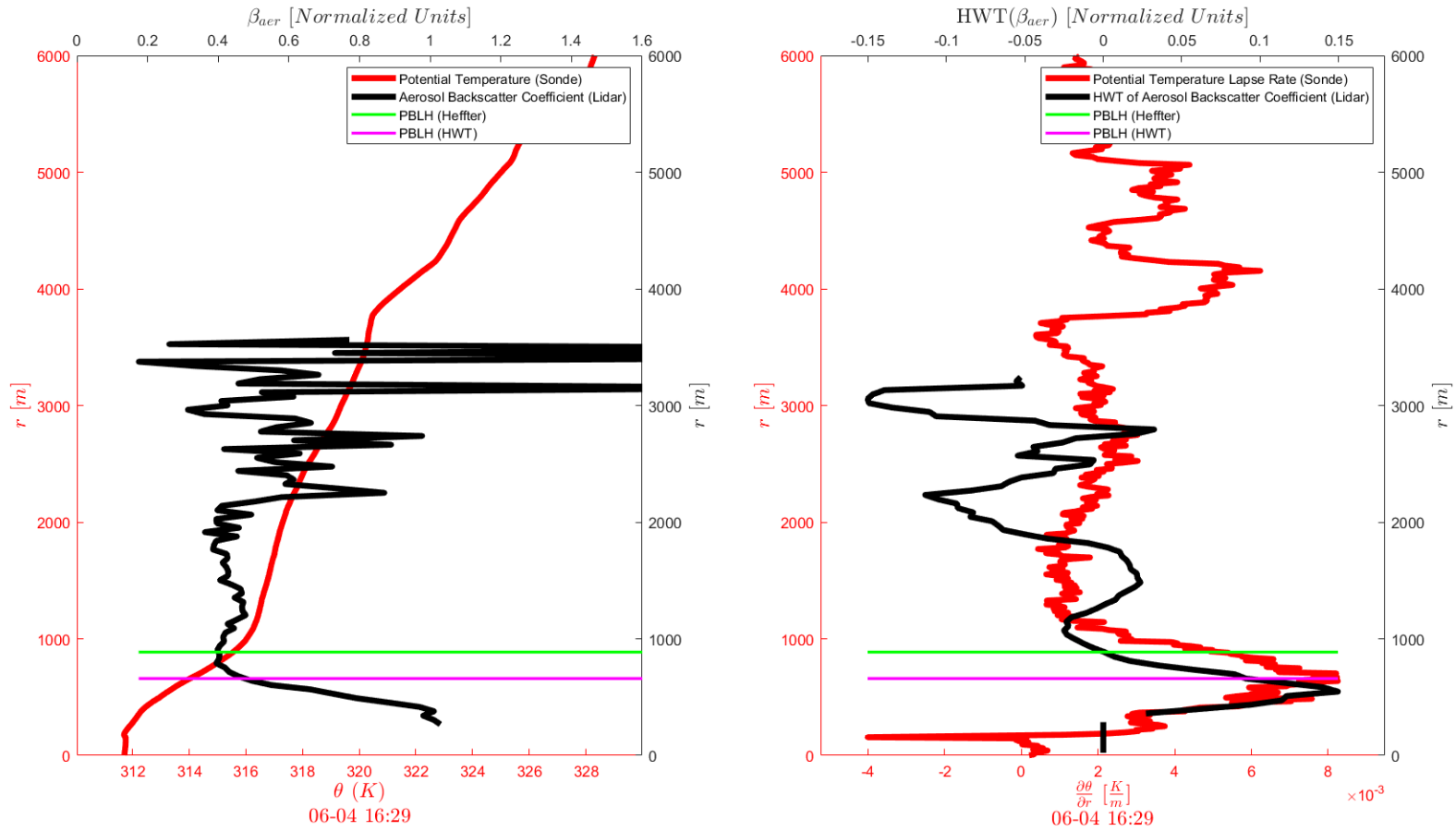


Figure B.30: Radiosonde comparison, June 4, 2021-16:29 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=886 m; PBLH-Lidar=659 m.

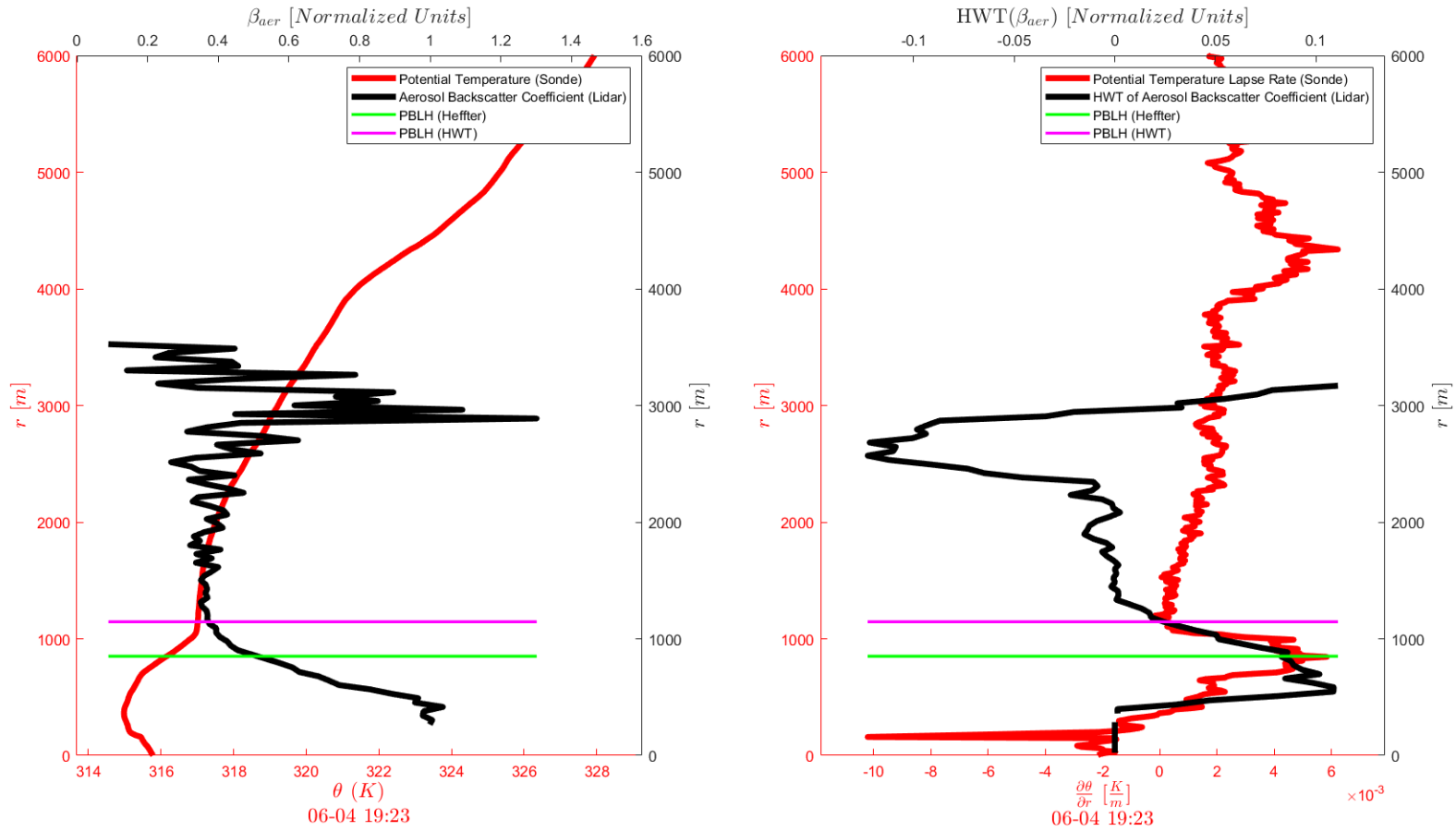


Figure B.31: Radiosonde comparison, June 4, 2021-19:23 (UTC); Left: Normalized aerosol backscatter coefficient and potential temperature. Right: HWT of normalized aerosol backscatter coefficient and potential temperature lapse rate. PBLH-Heffter=850 m; PBLH-Lidar=1146 m.

APPENDIX C

MATLAB CODE

Aerosol Optical Property and PBLH Retrieval Code

Main Function

```
% This code is the main function of the aerosol optical property and
% boundary layer retrieval for the Montana State combined O2-DIAL and HSRL
% The boundary layer retrieval requires that the day before and the day
% after are also used in the retrieval
% These 3 days are loaded in, then the data is trimmed to the second day
% and the value added products are stored for the second day only
%% 0. Precode
clear; close all;
%Define Parameters
% Select the days of data to process; make sure that the days are
% consecutive and to add an extra day on the beginning and end
days=[(13:31),(1:30),(1:31),(1:11)];
months=[3*ones(1,19),4*ones(1,30),5*ones(1,31),6*ones(1,11)];
years=([2021*ones(1,154)]);
if length(days)~=length(months)
    error('Make the variables "days" and "months" the same length')
end
if length(days)>length(years)
    error('Make the variable "years" longer than "days"')
end

% for loop surrounding the entire function
for j=2:length(days)-1
    day=[days(j-1),days(j),days(j+1)];
    month=[months(j-1),months(j),months(j+1)];
    year=[years(j-1),years(j),years(j+1)];
    Parameters.Day=strings(1,length(day));
    Parameters.Month=strings(1,length(day));
    Parameters.Year=strings(1,length(day));
    %convert date numbers to strings
    for i=1:length(day)
        if length(char(string(day(i))))==1
            Parameters.Day(i)=strcat('0',string(day(i)));
        else Parameters.Day(i)=string(day(i));
        end
        if length(char(string(month(i))))==1
            Parameters.Month(i)=strcat('0',string(month(i)));
        else Parameters.Month(i)=string(month(i));
        end
        Parameters.Year(i)=string(year(i));
    end

    %define file paths; these are specific to where the files are stored on the
    %computer
    Path.Code=pwd;
    Path.NetCDF='F:\MSU DIAL\Data\NetCDFOutput';
    Path.Weather='F:\MSU DIAL\Data\WeatherStation';
    Path.Sunrise='F:\MSU DIAL\Data\SunriseSunsetData';
    Path.Radiosondes='F:\MSU DIAL\Sondes';
    Path.ValueAddedProducts='F:\MSU DIAL\VAP\_08252021';

    %Range Parameters
    Parameters.MinRange=250; %m  min range for acceptable data; below this range afterpulsing causes
        issuse
    %Time Parameters
    Parameters.dt=5; % timestep minutes; this controls temporal resolution
    %Weather Parameters
    Parameters.LapseRate=7; %K/km; This controls the temperature and pressure models
    %7 K/km was derived from averaging the lapse rate measured by radiosonde profiles

    %Delay; This is the delay in MCS recording set by the quantum composer
    % 500 ns is standard, but other times could be used for trouble shooting
    if month(2)==3 || month(2)==4
        Parameters.Delay=500; %ns
    else
            Parameters.Delay=300; %ns
    end
    clear day month year i
    %% 1. Load Data and correct for nonlinear effects
    %Load NetCDF files
    [MCSData]=LoadNetCDF(Path,Parameters);
    %correct for APD nonlinear effects
    [MCSData]=NonlinearEffects(MCSData);
    %Load Weather Data from excel file
    [WeatherData]=LoadWeatherFile(Path,Parameters);
    %Load Sunrise/Sunset Data from excel file
    [SunriseSunsetData]=LoadSunriseSunsetData(Path,Parameters);
    %Load Radiosondes from .m file
    [Sondes]=LoadRadiosondes(Path,Parameters);

    %% 2. Process MCS Data
```

```
    %Subtract Background Photons
    [MCSData]=BackgroundSubtraction(MCSData);
    %Trim Data to 200 range bins
    [MCSData]=TrimData(MCSData);
    %Overlap Correction; correct for range dependent overlap function
    %difference between combined and molecular channels
    [LidarData,MCSData]=OverlapCorrection(MCSData,Parameters);

    %% 3. Interpolate MCS Data, remove unnecessary time
    %TimeGrid- set all data to desired time grid
    [LidarData,WeatherData]=TimeGrid(MCSData,LidarData,Parameters,WeatherData);
    clear MCSData
    %Trim data to useful times
    [LidarData,WeatherData,SunriseSunsetData]=...
        TrimTimes(LidarData,WeatherData,SunriseSunsetData,Parameters);

    %% 4. Backscatter Retrieval
    %Create Temperature and Pressure Models
    [WeatherData]=TemperatureAndPressure(WeatherData,Parameters,LidarData);
    %Aerosol Optical Property Retrieval
    [LidarData]=BackscatterRetrieval(LidarData,WeatherData);

    %% 5. Create and Apply Masks
    %Use Sobel Operator to find Gradient
    [LidarData,CloudMaskData]=FindGradient(LidarData);
    %Find Variance
    [LidarData,CloudMaskData]=FindVariance(LidarData,CloudMaskData);
    %Find Clouds and Create MCS Cloud Mask
    [LidarData]=FindClouds(LidarData,CloudMaskData);
    clear CloudMaskData
    %Create SNR Mask
    [LidarData]=SNRMasking(LidarData);
    %Apply Mask
    [LidarData]=ApplyMasks(LidarData);
    %Remove Low Altitude Data
    [LidarData]=DataRemoval(LidarData,Parameters);

    %% 6. Haar Wavelet Transform and Smoothing
    %Find Cloud Height from cloud mask
    [BoundaryLayers]=FindCloudHeight(LidarData,Parameters);
    %Normalize Aerosol Optical Property
    [LidarData]=NormalizeData(LidarData,BoundaryLayers);
    %Backscatter Smoothing Smooth Aerosol Optical Properties
    [LidarData]=BackscatterSmoothing(LidarData,Parameters);
    %Haar Wavlet Transform of Normalized Aerosol Optical Properties
    [LidarData]=HaarWaveletTransform(LidarData,Parameters);

    %% 7. Boundary Layers
    %Main function that contains boundary layer retrieval solution
    [BoundaryLayers]=BoundaryLayersMainFunction(LidarData,BoundaryLayers,Parameters,SunriseSunsetData);

    %% 8. Radiosonde Processing
    %Process radiosonde and lidar data
    [Sondes]=RadiosondeProcessing(LidarData,Sondes,Parameters,BoundaryLayers);

    %% 9. Save Files
    %Save Data to Value-added products file
    VAPSaveFile(BoundaryLayers,Sondes,...
    LidarData,Parameters,SunriseSunsetData,Path,WeatherData);
    clear WeatherData SunriseSunsetRawData BoundaryLayers EtalonData ...
        HKeepData i LidarData LLData MCSData Parameters Sondes SunriseSunsetData

end
```

LoadNetCDF

```
function [MCSData]=LoadNetCDF(Path,Parameters)
% This function loads the netCDF files generated by the MPD instrument and
% concatenates them into one large data structure: MCSData
%% Create a MCS data structure for each of the three days
    for i=1:length(Parameters.Day)

        %create date string from Parameter structure yyyyymmdd
        date=strcat(Parameters.Year(i),Parameters.Month(i),Parameters.Day(i));

        %open folder containing the NetCDF files from the desired day
        cd(Path.NetCDF);  cd(date);

        %create directory containing all the files for the desired day
        Directory(i).MCS=dir('MCS_00_*');

        %Find Channel Assignment for each channel (-1 comes from python index
        %starting at 0 and matlab index starting at 1)
        MCSRawData(i).ChannelAssignment=h5read(Directory(i).MCS(1).name,'/ChannelAssignment');
        MCSRawData(i).OfflineCombinedChannel=find(contains(MCSRawData(i).ChannelAssignment,'O2OfflineComb'
            ))-1;
        MCSRawData(i).OfflineMolecularChannel=find(contains(MCSRawData(i).ChannelAssignment,'O2OfflineMol'
            ))-1;
        MCSRawData(i).OnlineCombinedChannel=find(contains(MCSRawData(i).ChannelAssignment,'O2OnlineComb'))
            -1;
        MCSRawData(i).OnlineMolecularChannel=find(contains(MCSRawData(i).ChannelAssignment,'O2OnlineMol'))
            -1;

        %Files loaded in for loop, each new matrix is concatenated onto the
        %previously loaded data
        %Create Empty Matrix to begin as well as define number of bins and time
        %per bin
        Channel=[];  NBins=560; nsPerBin=250;

        %for loop for 1 variable, concatenate each step
        for j=1:length(Directory(i).MCS)
            Channel=[Channel;h5read(Directory(i).MCS(j).name,'/Channel')];
        end

        %preallocate the other variables
        Data=zeros(numel(Channel),NBins);
        ProfilesPerHist=zeros(numel(Channel),1);
        Time=zeros(numel(Channel),1);
        %temporary variables for indexing
        temp1=1;        temp2=1;        temp3=1;
        for j=1:length(Directory(i).MCS)
            temp11=h5read(Directory(i).MCS(j).name,'/Data');
            sz=size(temp11);
            Data(temp1:sz(1)+temp1-1,:)=temp11;
            temp1=temp1+sz(1);
            temp22=h5read(Directory(i).MCS(j).name,'/ProfilesPerHist');
            sz=size(temp22);
            ProfilesPerHist(temp2:sz(1)+temp2-1)=temp22;
            temp2=temp2+sz(1);
            temp33=h5read(Directory(i).MCS(j).name,'/time');
            sz=size(temp33);
            Time(temp3:sz(1)+temp3-1)=temp33;
            temp3=temp3+sz(1);
        end

        %Put into structure as field
        MCSRawData(i).RawChannel=Channel';
        MCSRawData(i).RawData=Data';
        MCSRawData(i).NBins=NBins;
        MCSRawData(i).ProfilesPerHist=ProfilesPerHist';
        MCSRawData(i).nsPerBin=nsPerBin;
        MCSRawData(i).Time=Time';

        %Sort Data into groups for each channel
        %Find Indices
        IndexOfflineCombined=find(Channel==MCSRawData(i).OfflineCombinedChannel);
        IndexOfflineMolecular=find(Channel==MCSRawData(i).OfflineMolecularChannel);
        IndexOnlineCombined=find(Channel==MCSRawData(i).OnlineCombinedChannel);
        IndexOnlineMolecular=find(Channel==MCSRawData(i).OnlineMolecularChannel);

        %Build Fields
        MCSRawData(i).OfflineCombinedData=Data(IndexOfflineCombined,:)';
        MCSRawData(i).OfflineCombinedTime=Time(IndexOfflineCombined,1)';
        MCSRawData(i).OfflineMolecularData=Data(IndexOfflineMolecular,:)';
        MCSRawData(i).OfflineMolecularTime=Time(IndexOfflineMolecular,1)';
        MCSRawData(i).OnlineCombinedData=Data(IndexOnlineCombined,:)';
        MCSRawData(i).OnlineCombinedTime=Time(IndexOnlineCombined,1)';
        MCSRawData(i).OnlineMolecularData=Data(IndexOnlineMolecular,:)';
        MCSRawData(i).OnlineMolecularTime=Time(IndexOnlineMolecular,1)';

    end
    cd(Path.Code)
```

```
%%  Create MCS data structure for three day span
    MCSData.NBins=MCSRawData(1).NBins;
    MCSData.nsPerBin=MCSRawData(1).nsPerBin;

    %create empty structure variables
    MCSData.OfflineCombinedTime=[];
    MCSData.OfflineMolecularTime=[];
    MCSData.OnlineCombinedTime=[];
    MCSData.OnlineMolecularTime=[];
    %concatenate structures from each day into one structure
    for i=1:length(Parameters.Day)
        MCSData.OfflineCombinedTime=[MCSData.OfflineCombinedTime,MCSRawData(i).OfflineCombinedTime+24*(i
            -1)];
        MCSData.OfflineMolecularTime=[MCSData.OfflineMolecularTime,MCSRawData(i).OfflineMolecularTime+24*(
            i-1)];
        MCSData.OnlineCombinedTime=[MCSData.OnlineCombinedTime,MCSRawData(i).OnlineCombinedTime+24*(i-1)];
        MCSData.OnlineMolecularTime=[MCSData.OnlineMolecularTime,MCSRawData(i).OnlineMolecularTime+24*(i
            -1)];
    end
    %preallocate structure variables
    MCSData.OfflineCombinedData=zeros(MCSRawData(i).NBins,length(MCSData.OfflineCombinedTime));
    MCSData.OfflineMolecularData=zeros(MCSRawData(i).NBins,length(MCSData.OfflineMolecularTime));
    MCSData.OnlineCombinedData=zeros(MCSRawData(i).NBins,length(MCSData.OnlineCombinedTime));
    MCSData.OnlineMolecularData=zeros(MCSRawData(i).NBins,length(MCSData.OnlineMolecularTime));
    %create temporaray indexing variables
    temp1=1;    temp2=1;    temp3=1;    temp4=1;
    %for loop for each day in dataset
    for i=1:length(Parameters.Day)
        temp11=MCSRawData(i).OfflineCombinedData;
        sz=size(temp11);
        MCSData.OfflineCombinedData(:,temp1:sz(2)+temp1-1)=MCSRawData(i).OfflineCombinedData;
        temp1=temp1+sz(2);

        temp22=MCSRawData(i).OfflineMolecularData;
        sz=size(temp22);
        MCSData.OfflineMolecularData(:,temp2:sz(2)+temp2-1)=MCSRawData(i).OfflineMolecularData;
        temp2=temp2+sz(2);

        temp33=MCSRawData(i).OnlineCombinedData;
        sz=size(temp33);
        MCSData.OnlineCombinedData(:,temp3:sz(2)+temp3-1)=MCSRawData(i).OnlineCombinedData;
        temp3=temp3+sz(2);

        temp44=MCSRawData(i).OnlineMolecularData;
        sz=size(temp44);
        MCSData.OnlineMolecularData(:,temp4:sz(2)+temp4-1)=MCSRawData(i).OnlineMolecularData;
        temp4=temp4+sz(2);
    end
end
```

NonLinearEffects

```
function [MCSData]=NonlinearEffects(MCSData)
% This function applies the nonlinear correction factor to the data
% collected by the MCS

    % Nonlinear Correction
    shotsperpulse=14000/2; %lidar profiles summed per MCS profile
    tbin=MCSData.nsPerBin*10^-9; %time per bin in s
    %Find photon count rate for every MCS bin
    OfflineCombinedCountRate=MCSData.OfflineCombinedData/(shotsperpulse*tbin);
    OfflineMolecularCountRate=MCSData.OfflineMolecularData/(shotsperpulse*tbin);
    OnlineCombinedCountRate=MCSData.OnlineCombinedData/(shotsperpulse*tbin);
    OnlineMolecularCountRate=MCSData.OnlineMolecularData/(shotsperpulse*tbin);
    %Values from APD Datasheet
    deadtime=60*10^-9;
    OfflineCombinedCorrectionFactor=1./(1-deadtime*OfflineCombinedCountRate);
    OfflineMolecularCorrectionFactor=1./(1-deadtime*OfflineMolecularCountRate);
    OnlineCombinedCorrectionFactor=1./(1-deadtime*OnlineCombinedCountRate);
    OnlineMolecularCorrectionFactor=1./(1-deadtime*OnlineMolecularCountRate);
    % Find Correction Factor
    OfflineCombinedCorrectionFactor(OfflineCombinedCorrectionFactor<=0)=1;
    OfflineMolecularCorrectionFactor(OfflineMolecularCorrectionFactor<=0)=1;
    OnlineCombinedCorrectionFactor(OnlineCombinedCorrectionFactor<=0)=1;
    OnlineMolecularCorrectionFactor(OnlineMolecularCorrectionFactor<=0)=1;
    % Apply Correction Factor
    MCSData.OfflineCombinedData=MCSData.OfflineCombinedData./(OfflineCombinedCorrectionFactor);
    MCSData.OfflineMolecularData=MCSData.OfflineMolecularData./(OfflineMolecularCorrectionFactor);
    MCSData.OnlineCombinedData=MCSData.OnlineCombinedData./(OnlineCombinedCorrectionFactor);
    MCSData.OnlineMolecularData=MCSData.OnlineMolecularData./(OnlineMolecularCorrectionFactor);

end
```

LoadWeatherFile

```
function [WeatherData,WeatherRawData]=LoadWeatherFile(Path,Parameters)
% This function loads the data from the weather station from a csv file
% csv format is
% Date,       Time,       Pressure,   Temperature
% 1/1/2021,   00:00:00,   847,        1.8

        %Set csv file name
        csvfile=strcat(Parameters.Year(1),'.csv');
        %Set points as start of day and end of day from parameters
        DayStart=datetime(str2double(Parameters.Year),str2double(Parameters.Month),str2double(Parameters.
            Day),0,0,0);
        DayStart.TimeZone='UTC';
        DayEnd=datetime(str2double(Parameters.Year),str2double(Parameters.Month),str2double(Parameters.Day
            ),24,0,0);
        DayEnd.TimeZone='UTC';
        cd(Path.Weather)
        % Read csv file into matlab character variable
        weatherfile = fileread(csvfile);
        % Remove unwanted characters
        weatherfile = strrep(weatherfile,'''','');
        weatherfile = strrep(weatherfile,' K','');
        weatherfile = strrep(weatherfile,'"','');
        % Read file into cell arrays
        formatSpec='%s %s %f %f';
        weathercell=textscan(weatherfile,formatSpec,'Delimiter',',','HeaderLines',1,'TreatAsEmpty','-');
        %Convert strings to datetime in UTC
        temp=[string(weathercell{1,1}),string(weathercell{1,2})];
        temp=[append(temp(:,1),' ',temp(:,2))];
        Date_Time=datetime(temp);
        Date_Time.TimeZone='UTC';

    %for loop for each day in data series
    for i=1:length(Parameters.Day)
        %Take Temperature and Pressure from cell arrays
        indx=Date_Time>=DayStart(i) & Date_Time<=DayEnd(i);
        WeatherRawData(i).GroundTemperature=weathercell{1,4}(indx);
        WeatherRawData(i).GroundTemperature(isempty(WeatherRawData(i).GroundTemperature))=10;
        WeatherRawData(i).GroundPressure=0.1*weathercell{1,3}(indx);
        WeatherRawData(i).GroundPressure(isempty(WeatherRawData(i).GroundPressure))=85;
        WeatherRawData(i).DateTime=Date_Time(indx);
        dt=24/(numel(WeatherRawData(i).GroundTemperature));
        WeatherRawData(i).Time=dt:dt:24;
        %correct length to be the same for all weather variables
        if length(WeatherRawData(i).GroundTemperature)<length(WeatherRawData(i).Time)
            WeatherRawData(i).GroundTemperature(length(WeatherRawData(i).GroundTemperature)+1:length(
                WeatherRawData(i).Time))=WeatherRawData(i).GroundTemperature(length(WeatherRawData(i).
                GroundTemperature));
        end
        if length(WeatherRawData(i).GroundPressure)<length(WeatherRawData(i).Time)
            WeatherRawData(i).GroundPressure(length(WeatherRawData(i).GroundPressure)+1:length(
                WeatherRawData(i).Time))=WeatherRawData(i).GroundPressure(length(WeatherRawData(i).
                GroundPressure));
        end
        %remove bad data points that have unrealistic temperatures or pressures
        medianpressure=nanmedian(WeatherRawData(i).GroundPressure);
        WeatherRawData(i).GroundPressure(abs(WeatherRawData(i).GroundPressure-medianpressure)>30)=NaN;
        WeatherRawData(i).GroundPressure=fillmissing(WeatherRawData(i).GroundPressure,'nearest');
        mediantemperature=nanmedian(WeatherRawData(i).GroundTemperature);
        WeatherRawData(i).GroundTemperature(abs(WeatherRawData(i).GroundTemperature-mediantemperature)>50)
            =NaN;
        WeatherRawData(i).GroundTemperature=fillmissing(WeatherRawData(i).GroundTemperature,'nearest');
    end
    cd(Path.Code)
    %Create empty variables
    WeatherData.GroundTemperature=[];
    WeatherData.GroundPressure=[];
    WeatherData.Time=[];
    WeatherData.DateTime=[];
    %concatenate weather data for three day span
    for i=1:length(Parameters.Day)
        WeatherData.GroundTemperature=[WeatherData.GroundTemperature;WeatherRawData(i).GroundTemperature];
        WeatherData.GroundPressure=[WeatherData.GroundPressure;WeatherRawData(i).GroundPressure];
        WeatherData.Time=[WeatherData.Time,WeatherRawData(i).Time+24*(i-1)];
        WeatherData.DateTime=[WeatherData.DateTime;WeatherRawData(i).DateTime];
    end
end
```

LoadSunriseSunsetData

```
function [SunriseSunsetData]=LoadSunriseSunsetData(Path,Parameters)
%This function loads the sunrise and sunset times from a csv file for each
%the three days
%csv format is
%Date,Sunrise (UTC),Sunset (UTC),Sunrise (MT),Sunset(MT)
%6/1/2020,11:37,3:06,5:37,21:06
    cd(Path.Sunrise)
    %Load csv file in as a character variable
    sunrisesunsetfile=fileread('SunriseSunsetBozeman.csv');
    %remove unwanted characters
    sunrisesunsetfile = strrep(sunrisesunsetfile,'''','');
    sunrisesunsetfile = strrep(sunrisesunsetfile,'"','');
    %create cell array from character variable
    formatSpec='%s %s %s %s %s';
    sunrisesunsetcell=textscan(sunrisesunsetfile,formatSpec,'Delimiter',',','HeaderLines',1,'TreatAsEmpty'
        ,'-');
    date=string(sunrisesunsetcell{1,1});
    %Bring times into character cell
    sunrise=string(sunrisesunsetcell{2});
    sunset=string(sunrisesunsetcell{3});
     %for loop for each day
    for i=1:length(Parameters.Day)
        Year=Parameters.Year(i);
        tempmonth=char(Parameters.Month(i));
        if tempmonth(1)=='0'
            Month=string(tempmonth(2));
        else
            Month=string(tempmonth);
        end
        tempday=char(Parameters.Day(i));
        if tempday(1)=='0'
            Day=string(tempday(2));
        else
            Day=string(tempday);
        end
        %create sunrise and sunset time variables as decimals
        dateindex=find(date==strcat(Month,'/',Day,'/',Year),1);
        sunrisetime=char(sunrise(dateindex));
        sunsettime=char(sunset(dateindex));
        if numel(sunrisetime)==5
            SunriseSunsetRawData(i).SunriseTime=str2num(sunrisetime(1:2))+1/60*str2num(sunrisetime(4:5));
        else
            SunriseSunsetRawData(i).SunriseTime=str2num(sunrisetime(1))+1/60*str2num(sunrisetime(3:4));
        end
        if numel(char(sunsettime))==5
            SunriseSunsetRawData(i).SunsetTime=str2num(sunsettime(1:2))+1/60*str2num(sunsettime(4:5));
        else
            SunriseSunsetRawData(i).SunsetTime=str2num(sunsettime(1))+1/60*str2num(sunsettime(3:4));
        end

        %Create Night/Day Vector
        %0=Night, 1=Day
        SunriseSunsetRawData(i).Day_Night=ones(1,1440/Parameters.dt);
        SunriseSunsetRawData(i).Time=0:Parameters.dt/60:24-Parameters.dt/60;
        %if Sunset<Sunrise, Night is between sunset and sunrise
        %if Sunrise<Sunset, Night is before sunrise and after sunset
        if SunriseSunsetRawData(i).SunsetTime<SunriseSunsetRawData(i).SunriseTime
            SunriseSunsetRawData(i).Day_Night(round(60/(Parameters.dt)*SunriseSunsetRawData(i).SunsetTime)
                :round(60/(Parameters.dt)*SunriseSunsetRawData(i).SunriseTime))=0;
        end
        if SunriseSunsetRawData(i).SunsetTime>SunriseSunsetRawData(i).SunriseTime
            SunriseSunsetRawData(i).Day_Night(1:round(60*SunriseSunsetRawData(i).SunriseTime))=0;
            SunriseSunsetRawData(i).Day_Night(round(60*SunriseSunsetRawData(i).SunsetTime):end)=0;
        end
        SunriseSunsetRawData(i).SunriseIndex=find(SunriseSunsetRawData(i).Day_Night==0,1,'last');
        SunriseSunsetRawData(i).SunsetIndex=find(SunriseSunsetRawData(i).Day_Night==0,1,'first')-1;
    end

    %Create Empty Variables
    SunriseSunsetData.SunriseTime=[];     SunriseSunsetData.SunsetTime=[];
    SunriseSunsetData.Day_Night=[];     SunriseSunsetData.Time=[];
    SunriseSunsetData.SunriseIndex=[];     SunriseSunsetData.SunsetIndex=[];

    %Create total sunrise/sunset structure
    for i=1:length(Parameters.Day)
        SunriseSunsetData.SunriseTime=[SunriseSunsetData.SunriseTime;SunriseSunsetRawData(i).SunriseTime];
        SunriseSunsetData.SunsetTime=[SunriseSunsetData.SunsetTime;SunriseSunsetRawData(i).SunsetTime];
        SunriseSunsetData.Day_Night=[SunriseSunsetData.Day_Night,SunriseSunsetRawData(i).Day_Night];
        SunriseSunsetData.Time=[SunriseSunsetData.Time,SunriseSunsetRawData(i).Time+24*(i-1)];
        SunriseSunsetData.SunriseIndex=[SunriseSunsetData.SunriseIndex,SunriseSunsetRawData(i).
            SunriseIndex+length(SunriseSunsetRawData(i).Time)*(i-1)];
        SunriseSunsetData.SunsetIndex=[SunriseSunsetData.SunsetIndex,SunriseSunsetRawData(i).SunsetIndex+
            length(SunriseSunsetRawData(i).Time)*(i-1)];
    end
    cd(Path.Code)
end
```

LoadRadiosondes

```
function [Sondes]=LoadRadiosondes(Path,Parameters)
%This function loads radiosonde data from m files and sorts he data into
%structures
cd(Path.Radiosondes)
for i=2:length(Parameters.Day)-1
    filename=strcat('Sondes_',Parameters.Year(i),Parameters.Month(i),Parameters.Day(i),'.mat');
    try
        Sondes(i-1)=load(filename);
    catch
        Sondes(i-1).RadiosondeData=struct;
    end
end
cd(Path.Code)
end
```

BackgroundSubtraction

```
function [MCSData]=BackgroundSubtraction(MCSData)
%this function determines the background photon counts for each 2s
%interval that the MCS records data and subtracts the background from the
%recorded lidar profile
    %Find Background and build into matrix
    OfflineCombinedBackground=nanmean(MCSData.OfflineCombinedData(301:560,:),1);
    OfflineCombinedBackground=repmat(OfflineCombinedBackground,560,1);
    OfflineMolecularBackground=nanmean(MCSData.OfflineMolecularData(301:560,:),1);
    OfflineMolecularBackground=repmat(OfflineMolecularBackground,560,1);
    OnlineCombinedBackground=nanmean(MCSData.OnlineCombinedData(301:560,:),1);
    OnlineCombinedBackground=repmat(OnlineCombinedBackground,560,1);
    OnlineMolecularBackground=nanmean(MCSData.OnlineMolecularData(301:560,:),1);
    OnlineMolecularBackground=repmat(OnlineMolecularBackground,560,1);
    %Subtract Background From Data
    MCSData.BackgroundSubtractedOfflineCombinedData=MCSData.OfflineCombinedData-OfflineCombinedBackground;
    MCSData.BackgroundSubtractedOfflineMolecularData=MCSData.OfflineMolecularData-
        OfflineMolecularBackground;
    MCSData.BackgroundSubtractedOnlineCombinedData=MCSData.OnlineCombinedData-OnlineCombinedBackground;
    MCSData.BackgroundSubtractedOnlineMolecularData=MCSData.OnlineMolecularData-OnlineMolecularBackground;
    %remove unnecessary fields for efficiency
    fields={'OnlineMolecularData','OfflineCombinedData','OfflineMolecularData','OnlineCombinedData'};
    MCSData=rmfield(MCSData,fields);
end
```

TrimData

```
function [MCSData]=TrimData(MCSData)
%this function removes data past the 200th range bin, which is at 7.5 km,
%or about where the lidar signal reaches its maximum usable range
maxindex=200;
MCSData.BackgroundSubtractedOfflineCombinedData=...
    MCSData.BackgroundSubtractedOfflineCombinedData(1:maxindex,:);
MCSData.BackgroundSubtractedOfflineMolecularData=...
    MCSData.BackgroundSubtractedOfflineMolecularData(1:maxindex,:);
MCSData.BackgroundSubtractedOnlineCombinedData=...
    MCSData.BackgroundSubtractedOnlineCombinedData(1:maxindex,:);
MCSData.BackgroundSubtractedOnlineMolecularData=...
    MCSData.BackgroundSubtractedOnlineMolecularData(1:maxindex,:);
end
```

OverlapCorrection

```
function [LidarData,MCSData]=OverlapCorrection(MCSData,Parameters)
%This function finds the average overlap for 1 day and corrects the offline
%molecular data

%find ratio: Nconline/Nmonline
    if length(MCSData.BackgroundSubtractedOnlineCombinedData(1,:))~=length(MCSData.
        BackgroundSubtractedOnlineMolecularData(1,:))
        templength=min([length(MCSData.BackgroundSubtractedOnlineCombinedData(1,:)),...
            length(MCSData.BackgroundSubtractedOnlineMolecularData(1,:))]);
            ratio=MCSData.BackgroundSubtractedOnlineCombinedData(:,1:templength)./...
                MCSData.BackgroundSubtractedOnlineMolecularData(:,1:templength);
    else ratio=MCSData.BackgroundSubtractedOnlineCombinedData./MCSData.
        BackgroundSubtractedOnlineMolecularData;
    end
    %create overlap correction matrix
    ratioprofile=nanmedian(ratio,2);
```

```
    ratioprofile=ratioprofile/nanmedian(ratioprofile(35:45));
    ratioprofile(35:end)=1;
    ratioprofile(1:6)=1;
    Overlap.overlapcorrection=ratioprofile;
    LidarData.OverlapCorrection=Overlap.overlapcorrection;
    OverlapMatrix=repmat(LidarData.OverlapCorrection,1,length(MCSData.
        BackgroundSubtractedOfflineMolecularData(1,:)));
    %Apply Overlap
    MCSData.BackgroundSubtractedOfflineMolecularData=MCSData.BackgroundSubtractedOfflineMolecularData.*
        OverlapMatrix;
    %Create Range Vector
    c=299704644.54; %m/s
    LidarData.dr=(MCSData.nsPerBin)*10^(-9)*c/2; % size of bin
    StartRange=(Parameters.Delay-500)*10^(-9)*c/2; %start distance
    LidarData.Range=((LidarData.dr+StartRange)/2:LidarData.dr:LidarData.dr*MCSData.NBins)'; %m
    LidarData.Range=LidarData.Range(1:200);
end
```

TimeGrid

```
function [LidarData,WeatherData]=TimeGrid(MCSData,LidarData,Parameters,WeatherData)
% This function interpolates all the data to the same time grid

    %Create Time Vector
    LidarData.Time=0:Parameters.dt/60:24*length(Parameters.Day)-(Parameters.dt/60);
    %Create Time High and Time Low vectors for limits
    timelow=0:Parameters.dt/60:24*length(Parameters.Day)-Parameters.dt/60;
    timehigh=Parameters.dt/60:Parameters.dt/60:24*length(Parameters.Day);
    %Find the indices for each minute
    %Preallocate as NaN so that there are no false 0 counts
    OfflineCombinedLow=NaN(1,length(LidarData.Time));
    OfflineCombinedHigh=NaN(1,length(LidarData.Time));
    OfflineMolecularLow=NaN(1,length(LidarData.Time));
    OfflineMolecularHigh=NaN(1,length(LidarData.Time));
    OnlineCombinedLow=NaN(1,length(LidarData.Time));
    OnlineCombinedHigh=NaN(1,length(LidarData.Time));
    OnlineMolecularLow=NaN(1,length(LidarData.Time));
    OnlineMolecularHigh=NaN(1,length(LidarData.Time));
    LidarData.OfflineCombinedTotalCounts=NaN(length(LidarData.Range),length(LidarData.Time));
    LidarData.OfflineMolecularTotalCounts=NaN(length(LidarData.Range),length(LidarData.Time));
    LidarData.OnlineCombinedTotalCounts=NaN(length(LidarData.Range),length(LidarData.Time));
    LidarData.OnlineMolecularTotalCounts=NaN(length(LidarData.Range),length(LidarData.Time));
    LidarData.CloudMask=zeros(length(LidarData.Range),length(LidarData.Time));

    %for loop that sums data for each time step
    for i=1:length(LidarData.Time)
        tempindex=find(MCSData.OfflineCombinedTime>LidarData.Time(i) & MCSData.OfflineCombinedTime<(
            LidarData.Time(i)+Parameters.dt/60));
        tempofflinecombinedcounts=MCSData.BackgroundSubtractedOfflineCombinedData(:,tempindex);
        LidarData.OfflineCombinedTotalCounts(:,i)=nansum(tempofflinecombinedcounts,2);
        tempindex=find(MCSData.OfflineMolecularTime>LidarData.Time(i) & MCSData.OfflineMolecularTime<(
            LidarData.Time(i)+Parameters.dt/60));
        tempofflinemolecularcounts=MCSData.BackgroundSubtractedOfflineMolecularData(:,tempindex);
        LidarData.OfflineMolecularTotalCounts(:,i)=nansum(tempofflinemolecularcounts,2);
        tempindex=find(MCSData.OnlineCombinedTime>LidarData.Time(i) & MCSData.OnlineCombinedTime<(
            LidarData.Time(i)+Parameters.dt/60));
        temponlinecombinedcounts=MCSData.BackgroundSubtractedOnlineCombinedData(:,tempindex);
        LidarData.OnlineCombinedTotalCounts(:,i)=nansum(temponlinecombinedcounts,2);
        tempindex=find(MCSData.OnlineMolecularTime>LidarData.Time(i) & MCSData.OnlineMolecularTime<(
            LidarData.Time(i)+Parameters.dt/60));
        temponlinemolecularcounts=MCSData.BackgroundSubtractedOnlineMolecularData(:,tempindex);
        LidarData.OnlineMolecularTotalCounts(:,i)=nansum(temponlinemolecularcounts,2);
    end

    %same for weather data
    if numel(WeatherData.Time)<numel(WeatherData.GroundTemperature)
        WeatherData.GroundTemperature=WeatherData.GroundTemperature(1:numel(WeatherData.Time));
    end
    if numel(WeatherData.Time)<numel(WeatherData.GroundPressure)
        WeatherData.GroundPressure=WeatherData.GroundPressure(1:numel(WeatherData.Time));
    end
    %interpolate weather data from weather file data
    WeatherData.GroundTemperature=interp1(WeatherData.Time,WeatherData.GroundTemperature,LidarData.Time);
    WeatherData.GroundTemperature=fillmissing(WeatherData.GroundTemperature,'Nearest');
    WeatherData.GroundPressure=interp1(WeatherData.Time,WeatherData.GroundPressure,LidarData.Time);
    WeatherData.GroundPressure=fillmissing(WeatherData.GroundPressure,'Nearest');
    WeatherData.DateTime=interp1(WeatherData.Time,WeatherData.DateTime,LidarData.Time);
    WeatherData.DateTime=fillmissing(WeatherData.DateTime,'Nearest');
    WeatherData.Time=LidarData.Time;
    % shift time to middle of bin from start of bin
    LidarData.Time=(Parameters.dt/60)/2:Parameters.dt/60:24*length(Parameters.Day)-(Parameters.dt/60)/2;
end
```

TrimTimes

```
function [LidarData,WeatherData,SunriseSunsetData]=TrimTimes(LidarData,WeatherData,SunriseSunsetData,
    Parameters)
%this function trims the data from the sunrise before the start of te second
%day to the sunset after the end of the second day this is done to speed up
%the program. the boundary laye program requires that full daytime periods
%be examined instead of the interrupted days caused by using 1 single UTC
%day
%create indices for start and end of day 2
    StartIndex=1+1440/Parameters.dt*(1);
    EndIndex=1440/Parameters.dt+1440/Parameters.dt*(1);
    %find sunrise and sunset
    sunrise=find(SunriseSunsetData.SunriseIndex>=StartIndex  &  SunriseSunsetData.SunriseIndex<=EndIndex);
    sunset=find(SunriseSunsetData.SunsetIndex>=StartIndex  &  SunriseSunsetData.SunsetIndex<=EndIndex);
    SunriseIndex=SunriseSunsetData.SunriseIndex(sunrise);
    SunsetIndex=SunriseSunsetData.SunsetIndex(sunset);

if SunriseIndex>SunsetIndex
    daystart=SunriseSunsetData.SunriseIndex(sunrise-1);
    dayend=SunriseSunsetData.SunsetIndex(sunset+1);
else
    daystart=StartIndex;
    dayend=EndIndex;
end

%Trim data to new start and end indices
LidarData.Time=LidarData.Time(daystart:dayend);
LidarData.OfflineCombinedTotalCounts=LidarData.OfflineCombinedTotalCounts(:,daystart:dayend);
LidarData.OfflineMolecularTotalCounts=LidarData.OfflineMolecularTotalCounts(:,daystart:dayend);
LidarData.OnlineCombinedTotalCounts=LidarData.OnlineCombinedTotalCounts(:,daystart:dayend);
LidarData.OnlineMolecularTotalCounts=LidarData.OnlineMolecularTotalCounts(:,daystart:dayend);
LidarData.CloudMask=LidarData.CloudMask(:,daystart:dayend);
WeatherData.GroundTemperature=WeatherData.GroundTemperature(daystart:dayend);
WeatherData.GroundPressure=WeatherData.GroundPressure(daystart:dayend);
WeatherData.Time=WeatherData.Time(daystart:dayend);
WeatherData.DateTime=WeatherData.DateTime(daystart:dayend);
SunriseSunsetData.Day_Night=SunriseSunsetData.Day_Night(daystart:dayend);
SunriseSunsetData.Time=SunriseSunsetData.Time(daystart:dayend);
SunriseSunsetData.SunriseIndex=SunriseSunsetData.SunriseIndex-daystart+1;
SunriseSunsetData.SunriseIndex(SunriseSunsetData.SunriseIndex<1)=[];
SunriseSunsetData.SunriseIndex(SunriseSunsetData.SunriseIndex>length(LidarData.Time))=[];
SunriseSunsetData.SunsetIndex=SunriseSunsetData.SunsetIndex-daystart+1;
SunriseSunsetData.SunsetIndex(SunriseSunsetData.SunsetIndex<1)=[];
SunriseSunsetData.SunsetIndex(SunriseSunsetData.SunsetIndex>length(LidarData.Time))=[];
end
```

TemperatureAndPressure

```
function [WeatherData]=TemperatureAndPressure(WeatherData,Parameters,LidarData)
%This function generates temperature and pressures for the atmosphere using
%a constant lapse rate temperature model and exponential atmosphere
%pressure model
%Constants
m_air=28.97e-3; %kg/mol
g=9.8; %m/s^2
R_universal=8.314; %J/(mol*K)
gamma=-Parameters.LapseRate/1000; %K/m
pressureconstant=(m_air*g)/(R_universal*gamma);

    WeatherData.Temperature=zeros(size(LidarData.OfflineCombinedTotalCounts));
    WeatherData.Pressure=zeros(size(LidarData.OfflineCombinedTotalCounts));
    limit=find(LidarData.Range<=10500,1,'last');
    %Temperature estimated to be linear until 12 km then constant
    WeatherData.Temperature(1,:)=WeatherData.GroundTemperature+273.15;
    WeatherData.Temperature(2:limit,:)=repmat(WeatherData.Temperature(1,:),limit-1,1)-Parameters.LapseRate
        /1000*(repmat(LidarData.Range(2:limit),1,length(LidarData.Time)));
    WeatherData.Temperature(limit+1:end,:)=repmat(WeatherData.Temperature(limit,:),length(LidarData.Range)
        -limit,1);
    %Pressure from exponential atmospheric pressure model
    WeatherData.Pressure(1,:)=WeatherData.GroundPressure(:);
        WeatherData.Pressure(2:end,:)=WeatherData.Pressure(1,:).*(WeatherData.Temperature(1,:)./...
        WeatherData.Temperature(2:end,:)).^pressureconstant;
end
```

BackscatterRetrieval

```
function [LidarData]=BackscatterRetrieval(LidarData,WeatherData)
%This function finds the molecular backscatter coefficient, the aerosol
%backscatter coefficient, and the backscatter ratio from the lidar
%profiles, the modeled temperature and pressure, and stored calibration
%constants
%% Load in calibration data
    file=pwd;
    cd("F:\Research\Calibration_Data"); load('CalibrationTables0702.mat');
    cd(file);
    P=Results.Pressure;      T=Results.Temperature;
    Eta_m=Results.MolecularEfficiency;
    Eta_c=Results.CombinedEfficiency;
    Cmm=Results.Cmm;     Cmc=Results.Cmc;
    Cam=Results.Cam(1);     Cac=Results.Cac(1);
%preallocate
    LidarData.Cmm=zeros(length(LidarData.Range),length(LidarData.Time));
    LidarData.Cmc=zeros(length(LidarData.Range),length(LidarData.Time));
%interpolate Cmm and Cmc from tables
    for i=1:length(LidarData.Time)
        for j=1:length(LidarData.Range)
            LidarData.Cmm(j,i)=interp2(P,T,Cmm,WeatherData.Pressure(j,i),WeatherData.Temperature(j,i));
            LidarData.Cmc(j,i)=interp2(P,T,Cmc,WeatherData.Pressure(j,i),WeatherData.Temperature(j,i));
        end
    end
    % Store Calibration Constants
    LidarData.Cac=Cac; %Aerosol in Combined
    LidarData.Cam=Cam; %Aerosol in Molecular
    %Store Beamsplitter and Detector Efficiencies
    LidarData.EtaCombined=Eta_c;
    LidarData.EtaMolecular=Eta_m;

    %Create Molecular Backscatter Coefficient
    LidarData.MolecularBackscatterCoefficient=9.94266e-7*(WeatherData.Pressure)./...
        (WeatherData.Temperature);

    %Aerosol Backscatter Coefficient
    LidarData.UnmaskedAerosolBackscatterCoefficient=LidarData.MolecularBackscatterCoefficient.*...
        (LidarData.Cmm*LidarData.EtaMolecular.*(LidarData.OfflineCombinedTotalCounts)./...
        (LidarData.OfflineMolecularTotalCounts)-LidarData.Cmc*LidarData.EtaCombined)./...
        (LidarData.Cac*LidarData.EtaCombined-LidarData.Cam*LidarData.EtaMolecular*...
        (LidarData.OfflineCombinedTotalCounts)./(LidarData.OfflineMolecularTotalCounts));

    %Backscatter Ratio
    LidarData.UnmaskedBackscatterRatio=(LidarData.UnmaskedAerosolBackscatterCoefficient./...
        LidarData.MolecularBackscatterCoefficient)+1;

    fields={'Cmm','Cmc','Cac','Cam','EtaCombined','EtaMolecular'};
        LidarData=rmfield(LidarData,fields);
end
```

FindGradient

```
function [LidarData,CloudMaskData]=FindGradient(LidarData)
%This function uses the sobel operator to find the 2D gradient of the
%Offline Combined Counts
    %Create Sobel Operator
    Gx = [1,2,1;0,0,0;-1,-2,-1]; Gy = Gx';
    temp_x = conv2(LidarData.OfflineCombinedTotalCounts,Gx,'same');
    temp_y = conv2(LidarData.OfflineCombinedTotalCounts,Gy,'same');
    CloudMaskData.OfflineCombinedCountGradient = sqrt(temp_x.^2 + temp_y.^2);
    %Fix data at edges of matrix
    CloudMaskData.OfflineCombinedCountGradient(1,:)=...
        CloudMaskData.OfflineCombinedCountGradient(2,:);
    CloudMaskData.OfflineCombinedCountGradient(:,1)=...
        CloudMaskData.OfflineCombinedCountGradient(:,2);
    CloudMaskData.OfflineCombinedCountGradient(end,:)=...
        CloudMaskData.OfflineCombinedCountGradient(end-1,:);
    CloudMaskData.OfflineCombinedCountGradient(:,end)=...
        CloudMaskData.OfflineCombinedCountGradient(:,end-1);
    %pad bottom 10 cells
    CloudMaskData.OfflineCombinedCountGradient(1:10,:)=1;
end
```

FindVariance

```
function [LidarData,CloudMaskData]=FindVariance(LidarData,CloudMaskData)
%This function finds the variance of 5x5 cells with the variance assigned
%to the center cell of the 5x5 bin. Believe it or not, this is the fastest
%method.
    %Preallocate
    CloudMaskData.OfflineCombinedCountVariance=...
        NaN(size(LidarData.OfflineCombinedTotalCounts));
    %Find Variance of 5x5 matrix surrounding each cell
    temp(:,:,1)=LidarData.OfflineCombinedTotalCounts(1:end-4,1:end-4);
    temp(:,:,2)=LidarData.OfflineCombinedTotalCounts(1:end-4,2:end-3);
    temp(:,:,3)=LidarData.OfflineCombinedTotalCounts(1:end-4,3:end-2);
    temp(:,:,4)=LidarData.OfflineCombinedTotalCounts(1:end-4,4:end-1);
    temp(:,:,5)=LidarData.OfflineCombinedTotalCounts(1:end-4,5:end);
    temp(:,:,6)=LidarData.OfflineCombinedTotalCounts(2:end-3,1:end-4);
    temp(:,:,7)=LidarData.OfflineCombinedTotalCounts(2:end-3,2:end-3);
    temp(:,:,8)=LidarData.OfflineCombinedTotalCounts(2:end-3,3:end-2);
    temp(:,:,9)=LidarData.OfflineCombinedTotalCounts(2:end-3,4:end-1);
    temp(:,:,10)=LidarData.OfflineCombinedTotalCounts(2:end-3,5:end);
    temp(:,:,11)=LidarData.OfflineCombinedTotalCounts(3:end-2,1:end-4);
    temp(:,:,12)=LidarData.OfflineCombinedTotalCounts(3:end-2,2:end-3);
    temp(:,:,13)=LidarData.OfflineCombinedTotalCounts(3:end-2,3:end-2);
    temp(:,:,14)=LidarData.OfflineCombinedTotalCounts(3:end-2,4:end-1);
    temp(:,:,15)=LidarData.OfflineCombinedTotalCounts(3:end-2,5:end);
    temp(:,:,16)=LidarData.OfflineCombinedTotalCounts(4:end-1,1:end-4);
    temp(:,:,17)=LidarData.OfflineCombinedTotalCounts(4:end-1,2:end-3);
    temp(:,:,18)=LidarData.OfflineCombinedTotalCounts(4:end-1,3:end-2);
    temp(:,:,19)=LidarData.OfflineCombinedTotalCounts(4:end-1,4:end-1);
    temp(:,:,20)=LidarData.OfflineCombinedTotalCounts(4:end-1,5:end);
    temp(:,:,21)=LidarData.OfflineCombinedTotalCounts(5:end,1:end-4);
    temp(:,:,22)=LidarData.OfflineCombinedTotalCounts(5:end,2:end-3);
    temp(:,:,23)=LidarData.OfflineCombinedTotalCounts(5:end,3:end-2);
    temp(:,:,24)=LidarData.OfflineCombinedTotalCounts(5:end,4:end-1);
    temp(:,:,25)=LidarData.OfflineCombinedTotalCounts(5:end,5:end);

    Variancetemp=var(temp,0,3);
    CloudMaskData.OfflineCombinedCountVariance(3:end-2,3:end-2)=Variancetemp;

    CloudMaskData.OfflineCombinedCountVariance=fillmissing(CloudMaskData.OfflineCombinedCountVariance,'
        nearest',1);
    CloudMaskData.OfflineCombinedCountVariance=fillmissing(CloudMaskData.OfflineCombinedCountVariance,'
        nearest',2);

    %pad bottom 10 cells
    CloudMaskData.OfflineCombinedCountVariance(1:10,:)=1;
end
```

FindClouds

```
function [LidarData]=FindClouds(LidarData,CloudMaskData)
%This function finds areas in the Lidar Profiles with high values for the
%NcOffline count gradient and creates a cloud mask

%set threshold for MCS count gradient
gradientthreshold=1e7; variancethreshold=1e9;
%find clouds and create mask
LidarData.CloudMask=(CloudMaskData.OfflineCombinedCountGradient>gradientthreshold...
| CloudMaskData.OfflineCombinedCountVariance>variancethreshold);
%convert from logical to double
LidarData.CloudMask=double(LidarData.CloudMask);
%also mask above clouds
tempmask=LidarData.CloudMask;
for i=1:length(tempmask(1,:))
    cloudheight=find(tempmask(:,i)==1,1);
    if ~isempty(cloudheight)
        LidarData.CloudMask(cloudheight:end,i)=NaN;
    end
end
LidarData.CloudMask(tempmask==1)=1;
end
```

SNRMasking

```
function [LidarData]=SNRMasking(LidarData)
%This function masks data with too low of a signal-to-noise ratio. It also
%masks data above the lowest point with too few counts
SNROfflineCombined=abs(LidarData.OfflineCombinedTotalCounts.^0.5);
SNROfflineMolecular=abs(LidarData.OfflineMolecularTotalCounts.^0.5);
%preallocate
SNRMaskOfflineCombined=zeros(size(LidarData.OfflineCombinedTotalCounts));
```

```
SNRMaskOfflineMolecular=zeros(size(LidarData.OfflineCombinedTotalCounts));
%create mask
SNRMaskOfflineCombined(SNROfflineCombined<3)=1;
SNRMaskOfflineCombined(LidarData.OfflineCombinedTotalCounts<0)=1;
SNRMaskOfflineMolecular(SNROfflineMolecular<3)=1;
SNRMaskOfflineMolecular(LidarData.OfflineMolecularTotalCounts<0)=1;
%add SNR masks for both channels
LidarData.SNRMask=SNRMaskOfflineCombined+...
    SNRMaskOfflineMolecular;
%create mask
LidarData.SNRMask(LidarData.SNRMask>0)=1;
%find lowest SNR location and mask above it
for i=1:length(LidarData.Time)
    ind=find(LidarData.SNRMask(:,i)==1,1,'first');
    LidarData.SNRMask(ind:end,i)=1;
end
end
```

ApplyMasks

```
function [LidarData]=ApplyMasks(LidarData)
%This function applies the SNR and cloud masks and creates a combined mask
%Create new variables
LidarData.MaskedOfflineCombinedTotalCounts=LidarData.OfflineCombinedTotalCounts;
LidarData.MaskedOfflineMolecularTotalCounts=LidarData.OfflineMolecularTotalCounts;
LidarData.MaskedOnlineCombinedTotalCounts=LidarData.OnlineCombinedTotalCounts;
LidarData.MaskedOnlineMolecularTotalCounts=LidarData.OnlineMolecularTotalCounts;
LidarData.MaskedBackscatterRatio=LidarData.UnmaskedBackscatterRatio;
LidarData.MaskedAerosolBackscatterCoefficient=LidarData.UnmaskedAerosolBackscatterCoefficient;
%Apply SNR Mask
LidarData.MaskedOfflineCombinedTotalCounts(LidarData.SNRMask==1)=NaN;
LidarData.MaskedOfflineMolecularTotalCounts(LidarData.SNRMask==1)=NaN;
LidarData.MaskedOnlineCombinedTotalCounts(LidarData.SNRMask==1)=NaN;
LidarData.MaskedOnlineMolecularTotalCounts(LidarData.SNRMask==1)=NaN;
LidarData.MaskedBackscatterRatio(LidarData.SNRMask==1)=NaN;
LidarData.MaskedAerosolBackscatterCoefficient(LidarData.SNRMask==1)=NaN;
%Apply Cloud Mask
LidarData.MaskedOfflineCombinedTotalCounts(LidarData.CloudMask==1)=NaN;
LidarData.MaskedOfflineMolecularTotalCounts(LidarData.CloudMask==1)=NaN;
LidarData.MaskedOnlineCombinedTotalCounts(LidarData.CloudMask==1)=NaN;
LidarData.MaskedOnlineMolecularTotalCounts(LidarData.CloudMask==1)=NaN;
LidarData.MaskedBackscatterRatio(LidarData.CloudMask==1)=NaN;
LidarData.MaskedAerosolBackscatterCoefficient(LidarData.CloudMask==1)=NaN;
%Create Combined Masks
LidarData.CombinedMask=zeros(size(LidarData.SNRMask));
LidarData.CombinedMask(isnan(LidarData.CloudMask))=NaN;
LidarData.CombinedMask(LidarData.SNRMask==1)=NaN;
LidarData.CombinedMask(LidarData.CloudMask==1)=1;
end
```

DataRemoval

```
function [LidarData]=DataRemoval(LidarData,Parameters)
%This function removes all data that is too low to be useful due to
%afterpulsing effects in the APDs
    %Find Minimum Range- closest point to
    [~,minrangeindex]=min(abs(LidarData.Range-Parameters.MinRange));
    %Remove Low Range Data only for offline channels
    LidarData.OfflineCombinedTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.OfflineMolecularTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.OnlineCombinedTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.OnlineCombinedTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedOfflineCombinedTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedOfflineMolecularTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedOnlineCombinedTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedOnlineMolecularTotalCounts(1:minrangeindex-1,:)=NaN;
    LidarData.MolecularBackscatterCoefficient(1:minrangeindex-1,:)=NaN;
    LidarData.UnmaskedAerosolBackscatterCoefficient(1:minrangeindex-1,:)=NaN;
    LidarData.UnmaskedBackscatterRatio(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedBackscatterRatio(1:minrangeindex-1,:)=NaN;
    LidarData.MaskedAerosolBackscatterCoefficient(1:minrangeindex-1,:)=NaN;
end
```

FindCloudHeight

```
function [BoundaryLayers]=FindCloudHeight(LidarData,Parameters)
 %This function finds the bin location of the lowest cloud and stores it as
 %a CloudHeight variable
%% Create Cloud Height Vector
BoundaryLayers.CloudHeight=NaN(1,length(LidarData.Time));
for i=1:length(LidarData.Time)
    temp=[];
   temp=find(LidarData.CloudMask(:,i)==1,1,'first');
   if ~isempty(temp)
       BoundaryLayers.CloudHeight(i)=temp;
   end
end
BoundaryLayers.CloudHeightOriginal=BoundaryLayers.CloudHeight;
BoundaryLayers.CloudHeightOriginalsmooth=...
    smoothdata(BoundaryLayers.CloudHeightOriginal,'g',round(60/Parameters.dt));
BoundaryLayers.CloudHeightOriginalsmooth(isnan(...
    BoundaryLayers.CloudHeightOriginal))=NaN;
end
```

NormalizeData

```
function [LidarData]=NormalizeData(LidarData,BoundaryLayers)
%This function normalizes the aerosol backscatter coefficient to the
%average value at low ranges
%% Unmasked
    %create normalized matrices for HWT computation
    [~,limithighformean]=min(abs(LidarData.Range-400));
    LidarData.UnmaskedNormalizedAerosolBackscatterCoefficient=...
        NaN(size(LidarData.UnmaskedAerosolBackscatterCoefficient));
    limithigh=200*ones(1,length(LidarData.Time));
    for i=1:length(LidarData.Time)
        temp=LidarData.UnmaskedAerosolBackscatterCoefficient(1:limithigh(i),i);
        temp(1:5)=NaN;
        tempmean=nanmean(temp(1:limithighformean));
        LidarData.UnmaskedNormalizedAerosolBackscatterCoefficient(1:length(temp),i)=temp./tempmean;
    end
    %% Masked
    %create normalized matrices for HWT computation
    LidarData.MaskedNormalizedAerosolBackscatterCoefficient=...
        NaN(size(LidarData.MaskedAerosolBackscatterCoefficient));
    limithigh=200*ones(1,length(LidarData.Time));
    limithigh(~isnan(BoundaryLayers.CloudHeight))=...
        BoundaryLayers.CloudHeight(~isnan(BoundaryLayers.CloudHeight));
    for i=1:length(LidarData.Time)
        temp=LidarData.MaskedAerosolBackscatterCoefficient(1:limithigh(i),i);
        temp(1:5)=NaN;
        tempmean=nanmean(temp(1:min([limithigh(i) limithighformean])));
        LidarData.MaskedNormalizedAerosolBackscatterCoefficient(1:length(temp),i)=temp./tempmean;
    end
    LidarData.MaskedNormalizedAerosolBackscatterCoefficient...
    (LidarData.CombinedMask~=0)=NaN;
end
```

BackscatterSmoothing

```
function [LidarData]=BackscatterSmoothing(LidarData,Parameters)
%This function applies a Gaussian smoothing filter to the aerosol optical
%properties
%% Masked Data
% LidarData with smoothed Backscatter Ratio and Cofficients
    LidarData.MolecularBackscatterCoefficientsmooth=...
        smoothdata(LidarData.MolecularBackscatterCoefficient,2,'gaussian',round(60/Parameters.dt));
    LidarData.MaskedAerosolBackscatterCoefficientsmooth=...
        smoothdata(LidarData.MaskedAerosolBackscatterCoefficient,2,'gaussian',round(60/Parameters.dt));
    LidarData.MaskedBackscatterRatiosmooth=...
        smoothdata(LidarData.MaskedBackscatterRatio,2,'gaussian',round(60/Parameters.dt));
    LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth=...
        smoothdata(LidarData.MaskedNormalizedAerosolBackscatterCoefficient,2,'gaussian',60/Parameters.dt);
    %Mask Smoothed Data
    LidarData.MolecularBackscatterCoefficientsmooth...
            (LidarData.CombinedMask(:,:)~=0)=NaN;
    LidarData.MaskedAerosolBackscatterCoefficientsmooth...
            (LidarData.CombinedMask(:,:)~=0)=NaN;
    LidarData.MaskedBackscatterRatiosmooth...
            (LidarData.CombinedMask(:,:)~=0)=NaN;
    LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth...
            (LidarData.CombinedMask(:,:)~=0)=NaN;
    LidarData.MaskedNormalizedBackscatterRatiosmooth...
            (LidarData.CombinedMask(:,:)~=0)=NaN;
```

```
%% Unmasked Data
LidarData.UnmaskedAerosolBackscatterCoefficientsmooth=...
        smoothdata(LidarData.UnmaskedAerosolBackscatterCoefficient,2,'gaussian',round(60/Parameters.dt));
    LidarData.UnmaskedBackscatterRatiosmooth=...
        smoothdata(LidarData.UnmaskedBackscatterRatio,2,'gaussian',round(60/Parameters.dt));
    LidarData.UnmaskedNormalizedAerosolBackscatterCoefficientsmooth=...
        smoothdata(LidarData.UnmaskedNormalizedAerosolBackscatterCoefficient,2,'gaussian',60/Parameters.dt
            );
end
```

HaarWaveletTransform

```
function [LidarData]=HaarWaveletTransform(LidarData,Parameters)
%This function performs the Haar Wavelet Transform on the normalized
%aerosol backscatter coefficient
%% Aerosol Backscatter Coefficient
% Masked
LidarData.HWTRange=(LidarData.Range(2:end)+LidarData.Range(1:end-1))/2;
dilation=(6:2:32);
dilationmeters=dilation*(LidarData.Range(2)-LidarData.Range(1));
UnmaskedHWTBa=NaN(length(LidarData.HWTRange),length(LidarData.Time),length(dilation));
for i=1:length(dilation)
    a=dilation(i);
    h=[]; h(1:a/2)=-1/a; h(a/2+1:a)=1/a;
    h=h';
    %perform HWT
    HWTmatrix=[];
    HWTmatrix=conv2(LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth,h);
    %trim matrix
    UnmaskedHWTBa(:,:,i)=HWTmatrix(1+a/2:end-a/2,:);
end

% use parabolic dilation equation
% range_HWT(km)^2=dilation(km)
scaledrangekm=(LidarData.HWTRange/1000)*.44;
dilationkm=(dilationmeters/1000);

%preallocate
LidarData.UnmaskedHWTBa=zeros(size(UnmaskedHWTBa(:,:,1)));
LidarData.dilation=NaN(size(LidarData.HWTRange));
indexlow=9;

for j=1:length(dilation)
    indexhigh=find(dilationkm(j)>scaledrangekm,1,'last');
    if indexhigh>indexlow
        LidarData.UnmaskedHWTBa(indexlow:indexhigh,:)=UnmaskedHWTBa(indexlow:indexhigh,:,j);
        LidarData.dilation(indexlow:indexhigh)=dilationmeters(j);
    else
        LidarData.UnmaskedHWTBa(indexlow,:)=UnmaskedHWTBa(indexlow,:,j);
        LidarData.dilation(indexlow:indexhigh)=dilationmeters(j);
    end
    indexlow=indexhigh+1;
end

%last rep
LidarData.UnmaskedHWTBa(indexlow:end,:)=UnmaskedHWTBa(indexlow:end,:,j);
LidarData.dilation(indexlow:end)=LidarData.dilation(indexhigh);
%create hwt mask
LidarData.HWTCombinedMask=LidarData.CombinedMask(2:end,:);
%mask hwt
LidarData.MaskedHWTBa=LidarData.UnmaskedHWTBa;
LidarData.MaskedHWTBa(LidarData.HWTCombinedMask~=0)=NaN;
%smooth
LidarData.MaskedHWTBasmooth=smoothdata(...
    LidarData.MaskedHWTBa,2,'g',round(15/Parameters.dt));
LidarData.UnmaskedHWTBasmooth=smoothdata(...
    LidarData.UnmaskedHWTBa,2,'g',round(15/Parameters.dt));
%mask hwt
LidarData.MaskedHWTBasmooth(LidarData.HWTCombinedMask~=0)=NaN;
end
```

BoundaryLayerMainFunction

```
function [BoundaryLayers]=BoundaryLayersMainFunction(LidarData,BoundaryLayers,Parameters,SunriseSunsetData
    )
%This function is the main function of the PBLh retrieval that calls other
%functions for each step in the retrieval algorithm
%find layers that have no time dependence
%Find approximate location of free atmosphere
[BoundaryLayers]=FindFreeAtmosphere(LidarData,BoundaryLayers,Parameters);
```

```
[BoundaryLayers]=FreeAtmosphereDiscontinuities(LidarData,BoundaryLayers,Parameters);
%find capping inversion layer
[BoundaryLayers]=FindCappingInversion(LidarData,BoundaryLayers,Parameters);
[BoundaryLayers]=CappingInversionDiscontinuities(LidarData,BoundaryLayers,Parameters);
%find convection point
[BoundaryLayers]=FindConvectionPoint(BoundaryLayers,Parameters,SunriseSunsetData);
%determine cloud type
[BoundaryLayers]=DetermineCloudType(BoundaryLayers,LidarData,Parameters,SunriseSunsetData);
%find residual layers
[BoundaryLayers]=FindResidualLayers(LidarData,BoundaryLayers,SunriseSunsetData,Parameters);
%find mixed layers
[BoundaryLayers]=FindMixedLayers(LidarData,BoundaryLayers,SunriseSunsetData,Parameters);
%iteratively solve for mixed layers
[BoundaryLayers]=MixedLayerDiscontinuities(LidarData,BoundaryLayers,Parameters,SunriseSunsetData);
%smooth all boundary layer value-added products
[BoundaryLayers]=BoundaryLayerSmoothing(BoundaryLayers,Parameters);
end
```

FindFreeAtmosphere

```
function [BoundaryLayers]=FindFreeAtmosphere(LidarData,BoundaryLayers,Parameters)
%This function finds the location of the free troposphere as part of the
%PBLH retrieval
%Create FreeAtmosphere Vector
BoundaryLayers.FreeAtmosphere=NaN(1,length(LidarData.Time));
%create threshold for minimum height of free atmosphere
threshold=40;
for i=1:numel(LidarData.Time)
    temp1=find(LidarData.UnmaskedAerosolBackscatterCoefficientsmooth(threshold:end,i)<5e-8,1,'first');
    temp=temp1+threshold-1+8;
    if ~isempty(temp)
            BoundaryLayers.FreeAtmosphere(i)=temp;
    end
end
BoundaryLayers.FreeAtmosphere(BoundaryLayers.FreeAtmosphere<threshold)=NaN;
%create smoothed free atmosphere
BoundaryLayers.FreeAtmospheresmooth=smoothdata(BoundaryLayers.FreeAtmosphere,'g',round(60/Parameters.dt));
%delete smoothed points that should be NaN
BoundaryLayers.FreeAtmospheresmooth(isnan(BoundaryLayers.FreeAtmosphere))=NaN;
%adjust Cloud Height to only include clouds below the capping inversion
BoundaryLayers.CloudHeight(BoundaryLayers.CloudHeight>BoundaryLayers.FreeAtmospheresmooth)=NaN;
end
```

FreeAtmosphereDiscontinuities

```
function [BoundaryLayers]=FreeAtmosphereDiscontinuities(LidarData,BoundaryLayers,Parameters)
%This function removes discontinuous points from the free atmosphere vector
%store original
BoundaryLayers.FreeAtmosphereOriginal=BoundaryLayers.FreeAtmosphere;
BoundaryLayers.FreeAtmospheresmoothOriginal=BoundaryLayers.FreeAtmospheresmooth;
%remove discontinuous points
diff=(abs(BoundaryLayers.FreeAtmosphere-BoundaryLayers.FreeAtmospheresmooth)>8 ...
    | (isnan(BoundaryLayers.FreeAtmosphere) & ~isnan(BoundaryLayers.FreeAtmospheresmooth)));
diffind=find(diff==1);
BoundaryLayers.FreeAtmosphere(diffind)=NaN;
%resmooth
BoundaryLayers.FreeAtmospheresmooth=...
    smoothdata(BoundaryLayers.FreeAtmosphere,'g',round(120/Parameters.dt));
iter=1;
while ~isempty(diffind) && iter<10
    if ~isempty(diffind)
        for i=1:length(diffind)
            try
                limitlow=floor(BoundaryLayers.FreeAtmospheresmooth(diffind(i)))-16;
                limithigh=ceil(BoundaryLayers.FreeAtmospheresmooth(diffind(i)));
                temp=LidarData.UnmaskedAerosolBackscatterCoefficientsmooth(limitlow:limithigh,diffind(i));
                tempind=find(temp<5e-8,1,'first');
                if ~isempty(tempind)
                        ind=tempind+limitlow-1+8;
                        BoundaryLayers.FreeAtmosphere(diffind(i))=ind;
                else
                        [~,ind]=min(temp);
                        ind=ind+limitlow-1+8;
                        BoundaryLayers.FreeAtmosphere(diffind(i))=ind;
                end
            catch
            end
        end
    end
iter=iter+1;
%resmooth
```

```
BoundaryLayers.FreeAtmospheresmooth=...
    smoothdata(BoundaryLayers.FreeAtmosphere,'g',round(120/Parameters.dt));
BoundaryLayers.FreeAtmospheresmooth(BoundaryLayers.FreeAtmospheresmooth==0)=NaN;

diff=(abs(BoundaryLayers.FreeAtmosphere-BoundaryLayers.FreeAtmospheresmooth)>8 ...
    | (isnan(BoundaryLayers.FreeAtmosphere) & ~isnan(BoundaryLayers.FreeAtmospheresmooth)));
diffind=find(diff==1);
end
%fill nan-values
BoundaryLayers.FreeAtmosphere=...
    fillmissing(BoundaryLayers.FreeAtmosphere,'nearest');
%last smoothing iteration
BoundaryLayers.FreeAtmospheresmooth=...
    smoothdata(BoundaryLayers.FreeAtmosphere,'g',round(120/Parameters.dt));
%adjust Cloud Height to only include clouds below the capping inversion
BoundaryLayers.CloudHeight(BoundaryLayers.CloudHeight>BoundaryLayers.FreeAtmospheresmooth)=NaN;
end
```

FindCappingInversion

```
function  [BoundaryLayers]=FindCappingInversion(LidarData,BoundaryLayers,Parameters)
%This function finds the location of the gradient at the top of either the
%mixed layer or the residual layer
%preallocate
BoundaryLayers.CappingInversion=NaN(1,length(LidarData.Time));
%create limits
limitlow=20*ones(1,length(LidarData.Time));
limithigh=160*ones(1,length(LidarData.Time));
limithigh(~isnan(BoundaryLayers.FreeAtmospheresmooth))=...
    round(BoundaryLayers.FreeAtmospheresmooth(~isnan(BoundaryLayers.FreeAtmospheresmooth)));
for i=1:length(LidarData.Time)
    if ~isnan(BoundaryLayers.CloudHeight(i))
        limitlow(i)=BoundaryLayers.CloudHeight(i)-10;
        limithigh(i)=BoundaryLayers.CloudHeight(i)+50;
    end
end
limitlow(limitlow<30)=30;
limithigh(limithigh>160)=160;
%Use Aerosol Backscatter Coefficient
    for i=1:length(LidarData.Time)
        if ~isnan(limithigh(i))
            if isnan(BoundaryLayers.CloudHeight(i))
                    ind=[];
                    temp=LidarData.UnmaskedHWTBasmooth(limitlow(i):limithigh(i),i);
                    log=temp>0.05;
                    ind=find(log==1,1,'last');
                    ind=ind+limitlow(i)-1;
                    if ~isempty(ind)
                        BoundaryLayers.CappingInversion(i)=ind;
                    elseif isempty(ind)
                        [~,I]=max(temp);
                        if ~isempty(I)
                            I=I+limitlow(i)-1;
                            BoundaryLayers.CappingInversion(i)=I;
                        end
                    end
            else
                temp=LidarData.UnmaskedHWTBasmooth(limitlow(i):limithigh(i),i);
                [~,I]=max(temp);
                if ~isnan(I)
                    I=I+limitlow(i)-1;
                    temp=LidarData.CloudMask(I:limithigh(i),i);
                    temp2=find(isnan(temp),1,'first');
                        if ~isnan(temp2)
                            temp3=temp2+I-1;
                            BoundaryLayers.CappingInversion(i)=temp3;
                        end
                end
            end
        end
    end
%create smoothed capping inversion plot
BoundaryLayers.CappingInversionsmooth=smoothdata(BoundaryLayers.CappingInversion,'g',round(60/Parameters.
    dt));
%delete smoothed points that should be NaN
BoundaryLayers.CappingInversionsmooth(isnan(BoundaryLayers.CappingInversion))=NaN;
%move smoothed capping inversion below smoothed free atmosphere
BoundaryLayers.CappingInversionsmooth(BoundaryLayers.CappingInversionsmooth>BoundaryLayers.
    FreeAtmospheresmooth)=...
BoundaryLayers.FreeAtmospheresmooth(BoundaryLayers.CappingInversionsmooth>BoundaryLayers.
    FreeAtmospheresmooth);
end
```

CappingInversionDiscontinuities

```
function [BoundaryLayers]=CappingInversionDiscontinuities(LidarData,BoundaryLayers,Parameters)
%This function removes discontinuous points from the cappin inversion
%vector and looks for new points new the smoothe =d average value
%store original
BoundaryLayers.CappingInversionOriginal=BoundaryLayers.CappingInversion;
BoundaryLayers.CappingInversionsmoothOriginal=BoundaryLayers.CappingInversionsmooth;
%remove discontinuous points
diff=(abs(BoundaryLayers.CappingInversion-BoundaryLayers.CappingInversionsmooth)>8 ...
    | (isnan(BoundaryLayers.CappingInversion) & ~isnan(BoundaryLayers.CappingInversionsmooth)));
diffind=find(diff==1);
BoundaryLayers.CappingInversion(diffind)=NaN;
%resmooth
BoundaryLayers.CappingInversionsmooth=...
    smoothdata(BoundaryLayers.CappingInversion,'g',round(120/Parameters.dt));
iter=1;
while ~isempty(diffind) && iter<10
    if ~isempty(diffind)
        for i=1:length(diffind)
            limitlow=round(BoundaryLayers.CappingInversionsmooth(diffind(i)))-8;
            limithigh=round(BoundaryLayers.CappingInversionsmooth(diffind(i)))+8;
            try
                if isnan(BoundaryLayers.CloudHeight(diffind(i)))
                    ind=[];
                    temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,diffind(i));
                    log=temp>0.05;
                    ind=find(log==1,1,'last');
                    ind=ind+limitlow-1;
                    if ~isempty(ind)
                        BoundaryLayers.CappingInversion(diffind(i))=ind;
                    elseif isempty(ind)
                        [~,I]=max(temp);
                        if ~isempty(I)
                            I=I+limitlow-1;
                            BoundaryLayers.CappingInversion(diffind(i))=I;
                        end
                    end

                else
                    limitlow=BoundaryLayers.CappingInversion(diffind(i))-8;
                    limithigh=BoundaryLayers.CappingInversion(diffind(i))+8;
                    temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,i);
                    [~,I]=max(temp);
                    if ~isnan(I)
                        I=I+limitlow-1;
                        temp=LidarData.CloudMask(I:limithigh,diffind(i));
                        temp2=find(isnan(temp),1,'first');
                            if ~isnan(temp2)
                                temp3=temp2+I-1;
                                BoundaryLayers.CappingInversion(diffind(i))=temp3;
                            end
                    end
                end
            catch
            end
        end
    end
iter=iter+1;
%resmooth
BoundaryLayers.CappingInversionsmooth=...
    smoothdata(BoundaryLayers.CappingInversion,'g',round(120/Parameters.dt));
BoundaryLayers.CappingInversionsmooth(BoundaryLayers.CappingInversionsmooth==0)=NaN;

diff=(abs(BoundaryLayers.CappingInversion-BoundaryLayers.CappingInversionsmooth)>8 ...
    | (isnan(BoundaryLayers.CappingInversion) & ~isnan(BoundaryLayers.CappingInversionsmooth)));
diffind=find(diff==1);
end
%last smoothing iteration
BoundaryLayers.CappingInversionsmooth=...
    smoothdata(BoundaryLayers.CappingInversion,'g',round(120/Parameters.dt));
%remove nan values
BoundaryLayers.CappingInversionsmooth(isnan(BoundaryLayers.CappingInversion))=...
    NaN;
%correct smoothed capping inversion to below smoothed free atmosphere
BoundaryLayers.CappingInversionsmooth(BoundaryLayers.CappingInversionsmooth>BoundaryLayers.
     FreeAtmospheresmooth)=...
    BoundaryLayers.FreeAtmospheresmooth(BoundaryLayers.CappingInversionsmooth>BoundaryLayers.
         FreeAtmospheresmooth);
end
```

FindConvectionPoint

```
function [BoundaryLayers]=FindConvectionPoint(BoundaryLayers,Parameters,SunriseSunsetData)
%This function finds the lowest point of the daytime capping inversion: the
%convection point
localdaylength=length(SunriseSunsetData.SunriseIndex);
BoundaryLayers.ConvectionHeight=NaN(1,localdaylength);
BoundaryLayers.ConvectionIndex=NaN(1,localdaylength);
for i=1:localdaylength
    daystart=SunriseSunsetData.SunriseIndex(i);
    dayend=SunriseSunsetData.SunsetIndex(i);
    diff=dayend-daystart;
    startindex=round(daystart+0.3*diff);
    endindex=round(daystart+0.7*diff);
    [BoundaryLayers.ConvectionHeight(i),BoundaryLayers.ConvectionIndex(i)]=...
        min(BoundaryLayers.CappingInversionsmooth(startindex:endindex));
    BoundaryLayers.ConvectionIndex(i)=BoundaryLayers.ConvectionIndex(i)+startindex-1;
end
end
```

DetermineCloudType

```
function [BoundaryLayers]=DetermineCloudType(BoundaryLayers,LidarData,Parameters,SunriseSunsetData)
%This function determines whether the clouds are beneath the PBLH or
%decoupled from the PBLH
%1=Boundary Layer; 0=no cloud
%Boundary Layer
%This vector only is 1 for the daytime time points
% create morning vector
localdaylength=length(Parameters.Day)-1;
morning=zeros(1,length(LidarData.Time));
morninglimithigh=zeros(1,length(LidarData.Time));
for i=1:localdaylength
    if i==1
        daystart=find(SunriseSunsetData.Day_Night==1,1,'first');
        dayend=find(SunriseSunsetData.Day_Night(daystart+1:end)==0,1,'first')-1+daystart;
    else
        daystart=find(SunriseSunsetData.Day_Night(dayend+1:end)==1,1,'first')+dayend;
        dayend=find(SunriseSunsetData.Day_Night(daystart+1:end)==0,1,'first')-1+daystart;
    end
    morning(daystart:BoundaryLayers.ConvectionIndex(i))=1;

    if isnan(BoundaryLayers.ConvectionHeight(i))
        ConvectionHeight=30;
        ConvectionIndex=200;
    else
        ConvectionHeight=BoundaryLayers.ConvectionHeight(i);
        ConvectionIndex=BoundaryLayers.ConvectionIndex(i);
    end
    morninglimithigh(SunriseSunsetData.SunriseIndex(i):ConvectionIndex)=ceil(12+...
        ((ConvectionHeight-12)/(ConvectionIndex-SunriseSunsetData.SunriseIndex(i)))...
        *((SunriseSunsetData.SunriseIndex(i):ConvectionIndex)-SunriseSunsetData.SunriseIndex(i)));
end
morninglimithigh(morning==0)=NaN;
temp0=NaN(1,length(BoundaryLayers.ConvectionIndex));
temp1=NaN(1,length(BoundaryLayers.ConvectionIndex));
temp2=NaN(1,length(BoundaryLayers.ConvectionIndex));
for k=1:length(BoundaryLayers.ConvectionIndex)
    temp0(k)=BoundaryLayers.ConvectionIndex(k);
    temp1(k)=SunriseSunsetData.SunriseIndex(find(SunriseSunsetData.SunriseIndex<BoundaryLayers.
        ConvectionIndex(k),1,'last'));
    temp2(k)=SunriseSunsetData.SunsetIndex(find(SunriseSunsetData.SunsetIndex>BoundaryLayers.
        ConvectionIndex(k),1,'first'));
end

beforeindex=NaN(1,length(LidarData.Time));
afterindex=NaN(1,length(LidarData.Time));
convectionheight=NaN(1,length(LidarData.Time));
for j=1:length(BoundaryLayers.ConvectionIndex)
    beforeindex(temp1(j):temp0(j))=1;
    afterindex(temp0(j)+1:temp2(j))=1;
    convectionheight(temp1(j):temp0(j))=BoundaryLayers.ConvectionHeight(j);
end
BoundaryLayers.CloudType=zeros(1,length(LidarData.Time));
for i=1:length(LidarData.Time)
    if beforeindex(i)==1
        if ~isnan(BoundaryLayers.CloudHeight(i))
            if (BoundaryLayers.CappingInversion(i)>BoundaryLayers.CloudHeight(i)) && (BoundaryLayers.
                CloudHeight(i)<morninglimithigh(i))
                BoundaryLayers.CloudType(i)=1;
            else
                BoundaryLayers.CloudType(i)=2;
            end
        end
```

```
    end
    if afterindex(i)==1
        if ~isnan(BoundaryLayers.CloudHeight(i))
            if BoundaryLayers.CappingInversion(i)>BoundaryLayers.CloudHeight(i)
                BoundaryLayers.CloudType(i)=1;
            else
                BoundaryLayers.CloudType(i)=2;
            end
        end
    end
end
end
```

FindResidualLayers

```
function [BoundaryLayers]=FindResidualLayers(LidarData,BoundaryLayers,SunriseSunsetData,Parameters)
%This function finds the negative HWT associated with the bottom of a
%lofted residual layer
%% create limit vectors
localdaylength=length(Parameters.Day)-1;
morning=zeros(1,length(LidarData.Time));
morninglimitlow=zeros(1,length(LidarData.Time));
for i=1:localdaylength
    daystart=SunriseSunsetData.SunriseIndex(i);
    dayend=SunriseSunsetData.SunsetIndex(i);
    morning(daystart:BoundaryLayers.ConvectionIndex(i))=1;
    morninglimitlow(daystart:dayend)=ceil(9+...
        ((16-8)/(BoundaryLayers.ConvectionIndex(i)-daystart))...
        *((daystart:dayend)-(daystart-1)));
end
morninglimitlow(morning==0)=NaN;
limitlow(SunriseSunsetData.Day_Night==0)=8;
limitlow(SunriseSunsetData.Day_Night==1)=16;
limitlow(~isnan(morninglimitlow))=morninglimitlow(~isnan(morninglimitlow));
limithigh=100*ones(1,length(LidarData.Time));
limithigh(~isnan(BoundaryLayers.CappingInversion))=BoundaryLayers.CappingInversion(~isnan(BoundaryLayers.
    CappingInversion));
limithigh(limithigh>100)=100;
%% Find Residual Layers
BoundaryLayers.ResidualLayer=NaN(1,length(LidarData.Time));
for i=1:length(LidarData.Time)
    if SunriseSunsetData.Day_Night(i)~=-1
        temp=LidarData.UnmaskedHWTBasmooth(limitlow(i):limithigh(i),i);
        log=temp<-.015;
        ind=find(log==1,1,'first');
        ind=ind+limitlow(i)-1;
            if ~isempty(ind)
                BoundaryLayers.ResidualLayer(i)=ind;
            end
    end
end
end
```

FindMixedLayers

```
function [BoundaryLayers]=FindMixedLayers(LidarData,BoundaryLayers,SunriseSunsetData,Parameters)
%This function finds the PBLH
% create morning vector
localdaylength=length(Parameters.Day)-1;
morning=zeros(1,length(LidarData.Time));
morninglimithigh=zeros(1,length(LidarData.Time));
for i=1:localdaylength
    daystart=SunriseSunsetData.SunriseIndex(i);
    dayend=SunriseSunsetData.SunsetIndex(i);
    morning(daystart:BoundaryLayers.ConvectionIndex(i))=1;
    if isnan(BoundaryLayers.ConvectionHeight(i))
        ConvectionHeight=30;
        ConvectionIndex=200;
    else
        ConvectionHeight=BoundaryLayers.ConvectionHeight(i);
        ConvectionIndex=BoundaryLayers.ConvectionIndex(i);
    end
    morninglimithigh(SunriseSunsetData.SunriseIndex(i):ConvectionIndex)=ceil(12+...
        ((ConvectionHeight-12)/(ConvectionIndex-SunriseSunsetData.SunriseIndex(i)))...
        *((SunriseSunsetData.SunriseIndex(i):ConvectionIndex)-SunriseSunsetData.SunriseIndex(i)));
end
morninglimithigh(morning==0)=NaN;
BoundaryLayers.limitlowvector=NaN(1,length(LidarData.Time));
BoundaryLayers.limithighvector=NaN(1,length(LidarData.Time));
BoundaryLayers.MixedLayer=NaN(1,length(LidarData.Time));
for i=1:length(LidarData.Time)
```

```
if SunriseSunsetData.Day_Night(i)==1
    if BoundaryLayers.CloudType(i)~=1
        if morning(i)==1
            limitlow=10;
            if ~isnan(BoundaryLayers.ResidualLayer(i))
                limithigh=BoundaryLayers.ResidualLayer(i);
            elseif ~isnan(BoundaryLayers.CappingInversion(i))
                limithigh=BoundaryLayers.CappingInversion(i);
            elseif ~isnan(BoundaryLayers.CloudHeight(i))
                limithigh=BoundaryLayers.CloudHeight(i);
            else
                limithigh=morninglimithigh(i);
            end
            if limithigh>morninglimithigh(i)
                limithigh=morninglimithigh(i);
            end
            if limithigh<limitlow+3
                limithigh=limitlow+3;
            end
            if limithigh>134
                limithigh=134;
            end
            temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,i);

            if isempty(temp(temp>0))
               BoundaryLayers.MixedLayer(i)=NaN;
            else
                [pks,locs]=findpeaks(temp);
                locs=locs-1+limitlow;
                if temp(1)>temp(2)
                    pks=[temp(1);pks];
                    locs=[limitlow;locs];
                end
                    locs(pks<0.025)=[];
                if isempty(locs)
                    [pks,locs]=findpeaks(temp);
                    locs=locs-1+limitlow;
                    [~,ind2]=max(pks);
                    pks=pks(ind2);
                end
                if ~isempty(locs)
                    pks=pks(1);
                    temp=LidarData.UnmaskedHWTBasmooth(locs(1):limithigh,i);
                    ind=find(temp<pks/1.5,1,'first');
                    if isempty(ind)
                        ind=find(temp>pks/1.5,1,'last');
                    end
                    ind=ind+locs(1)-1;
                    if ~isempty(ind)
                        BoundaryLayers.MixedLayer(i)=ind;
                    end
                end
            end
        else
            limitlow=10;
            templimitlow=[];
            templimitlow=find(LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth(:,i)<0.6 |
                ...
                LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth(:,i)>1.4,1,'first');
            if ~isempty(templimitlow)
                if templimitlow<100
                    limitlow=templimitlow;
                end
            end
            if ~isnan(BoundaryLayers.CappingInversion(i))
                limithigh=BoundaryLayers.CappingInversion(i);
            elseif ~isnan(BoundaryLayers.CloudHeight(i))
                limithigh=BoundaryLayers.CloudHeight(i);
            else
                limithigh=limitlow+100;
            end
            if limithigh<limitlow+3
                limithigh=limitlow+3;
            end
            temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,i);
            [pks,locs]=findpeaks(temp);
            locs(pks<0.04)=[];
            locs=locs-1+limitlow;
            if ~isempty(locs)
                pks=pks(1);
                temp=LidarData.UnmaskedHWTBasmooth(locs(1):limithigh,i);
                ind=find(temp<pks/1.333,1,'first');
                if isempty(ind)
                    ind=find(temp>pks/1.333,1,'last');
                end
                ind=ind+locs(1)-1;
                if ~isempty(ind)
                    BoundaryLayers.MixedLayer(i)=ind;
```

```
                    end
                end
            end
        end
        if BoundaryLayers.CloudType(i)==1
            limitlow=BoundaryLayers.CloudHeight(i)-10;
            limithigh=BoundaryLayers.CloudHeight(i)+20;
            if limitlow<10
                limitlow=10;
            end
            if morning(i)==1
                if limithigh>morninglimithigh(i)
                    limithigh=morninglimithigh(i);
                end
            end
            temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,i);
            [~,I]=max(temp);
            if ~isnan(I)
                I=I+limitlow-1;
                BoundaryLayers.MixedLayer(i)=I;
            end
        end
        BoundaryLayers.limitlowvector(i)=limitlow;
        BoundaryLayers.limithighvector(i)=limithigh;
    end
end
BoundaryLayers.MixedLayersmooth=...
    smoothdata(BoundaryLayers.MixedLayer,'g',round(60/Parameters.dt));
BoundaryLayers.MixedLayersmooth(isnan(...
    BoundaryLayers.MixedLayer))=NaN;
end
```

MixedLayerDiscontinuities

```
function [BoundaryLayers]=MixedLayerDiscontinuities(LidarData,BoundaryLayers,Parameters,SunriseSunsetData)
%This function removes discontinuos points from the PBLH vector and attempts
%to find new PBLH points near the smoothed average
%smooth
BoundaryLayers.MixedLayersmooth=...
    smoothdata(BoundaryLayers.MixedLayer,'g',round(60/Parameters.dt));
BoundaryLayers.MixedLayersmooth(isnan(...
    BoundaryLayers.MixedLayer))=NaN;
%store original
BoundaryLayers.MixedLayerOriginal=BoundaryLayers.MixedLayer;
BoundaryLayers.MixedLayersmoothOriginal=BoundaryLayers.MixedLayersmooth;
%remove discontinuous points
diff=(abs(BoundaryLayers.MixedLayer-BoundaryLayers.MixedLayersmooth)>8 ...
    | (isnan(BoundaryLayers.MixedLayer) & ~isnan(BoundaryLayers.MixedLayersmooth)));
diffind=find(diff==1);
BoundaryLayers.MixedLayer(diffind)=NaN;
BoundaryLayers.MixedLayersmooth(diffind)=NaN;
%resmooth
BoundaryLayers.MixedLayersmooth=...
    smoothdata(BoundaryLayers.MixedLayer,'g',round(60/Parameters.dt));
% create morning vector
localdaylength=length(Parameters.Day)-1;
morning=zeros(1,length(LidarData.Time));
morninglimithigh=zeros(1,length(LidarData.Time));
for i=1:localdaylength
    if i==1
        daystart=find(SunriseSunsetData.Day_Night==1,1,'first');
        dayend=find(SunriseSunsetData.Day_Night(daystart+1:end)==0,1,'first')-1+daystart;
    else
        daystart=find(SunriseSunsetData.Day_Night(dayend+1:end)==1,1,'first')+dayend;
        dayend=find(SunriseSunsetData.Day_Night(daystart+1:end)==0,1,'first')-1+daystart;
    end
    morning(daystart:BoundaryLayers.ConvectionIndex(i))=1;

    if isnan(BoundaryLayers.ConvectionHeight(i))
        ConvectionHeight=30;
        ConvectionIndex=200;
    else
        ConvectionHeight=BoundaryLayers.ConvectionHeight(i);
        ConvectionIndex=BoundaryLayers.ConvectionIndex(i);
    end
    morninglimithigh(SunriseSunsetData.SunriseIndex(i):ConvectionIndex)=ceil(12+...
        ((ConvectionHeight-12)/(ConvectionIndex-SunriseSunsetData.SunriseIndex(i)))...
        *((SunriseSunsetData.SunriseIndex(i):ConvectionIndex)-SunriseSunsetData.SunriseIndex(i)));
end
morninglimithigh(morning==0)=NaN;
iter=1;
while ~isempty(diffind) && iter<5
    if ~isempty(diffind)
        for i=1:length(diffind)
```

```
if SunriseSunsetData.Day_Night(diffind(i))==1
    if BoundaryLayers.CloudType(diffind(i))~=1
        if morning(diffind(i))==1
            limitlow=10;
            if ~isnan(BoundaryLayers.ResidualLayer(diffind(i)))
                limithigh=BoundaryLayers.ResidualLayer(diffind(i));
            elseif ~isnan(BoundaryLayers.CappingInversion(diffind(i)))
                limithigh=BoundaryLayers.CappingInversion(diffind(i));
            elseif ~isnan(BoundaryLayers.CloudHeight(diffind(i)))
                limithigh=BoundaryLayers.CloudHeight(diffind(i));
            else
                limithigh=morninglimithigh(diffind(i));
            end
            if limithigh>morninglimithigh(diffind(i))
                limithigh=morninglimithigh(diffind(i));
            end
            if limithigh>134
                limithigh=134;
            end
            newlimitlow=round(BoundaryLayers.MixedLayersmooth(diffind(i)))-iter;
            newlimithigh=round(BoundaryLayers.MixedLayersmooth(diffind(i)))+iter;
            if newlimitlow>limitlow
                limitlow=newlimitlow;
            end
            if newlimithigh<limithigh
                limithigh=newlimithigh;
            end
            if limithigh<limitlow+2
                limithigh=limitlow+2;
            end
            if BoundaryLayers.CappingInversion(diffind(i))<limithigh
                if BoundaryLayers.CappingInversion(diffind(i))>9
                    limithigh=BoundaryLayers.CappingInversion(diffind(i));
                    limitlow=limithigh-2;
                end
            end
                temp=LidarData.MaskedHWTBasmooth(limitlow:limithigh,diffind(i));
            if isempty(temp(temp>0))
                BoundaryLayers.MixedLayer(diffind(i))=NaN;
            else
            [pks,locs]=findpeaks(temp);
            locs=locs-1+limitlow;
            if temp(1)>temp(2)
                pks=[temp(1);pks];
                locs=[limitlow;locs];
            end
                locs(pks<0.025)=[];
            if isempty(locs)
                [pks,locs]=findpeaks(temp);
                locs=locs-1+limitlow;
                [~,ind2]=max(pks);
                pks=pks(ind2);
            end
        if ~isempty(locs)
            pks=pks(1);
            temp=LidarData.UnmaskedHWTBasmooth(locs(1):limithigh,diffind(i));
            ind=find(temp<pks/1.5,1,'first');
            if isempty(ind)
                ind=find(temp>pks/1.5,1,'last');
            end
            ind=ind+locs(1)-1;
            if ~isempty(ind)
                BoundaryLayers.MixedLayer(diffind(i))=ind;
            end
        end
    end
            %%
        else
            limitlow=10;
            templimitlow=[];
            templimitlow=find(LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth(:,
                diffind(i))<0.6 |...
                LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth(:,diffind(i))
                    >1.5,1,'first');
            if ~isempty(templimitlow)
                if templimitlow<100
                    limitlow=templimitlow;
                end
            end
            if ~isnan(BoundaryLayers.CappingInversion(diffind(i)))
                limithigh=BoundaryLayers.CappingInversion(diffind(i));
            elseif ~isnan(BoundaryLayers.CloudHeight(diffind(i)))
                limithigh=BoundaryLayers.CloudHeight(diffind(i));
            else
                limithigh=limitlow+100;
            end
            if limithigh>134
                limithigh=134;
```

```
                        end
                        newlimitlow=round(BoundaryLayers.MixedLayersmooth(diffind(i)))-iter;
                        newlimithigh=round(BoundaryLayers.MixedLayersmooth(diffind(i)))+iter;
                        if newlimitlow>limitlow
                            limitlow=newlimitlow;
                        end
                        if newlimithigh<limithigh
                            limithigh=newlimithigh;
                        end
                        if limithigh<limitlow+3
                            limithigh=limitlow+3;
                        end
                        if BoundaryLayers.CappingInversion(diffind(i))<limithigh
                            if BoundaryLayers.CappingInversion(diffind(i))>5
                                limithigh=BoundaryLayers.CappingInversion(diffind(i));
                                limitlow=limithigh-2;
                            end
                        end
                        temp=LidarData.MaskedHWTBasmooth(limitlow:limithigh,diffind(i));
                        [pks,locs]=findpeaks(temp);
                        locs(pks<0.04)=[];
                        locs=locs-1+limitlow;
                        if isempty(locs)
                            [pks,locs]=max(temp);
                            locs(pks<0.04)=[];
                            locs=locs-1+limitlow;
                        end
                        if ~isempty(locs)
                            pks=pks(1);
                            temp=LidarData.MaskedHWTBasmooth(locs(1):limithigh,diffind(i));
                            ind=find(temp<pks/1.333,1,'first');
                            if isempty(ind)
                                ind=find(temp>pks/1.333,1,'last');
                            end
                            ind=ind+locs(1)-1;
                            if ~isempty(ind)
                                BoundaryLayers.MixedLayer(diffind(i))=ind;
                            end
                        end
                    end
                end

                if BoundaryLayers.CloudType(diffind(i))==1
                    limitlow=BoundaryLayers.CloudHeight(diffind(i))-5;
                    limithigh=BoundaryLayers.CloudHeight(diffind(i))+25;
                    if limitlow<10
                        limitlow=10;
                    end
                    if morning(i)==1
                        if limithigh>morninglimithigh(diffind(i))
                            limithigh=morninglimithigh(diffind(i));
                        end
                    end
                    temp=LidarData.UnmaskedHWTBasmooth(limitlow:limithigh,(diffind(i)));
                    [~,I]=max(temp);
                    if ~isnan(I)
                        I=I+limitlow-1;
                        BoundaryLayers.MixedLayer(diffind(i))=I;
                    end
                end
            end
        end
    end
iter=iter+1;
%resmooth
BoundaryLayers.MixedLayersmooth=...
    smoothdata(BoundaryLayers.MixedLayer,'g',round(60/Parameters.dt));

diff=(abs(BoundaryLayers.MixedLayer-BoundaryLayers.MixedLayersmooth)>iter ...
    | (isnan(BoundaryLayers.MixedLayer) & ~isnan(BoundaryLayers.MixedLayersmooth)));
diffind=find(diff==1);
end
BoundaryLayers.MixedLayer(diffind)=NaN;
BoundaryLayers.MixedLayersmooth(diffind)=NaN;
%remove smoothing over discontinuous points
BoundaryLayers.MixedLayersmooth(isnan(BoundaryLayers.MixedLayer))=NaN;
end
```

BoundaryLayerSmoothing

```
function [BoundaryLayers]=BoundaryLayerSmoothing(BoundaryLayers,Parameters)
%This function applies a gaussian smoothing filter to VAPs related to the
%PBLH that have not yet been smoothed
BoundaryLayers.CloudHeightsmooth=...
    smoothdata(BoundaryLayers.CloudHeight,'g',round(60/Parameters.dt));
BoundaryLayers.CloudHeightsmooth(isnan(...
    BoundaryLayers.CloudHeight))=NaN;
BoundaryLayers.ResidualLayersmooth=...
    smoothdata(BoundaryLayers.ResidualLayer,'g',round(60/Parameters.dt));
BoundaryLayers.ResidualLayersmooth(isnan(...
    BoundaryLayers.ResidualLayer))=NaN;
end
```

RadiosondeProcessing

```
function [Sondes]=RadiosondeProcessing(LidarData,Sondes,Parameters,BoundaryLayers)
%This function loads the radiosonde m files and finds the aerosol optical
%properties that coincide with the radiosonde launch times
for z=1:length(Sondes)
    NewSondes(z,1).RadiosondeData=struct();
end
for z=1:length(Sondes)
    try
        RadiosondeData=[];
        RadiosondeData=Sondes(z).RadiosondeData;
            for i=1:length(RadiosondeData)
            name=char(RadiosondeData(i).name);
            RadiosondeData(i).month=name(10:11);
            RadiosondeData(i).day=name(13:14);
            hour=name(16:17);
            RadiosondeData(i).hour=hour;
            if hour(1)=='0'
                hour=hour(2);
            end
            hour=str2num(hour);
            minute=name(19:20);
            RadiosondeData(i).minute=minute;
            if minute(1)=='0'
                minute=minute(2);
            end
            minute=str2num(minute);

            second=name(21:22);
            RadiosondeData(i).second=second;
            if second(1)=='0'
                second=second(2);
            end
            second=str2num(second);
            start_time=hour+minute/60+second/3600+24;
            RadiosondeData(i).LidarTime=start_time+RadiosondeData(i).subsampledtime/3600;
            %Find Lidar Profile for each Radiosondes
            ind=find(RadiosondeData(i).subsampledheight>6000,1);
            if isempty(ind)
               ind=length(RadiosondeData(i).subsampledheight)-2;
            end
            LidarHeightShort=RadiosondeData(i).subsampledheight(1:ind);
            LidarGradientHeightShort=RadiosondeData(i).gradientheight(1:ind);
            LidarTimeShort=RadiosondeData(i).LidarTime(1:ind+1);
            [~,lowind]=min(abs(LidarData.Time-LidarTimeShort(1)));
            [~,highind]=min(abs(LidarData.Time-LidarTimeShort(end)));
            indices=lowind:highind;
            RadiosondeData(i).NormalizedAerosolBackscatterCoefficient=[];
            RadiosondeData(i).HaarWaveletTransform=[];
            for j=1:length(indices)-1
                [~,timeindlow]=min(abs(LidarData.Time(indices(j))-LidarTimeShort));
                [~,timeindhigh]=min(abs(LidarData.Time(indices(j+1))-LidarTimeShort));
                timeindhigh=timeindhigh-1;
                heightvector=LidarHeightShort(timeindlow:timeindhigh);
                gradientheightvector=LidarGradientHeightShort(timeindlow:timeindhigh);
                Ba=zeros(1,length(heightvector));
                HWT=zeros(1,length(heightvector));
                for k=1:length(heightvector)
                    [~,ind]=min(abs(heightvector(k)-LidarData.Range));
                    [~,gradind]=min(abs(gradientheightvector(k)-LidarData.Range));
                    try
                        Ba(k)=...
                            LidarData.UnmaskedNormalizedAerosolBackscatterCoefficientsmooth(ind,indices(j)
                                );
                        HWT(k)=...
                            LidarData.UnmaskedHWTBasmooth(gradind,indices(j));
                    catch
                        Ba(k)=NaN;
```

```
                        HWT(k)=NaN;
                    end
                end
            RadiosondeData(i).NormalizedAerosolBackscatterCoefficient=...
                [RadiosondeData(i).NormalizedAerosolBackscatterCoefficient,Ba];
            RadiosondeData(i).HaarWaveletTransform=...
                [RadiosondeData(i).HaarWaveletTransform,HWT];
            end
            RadiosondeData(i).NormalizedAerosolBackscatterCoefficient=...
                smoothdata(RadiosondeData(i).NormalizedAerosolBackscatterCoefficient,'g',15,'omitnan');
            RadiosondeData(i).HaarWaveletTransform=...
                smoothdata(RadiosondeData(i).HaarWaveletTransform,'g',15,'omitnan');
            end
    catch
        RadiosondeData=struct();
    end
        NewSondes(z,1).RadiosondeData=RadiosondeData;
end
Sondes=[];
Sondes=NewSondes;
end
```

VAPSaveFile

```
function VAPSaveFile(BoundaryLayers,Sondes,...
    LidarData,Parameters,SunriseSunsetData,Path,WeatherData)
%This function pulls out the data for the central day in the 3 day span and
%saves the results to an m file
i=2;
    Results.Day=Parameters.Day(i);
    Results.Month=Parameters.Month(i);
    Results.Year=Parameters.Year(i);
    StartIndex=find(LidarData.Time>24,1,'first');
    EndIndex=find(LidarData.Time<=48,1,'last');
    %% Boundary Layers
    BoundaryLayers.CloudHeight=BoundaryLayers.CloudHeight(StartIndex:EndIndex);
    BoundaryLayers.FreeAtmosphere=BoundaryLayers.FreeAtmosphere(StartIndex:EndIndex);
    BoundaryLayers.FreeAtmospheresmooth=BoundaryLayers.FreeAtmospheresmooth(StartIndex:EndIndex);
    BoundaryLayers.CappingInversion=BoundaryLayers.CappingInversion(StartIndex:EndIndex);
    BoundaryLayers.CappingInversionsmooth=BoundaryLayers.CappingInversionsmooth(StartIndex:EndIndex);
    ind=find(BoundaryLayers.ConvectionIndex>StartIndex & BoundaryLayers.ConvectionIndex<EndIndex);
    BoundaryLayers.ConvectionHeight=BoundaryLayers.ConvectionHeight(ind);
    BoundaryLayers.ConvectionIndex=BoundaryLayers.ConvectionIndex(ind)-1440/Parameters.dt*(i-1);
    BoundaryLayers.ResidualLayer=BoundaryLayers.ResidualLayer(StartIndex:EndIndex);
    BoundaryLayers.MixedLayer=BoundaryLayers.MixedLayer(StartIndex:EndIndex);
    BoundaryLayers.MixedLayersmooth=BoundaryLayers.MixedLayersmooth(StartIndex:EndIndex);
    BoundaryLayers.CloudHeightsmooth=BoundaryLayers.CloudHeightsmooth(StartIndex:EndIndex);
    BoundaryLayers.ResidualLayersmooth=BoundaryLayers.ResidualLayersmooth(StartIndex:EndIndex);
    %% LidarData
    LidarData.Time=0:Parameters.dt/60:24-Parameters.dt/60;
    LidarData.CloudMask=LidarData.CloudMask(:,StartIndex:EndIndex);
    LidarData.SNRMask=LidarData.SNRMask(:,StartIndex:EndIndex);
    LidarData.CombinedMask=LidarData.CombinedMask(:,StartIndex:EndIndex);
    LidarData.MolecularBackscatterCoefficient=LidarData.MolecularBackscatterCoefficient(:,StartIndex:
        EndIndex);
    LidarData.UnmaskedAerosolBackscatterCoefficient=LidarData.UnmaskedAerosolBackscatterCoefficient(:,
        StartIndex:EndIndex);
    LidarData.UnmaskedBackscatterRatio=LidarData.UnmaskedBackscatterRatio(:,StartIndex:EndIndex);
    LidarData.MaskedAerosolBackscatterCoefficient=LidarData.MaskedAerosolBackscatterCoefficient(:,
        StartIndex:EndIndex);
    LidarData.MaskedBackscatterRatio=LidarData.MaskedBackscatterRatio(:,StartIndex:EndIndex);
    LidarData.UnmaskedNormalizedAerosolBackscatterCoefficient=LidarData.
        UnmaskedNormalizedAerosolBackscatterCoefficient(:,StartIndex:EndIndex);
    LidarData.MaskedNormalizedAerosolBackscatterCoefficient=LidarData.
        MaskedNormalizedAerosolBackscatterCoefficient(:,StartIndex:EndIndex);
    LidarData.MolecularBackscatterCoefficientsmooth=LidarData.MolecularBackscatterCoefficientsmooth(:,
        StartIndex:EndIndex);
    LidarData.MaskedAerosolBackscatterCoefficientsmooth=LidarData.
        MaskedAerosolBackscatterCoefficientsmooth(:,StartIndex:EndIndex);
    LidarData.MaskedBackscatterRatiosmooth=LidarData.MaskedBackscatterRatiosmooth(:,StartIndex:EndIndex);
    LidarData.MaskedNormalizedAerosolBackscatterCoefficientsmooth=LidarData.
        MaskedNormalizedAerosolBackscatterCoefficientsmooth(:,StartIndex:EndIndex);
    LidarData.MaskedNormalizedBackscatterRatiosmooth=LidarData.MaskedNormalizedBackscatterRatiosmooth(:,
        StartIndex:EndIndex);
    LidarData.UnmaskedAerosolBackscatterCoefficientsmooth=LidarData.
        UnmaskedAerosolBackscatterCoefficientsmooth(:,StartIndex:EndIndex);
    LidarData.UnmaskedBackscatterRatiosmooth=LidarData.UnmaskedBackscatterRatiosmooth(:,StartIndex:
        EndIndex);
    LidarData.UnmaskedNormalizedAerosolBackscatterCoefficientsmooth=LidarData.
        UnmaskedNormalizedAerosolBackscatterCoefficientsmooth(:,StartIndex:EndIndex);
    LidarData.MaskedHWTBa=LidarData.MaskedHWTBa(:,StartIndex:EndIndex);
    LidarData.MaskedHWTBasmooth=LidarData.MaskedHWTBasmooth(:,StartIndex:EndIndex);
    LidarData.UnmaskedHWTBa=LidarData.UnmaskedHWTBa(:,StartIndex:EndIndex);
    LidarData.UnmaskedHWTBasmooth=LidarData.UnmaskedHWTBasmooth(:,StartIndex:EndIndex);
```

```
    LidarData.HWTCombinedMask=LidarData.HWTCombinedMask(:,StartIndex:EndIndex);
    %% Sondes
    for j=1:length(Sondes.RadiosondeData)
        try
        temp=Sondes.RadiosondeData(j).LidarTime;
        Sondes.RadiosondeData(j).LidarTime=temp-24;
        catch
        end
    end
%% Sunrise Sunset Data
    SunriseSunsetData.SunriseTime=SunriseSunsetData.SunriseTime(i);
    SunriseSunsetData.SunsetTime=SunriseSunsetData.SunsetTime(i);
    SunriseSunsetData.Time=SunriseSunsetData.Time(StartIndex:EndIndex);

    SunriseSunsetData.SunriseIndex=SunriseSunsetData.SunriseIndex(i)-1440/Parameters.dt*(i-1);
    SunriseSunsetData.SunsetIndex=SunriseSunsetData.SunsetIndex(i)-1440/Parameters.dt*(i-1);
    %% Weather Data
    WeatherData.GroundTemperature=WeatherData.GroundTemperature(StartIndex:EndIndex);
    WeatherData.GroundPressure=WeatherData.GroundPressure(StartIndex:EndIndex);
    WeatherData.Time=WeatherData.Time(StartIndex:EndIndex)-24*(i-1);
    WeatherData.DateTime=WeatherData.DateTime(StartIndex:EndIndex);
    WeatherData.Temperature=WeatherData.Temperature(:,StartIndex:EndIndex);
    WeatherData.Pressure=WeatherData.Pressure(:,StartIndex:EndIndex);
    %% Parameters
    Parameters.Day=Parameters.Day(i+1);
    Parameters.Month=Parameters.Month(i+1);
    Parameters.Year=Parameters.Year(i+1);
    %% Clear old variables
    fields={'limitlowvector','limithighvector',...
        'CappingInversionOriginal','CappingInversionsmoothOriginal'};
    BoundaryLayers=rmfield(BoundaryLayers,fields);
    fields={'Day_Night'};
    SunriseSunsetData=rmfield(SunriseSunsetData,fields);
    fields={'MinRange','LapseRate'};
    Parameters=rmfield(Parameters,fields);
    %% Store structures in results structure
    Results.BoundaryLayers=BoundaryLayers;
    Results.LidarData=LidarData;
    Results.Sondes=Sondes;
    Results.SunriseSunsetData=SunriseSunsetData;
    Results.WeatherData=WeatherData;
    Results.Parameters=Parameters;
    file_name=strcat('ValueAddedProducts_UTC_',string(Results.Month),string(Results.Day),string(Results.
        Year),'.mat');
    cd(Path.ValueAddedProducts)
    save(file_name,'Results')
    cd(Path.Code)
end
```

Calibration Code

Main Function

```
%Calibration Main Function
%The purpose of this code is to read scan data and retrieve the calibration
%constants of the MSU O2 Dial / HSRL
%% Precode
clear all; close all;
%Defines times that the scans were taken
%This example loads 2 scans
Parameters.OnlineScanStartTime=[5.89 6.48 7.44];
Parameters.OnlineScanEndTime=[6.39 6.97 8.5];
Parameters.OfflineScanStartTime=[4.03 4.4 4.78];
Parameters.OfflineScanEndTime=[4.37 4.74 5.78];
%Define Parameters
Parameters.OnlineWavelength=769.7958e-9; %m
Parameters.OnlineFrequency=299792458/Parameters.OnlineWavelength; %Hz
Parameters.OfflineWavelength=770.1085e-9; %m
Parameters.OfflineFrequency=299792458/Parameters.OfflineWavelength; %Hz
Parameters.ScanDate='20210702';
%Load scan data
[ScanData,Results]=ReadScanFiles(Parameters);
Parameters.Temperature=170:10:320; %K
Parameters.Pressure=10:10:100; %KPa
%Preallocate Calibration Constants
RBSData.SpontaneousSignalTentiS6Offline=zeros(length(Parameters.Temperature),length(Parameters.Pressure)
    ,1000);
RBSData.frequencyshiftOnline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
RBSData.frequencyOnline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
RBSData.wavelengthOnline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
RBSData.SpontaneousSignalTentiS6Online=zeros(length(Parameters.Temperature),length(Parameters.Pressure)
    ,1000);
RBSData.frequencyshiftOffline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
RBSData.frequencyOffline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
RBSData.wavelengthOffline=zeros(length(Parameters.Temperature),length(Parameters.Pressure),1000);
Results.Cmm=zeros(length(Parameters.Temperature),length(Parameters.Pressure));
Results.Cmc=zeros(length(Parameters.Temperature),length(Parameters.Pressure));
Results.Cam=zeros(length(Parameters.Temperature),length(Parameters.Pressure));
Results.Cac=zeros(length(Parameters.Temperature),length(Parameters.Pressure));
%For loop for each pressure and temperature point
for i=1:length(Parameters.Temperature)
    for j=1:length(Parameters.Pressure)
        % create profile from temperature and pressure
        [RBSData]=RayleighBrillouinSpectrum(Parameters,i,j,RBSData);
        % Find Calibration Constants
         [Results,ScanData]=CalibrationConstantsTables(Parameters,ScanData,RBSData,i,j,Results);
    end
end
```

ReadScanFiles

```
function [ScanData,Results]=ReadScanFiles(Parameters)
%This function reads the MCS files
%Create Paths
Path.Code = pwd;
Path.Data =['F:\MSU DIAL\Data\NetCDFOutput\',Parameters.ScanDate];

% Defining files to read
cd(Path.Data);
Path.EtalonFile     = dir('ReceiverScanEtalon*');
Path.LaserFile      = dir('ReceiverScanLaser*');
Path.MCSFile        = dir('ReceiverScanMCS*');
Path.WavemeterFile = dir('ReceiverScanWavemeter*');

% Loading data
Wave.time = []; Wave.wavelength = []; Wave.power = [];
for m=1:1:size(Path.WavemeterFile,1)
    Filename = Path.WavemeterFile(m,1).name;
    Wave.time      = [Wave.time;ncread(Filename,'time')];
    Wave.wavelength  = [Wave.wavelength;ncread(Filename,'Wavelength')];
    Wave.power     = [Wave.power;ncread(Filename,'SeedPower')];
end

MCS.time = []; MCS.chan = []; MCS.data = []; MCS.nsPerBin = [];
for m=1:1:size(Path.MCSFile,1)
    Filename = Path.MCSFile(m,1).name;
    MCS.time  = [MCS.time;ncread(Filename,'time')];
    MCS.chan  = [MCS.chan;ncread(Filename,'Channel')];
    MCS.data  = [MCS.data;ncread(Filename,'Data')];
    MCS.nsPerBin = [MCS.nsPerBin;ncread(Filename,'nsPerBin')];
end
```

```
cd(Path.Code)


% Processing wavemeter and MCS data
Wave.power(Wave.wavelength<=0) = nan;
Wave.wavelength(Wave.wavelength<=0) = nan;
Wave.power(Wave.wavelength>900) = nan;
Wave.wavelength(Wave.wavelength>900) = nan;

MCS.data                = mean(MCS.data,2);
MCS.timeWVOnline        = MCS.time(MCS.chan ==0);
MCS.timeWVOffline       = MCS.time(MCS.chan ==2);
MCS.timeWVOnlineLow       = MCS.time(MCS.chan ==8);
MCS.timeWVOfflineLow      = MCS.time(MCS.chan ==10);
MCS.WVOnline        = MCS.data(MCS.chan ==0);
MCS.WVOffline       = MCS.data(MCS.chan ==2);
MCS.WVOnlineLow         = MCS.data(MCS.chan ==8);
MCS.WVOfflineLow        = MCS.data(MCS.chan ==10);
[MCS]=DetectorCorrection(MCS);

%online scan
%Read Scans
ScanData.OnlineMolecularSignal=[];
ScanData.OnlineMolecularSignalTime=[];
ScanData.OnlineCombinedSignal=[];
ScanData.OnlineCombinedSignalTime=[];
ScanData.OnlineWavelength=[];
ScanData.OnlineWavelengthTime=[];
ScanData.OnlineTime=[];
ScanData.OnlineCombinedSignal=[];
ScanData.OnlineMolecularSignal=[];
ScanData.OnlineWavelength=[];
ScanData.AverageOnlineWavelength=[];
ScanData.AverageOnlineCombinedSignal=[];
ScanData.AverageOnlineMolecularSignal=[];
ScanData.NormalAverageOnlineCombinedSignal=[];
ScanData.NormalAverageOnlineMolecularSignal=[];
ScanData.NormalAverageOnlineCombinedSignal=[];
ScanData.NormalAverageOnlineMolecularSignal=[];

for i=1:length(Parameters.OnlineScanStartTime)

    cutoffindexlow=find(MCS.timeWVOnline>Parameters.OnlineScanStartTime(i),1);
    cutoffindexhigh=find(MCS.timeWVOnline<Parameters.OnlineScanEndTime(i),1,'last');
        OnlineMolecularSignal=MCS.WVOnline(cutoffindexlow:cutoffindexhigh);
        OnlineMolecularSignalTime=MCS.timeWVOnline(cutoffindexlow:cutoffindexhigh);
    cutoffindexlow=find(MCS.timeWVOffline>Parameters.OnlineScanStartTime(i),1);
    cutoffindexhigh=find(MCS.timeWVOffline<Parameters.OnlineScanEndTime(i),1,'last');
        OnlineCombinedSignal=MCS.WVOffline(cutoffindexlow:cutoffindexhigh);
        OnlineCombinedSignalTime=MCS.timeWVOffline(cutoffindexlow:cutoffindexhigh);
    cutoffindexlow=find(Wave.time>Parameters.OnlineScanStartTime(i),1);
    cutoffindexhigh=find(Wave.time<Parameters.OnlineScanEndTime(i),1,'last');
        OnlineWavelength=Wave.wavelength(cutoffindexlow:cutoffindexhigh)*1e-9;
        OnlineWavelengthTime=Wave.time(cutoffindexlow:cutoffindexhigh);

    OnlineTime=OnlineWavelengthTime(2):0.0001:OnlineWavelengthTime(end-1);
    OnlineCombinedSignal=interp1(OnlineCombinedSignalTime,OnlineCombinedSignal,OnlineTime);
    OnlineMolecularSignal=interp1(OnlineMolecularSignalTime,OnlineMolecularSignal,OnlineTime);
    OnlineWavelength=interp1(OnlineWavelengthTime,OnlineWavelength,OnlineTime);

    AverageOnlineWavelength=min(OnlineWavelength):0.0001e-9:max(OnlineWavelength);
    % averagesignal
    AverageOnlineCombinedSignal=zeros(length(AverageOnlineWavelength),1);
    AverageOnlineMolecularSignal=zeros(length(AverageOnlineWavelength),1);
    for j=1:length(AverageOnlineWavelength)
        temp1=OnlineCombinedSignal(find(OnlineWavelength==AverageOnlineWavelength(j)));
        AverageOnlineCombinedSignal(j)=mean(temp1);
        temp2=OnlineMolecularSignal(find(OnlineWavelength==AverageOnlineWavelength(j)));
        AverageOnlineMolecularSignal(j)=mean(temp2);
    end
    %normalize
    AverageOnlineCombinedSignal=smoothdata(AverageOnlineCombinedSignal,'g',5);
    AverageOnlineMolecularSignal=smoothdata(AverageOnlineMolecularSignal,'g',5);

    if i==1
        temp=AverageOnlineWavelength>7.6978e-7;
        AverageOnlineCombinedSignal(temp)=[];
        AverageOnlineMolecularSignal(temp)=[];
        AverageOnlineWavelength(temp)=[];
    elseif i==2
        temp=AverageOnlineWavelength<7.6981e-7;
        AverageOnlineCombinedSignal(temp)=[];
        AverageOnlineMolecularSignal(temp)=[];
        AverageOnlineWavelength(temp)=[];

    end

    ScanData.OnlineMolecularSignal=[ScanData.OnlineMolecularSignal,OnlineMolecularSignal];
```

```
    ScanData.OnlineMolecularSignalTime=[ScanData.OnlineMolecularSignalTime;OnlineMolecularSignalTime];
    ScanData.OnlineCombinedSignal=[ScanData.OnlineCombinedSignal,OnlineCombinedSignal];
    ScanData.OnlineCombinedSignalTime=[ScanData.OnlineCombinedSignalTime;OnlineCombinedSignalTime];
    ScanData.OnlineWavelength=[ScanData.OnlineWavelength,OnlineWavelength];
    ScanData.OnlineWavelengthTime=[ScanData.OnlineWavelengthTime;OnlineWavelengthTime];
    ScanData.OnlineTime=[ScanData.OnlineTime,OnlineTime];
    ScanData.OnlineCombinedSignal=[ScanData.OnlineCombinedSignal,OnlineCombinedSignal];
    ScanData.OnlineMolecularSignal=[ScanData.OnlineMolecularSignal,OnlineMolecularSignal];
    ScanData.OnlineWavelength=[ScanData.OnlineWavelength,OnlineWavelength];
    ScanData.AverageOnlineWavelength=[ScanData.AverageOnlineWavelength,AverageOnlineWavelength];
    ScanData.AverageOnlineCombinedSignal=[ScanData.AverageOnlineCombinedSignal;AverageOnlineCombinedSignal
        ];
    ScanData.AverageOnlineMolecularSignal=[ScanData.AverageOnlineMolecularSignal;
        AverageOnlineMolecularSignal];

end

ScanData.NormalAverageOnlineCombinedSignal=ScanData.AverageOnlineCombinedSignal;
ScanData.NormalAverageOnlineMolecularSignal=ScanData.AverageOnlineMolecularSignal;
ScanData.NormalAverageOnlineCombinedSignal=fillmissing(ScanData.NormalAverageOnlineCombinedSignal,'nearest
    ');
ScanData.NormalAverageOnlineMolecularSignal=fillmissing(ScanData.NormalAverageOnlineMolecularSignal,'
    nearest');
[ScanData.AverageOnlineWavelength,I]=sort(ScanData.AverageOnlineWavelength,'ascend');
ScanData.AverageOnlineCombinedSignal=ScanData.AverageOnlineCombinedSignal(I);
ScanData.AverageOnlineMolecularSignal=ScanData.AverageOnlineMolecularSignal(I);
ScanData.AverageOnlineCombinedSignal=smoothdata(ScanData.AverageOnlineCombinedSignal,'movmean',5);
ScanData.AverageOnlineMolecularSignal=smoothdata(ScanData.AverageOnlineMolecularSignal,'movmean',5);
ScanData.NormalAverageOnlineCombinedSignal=ScanData.NormalAverageOnlineCombinedSignal(I);
ScanData.NormalAverageOnlineMolecularSignal=ScanData.NormalAverageOnlineMolecularSignal(I);
ScanData.NormalAverageOnlineCombinedSignal=smoothdata(ScanData.NormalAverageOnlineCombinedSignal,'movmean'
    ,5);
ScanData.NormalAverageOnlineMolecularSignal=smoothdata(ScanData.NormalAverageOnlineMolecularSignal,'
    movmean',5);
C_M_Efficiency_online=nanmedian(ScanData.NormalAverageOnlineCombinedSignal./ScanData.
    NormalAverageOnlineMolecularSignal);
ScanData.NormalAverageOnlineCombinedSignal=ScanData.NormalAverageOnlineCombinedSignal/max(ScanData.
    NormalAverageOnlineCombinedSignal);
ScanData.NormalAverageOnlineMolecularSignal=ScanData.NormalAverageOnlineMolecularSignal/max(ScanData.
    NormalAverageOnlineMolecularSignal);

%offline scan
%Read Scans
ScanData.OfflineMolecularSignal=[];
ScanData.OfflineMolecularSignalTime=[];
ScanData.OfflineCombinedSignal=[];
ScanData.OfflineCombinedSignalTime=[];
ScanData.OfflineWavelength=[];
ScanData.OfflineWavelengthTime=[];
ScanData.OfflineTime=[];
ScanData.OfflineCombinedSignal=[];
ScanData.OfflineMolecularSignal=[];
ScanData.OfflineWavelength=[];
ScanData.AverageOfflineWavelength=[];
ScanData.AverageOfflineCombinedSignal=[];
ScanData.AverageOfflineMolecularSignal=[];
ScanData.NormalAverageOfflineCombinedSignal=[];
ScanData.NormalAverageOfflineMolecularSignal=[];
ScanData.NormalAverageOfflineCombinedSignal=[];
ScanData.NormalAverageOfflineMolecularSignal=[];

%channel efficiencies
Results.MolecularEfficiency=1/(1+C_M_Efficiency_online);
Results.CombinedEfficiency=1-Results.MolecularEfficiency;

for i=1:length(Parameters.OfflineScanStartTime)
    cutoffindexlow=find(MCS.timeWVOfflineLow>Parameters.OfflineScanStartTime(i),1);
    cutoffindexhigh=find(MCS.timeWVOfflineLow<Parameters.OfflineScanEndTime(i),1,'last');
        OfflineMolecularSignal=MCS.WVOnlineLow(cutoffindexlow:cutoffindexhigh);
        OfflineMolecularSignalTime=MCS.timeWVOnlineLow(cutoffindexlow:cutoffindexhigh);
    cutoffindexlow=find(MCS.timeWVOfflineLow>Parameters.OfflineScanStartTime(i),1);
    cutoffindexhigh=find(MCS.timeWVOfflineLow<Parameters.OfflineScanEndTime(i),1,'last');
        OfflineCombinedSignal=MCS.WVOfflineLow(cutoffindexlow:cutoffindexhigh);
        OfflineCombinedSignalTime=MCS.timeWVOfflineLow(cutoffindexlow:cutoffindexhigh);
    cutoffindexlow=find(Wave.time>Parameters.OfflineScanStartTime(i),1);
    cutoffindexhigh=find(Wave.time<Parameters.OfflineScanEndTime(i),1,'last');
        OfflineWavelength=Wave.wavelength(cutoffindexlow:cutoffindexhigh)*1e-9;
        OfflineWavelengthTime=Wave.time(cutoffindexlow:cutoffindexhigh);

    OfflineTime=OfflineWavelengthTime(2):0.0001:OfflineWavelengthTime(end-1);
    OfflineCombinedSignal=interp1(OfflineCombinedSignalTime,OfflineCombinedSignal,OfflineTime);
    OfflineMolecularSignal=interp1(OfflineMolecularSignalTime,OfflineMolecularSignal,OfflineTime);
    OfflineWavelength=interp1(OfflineWavelengthTime,OfflineWavelength,OfflineTime);

    AverageOfflineWavelength=min(OfflineWavelength):0.0001e-9:max(OfflineWavelength);
    % averagesignal
    AverageOfflineCombinedSignal=zeros(length(AverageOfflineWavelength),1);
    AverageOfflineMolecularSignal=zeros(length(AverageOfflineWavelength),1);
```

```
    for j=1:length(AverageOfflineWavelength)
        temp1=OfflineCombinedSignal(find(OfflineWavelength==AverageOfflineWavelength(j)));
        AverageOfflineCombinedSignal(j)=mean(temp1);
        temp2=OfflineMolecularSignal(find(OfflineWavelength==AverageOfflineWavelength(j)));
        AverageOfflineMolecularSignal(j)=mean(temp2);
    end
    %normalize
    AverageOfflineCombinedSignal=smoothdata(AverageOfflineCombinedSignal,'g',5);
    AverageOfflineMolecularSignal=smoothdata(AverageOfflineMolecularSignal,'g',5);

    if i==1
        temp=AverageOfflineWavelength>7.70096e-7;
        AverageOfflineCombinedSignal(temp)=[];
        AverageOfflineMolecularSignal(temp)=[];
        AverageOfflineWavelength(temp)=[];
    elseif i==2
        temp=AverageOfflineWavelength<7.7012e-7;
        AverageOfflineCombinedSignal(temp)=[];
        AverageOfflineMolecularSignal(temp)=[];
        AverageOfflineWavelength(temp)=[];

    end

    ScanData.OfflineMolecularSignal=[ScanData.OfflineMolecularSignal,OfflineMolecularSignal];
    ScanData.OfflineMolecularSignalTime=[ScanData.OfflineMolecularSignalTime;OfflineMolecularSignalTime];
    ScanData.OfflineCombinedSignal=[ScanData.OfflineCombinedSignal,OfflineCombinedSignal];
    ScanData.OfflineCombinedSignalTime=[ScanData.OfflineCombinedSignalTime;OfflineCombinedSignalTime];
    ScanData.OfflineWavelength=[ScanData.OfflineWavelength,OfflineWavelength];
    ScanData.OfflineWavelengthTime=[ScanData.OfflineWavelengthTime;OfflineWavelengthTime];
    ScanData.OfflineTime=[ScanData.OfflineTime,OfflineTime];
    ScanData.OfflineCombinedSignal=[ScanData.OfflineCombinedSignal,OfflineCombinedSignal];
    ScanData.OfflineMolecularSignal=[ScanData.OfflineMolecularSignal,OfflineMolecularSignal];
    ScanData.OfflineWavelength=[ScanData.OfflineWavelength,OfflineWavelength];
    ScanData.AverageOfflineWavelength=[ScanData.AverageOfflineWavelength,AverageOfflineWavelength];
    ScanData.AverageOfflineCombinedSignal=[ScanData.AverageOfflineCombinedSignal;
        AverageOfflineCombinedSignal];
    ScanData.AverageOfflineMolecularSignal=[ScanData.AverageOfflineMolecularSignal;
        AverageOfflineMolecularSignal];


end

ScanData.NormalAverageOfflineCombinedSignal=ScanData.AverageOfflineCombinedSignal;
ScanData.NormalAverageOfflineMolecularSignal=ScanData.AverageOfflineMolecularSignal;
ScanData.NormalAverageOfflineCombinedSignal=fillmissing(ScanData.NormalAverageOfflineCombinedSignal,'
    nearest');
ScanData.NormalAverageOfflineMolecularSignal=fillmissing(ScanData.NormalAverageOfflineMolecularSignal,'
    nearest');
[ScanData.AverageOfflineWavelength,I]=sort(ScanData.AverageOfflineWavelength,'ascend');
ScanData.AverageOfflineCombinedSignal=ScanData.AverageOfflineCombinedSignal(I);
ScanData.AverageOfflineMolecularSignal=ScanData.AverageOfflineMolecularSignal(I);
ScanData.AverageOfflineCombinedSignal=smoothdata(ScanData.AverageOfflineCombinedSignal,'movmean',5);
ScanData.AverageOfflineMolecularSignal=smoothdata(ScanData.AverageOfflineMolecularSignal,'movmean',5);
ScanData.NormalAverageOfflineCombinedSignal=ScanData.NormalAverageOfflineCombinedSignal(I);
ScanData.NormalAverageOfflineMolecularSignal=ScanData.NormalAverageOfflineMolecularSignal(I);
ScanData.NormalAverageOfflineCombinedSignal=smoothdata(ScanData.NormalAverageOfflineCombinedSignal,'
    movmean',5);
ScanData.NormalAverageOfflineMolecularSignal=smoothdata(ScanData.NormalAverageOfflineMolecularSignal,'
    movmean',5);

ScanData.NormalAverageOfflineMolecularSignal=ScanData.NormalAverageOfflineMolecularSignal/Results.
    MolecularEfficiency;
ScanData.NormalAverageOfflineCombinedSignal=ScanData.NormalAverageOfflineCombinedSignal/Results.
    CombinedEfficiency;
ScanData.NormalAverageOfflineMolecularSignal=ScanData.NormalAverageOfflineMolecularSignal/max(ScanData.
    NormalAverageOfflineCombinedSignal);
ScanData.NormalAverageOfflineCombinedSignal=ScanData.NormalAverageOfflineCombinedSignal/max(ScanData.
    NormalAverageOfflineCombinedSignal);
%convert wavelength to frequency
ScanData.AverageOnlineFrequency=299792458./ScanData.AverageOnlineWavelength;
ScanData.AverageOfflineFrequency=299792458./ScanData.AverageOfflineWavelength;

%channel efficiencies
Results.MolecularEfficiency=1/(1+C_M_Efficiency_online);
Results.CombinedEfficiency=1-Results.MolecularEfficiency;

%transmission
Results.Wavelength=[ScanData.AverageOnlineWavelength,ScanData.AverageOfflineWavelength];
[Results.Wavelength,ia,~]=unique(Results.Wavelength);
Results.Frequency=299792458./Results.Wavelength;
Results.CombinedTransmission=[ScanData.NormalAverageOnlineCombinedSignal;ScanData.
    NormalAverageOfflineCombinedSignal];
Results.MolecularTransmission=[ScanData.NormalAverageOnlineMolecularSignal;ScanData.
    NormalAverageOfflineMolecularSignal];
Results.CombinedTransmission=Results.CombinedTransmission(ia);
Results.MolecularTransmission=Results.MolecularTransmission(ia);
end
```

DetectorCorrection

```
function [MCS]=DetectorCorrection(MCS)
%APD correction factor
deadtime=22e-9;
WVOnlineCountRate=MCS.WVOnline/(7000*250e-9);
WVOnlineCorrectionFactor=1./(1-deadtime*WVOnlineCountRate);
MCS.WVOnline=WVOnlineCorrectionFactor.*MCS.WVOnline;

WVOnlineLowCountRate=MCS.WVOnlineLow/(7000*250e-9);
WVOnlineLowCorrectionFactor=1./(1-deadtime*WVOnlineLowCountRate);
MCS.WVOnlineLow=WVOnlineLowCorrectionFactor.*MCS.WVOnlineLow;

WVOfflineCountRate=MCS.WVOffline/(7000*250e-9);
WVOfflineCorrectionFactor=1./(1-deadtime*WVOfflineCountRate);
MCS.WVOffline=WVOfflineCorrectionFactor.*MCS.WVOffline;

WVOfflineLowCountRate=MCS.WVOfflineLow/(7000*250e-9);
WVOfflineLowCorrectionFactor=1./(1-deadtime*WVOfflineLowCountRate);
MCS.WVOfflineLow=WVOfflineLowCorrectionFactor.*MCS.WVOfflineLow;
end
```

RayleighBrillouinSpectrum

```
%% Rayleigh Brillouin Spectrum
function [RBSData]=RayleighBrillouinSpectrum(Parameters,i,j,RBSData)
T=Parameters.Temperature(i);
P=Parameters.Pressure(j);
lambda=Parameters.OnlineWavelength;
m=4.789e-26; %molecular mass of air in g/mol
kb=1.38064852e-23; %Boltzmann's Constant

% create the domain of xi
% 'xi' is dimensionless and should be scaled by k*v0/(2*pi) [Hz],
% where 'v0' is the most probable gas velocity, or sqrt(2*kb*T/m),
% and 'k' is 4*pi/lambda*sin(scatter_angle/2)
N=1000;
xi_lef=-5.0d0;
xi_rgt=5.0d0;
xi=linspace(xi_lef,xi_rgt,N);

scatter_angle=pi;
k=4*pi/lambda*sin(scatter_angle/2);
v0=sqrt(2*kb*T/m);
%scale x
nushift=xi*k*v0/(2*pi);

%dynamic viscosity of air
%Sutherland Equation
eta0=1.716e-5; %kg/(m*s) reference shear viscosity
k0=0.0241; %W/(K*m) reference thermal conductivity
T0=273; %K reference temperature
Seta=111; %K
Sk=194; %K
shearviscosity=eta0*(T/T0)^(3/2)*((T0+Seta)/(T+Seta));
thermalconductivity=k0*(T/T0)^(3/2)*((T0+Sk)/(T+Sk));
% bulk_vis=0.86e-5+1.29e-7*(T-250);
bulk_vis=shearviscosity*0.71; %This is a poor approximation
c_int=1.0; %internal specific heat capacity

% convert pressures and densities
p_pa=P*1000;
n0=p_pa/(T*kb);

%% Don't Change This Section of Code
% compute and set RBS model input parameters
c_tr=3/2;
y=n0*kb*T/(k*v0*shearviscosity);
gamma_int=c_int/(c_tr+c_int);
rlx_int=1.5*bulk_vis/(shearviscosity*gamma_int);
eukenf=m*thermalconductivity/(shearviscosity*kb*(c_tr+c_int));
% run the code

[cohsig6,sptsig6]=crbs6(y,rlx_int,eukenf,c_int,c_tr,xi);

RBSData.frequencyshiftOnline(i,j,:)=nushift;
RBSData.frequencyOnline(i,j,:)=Parameters.OnlineFrequency+nushift;
RBSData.SpontaneousSignalTentiS6Online(i,j,:)=sptsig6;
RBSData.wavelengthOnline(i,j,:)=299792458./RBSData.frequencyOnline(i,j,:);

lambda=Parameters.OfflineWavelength;
m=4.81e-26; %molecular mass of air in kg
kb=1.38064852e-23; %Boltzmann's Constant
```

```
% create the domain of xi
% 'xi' is dimensionless and should be scaled by k*v0/(2*pi) [Hz],
% where 'v0' is the most probable gas velocity, or sqrt(2*kb*T/m),
% and 'k' is 4*pi/lambda*sin(scatter_angle/2)
N=1000;
xi_lef=-5.0d0;
xi_rgt=5.0d0;
xi=linspace(xi_lef,xi_rgt,N);

scatter_angle=pi;
k=4*pi/lambda*sin(scatter_angle/2);
v0=sqrt(2*kb*T/m);
%scale x
nushift=xi*k*v0/(2*pi);

%dynamic viscosity of air
%Sutherland Equation
eta0=1.716e-5; %kg/(m*s) reference shear viscosity
k0=0.0241; %W/(K*m) reference thermal conductivity
T0=273; %K reference temperature
Seta=111; %K
Sk=194; %K
shearviscosity=eta0*(T/T0)^(3/2)*((T0+Seta)/(T+Seta));
thermalconductivity=k0*(T/T0)^(3/2)*((T0+Sk)/(T+Sk));
bulk_vis=shearviscosity*0.71; %This is a poor approximation
c_int=1.0; %internal specific heat capacity

% convert pressures and densities
p_pa=P*1000;
n0=p_pa/(T*kb);

%% Don't Change This Section of Code
% compute and set RBS model input parameters
c_tr=3/2;
y=n0*kb*T/(k*v0*shearviscosity);
gamma_int=c_int/(c_tr+c_int);
rlx_int=1.5*bulk_vis/(shearviscosity*gamma_int);
eukenf=m*thermalconductivity/(shearviscosity*kb*(c_tr+c_int));
% run the code

[cohsig6,sptsig6]=crbs6(y,rlx_int,eukenf,c_int,c_tr,xi);

RBSData.frequencyshiftOffline(i,j,:)=nushift;
RBSData.frequencyOffline(i,j,:)=Parameters.OfflineFrequency+nushift;
RBSData.SpontaneousSignalTentiS6Offline(i,j,:)=sptsig6;
RBSData.wavelengthOffline(i,j,:)=299792458./RBSData.frequencyOffline(i,j,:);

end
```

crbs6

```
function [cohsig,sptsig,c] = crbs6(y,rlx_int,eukenf,c_int,c_tr,xi)
% Written by John Smith
% October 21st, 2010
% University of Colorado at Boulder, CIRES
% John.A.Smith@Colorado.EDU
% MATLAB version 7.10.0.59 (R2010a) 64-bit
% Adapted from "Coherent Rayleigh-Brillouin Scattering"
% by Xingguo Pan

% Computes the coherent and spontaneous
% RBS spectrum given the parameters
% in crbs_molecular using the s6 model
% by G. Tenti, 1974

% Called by: crbs_molecular.m

n_xi=numel(xi);

n=6;
a=zeros(n,n);
b=zeros(n,2);

cohsig=zeros(1,n_xi);
sptsig=zeros(1,n_xi);

cpxunit=sqrt(-1);

%y7=1.5*y;
gamma_int=c_int/(c_tr+c_int);
j020=-y;
%j030=1.5*j020;
j100=-gamma_int*y/rlx_int;
j001=j100*c_tr/c_int;
```

```
j100001=j100*sqrt(c_tr/c_int);
j110=j100*5/6+j020*2/3;
j011110=j100*sqrt(5/(8*c_int));
j_nu=0.4*(1.5+c_int)+(3+c_int)/(2*rlx_int)+9*eukenf/(16*rlx_int^2);
j_de=-1+(4/15)*eukenf*(1.5+c_int)+(c_int/3)*eukenf/rlx_int;
j_co=-y*(2*gamma_int/3);
j011=j_co*j_nu/j_de;

%coharea=0;
%sptarea=0;

for i=1:n_xi
    z=xi(i)+y*cpxunit;
        w0=w0_func(z);
        w1=-sqrt(pi)+z*w0;
        w2=z*w1;
        w3=-0.5*sqrt(pi)+z*w2;
        w4=z*w3;
        w5=-3*sqrt(pi)/4+z*w4;
        w6=z*w5;

    i0000=w0/(sqrt(pi));
        i0100=(z*w0-sqrt(pi))*sqrt(2/pi);
        i0001=i0100;
        i0010=(2*w2-w0)/(sqrt(6*pi));
        i1000=i0010;
        i0011=(2*w3-3*w1)/(sqrt(5*pi));
        i1100=i0011;
        i0101=2*w2/sqrt(pi);
        i0110=(-w1+2*w3)/sqrt(3*pi);
        i1001=i0110;
        i0111=(-3*w2+2*w4)*sqrt(2/(5*pi));
        i1101=i0111;
        i1111=(13*w2-12*w4+4*w6)/(5*sqrt(pi));
        %i0002=(-w0+2*w2)/sqrt(3*pi);
        %i0200=i0002;
        %i0211=(-w1+8*w3-4*w5)/sqrt(15*pi);
        %i1102=i0211;
        %i0202=2*(w0-2*w2+2*w4)/(3*sqrt(pi));
        %i0210=(w0+4*w2-4*w4)/(3*sqrt(2*pi));
        %i1002=i0210;
        %i0102=(-w1+2*w3)*sqrt(2/(3*pi));
        %i0201=i0102;
        i1010=(5*w0-4*w2+4*w4)/(6*sqrt(pi));
        i1110=(7*w1-8*w3+4*w5)/sqrt(30*pi);
        i1011=i1110;

    %a_factor=1;


    a(:,1)=-j020*[i0000 i0001 i0011 i0010 0 0]+[cpxunit 0 0 0 0 0];
    a(:,2)=-j020*[i0100 i0101 i0111 i0110 0 0]+[0 cpxunit 0 0 0 0];
    a(:,3)=(j020-j110)*[i1100 i1101 i1111 i1110 0 0]+j011110*[0 0 0 0 -i0100 -i0101]+[0 0 -cpxunit 0 0 0];
    a(:,4)=(j020-j100)*[i1000 i1001 i1011 i1010 0 0]+j100001*[0 0 0 0 -i0000 -i0001]+[0 0 0 -cpxunit 0 0];
    a(:,5)=j100001*[i1000 i1001 i1011 i1010 0 0]+(j001-j020)*[0 0 0 0 i0000 i0001]+[0 0 0 0 cpxunit 0];
    a(:,6)=j011110*[i1100 i1101 i1111 i1110 0 0]+(j011-j020)*[0 0 0 0 i0100 i0101]+[0 0 0 0 0 cpxunit];

    b(:,1)=-[i0100 i0101 i0111 i0110 0 0];
    b(:,2)=-[i0000 i0001 i0011 i0010 0 0];

    c=linsolve(a,b);

    cohsig(i)=c(1,1)*conj(c(1,1));
    sptsig(i)=2*real(c(1,2));
%    coharea=coharea+cohsig(i);
%    sptarea=sptarea+sptsig(i);
end
end
```

w0_func

```
function w = w0_func(z,N)
% Based on:
% FADDEEVA    Faddeeva function
%   W = FADDEEVA(Z) is the Faddeeva function, aka the plasma dispersion
%   function, for each element of Z. The Faddeeva function is defined as:
%
%     w(z) = exp(-z^2) * erfc(-j*z)
%
%   where erfc(x) is the complex complementary error function.
%
%   W = FADDEEVA(Z,N) can be used to explicitly specify the number of terms
%   to truncate the expansion (see (13) in [1]). N = 16 is used as default.
%
```

```
%   Example:
%       x = linspace(-10,10,1001); [X,Y] = meshgrid(x,x);
%       W = faddeeva(complex(X,Y));
%       figure;
%       subplot(121); imagesc(x,x,real(W)); axis xy square; caxis([-1 1]);
%       title('re(faddeeva(z))'); xlabel('re(z)'); ylabel('im(z)');
%       subplot(122); imagesc(x,x,imag(W)); axis xy square; caxis([-1 1]);
%       title('im(faddeeva(z))'); xlabel('re(z)'); ylabel('im(z)');
%
%   Reference:
%   [1] J.A.C. Weideman, "Computation of the Complex Error Function," SIAM
%       J. Numerical Analysis, pp. 1497-1518, No. 5, Vol. 31, Oct., 1994
%       Available Online: http://www.jstor.org/stable/2158232

% Called by: crbs6.m, crbs7.m


if nargin<2, N = []; end
if isempty(N), N = 50; end

w = zeros(size(z)); % initialize output

%%%%
% for purely imaginary-valued inputs, use erf as is if z is real
idx = real(z)==0; %
w(idx) = exp(-z(idx).^2).*erfc(imag(z(idx)));

if all(idx), return; end
idx = ~idx;

%%%%
% for complex-valued inputs

% make sure all points are in the upper half-plane (positive imag. values)
idx1 = idx & imag(z)<0;
z(idx1) = conj(z(idx1));

M = 2*N;
M2 = 2*M;
k = (-M+1:1:M-1)'; % M2 = no. of sampling points.
L = sqrt(N/sqrt(2)); % Optimal choice of L.

theta = k*pi/M;
t = L*tan(theta/2); % Variables theta and t.
f = exp(-t.^2).*(L^2+t.^2);
f = [0; f]; % Function to be transformed.
a = real(fft(fftshift(f)))/M2; % Coefficients of transform.
a = flipud(a(2:N+1)); % Reorder coefficients.

Z = (L+1i*z(idx))./(L-1i*z(idx));
p = polyval(a,Z); % Polynomial evaluation.
w(idx) = 2*p./(L-1i*z(idx)).^2 + (1/sqrt(pi))./(L-1i*z(idx)); % Evaluate w(z).

% convert the upper half-plane results to the lower half-plane if necesary
w(idx1) = conj(2*exp(-z(idx1).^2) - w(idx1));

w=-w*sqrt(-1)*pi; % converts to 'w0_func' used in crbs_molecular.m
```

CalibrationConstantsTables

```
function [Results,ScanData]=CalibrationConstantsTables(Parameters,ScanData,RBSData,i,j,Results)

%% set up new frequency grid for offline scan

RBSsignal=squeeze(RBSData.SpontaneousSignalTentiS6Offline(i,j,:));
RBSwavelength=squeeze(RBSData.wavelengthOffline(i,j,:));

%% Offline Laser Combined Channel(molecular and aerosol return in combined channel)
[ScanData.AverageOfflineWavelength,index]=unique(ScanData.AverageOfflineWavelength);
ScanData.NormalAverageOfflineCombinedSignal=ScanData.NormalAverageOfflineCombinedSignal(index);
combinedtransmittance=interp1(ScanData.AverageOfflineWavelength,ScanData.
    NormalAverageOfflineCombinedSignal,RBSwavelength);
combinedtransmittance=fillmissing(combinedtransmittance,'nearest');

int1=trapz(RBSwavelength,RBSsignal);
int2=trapz(RBSwavelength,RBSsignal.*combinedtransmittance);

Cmc=int2/int1;
[~,ind]=min(abs(RBSwavelength-Parameters.OfflineWavelength));
Cac=combinedtransmittance(ind);

%% Offline Laser Molecular Channel (Molecular and aerosol return in molecular channel)
ScanData.NormalAverageOfflineMolecularSignal=ScanData.NormalAverageOfflineMolecularSignal(index);
moleculartransmittance=interp1(ScanData.AverageOfflineWavelength,ScanData.
    NormalAverageOfflineMolecularSignal,RBSwavelength);
```

```
moleculartransmittance=fillmissing(moleculartransmittance,'nearest');

int1=trapz(RBSwavelength,RBSsignal);
int2=trapz(RBSwavelength,RBSsignal.*moleculartransmittance);

Cmm=int2/int1;
[~,ind]=min(abs(RBSwavelength-Parameters.OfflineWavelength));
Cam=moleculartransmittance(ind);

%% Normalize to Cac
Cmm=Cmm/Cac;
Cmc=Cmc/Cac;
Cam=Cam/Cac;
Cac=1;

%% Store Results
Results.Cmm(i,j)=Cmm;
Results.Cmc(i,j)=Cmc;
Results.Cam(i,j)=Cam;
Results.Cac(i,j)=Cac;

Results.Temperature(i)=Parameters.Temperature(i);
Results.Pressure(j)=Parameters.Pressure(j);


end
```

Radiosonde Code

Main Function

```
% This function processes the radiosonde files and finds the potential
% temperature as well as the boundary layer height
% This function also processes the standard HSRL data for the selected days

%% Radisonde Precode
clc; clear all;
close all;
%Define Parameters
%Date
    Parameters.Day='02'; %make sure to use 2 digits (02=2)
    Parameters.Month='06'; %make sure to use 2 digits (02=2)
    Parameters.Year='2021';
%Define file paths
    Path.Code=pwd;
    Path.Radiosondes='F:\MSU DIAL\Sondes';

%% Radiosonde Functions
%load radiosonde data
    [RadiosondeData]=LoadSonde(Parameters,Path);
% Check for bad sonde
     [Parameters]=SondeCheck(RadiosondeData,Parameters);
%remove errant data points
    [RadiosondeData]=RemoveSondeData(RadiosondeData);
%find height above surface
    [RadiosondeData]=FindHeight(RadiosondeData);
%find potential temperature
    [RadiosondeData]=FindPotentialTemperature(RadiosondeData);
%find absolute humidity
    [RadiosondeData]=FindAbsoluteHumidity(RadiosondeData);
%smooth data
    [RadiosondeData]=SmoothSondeData(RadiosondeData);
%find gradients in the humidity and potential temperature
    [RadiosondeData]=FindSondeGradients(RadiosondeData);

for i=1:length(RadiosondeData)
    %create plots for reference
        CreateBoundaryLayerPlots(RadiosondeData,i);
    %find pblh by Heffter Method
        [RadiosondeData]=HefftерMethodRetrieval(RadiosondeData,i);
    %input pblh_lidar
        RadiosondeData(i).pblh_Lidar=input('Input Lidar boundary layer height. (NaN if mixed layer DNE) \n
            ');
end
        %save results
         WriteSondeFile(RadiosondeData,Parameters,Path);
```

LoadSonde

```
function [RadiosondeData]=LoadSonde(Parameters,Path)
%This function loads the radiosonde tsv files generated by the radiosonde
%station
cd(Path.Radiosondes)
directory=dir(['COB_',Parameters.Year,'-',Parameters.Month,'-',Parameters.Day,'*']);
cd(Path.Code)
for i=1:length(directory)
    name=string(directory(i).name);
    [data]=tsvread(name,Path);
    data.name=name;
    if i==1
        RadiosondeData=data;
    else
        RadiosondeData=[RadiosondeData;data];
    end
end
end
```

tsvread

```
function [SondeData]=...
    tsvread(varargin,Path)
%[data, header, raw] = tsvread( file ) reads in text file with tab-seperated variables. default value for
    data is nan.
%alternative input/output option is suppluying header strings
%[col1, col2, col3, ..., header, raw] = tsvread( file, header1, header2, header3, ... )
%header is the first row (assumed to have header names) and raw is the imported text
%if a vector is supplied, this specifies the number rows to be imported.
%examples:
%[col1, col2,col3,header,raw] = tsvread( 'example.tsv', 'header1', 'header2', 'header3', 1:5 )
%will import data from example.tsv, and cols corresponding to header1,
%header2, header3, cols 1 t0 5.
%if no outputs are requested, then a portion of the rquested table is
%displayed -- good idea to see how the import and header requests are
%working!
%If there is a tsv file in local directory, then just running "tsvread" at
%the command line will read the newest tsv file and display first ten rows to screen.
%sak 9/2/11
cd(Path.Radiosondes)
if nargin == 0
    disp( 'importing most recent tsv file in local directory' );
    d = dir( '*.tsv' );
    [~,i] = sort( [d.datenum], 'descend' );
    fprintf( 'tsvread( ''%s'', 1:10 )\n', d(i(1)).name );
    eval( sprintf( 'tsvread( ''%s'', 1:10 )', d(i(1)).name ) );
    return;
end
fid = fopen( varargin{1}, 'r' );
if nargout == 0
    fprintf( 'loading %s, and displaying sideways\n', varargin{1} );
end
varargin(1) = [];
stuff = textscan( fid, '%s', 'delimiter', '\n');
stuff = stuff{1};
fclose(fid);
numrows = 1:size(stuff,1);
ind = cellfun( 'isclass', varargin, 'double' );
if any( ind )
    numrows = varargin{ind};
    varargin(ind) = [];
    if numel(numrows) == 1
        numrows = 1:min(numrows, size( stuff, 1) );
    end
end
numrows = intersect( numrows, 1:size( stuff, 1 ) );
header = regexprep( regexp( stuff{1}, '[^\t]*\t', 'match' ), '\t', '' );
raw = repmat( {}, numel(numrows), numel(header) );
for i=numrows
    stuff{i}(end+1) = 9;
    tmp = regexprep( regexp( stuff{i}, '[^\t]*\t', 'match' ), '\t', '');
    raw(i,1:numel(tmp)) = tmp;
end

raw=raw(40:end,:);
SondeData.time=str2double(raw(:,1));
SondeData.temperature=str2double(raw(:,3));
SondeData.relativehumidity=str2double(raw(:,4));
SondeData.altitude=str2double(raw(:,7));
SondeData.pressure=str2double(raw(:,8));

cd(Path.Code)
end
```

SondeCheck

```
function  [Parameters]=SondeCheck(RadiosondeData,Parameters)
%This function checks for sondes with poor data
for i=1:length(RadiosondeData)
        RadiosondeData(i).BadSonde=0;
        % check for enough data points
        datapoints=numel(RadiosondeData(i).time);
        if datapoints<1
            RadiosondeData(i).BadSonde=1;
        end
        %check that maximum height is greater than 1000m
        if max(RadiosondeData(i).altitude)<2524
            RadiosondeData(i).BadSonde=1;
        end
        %check that maximum pressure is greater than 200 mbar
        if max(RadiosondeData(i).pressure)<200
            RadiosondeData(i).BadSonde=1;
        end
```

```
        %check that temperature does not change by more than 30 degrees i the first
        %10 seconds
        ind=find(RadiosondeData(i).time>30);
        maxtemp=max(RadiosondeData(i).temperature(1:ind-1));
        mintemp=min(RadiosondeData(i).temperature(1:ind-1));
        if (maxtemp-mintemp)>30
            RadiosondeData(i).BadSonde=1;
        end
        %check that maximum temperature is less than 50C and minimum is greater
        %than -90C
        if max(RadiosondeData(i).temperature)>323 || min(RadiosondeData(i).temperature)<183
            RadiosondeData(i).BadSonde=1;
        end
        %check that the pressure in the first two levels is present
        if sum(isempty(RadiosondeData(i).pressure(1:2)))>0
            RadiosondeData(i).BadSonde=1;
        end
    end
end
```

RemoveSondeData

```
function [RadiosondeData]=RemoveSondeData(RadiosondeData)
%This function looks for unrealistic data points and removes them
for i=1:length(RadiosondeData)

    %trim data on descent of sonde
    [~,cutoff]=max(RadiosondeData(i).altitude);
    RadiosondeData(i).time=RadiosondeData(i).time(1:cutoff);
    RadiosondeData(i).temperature=RadiosondeData(i).temperature(1:cutoff);
    RadiosondeData(i).relativehumidity=RadiosondeData(i).relativehumidity(1:cutoff);
    RadiosondeData(i).altitude=RadiosondeData(i).altitude(1:cutoff);
    RadiosondeData(i).pressure=RadiosondeData(i).pressure(1:cutoff);
    %trim unrealistic temperatures
    temp=RadiosondeData(i).temperature;
    temp(temp<0)=NaN;
    temp=fillmissing(temp,'nearest');
    RadiosondeData(i).temperature=temp;
    %trim unrealistic humidity
    temp=RadiosondeData(i).relativehumidity;
    temp(temp<0)=NaN;
    temp=fillmissing(temp,'nearest');
    RadiosondeData(i).relativehumidity=temp;
    %trim unrealistic pressure
    temp=RadiosondeData(i).pressure;
    temp(temp<0)=NaN;
    temp=fillmissing(temp,'nearest');
    RadiosondeData(i).pressure=temp/10;
    %limit maximum height of sonde
    ind=find(RadiosondeData(i).pressure<20);
    if ~isempty(ind)
        RadiosondeData(i).time=RadiosondeData(i).time(1:ind);
        RadiosondeData(i).temperature=RadiosondeData(i).temperature(1:ind);
        RadiosondeData(i).relativehumidity=RadiosondeData(i).relativehumidity(1:ind);
        RadiosondeData(i).altitude=RadiosondeData(i).altitude(1:ind);
        RadiosondeData(i).pressure=RadiosondeData(i).pressure(1:ind);
    end
end

end
```

FindHeight

```
function [RadiosondeData]=FindHeight(RadiosondeData)
%This funcion converts altiude to height
for i=1:length(RadiosondeData)
    RadiosondeData(i).height=RadiosondeData(i).altitude-...
        RadiosondeData(i).altitude(1);
end
end
```

FindPotentialTemperature

```
function [RadiosondeData]=FindPotentialTemperature(RadiosondeData)
%this function converts temperature and pressure to potential temperature
pnaught=100;
for i=1:length(RadiosondeData)
    RadiosondeData(i).potentialtemperature=RadiosondeData(i).temperature.*...
        (pnaught./RadiosondeData(i).pressure).^(0.286);
end
end
```

FindAbsoluteHumidity

```
function [RadiosondeData]=FindAbsoluteHumidity(RadiosondeData)
%find absolute humidity
for i=1:length(RadiosondeData)

    RadiosondeData(i).absolutehumidity=...
        (6.112*exp(...
        (17.67*RadiosondeData(i).temperature)./...
        (RadiosondeData(i).temperature+243.5))...
        .*RadiosondeData(i).relativehumidity*18.02...
        )./(273.15+RadiosondeData(i).temperature);

    %convert to grams/cm^3
    RadiosondeData(i).absolutehumidity=RadiosondeData(i).absolutehumidity;

end
end
```

SmoothSondeData

```
function [RadiosondeData]=SmoothSondeData(RadiosondeData)
%This function smooths the radiosonde data to 15 mbar
for i=1:length(RadiosondeData)
    %apply gaussian smoothing filter to radiosonde temperature and relative
    %humidity
    pressure=(RadiosondeData(i).pressure*10)';
    [sortedpressure,I]=sort(pressure);
    [uniquesortedpressure,II]=unique(sortedpressure);
    newpressure=pressure(1):-1:pressure(end);
    RadiosondeData(i).subsampledpressure=newpressure;

    sortedheight=(RadiosondeData(i).height(I))';
    uniquesortedheight=sortedheight(II);
    newheight=interp1(uniquesortedpressure,uniquesortedheight,newpressure);
    RadiosondeData(i).subsampledheight=newheight;

    sortedtemperature=(RadiosondeData(i).temperature(I))';
    uniquesortedtemperature=sortedtemperature(II);
    newtemperature=interp1(uniquesortedpressure,uniquesortedtemperature,newpressure);
    RadiosondeData(i).subsampledtemperature=newtemperature;

    sortedpotentialtemperature=(RadiosondeData(i).potentialtemperature(I))';
    uniquesortedpotentialtemperature=sortedpotentialtemperature(II);
    newpotentialtemperature=interp1(uniquesortedpressure,uniquesortedpotentialtemperature,newpressure);
    RadiosondeData(i).subsampledpotentialtemperature=newpotentialtemperature;

    sortedrelativehumidity=(RadiosondeData(i).relativehumidity(I))';
    uniquesortedrelativehumidity=sortedrelativehumidity(II);
    newrelativehumidity=interp1(uniquesortedpressure,uniquesortedrelativehumidity,newpressure);
    RadiosondeData(i).subsampledrelativehumidity=newrelativehumidity;

    sortedabsolutehumidity=(RadiosondeData(i).absolutehumidity(I))';
    uniquesortedabsolutehumidity=sortedabsolutehumidity(II);
    newabsolutehumidity=interp1(uniquesortedpressure,uniquesortedabsolutehumidity,newpressure);
    RadiosondeData(i).subsampledabsolutehumidity=newabsolutehumidity;


    sortedtime=(RadiosondeData(i).time(I))';
    uniquesortedtime=sortedtime(II);
    newtime=interp1(uniquesortedpressure,uniquesortedtime,newpressure);
    RadiosondeData(i).subsampledtime=newtime;

    %smooth to 15 mbar
        RadiosondeData(i).heightsmooth=newheight;
        RadiosondeData(i).temperaturesmooth=smoothdata(newtemperature,'movmean',31);
        RadiosondeData(i).relativehumiditysmooth=smoothdata(newrelativehumidity,'movmean',31);
        RadiosondeData(i).potentialtemperaturesmooth=smoothdata(newpotentialtemperature,'movmean',31);
        RadiosondeData(i).absolutehumiditysmooth=smoothdata(newabsolutehumidity,'movmean',31);
end
end
```

FindSondeGradients

```
function [RadiosondeData]=FindSondeGradients(RadiosondeData)
%This function finds the temperature lapse rate and humidity gradient
for i=1:length(RadiosondeData)

    %create gradient height
    RadiosondeData(i).gradientheight=...
        (RadiosondeData(i).heightsmooth(2:end)-RadiosondeData(i).heightsmooth(1:end-1))/(2)...
        +RadiosondeData(i).heightsmooth(1:end-1);
    diffheight=RadiosondeData(i).heightsmooth(2:end)-RadiosondeData(i).heightsmooth(1:end-1);
    %find gradient for potential temperature
    diff=RadiosondeData(i).potentialtemperaturesmooth(2:end)-RadiosondeData(i).potentialtemperaturesmooth
        (1:end-1);
    RadiosondeData(i).potentialtemperaturesmoothgradient=diff./diffheight;
    %find potential temperature gradient from subsampled data
    diff=RadiosondeData(i).subsampledpotentialtemperature(2:end)-RadiosondeData(i).
        subsampledpotentialtemperature(1:end-1);
    RadiosondeData(i).subsampledpotentialtemperaturegradient=diff./diffheight;
    %find gradient for absolute humidity
    diff=RadiosondeData(i).absolutehumiditysmooth(2:end)-RadiosondeData(i).absolutehumiditysmooth(1:end-1)
        ;
    RadiosondeData(i).absolutehumiditysmoothgradient=diff./diffheight;
    %find gradient for relative humidity
    diff=RadiosondeData(i).relativehumiditysmooth(2:end)-RadiosondeData(i).relativehumiditysmooth(1:end-1)
        ;
    RadiosondeData(i).relativehumiditysmoothgradient=diff./diffheight;

    %remove NaN and inf values
    RadiosondeData(i).potentialtemperaturesmoothgradient(...
        isinf(RadiosondeData(i).potentialtemperaturesmoothgradient))=NaN;
    RadiosondeData(i).potentialtemperaturesmoothgradient=...
        fillmissing(RadiosondeData(i).potentialtemperaturesmoothgradient,'nearest');
    RadiosondeData(i).subsampledpotentialtemperaturegradient(...
        isinf(RadiosondeData(i).subsampledpotentialtemperaturegradient))=NaN;
    RadiosondeData(i).subsampledpotentialtemperaturegradient=...
        fillmissing(RadiosondeData(i).subsampledpotentialtemperaturegradient,'nearest');
    RadiosondeData(i).absolutehumiditysmoothgradient(...
        isinf(RadiosondeData(i).absolutehumiditysmoothgradient))=NaN;
    RadiosondeData(i).absolutehumiditysmoothgradient=...
        fillmissing(RadiosondeData(i).absolutehumiditysmoothgradient,'nearest');
    RadiosondeData(i).relativehumiditysmoothgradient=...
        fillmissing(RadiosondeData(i).relativehumiditysmoothgradient,'nearest');

end


end
```

CreateBoundaryLayerPlots

```
function []=CreateBoundaryLayerPlots(RadiosondeData,i)
%This function creates plots of the potential temperature and lidar
%profiles
    ind=find(RadiosondeData(i).heightsmooth>4500,1,'first');

    if isempty(ind)
        ind=numel(RadiosondeData(i).heightsmooth)-1;
    end

    close all

    figure(1);
    clf;
    set(gcf,'Units', 'Normalized', 'OuterPosition', [0 0 0.5 1]);
    movegui('west');
    pos = get(gca, 'Position');
    yscale=1;
    xscale=1;
    set(gca,'Position',[pos(1), pos(2), (pos(3)-pos(1))*xscale+pos(1), (pos(4)-pos(2))*yscale+pos(1)])
    subplot(2,1,1); hold on
    plot(RadiosondeData(i).potentialtemperaturesmooth(1:ind),RadiosondeData(i).heightsmooth(1:ind));


    subplot(2,1,2); hold on
    plot(RadiosondeData(i).potentialtemperaturesmoothgradient(1:ind),RadiosondeData(i).gradientheight(1:
        ind));
    xline(0.005)
    xline(0.004)

    figure(2);
    clf;
    set(gcf,'Units', 'Normalized', 'OuterPosition', [0 0 0.5 1]);
    movegui('east');
```

```
    pos = get(gca, 'Position');
    yscale=1;
    xscale=1;
    set(gca,'Position',[pos(1), pos(2), (pos(3)-pos(1))*xscale+pos(1), (pos(4)-pos(2))*yscale+pos(1)])
    subplot(2,1,1); hold on
    plot(RadiosondeData(i).relativehumiditysmooth(1:ind),RadiosondeData(i).heightsmooth(1:ind));

    subplot(2,1,2); hold on
    plot(RadiosondeData(i).relativehumiditysmoothgradient(1:ind),RadiosondeData(i).gradientheight(1:ind));
end
```

HeffterMethodRetrieval

```
function [RadiosondeData]=HeffterMethodRetrieval(RadiosondeData,i)
%This function finds the potential temperature pblh using the Heffter
%method
    %find potential temperature lapse rate greater than 5K/km
        ind=find(RadiosondeData(i).potentialtemperaturesmoothgradient>0.005);
        if ~isempty(ind) && numel(ind)>1
            diff=ind(2:end)-ind(1:end-1);
            diff=[2,diff];
            diff2=ind(3:end)-ind(2:end-1);
            diff2=[1,diff2,1];
            log=logical([diff~=1 | diff2~=1]);
            ind2=ind(log);
            % pick out top and bottom of inversion layers
            if RadiosondeData(i).potentialtemperaturesmoothgradient(1)>0.005
                ind2(1)=[];
            end
            inversionbottomheight=RadiosondeData(i).gradientheight(ind2(1:2:end));
            inversiontopheight=RadiosondeData(i).gradientheight(ind2(2:2:end));
            inversiontopheight(inversiontopheight>4000)=[];

            if length(inversionbottomheight)>length(inversiontopheight)
                inversionbottomheight(length(inversiontopheight)+1:end)=[];
            end

            inversionbottompotentialtemperature=RadiosondeData(i).potentialtemperaturesmooth(ind2(1:2:end)
                );
            if length(inversionbottompotentialtemperature)>length(inversiontopheight)
                inversionbottompotentialtemperature(length(inversiontopheight)+1:end)=[];
            end
            inversiontoppotentialtemperature=RadiosondeData(i).potentialtemperaturesmooth(ind2(2:2:end));
            if length(inversiontoppotentialtemperature)>length(inversiontopheight)
                inversiontoppotentialtemperature(length(inversiontopheight)+1:end)=[];
            end

            temperaturedifference=inversiontoppotentialtemperature-...
                inversionbottompotentialtemperature(1:length(inversiontopheight));
            if ~isnan(inversiontopheight)
                if sum(temperaturedifference>=2)>=1
                    pblh_Heffter=inversiontopheight(find(temperaturedifference>=2,1,'first'));
                else
                    temp=find(RadiosondeData(i).gradientheight>inversiontopheight(end),1);
                    ind(ind>temp)=[];
                    [~,I]=max(RadiosondeData(i).potentialtemperaturesmoothgradient(ind));
                    pblh_Heffter=RadiosondeData(i).gradientheight(ind(I));
                end
            else
                pblh_Heffter=NaN;
            end
        else
            pblh_Heffter=NaN;
        end
        disp(['The pblh height by the Heftter Method is ',num2str(pblh_Heffter)])

        RadiosondeData(i).pblh_Heffter=input('Input Heffter method boundary layer height. (NaN if mixed
            layer DNE) \n');

end
```

WriteSondeFile

```
function []=WriteSondeFile(RadiosondeData,Parameters,Path)
%This function stores the pblh values as an m file
filename=string(['Sondes_',Parameters.Year,Parameters.Month,Parameters.Day,'.mat']);

cd(Path.Radiosondes)
save(filename,'RadiosondeData')
cd(Path.Code)
end
```

APPENDIX D

PERMISSIONS FOR COPYRIGHTED MATERIAL

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Oct 28, 2021

This Agreement between Montana State University -- Luke Colberg ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5170350631790
License date	Oct 15, 2021
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Mean Boundary Layer Characteristics
Licensed Content Author	Roland B. Stull
Licensed Content Date	Jan 1, 1988
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic

Portion	figures/tables/illustrations
Number of figures/tables/illustrations	1
Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	no
Title	Diode-Laser-Based High Spectral Resolution Lidar
Institution name	Montana State University
Expected presentation date	Oct 2021
Portions	Figure 1.7
Requestor Location	Montana State University 405 North 16th Ave APT A BOZEMAN, MT 59715 United States Attn: Montana State University
Total	0.00 USD

Terms and Conditions

Springer Nature Customer Service Centre GmbH
Terms and Conditions

This agreement sets out the terms and conditions of the licence (the **Licence**) between you and **Springer Nature Customer Service Centre GmbH** (the **Licensor**). By clicking 'accept' and completing the transaction for the material (**Licensed Material**), you also confirm your acceptance of these terms and conditions.

1. Grant of License

1. 1. The Licensor grants you a personal, non-exclusive, non-transferable, world-wide licence to reproduce the Licensed Material for the purpose specified in your order only. Licences are granted for the specific use requested in the order and for no other use, subject to the conditions below.

1. 2. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. However, you should ensure that the material you are requesting is original to the Licensor and does not carry the copyright of another entity (as credited in the published version).

1. 3. If the credit line on any part of the material you have requested indicates that it was reprinted or adapted with permission from another source, then you should also seek permission from that source to reuse the material.

2. Scope of Licence

2. 1. You may only use the Licensed Content in the manner and to the extent permitted by these Ts&Cs and any applicable laws.

2. 2. A separate licence may be required for any additional use of the Licensed Material, e.g. where a licence has been purchased for print only use, separate permission must be obtained for electronic re-use. Similarly, a licence is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the licence. Any content owned by third parties are expressly excluded from the licence.

2. 3. Similarly, rights for additional components such as custom editions and derivatives require additional permission and may be subject to an additional fee. Please apply to Journalpermissions@springernature.com/bookpermissions@springernature.com for these rights.

2. 4. Where permission has been granted **free of charge** for material in print, permission may also be granted for any electronic version of that work, provided that the material is incidental to your work as a whole and that the electronic version is essentially equivalent to, or substitutes for, the print version.

2. 5. An alternative scope of licence may apply to signatories of the STM Permissions Guidelines, as amended from time to time.

3. Duration of Licence

3. 1. A licence for is valid from the date of purchase ('Licence Date') at the end of the relevant period in the below table:

Scope of Licence	Duration of Licence
Post on a website	12 months
Presentations	12 months
Books and journals	Lifetime of the edition in the language purchased

4. Acknowledgement

4. 1. The Licensor's permission must be acknowledged next to the Licenced Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract, and must be hyperlinked to the journal/book's homepage. Our required acknowledgement format is in the Appendix below.

5. Restrictions on use

5. 1. Use of the Licensed Material may be permitted for incidental promotional use and minor editing privileges e.g. minor adaptations of single figures, changes of format, colour and/or style where the adaptation is credited as set out in Appendix 1 below. Any other changes including but not limited to, cropping, adapting, omitting material that affect the meaning, intention or moral rights of the author are strictly prohibited.

5. 2. You must not use any Licensed Material as part of any design or trademark.

5. 3. Licensed Material may be used in Open Access Publications (OAP) before publication by Springer Nature, but any Licensed Material must be removed from OAP sites prior to final publication.

6. Ownership of Rights

6. 1. Licensed Material remains the property of either Licensor or the relevant third party and any rights not explicitly granted herein are expressly reserved.

7. Warranty

IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND
WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

8. Limitations

8. 1. ***BOOKS ONLY:***Where **'reuse in a dissertation/thesis'** has been selected the following terms apply: Print rights of the final author's accepted manuscript (for clarity, NOT the published version) for up to 100 copies, electronic rights for use only on a personal website or institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/).

8. 2. For content reuse requests that qualify for permission under the STM Permissions Guidelines, which may be updated from time to time, the STM Permissions Guidelines supersede the terms and conditions contained in this licence.

9. Termination and Cancellation

9. 1. Licences will expire after the period shown in Clause 3 (above).

9. 2. Licensee reserves the right to terminate the Licence in the event that payment is not received in full or if there has been a breach of this agreement by you.

Appendix 1 — Acknowledgements:

For Journal Content:
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g.

Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

For **Advance Online Publication papers:**
Reprinted by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM].)

For Adaptations/Translations:
Adapted/Translated by permission from [**the Licensor**]: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication)

Note: For any republication from the British Journal of Cancer, the following credit line style applies:

Reprinted/adapted/translated by permission from [**the Licensor**]: on behalf of Cancer Research UK: : [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT]** (year of publication)

For **Advance Online Publication** papers:
Reprinted by permission from The [**the Licensor**]: on behalf of Cancer Research UK: [**Journal Publisher** (e.g. Nature/Springer/Palgrave)] [**JOURNAL NAME**] [**REFERENCE CITATION** (Article name, Author(s) Name), [**COPYRIGHT**] (year of publication), advance online publication, day month year (doi: 10.1038/sj.[JOURNAL ACRONYM])

For Book content:
Reprinted/adapted by permission from [**the Licensor**]: [**Book Publisher** (e.g. Palgrave Macmillan, Springer etc) [**Book Title**] by [**Book author**(s)] [**COPYRIGHT**] (year of publication)

Other Conditions:

Version 1.3

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.

JOHN WILEY AND SONS LICENSE
TERMS AND CONDITIONS

Oct 28, 2021

This Agreement between Montana State University -- Luke Colberg ("You") and John Wiley and Sons ("John Wiley and Sons") consists of your license details and the terms and conditions provided by John Wiley and Sons and Copyright Clearance Center.

The publisher has provided special terms related to this request that can be found at the end of the Publisher's Terms and Conditions.

License Number	5171981103709
License date	Oct 18, 2021
Licensed Content Publisher	John Wiley and Sons
Licensed Content Publication	Wiley Books
Licensed Content Title	Aerosol Technology: Properties, Behavior, and Measurement of Airborne Particles, 2nd Edition
Licensed Content Author	William C. Hinds
Licensed Content Date	Jan 1, 1999

Licensed Content Pages	1
Type of use	Dissertation/Thesis
Requestor type	University/Academic
Format	Print and electronic
Portion	Figure/table
Number of figures/tables	5
Will you be translating?	No
Title	Diode-Laser-Based High Spectral Resolution Lidar
Institution name	Montana State University
Expected presentation date	Oct 2021
Portions	Figures 1.1,1.2,1.3,1.4,1.5
Requestor Location	Montana State University 405 North 16th Ave APT A BOZEMAN, MT 59715 United States Attn: Montana State University

Publisher Tax ID	EU826007151
Billing Type	Invoice
Billing Address	Montana State University 405 North 16th Ave APT A BOZEMAN, MT 59715 United States Attn: Montana State University
Total	0.00 USD

Terms and Conditions

TERMS AND CONDITIONS

This copyrighted material is owned by or exclusively licensed to John Wiley & Sons, Inc. or one of its group companies (each a"Wiley Company") or handled on behalf of a society with which a Wiley Company has exclusive publishing rights in relation to a particular work (collectively "WILEY"). By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the billing and payment terms and conditions established by the Copyright Clearance Center Inc., ("CCC's Billing and Payment terms and conditions"), at the time that you opened your RightsLink account (these are available at any time at http://myaccount.copyright.com).

Terms and Conditions

- The materials you have requested permission to reproduce or reuse (the "Wiley Materials") are protected by copyright.
- You are hereby granted a personal, non-exclusive, non-sub licensable (on a stand-alone basis), non-transferable, worldwide, limited license to reproduce the Wiley Materials for the purpose specified in the licensing process. This license, **and any CONTENT (PDF or image file) purchased as part of your order,** is for a one-time use only and limited to any maximum distribution number specified in the license. The first instance of republication or reuse granted by this license must be completed within two years of the date of the grant of this license (although copies prepared

before the end date may be distributed thereafter). The Wiley Materials shall not be used in any other manner or for any other purpose, beyond what is granted in the license. Permission is granted subject to an appropriate acknowledgement given to the author, title of the material/book/journal and the publisher. You shall also duplicate the copyright notice that appears in the Wiley publication in your use of the Wiley Material. Permission is also granted on the understanding that nowhere in the text is a previously published source acknowledged for all or part of this Wiley Material. Any third party content is expressly excluded from this permission.

- With respect to the Wiley Materials, all rights are reserved. Except as expressly granted by the terms of the license, no part of the Wiley Materials may be copied, modified, adapted (except for minor reformatting required by the new Publication), translated, reproduced, transferred or distributed, in any form or by any means, and no derivative works may be made based on the Wiley Materials without the prior permission of the respective copyright owner.**For STM Signatory Publishers clearing permission under the terms of the STM Permissions Guidelines only, the terms of the license are extended to include subsequent editions and for editions in other languages, provided such editions are for the work as a whole in situ and does not involve the separate exploitation of the permitted figures or extracts,** You may not alter, remove or suppress in any manner any copyright, trademark or other notices displayed by the Wiley Materials. You may not license, rent, sell, loan, lease, pledge, offer as security, transfer or assign the Wiley Materials on a stand-alone basis, or any of the rights granted to you hereunder to any other person.

- The Wiley Materials and all of the intellectual property rights therein shall at all times remain the exclusive property of John Wiley & Sons Inc, the Wiley Companies, or their respective licensors, and your interest therein is only that of having possession of and the right to reproduce the Wiley Materials pursuant to Section 2 herein during the continuance of this Agreement. You agree that you own no right, title or interest in or to the Wiley Materials or any of the intellectual property rights therein. You shall have no rights hereunder other than the license as provided for above in Section 2. No right, license or interest to any trademark, trade name, service mark or other branding ("Marks") of WILEY or its licensors is granted hereunder, and you agree that you shall not assert any such right, license or interest with respect thereto

- NEITHER WILEY NOR ITS LICENSORS MAKES ANY WARRANTY OR REPRESENTATION OF ANY KIND TO YOU OR ANY THIRD PARTY, EXPRESS, IMPLIED OR STATUTORY, WITH RESPECT TO THE MATERIALS OR THE ACCURACY OF ANY INFORMATION CONTAINED IN THE MATERIALS, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTY OF MERCHANTABILITY, ACCURACY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, USABILITY, INTEGRATION OR NON-INFRINGEMENT AND ALL SUCH WARRANTIES ARE HEREBY EXCLUDED BY WILEY AND ITS LICENSORS AND WAIVED BY YOU.

- WILEY shall have the right to terminate this Agreement immediately upon breach of this Agreement by you.

- You shall indemnify, defend and hold harmless WILEY, its Licensors and their respective directors, officers, agents and employees, from and against any actual or threatened claims, demands, causes of action or proceedings arising from any breach of this Agreement by you.

- IN NO EVENT SHALL WILEY OR ITS LICENSORS BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR ENTITY FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, EXEMPLARY OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, PROVISIONING, VIEWING OR USE OF THE MATERIALS REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

- Should any provision of this Agreement be held by a court of competent jurisdiction to be illegal, invalid, or unenforceable, that provision shall be deemed amended to achieve as nearly as possible the same economic effect as the original provision, and the legality, validity and enforceability of the remaining provisions of this Agreement shall not be affected or impaired thereby.

- The failure of either party to enforce any term or condition of this Agreement shall not constitute a waiver of either party's right to enforce each and every term and condition of this Agreement. No breach under this agreement shall be deemed waived or excused by either party unless such waiver or consent is in writing signed by the party granting such waiver or consent. The waiver by or consent of a party to a breach of any provision of this Agreement shall not operate or be construed as a waiver of or consent to any other or subsequent breach by such other party.

- This Agreement may not be assigned (including by operation of law or otherwise) by you without WILEY's prior written consent.

- Any fee required for this permission shall be non-refundable after thirty (30) days from receipt by the CCC.

- These terms and conditions together with CCC's Billing and Payment terms and conditions (which are incorporated herein) form the entire agreement between you and WILEY concerning this licensing transaction and (in the absence of fraud) supersedes

all prior agreements and representations of the parties, oral or written. This Agreement may not be amended except in writing signed by both parties. This Agreement shall be binding upon and inure to the benefit of the parties' successors, legal representatives, and authorized assigns.

- In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall prevail.

- WILEY expressly reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.

- This Agreement will be void if the Type of Use, Format, Circulation, or Requestor Type was misrepresented during the licensing process.

- This Agreement shall be governed by and construed in accordance with the laws of the State of New York, USA, without regards to such state's conflict of law rules. Any legal action, suit or proceeding arising out of or relating to these Terms and Conditions or the breach thereof shall be instituted in a court of competent jurisdiction in New York County in the State of New York in the United States of America and each party hereby consents and submits to the personal jurisdiction of such court, waives any objection to venue in such court and consents to service of process by registered or certified mail, return receipt requested, at the last known address of such party.

WILEY OPEN ACCESS TERMS AND CONDITIONS

Wiley Publishes Open Access Articles in fully Open Access Journals and in Subscription journals offering Online Open. Although most of the fully Open Access journals publish open access articles under the terms of the Creative Commons Attribution (CC BY) License only, the subscription journals and a few of the Open Access Journals offer a choice of Creative Commons Licenses. The license type is clearly identified on the article.

The Creative Commons Attribution License

The Creative Commons Attribution License (CC-BY) allows users to copy, distribute and transmit an article, adapt the article and make commercial use of the article. The CC-BY license permits commercial and non-

Creative Commons Attribution Non-Commercial License

The Creative Commons Attribution Non-Commercial (CC-BY-NC)License permits use, distribution and reproduction in any medium, provided the original work is properly cited and is not used for commercial purposes.(see below)

Creative Commons Attribution-Non-Commercial-NoDerivs License

The Creative Commons Attribution Non-Commercial-NoDerivs License (CC-BY-NC-ND) permits use, distribution and reproduction in any medium, provided the original work is properly cited, is not used for commercial purposes and no modifications or adaptations are made. (see below)

Use by commercial "for-profit" organizations

Use of Wiley Open Access articles for commercial, promotional, or marketing purposes requires further explicit permission from Wiley and will be subject to a fee.

Further details can be found on Wiley Online Library
http://olabout.wiley.com/WileyCDA/Section/id-410895.html

Other Terms and Conditions:
null

v1.10 Last updated September 2015

Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.