SPECTRAL PROCESSING AND WIND ESTIMATION WITH JICAMARCA MESOSPHERIC RADAR DATA

BY

JENNIFER ANN SMITH

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Erhan Kudeki

ABSTRACT

Since the first radar measurement of the mesosphere above the Jicamarca Radio Observatory in the 1970s, advancement in computing has allowed for increasingly complex processing on increasingly large sets of data. These advances have allowed for more accurate processing techniques to be applied to more data than was possible in the past. Presented in this thesis is an improved method of spectral processing using least-squares nonlinear curve fitting techniques. Using a constrained generalized Gaussian model, the spectral parameters are found for five years of data from Jicamarca's mesosphere-stratosphere-troposphere (MST) radar campaigns. The Doppler velocity from the spectral parameters is then used to estimate the zonal, meridional, and vertical wind velocities. The winds and spectral parameters will be uploaded to the CEDAR Archival Madrigal Database. Winds and spectral data are also displayed at http://remote2.csl.illinois.edu/MSTISR/showmaps_2 utilizing dynamic javascript tools.

This thesis also discusses the detection and fitting of two peaked spectra, known as double Gaussians. An algorithm is described to detect when they occur, based on recognizing when there is a separation of spectral data points above a threshold. Knowing the location of the double peaked spectra allows for fitting them using a double Gaussian model, as well as facilitating the analysis of their causes.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Erhan Kudeki for all of his guidance, Dr. Marco Milla for his guidance during my time at Jicamarca Radio Observatory, and Pablo Reyes for his technical support and advice. Also I would like to thank my parents, Steve and Ann, my aunt, Sue, and David for their support during my time at the University of Illinois at Urbana-Champaign.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 THE MESOSPHERE 4
2.1 Turbulence in the Mesosphere 5
2.2 Radar Detection of the Mesosphere 6
2.3 Mesospheric Neutral Dynamics 9

CHAPTER 3 THE MST-ISR EXPERIMENT 10
3.1 The Jicamarca Antenna 10
3.2 Transmission and Reception 15
3.3 Experiment Dates and Notes 18

CHAPTER 4 ESTIMATION OF SPECTRAL PARAMETERS 20
4.1 Fitting 20
4.1.1 The Spectrum Model 21
4.1.2 Linear and Logarithmic Misfits 23
4.1.3 Evaluation of Best Model 25
4.1.4 Constraint on p 27
4.2 Estimation of Measurement Error 27
4.3 Fitting and Spectral Parameter Results 29
4.4 Double Gaussians 31
4.4.1 Location and Cause 38

CHAPTER 5 WIND ESTIMATION 45
5.1 Determining Jicamarca Unit Vectors 46
5.2 Antenna Beam Directions 48
5.3 Estimation of UVW 49
5.4 Propagation of Measurement Error 51
5.5 Results 52

CHAPTER 6 CONCLUSION 60

APPENDIX A MADRIGAL DATA STRUCTURE 62
A.1 Creating a Madrigal File 62
A.1.1 Creating a New Parameter Code 67
A.2 Creating an Experiment in Madrigal 67
A.2.1 Adding a New Experiment 67
A.3 Adding and Replacing Files an in Experiment 68

APPENDIX B FITTING AND DOUBLE GAUSSIAN DETECTION CODE 69

APPENDIX C WIND ESTIMATION CODE 82

APPENDIX D FILE LOCATION 85

REFERENCES 86

CHAPTER 1

INTRODUCTION

In the over forty years since the first radar observation of the mesosphere above the Jicamarca Radio Observatory (JRO), technology, especially computing, has improved immensely. Not only has the processing power of computers increased dramatically, but the ability to easily store and retrieve large amounts of data has also drastically improved. This thesis shows more complex processing techniques applied to larger amounts of data than was possible to do previously due to processing and data storage limitations, in particular to processing of spectral parameters and wind estimation of mesospheric radar data taken at JRO.

The first paper on radar observation of the mesosphere by *Woodman and Guillen* [1974] stated "This computer is an old and very slow machine according to modern standards, limiting observations to only one height at a time, even though we have used some efficient processing techniques ... These same techniques will allow us to process in real time all the observable heights simultaneously when a third-generation computer already available in the Observatory is connected to the system." The efficient processing techniques referred to are using the autocorrelation function to determine the spectral moments. In *Woodman* [1985], even more complex methods were introduced, primarily spectral parameter estimation using nonlinear curve fitting techniques. *Woodman* noted that a least squares parameter estimation, such as the one used in this thesis, would probably be the best approach. This least squares fitting was first applied to MST radar data in *Yamamoto*

et al. [1988], where it was shown that using a Gaussian as a model for the spectral shape, does indeed provide better results than the earlier spectral moment estimation techniques, particularly in cases of low SNR. This technique remained largely unchanged until *Sheth et al.* [2006], where a generalized Gaussian was applied to take into account the deviation of the spectra from the standard Gaussian shape. Here, this work is continued and improved upon by constraining the value of the Gaussian exponent, p, and fitting some of the spectra with two Gaussian peaks. Additionally, the new fitting techniques are applied to a large set of data, spanning over 50 days across several years and a new way to estimate the winds is developed.

This thesis has six chapters:

Chapter 2 introduces the region of study, the equatorial mesosphere, and the accompanying D-region of the ionosphere. It continues to describe how turbulence in the mesosphere impacts radar backscatter and derives a model for the backscattered electric field of a radar experiment. Finally, some important concepts related to the neutral dynamics of the region are introduced.

Chapter 3 introduces features of the JRO radar used in mesospheric experiments and important experiment parameters, such as the antenna pattern and pulse patterns.

Chapter 4 describes the spectral processing procedure and results of the Jicamarca mesospheric radar observations. The generalized Gaussian model of the mesospheric backscatter spectra is developed and implemented, and the estimation results of Doppler velocity, spectral width, and the general Gaussian parameter, p, are presented. This chapter also includes the determination and spectral processing of the double Gaussians. It ends with a discussion of the causes of double Gaussians.

In Chapter 5, the process for calculating the mesospheric winds from the Doppler velocity obtained in Chapter 4 is developed, and the results are shown.

Finally in Chapter 6, conclusions are drawn and remaining work is discussed.

Appendix A describes the format and structure of the reduced data deposited in the

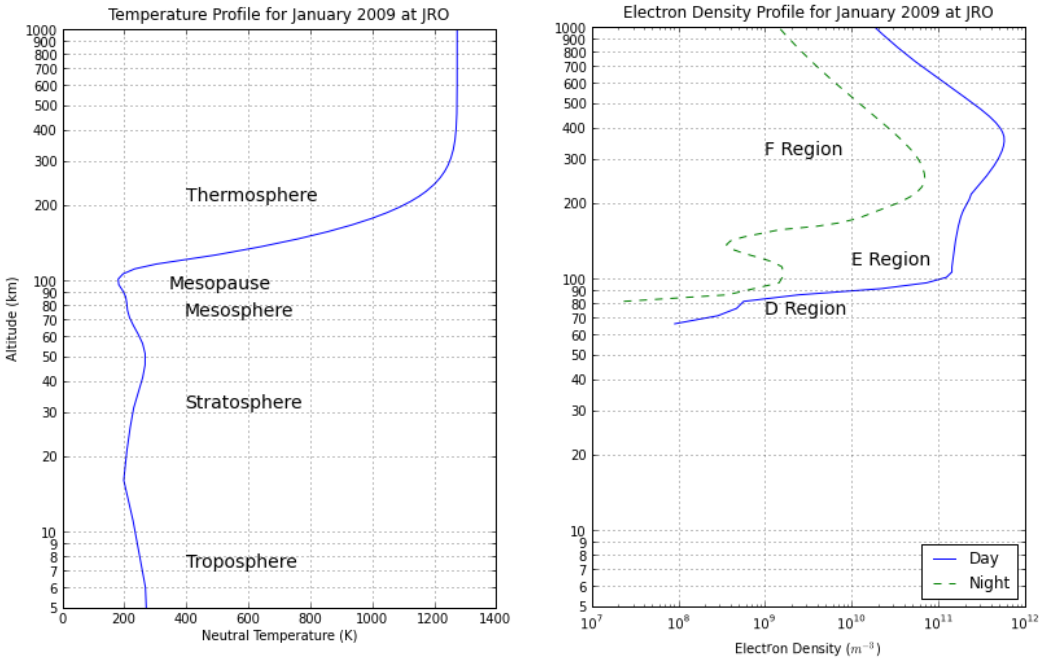
Madrigal data base. Appendices B and C contain the python code for the double Gaussian detection algorithm, spectra fitting, and wind estimation. Appendix D shows the files location on the server.

CHAPTER 2

THE MESOSPHERE

Earth's atmosphere is stratified in to fours layers based on its temperature, as shown in Figure 2.1. In the lowest layer, the troposphere, the temperature decreases with altitude. Next is the stratosphere which stretches from the troposphere to an altitude of roughly 50 km, and is characterized by an increase in temperature. The temperature decreases with altitude again in the mesosphere, which ends at the coldest region of the atmosphere, the mesopause. In the highest layer, the thermosphere, the temperature increases again until about 200 km where it flattens out. The mesosphere, located roughly 50 to 90 km above the surface of earth is central to this thesis.

The atmosphere is also layered according to the density of ionized particles in what is called the ionosphere, shown in Figure 2.1. During the the day, the mesosphere is ionized by solar radiation to form the lowest region of the ionosphere, the D-region. Here, neutral particles are more abundant than in any other part of the ionosphere. Meanwhile electrons have a concentration of $10^8 - 10^9 \mathrm{m}^{-3}$ which is one hundred to two hundred times lower than the typical max of the ionosphere, the F region peak. The neutral particles in the D-region can be ionized in several ways [*Hargreaves*, 1992]. The Lyman-α spectral hydrogen line ionizes NO, which is most abundant in the D-region. EUV radiation creates O_2^+ ions in the upper level of the mesosphere. In the lower levels the neutral water vapor can undergo chemical reactions to create water cluster ions such as H_3O^+. Negative ions are formed when free electrons collide with neutral particles. At night, due to the lack of solar



(a) (b)

Figure 2.1 – Figure (a) shows the layers of the atmosphere according to temperature. Figure (b) shows the layers of the ionosphere. These plots were created using the mass spectrometer and incoherent scatter data (MSIS-E 90) model [*Hedin*, 1991] for the temperatures, and the international reference ionosphere (IRI) model [*Bilitza et al.*, 2011] for the electron densities.

radiation, the ionized particles recombine and the D-region disappears. The ionization that forms during the day is too weak to affect the many neutral particles, and consequently does not affect the neutral dynamics of the region. Free electrons and neutralizing ions are advected by neutral dynamics and behave as "passive scalars." The ionization key component of the mesosphere that allows for VHF radar detection of the region.

2.1 Turbulence in the Mesosphere

It is has been known for some time that VHF radar signals from the mesosphere are greatly dependent on the presence of mesospheric turbulence [*Röttger et al.*, 1979], and the causes of this turbulence has been studied in detail [*Gage and Green*, 1978; *Lehmacher and*

Kudeki, 2003; *Lehmacher et al.*, 2007; *Kudeki*, 1988]. Much of the turbulence is thought to be caused by dynamic instabilities created by large winds shears [*Kelley*, 2009], though convective instabilities also play a role [*Fritts and Rastogi*, 1985].

Dynamic instabilities such as the Kelvin-Helmhotz instability (KHI) occur when the Richardson number, defined as

$$R_i = \frac{\omega_B^2}{(\frac{dU}{dz})^2 + (\frac{dV}{dz})^2}, \tag{2.1}$$

falls below 0.25, where ω_B is the Brunt-Vaisala frequency, the oscillation frequency of an air parcel in a stratified atmosphere, and the denominator is the vertical shear of horizontal wind with U and V components in zonal (eastward) and meridional (northward) directions, respectively. Mesospheric KHI [*Lehmacher et al.*, 2007] produces scattering structures which show up as braids and cat's eyes in VHF radar maps [*Fukao et al.*, 1980].

Most importantly, the instabilities in the mesosphere lead to turbulent mixing of the electron density gradients. The turbulence of the neutrals is transferred by collisions to the ions, and the electrons closely follow the ions due to electrostatic forces and are shuffled around to create electron density gradients [*Chandra et al.*, 2012]. This creates enhanced electron density and refractive index fluctuations that cause VHF radar scattering as described in Section 2.2.

2.2 Radar Detection of the Mesosphere

The first measurement of the mesosphere by a radar was made at JRO outside Lima, Peru in 1970 [*Woodman and Guillen*, 1974]. The region is difficult to measure as it is too high for methods such as high altitude balloons, too low for satellites, and too weakly ionized for small low power VHF radars. Rocket borne in-situ probing and meteor radars can be used for spot measurements. Additionally VHF radars that have large power/aperture products (have large antenna aperture and high transmitter power) can be used. Such

radars are known as mesosphere-stratosphere-troposphere (MST) radars.

Enhanced electron density fluctuations in the mesosphere caused by turbulent mixing allow MST radars to detect backscatter and the dynamics of the region. Following *Kudeki* [2013], the free electrons in the mesosphere will oscillate when illuminated by an AC electric field from the radar. This oscillation will radiate another AC electric field at the same frequency in a process known as Thompson scattering. In Thompson scattering an electron radiates like a Hertzian dipole, producing the scattered electric field as

$$E_S = -\frac{r_e}{r} E_i e^{-jk_o r}, \tag{2.2}$$

where the the incident electric field, E_i, is

$$E_i = E_o e^{-jk_o r} \tag{2.3}$$

so that

$$E_s = -\frac{r_e}{r} E_o e^{-j2k_o r}. \tag{2.4}$$

In these equations k_o is the wave number of the transmitted wave, r is the distance from the source (in this case the radar antenna), and *classical electron radius*,

$$r_e = \frac{e^2}{4\pi\epsilon_0 mc^2} \approx 2.8 \times 10^{-15} \text{ m}. \tag{2.5}$$

In the mesosphere there will be many scattering electrons in a volume of a size ΔV. If ΔV is taken to be much smaller than the radar range, over 60 kilometers, a plane wave approximation can be invoked to model the total scattered field as a superposition

$$E_s = -\sum_{p=1}^{N\Delta V} \frac{r_e}{r_p} E_{op} e^{-j2k_p r_p} \approx -\frac{r_e}{r} E_o \sum_{p=1}^{N\Delta V} e^{j\mathbf{k}\bullet\mathbf{r}_p(t)}, \tag{2.6}$$

where N is the average electron density, $\mathbf{r}_p(t)$ is the trajectory of electrons within ΔV, and

$$\mathbf{k} \equiv -2k_o \frac{\mathbf{r}}{r} \equiv -2k_o \hat{r} = -\frac{2\pi}{\lambda/2}\hat{r}, \tag{2.7}$$

is known as the Bragg vector. Also, scattered field (2.6) from volume ΔV is known as the Bragg scattered field.

The density distribution of electrons

$$n_e(\mathbf{r}, t) = \sum_{p=1}^{N\Delta V} \delta(\mathbf{r} - \mathbf{r}_p(t)) \tag{2.8}$$

within the volume ΔV given in terms of individual electrons trajectories can be spatial Fourier transformed as

$$n_e(\mathbf{k}, t) = \int_{-\infty}^{\infty} n_e(\mathbf{r}, t) e^{j\mathbf{k}\bullet\mathbf{r}} d\mathbf{r} = \int_{-\infty}^{\infty} \sum_{p=1}^{N\Delta V} \delta(r - r_p(t)) e^{j\mathbf{k}\bullet\mathbf{r}} d\mathbf{r} = \sum_{p=1}^{N\Delta V} e^{j\mathbf{k}\bullet\mathbf{r}_p} \tag{2.9}$$

so that the Bragg scattered field from volume ΔV can be expressed as

$$E_s \approx -\frac{r_e}{r} E_o n_e(\mathbf{k}, t). \tag{2.10}$$

This result can be used in a superposition method to model the radar scatter from bigger scattering volumes than ΔV determined by antenna beam sizes. Note that the scattered field, E_s, is a scaled version of the "electron density Fourier amplitude" at the "Bragg scale."

For plasmas in thermal equilibrium, statistical models of the spectra and variance of $n_e(\mathbf{k}, t)$ and hence E_s can be developed as described the incoherent scatter theory (e.g. *Kudeki and Milla* [2011]). However, in mesospheric scattering, the plasma is turbulent and not in thermal equilibrium, and as a consequence there are no first principle spectral models for E_s.

2.3 Mesospheric Neutral Dynamics

As mentioned before, the ionization of mesosphere does not affect the neutral dynamics where the electrons and ions move together with the neutrals. When the Doppler velocity is measured via radar what is effectively measured is the velocity of the neutrals and consequently the *wind velocity*. Several geophysical phenomena can be observed in the mean zonal (east to west) wind in the mesosphere. The quasi-biennial oscillation (QBO) refers to irregular oscillation in the mean zonal wind between easterly and westerly. The period varies, but has a mean of roughly 26 months. The semiannual oscillation (SAO) is similar to the QBO, but with a six month period. The diurnal and semi-diurnal tides occur on day and half-day cycles, respectively. Gravity waves that propagate up through the troposphere and stratosphere can be observed in the Doppler velocity and in the wind almost continuously. Further discussion of the causes of these oscillations and waves are beyond the scope of this thesis; for more information see [*Buriti et al.*, 2008; *Venkateswara Rao et al.*, 2012; *Li et al.*, 2012; *Hitchman et al.*, 1997; *Lieberman et al.*, 1993].

CHAPTER 3

THE MST-ISR EXPERIMENT

The Jicamarca Radio Observatory (JRO) is located at 76.87 W, 11.95 S, just east of Lima, Peru. It was built in 1960 by the Central Radio Propagation Laboratory (CRPL) of the Nation Bureau of Standards (NBS), later part of the National Oceanic and Atmospheric Administration (NOAA), to study the equatorial ionosphere. Today, it is jointly operated by Cornell University and Instituto Geofísico del Perú (IGP). The location of JRO is central to its operation; it is located near the magnetic equator, where earth's magnetic field lines are parallel to the ground. This is very important for the studies of magnetized plasmas in the E and F regions of the ionosphere, which are probed in the "ISR part" of the MST-ISR experiment and by many other experiments carried out at Jicamarca.

3.1 The Jicamarca Antenna

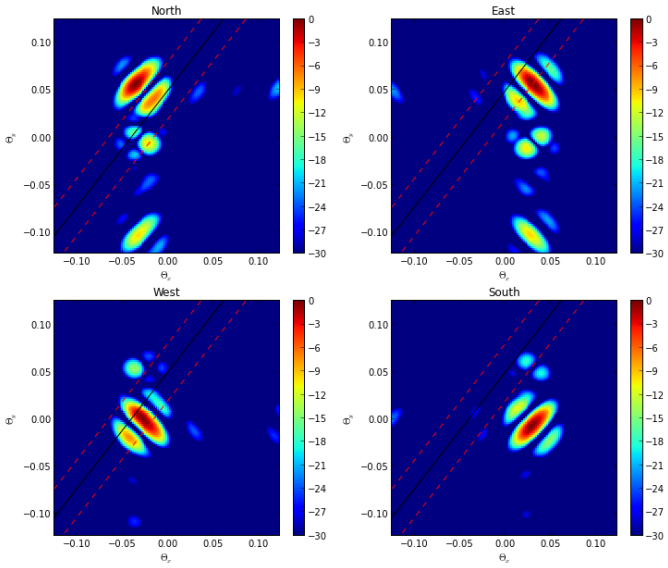
The JRO main antenna, known as the main array, is a large crossed dipole antenna array. Shown in Figure 3.1, the array is approximately 288 x 288 m^2 in area and is capable of transmitting up to 1.5 MW of power at roughly 50 MHz. The main array is divided into quarters (north, south, east and west), each quarter has sixteen modules, and each module has 144 half-wavelength (3 meters) dipoles, for a total of 18432 total dipoles in the array. The dipoles are part of a coaxial-collinear (COCO) structure, where the outer conductor of one coaxial line of $\lambda/2$ length, acting as a radiating dipole, is connected to

Figure 3.1 – The main array in 1966; it looks much the same today.

the inner conductor of the adjacent $\lambda/2$ dipole of an identical coaxial configuration. In this way all $\lambda/2$ dipoles in to COCO structure radiate with "in phase" currents of the outer conductors. Rows of dipoles are crossed at a 90° angle to give orthogonal pairs of dipoles; the top dipoles are known as the up polarized dipoles and bottom dipoles are called the down polarized dipoles.

Each of the modules of the main array can be phased independently to steer and shape the antenna beam pattern. Additionally, within each module the up and down polarizations can have different phasing. Several different MST-ISR experiments, between the years 2005 and 2014, are considered in this thesis. Three different antenna phasing configurations have been used in this time span which are shown in Figures 3.2, 3.3, and 3.4, along with the resulting antenna beam patterns.

The 2005-2007 experiment beam pattern is more symmetric about the zenith of Jicamarca. The east and west beams in the 2009 experiment were moved to be better aligned



(a) The antenna pattern for the 2005-2007 experiments, the dashed red and black lines correspond to the magnetic equator for the year 2006. Θ_X and Θ_y are direction cosines that are described in Section 5.2

North Quarter

2.00 4.29	2.73 3.55	3.47 2.82	4.20 2.08
2.12 2.94	2.85 2.20	3.59 5.44	4.32 4.70
2.24 5.56	2.97 4.82	3.71 4.09	4.44 3.35
2.35 4.20	3.09 3.47	3.82 2.73	4.56 2.00

East Quarter

2.08 4.20	2.82 3.47	3.55 2.73	4.29 2.00
4.70 4.32	5.44 3.59	2.20 2.85	2.94 2.12
3.35 4.44	4.09 3.71	4.82 2.97	5.56 2.24
2.00 4.56	2.73 3.82	3.47 3.09	4.20 2.35

West Quarter

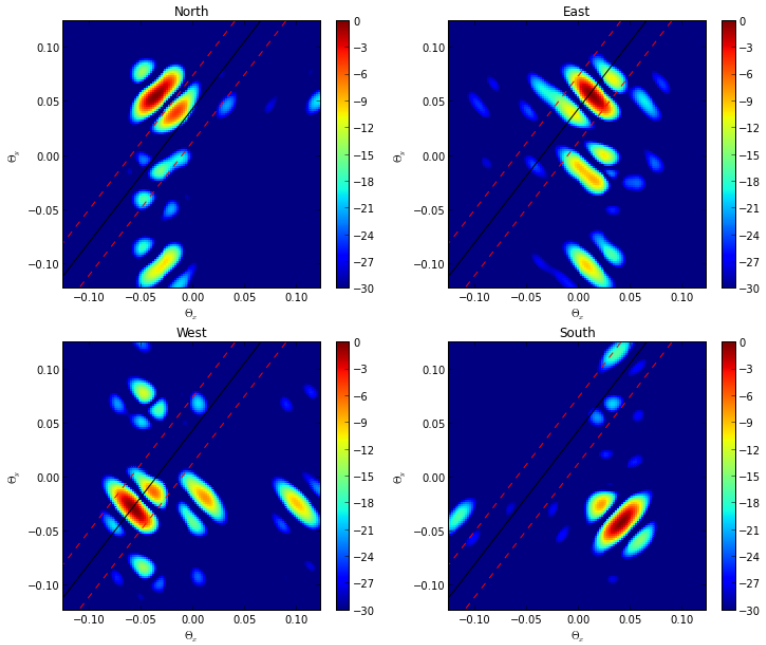
2.08 4.20	2.82 3.47	3.55 2.73	4.29 2.00
4.70 4.32	5.44 3.59	2.20 2.85	2.94 2.12
3.35 4.44	4.09 3.71	4.82 2.97	5.56 2.24
2.00 4.56	2.73 3.82	3.47 3.09	4.20 2.35

South Quarter

2.00 4.29	2.73 3.55	3.47 2.82	4.20 2.08
2.12 2.94	2.85 2.20	3.59 5.44	4.32 4.70
2.24 5.56	2.97 4.82	3.71 4.09	4.44 3.35
2.35 4.20	3.09 3.47	3.82 2.73	4.56 2.00

(b) Antenna phasing for 2005-2007, the large squares show all four of the quarters. Each of the smaller squares corresponds to one of the 64 modules. The top number in the for each modules corresponds to the up polarization phasing (expressed in $\lambda/4$ units) while the bottom number corresponds to the down polarization phasing.

Figure 3.2 – The antenna pattern and phasing used in the 2005-2007 MST-ISR experiments.



(a) The two-way antenna beam pattern for each of the quadrants for the 2009 MST-ISR experiment, the magnetic equator is shown for January 2009.

North Quarter

4 4.29	5 3.55	2 2.82	3 2.08
5 2.94	2 2.20	3 5.44	4 4.70
2 5.56	3 4.82	4 4.09	5 3.35
3 4.20	4 3.47	5 2.73	2 2.00

East Quarter

2 2	3 5	3 3	3 2
5 3	2 2	2 4	2 3
3 3	4 2	4 4	4 3
2 4	3 3	3 5	3 4

West Quarter

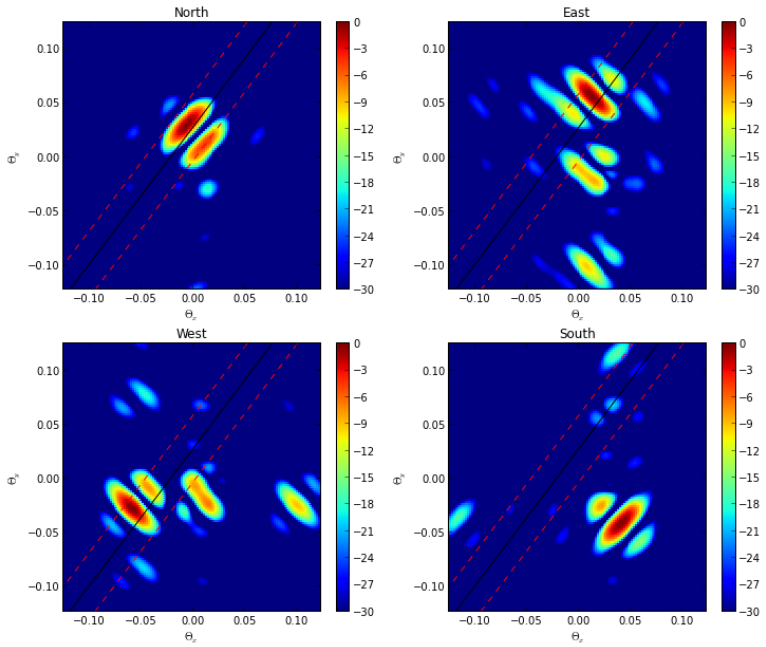
4 4	5 3	5 5	5 4
3 5	4 4	4 2	4 5
5 5	2 4	2 2	2 5
4 2	5 5	5 3	5 2

South Quarter

4 4.29	5 3.55	2 2.82	3 2.08
5 2.94	2 2.20	3 5.44	4 4.70
2 5.56	3 4.82	4 4.09	5 3.35
3 4.20	4 3.47	5 2.73	2 2.00

(b) The antenna phasing for the 2009 MST-ISR experiment.

Figure 3.3 – The antenna pattern and phasing used in the 2009 MST-ISR experiment.



(a) The two-way antenna beam pattern for each of the quadrants for the 2014 MST-ISR experiment, the magnetic equator is shown for January 2014.

North Quarter

4 3.41	5 3.41	2 3.41	3 3.41
5 2.78	2 2.78	3 2.78	4 2.78
2 2.15	3 2.15	4 2.15	5 2.15
3 5.52	4 5.52	5 5.52	2 5.52

East Quarter

2 2	3 5	3 3	3 2
5 3	2 2	2 4	2 3
3 3	4 2	4 4	4 3
2 4	3 3	3 5	3 4

West Quarter

4 4	5 3	5 5	5 4
3 5	4 4	4 2	4 5
5 5	2 4	2 2	2 5
4 2	5 5	5 3	5 2

South Quarter

4 4.89	5 4.89	2 4.89	3 4.89
5 4.26	2 4.26	3 4.26	4 4.26
2 3.63	3 3.63	4 3.63	5 3.63
3 3.00	4 3.00	5 3.00	2 3.00

(b) The antenna phasing for the 2014 MST-ISR experiment.

Figure 3.4 – The antenna pattern and phasing used in the 2014 MST-ISR experiment.

with the magnetic equator. In the 2014 experiment, the north beam was also moved to be closer to the magnetic equator.

3.2 Transmission and Reception

A finer range resolution is needed/desired for probing the mesosphere than the higher regions of the ionosphere where altitudinal variations are more gradual. For this reason, complementary coded pulses with short baud lengths, shown in Figure 3.5, were used for mesospheric probing. These are 64 baud complementary coded pulses with a baud length of one microsecond. In 2009, each one of the 64 baud complementary coded pulses shown in Figure 3.5 were transmitted with a pattern of code A, code B, code -A, code -B, repeated five times.

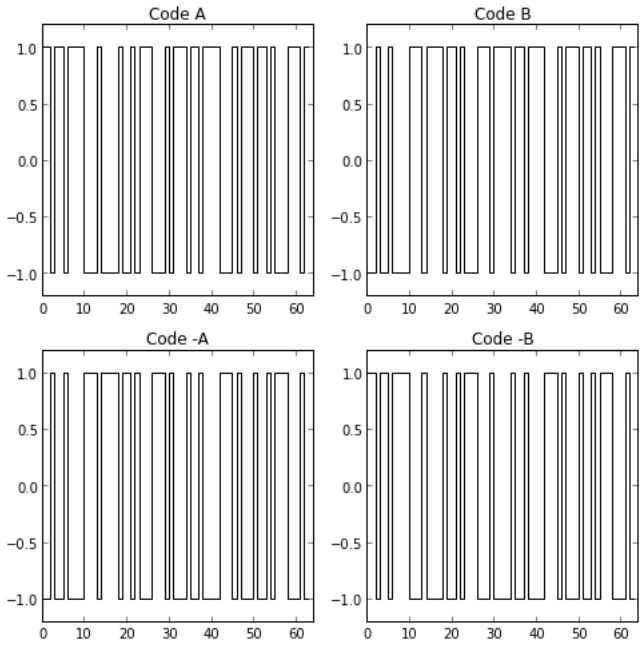


Figure 3.5 – The four MST complementary codes used in 2009.

In Chapter 2, the scattering electric field was derived as Equation (2.10). The receiving antenna of the radar will effectively convert the scattered electric field into a voltage phasor

$$v(t) \equiv I(t) + jQ(t) = lE_s(t) = -\frac{r_e}{r} E_o l n_e(\mathbf{k}, t), \tag{3.1}$$

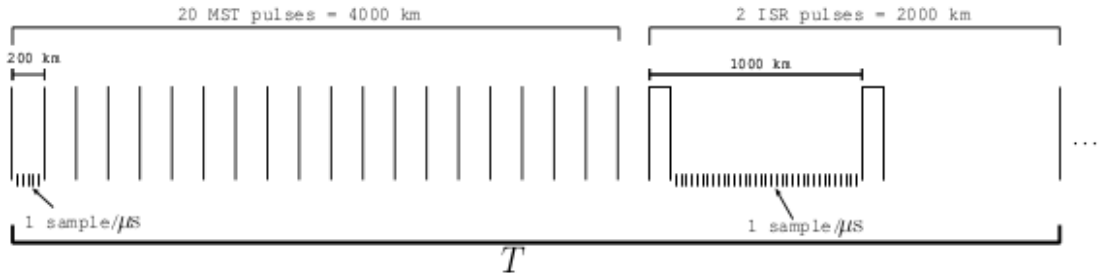
where l is the effective length of the antenna in the direction of the scattering volume, and $I(t)$ and $Q(t)$ are the *in-phase* and *quadrature* components of $v(t)$ respectively. The sequence of twenty complementary coded transmitter pulses are embedded in the electric field amplitude E_o so that the received signal has a similar shape (a delayed replica) as the transmitted signal. At the receiver, the pulses are sampled with a sampling period of one microsecond, giving one sample per transmitted baud. The MST section of the received and sampled signal is then decoded by *matched filtering*. An analog matched filter is given as

$$h(t) = f^*(-t), \tag{3.2}$$

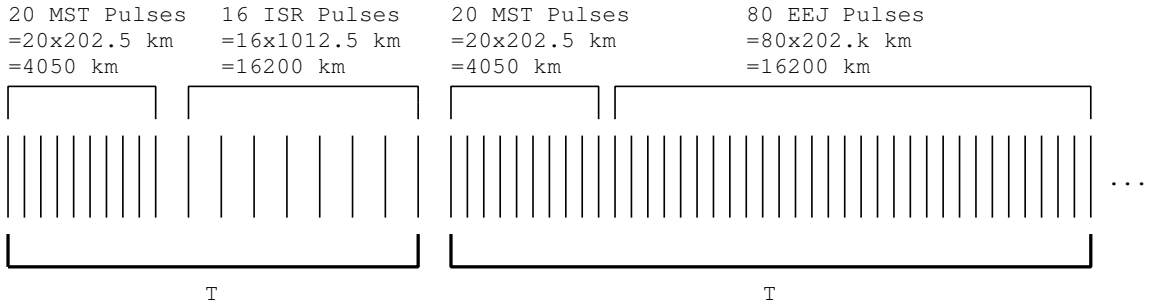
the complex conjugation and time reversal of the original pulse shape $f(t)$. Matched filtering maximizes the SNR of the signal. Additionally when complementary coded pulses are matched filtered and then summed, ideally their sidelobes cancel out. This summing of complementary coded pulses after matched filtering is called coherent integration. In the MST-ISR experiment, typically twenty complementary coded pulses are summed together to get twenty coherent integrations.

Figure 3.6 shows the pulse configurations for the 2005-2007 experiments and for the 2014 experiment. The 2009 experiment's pulse configuration is very similar to the first period of the 2014 experiment. The pulse configuration information can be used for determining *interpulse period* (IPP) and the *Doppler velocity range* for each experiment. For example, in the 2014 MST experiment the IPP is 135 ms. The Nyquist frequency of the voltage sample time series is

$$f_{Nyq} = \frac{1}{2 \times \mathrm{IPP(sec)}}, \tag{3.3}$$



(a) Pulse configuration for the 2005-2007 MST-ISR experiments. $T = \frac{6000 \text{ km}}{c/2} = 40$ ms the period of the transmitted signal. The MST pulses are shorter for higher range resolution. From *Akgiray* [2007].



(b) Pulse configuration for the 2014 MST-ISR experiment with $T = 135$ ms. The sampling is the same as above.

Figure 3.6 – Examples of MST-ISR pulse configurations.

and using the relationship between Doppler velocity, Doppler frequency and Bragg wavelength, the velocity range for the 2014 MST experiment is found by

$$v_d = \frac{\pm\lambda_B}{2 \times \text{IPP(sec)}} = \pm 11.11 \frac{\text{m}}{\text{s}}. \tag{3.4}$$

The Bragg wavelength λ_B, half the wavelength $\lambda_o = c/f_o$ of the operating frequency $f_o = 49.98$ MHz of the Jicamarca radar is approximately 3 meters. For all the experiments, the radar range resolution is calculated as

$$\delta r = \frac{c\delta t}{2} = 0.15 \text{ km}, \tag{3.5}$$

where c is the speed of light in m/s, δt is the one microsecond baud length, and division by two stems from the two-way travel distance of the radar pulse to the scattering volume.

3.3 Experiment Dates and Notes

Table 3.1 shows the dates and some key parameters of the MST part of the MST-ISR experiment. All the experiments have the same range resolution, but 2009 and 2014 have a longer IPP (km) and therefore a smaller usable velocity range. There are differing numbers of available heights, but all cover the required range of 60 to 90 km.

Table 3.1 – Key parameters of the MST experiments.

Day of Year	75	74,76, 105, 115-117, 164-168, 248-251, 346-348	354-356
Date	March 16	March 15, March 17, April 15, April 25-27, June 13-17, September 5-8, December 12-14	December 20-22
Number of Heights	937	1137	937
Height Range (km)	10.95-151.35	9.6-180.0	10.95-151.35
IPP (s)	0.040		

(a) Experiment dates for 2005. There are 22 days of data available.

Day of Year	93-96, 213-215, 248-250, 338-341
Date	April 3-6, August 1-3, September 5-7, December 4-7
Number of Heights	1137
Height Range (km)	9.6-180.0
IPP (s)	0.040

(b) Experiment dates for 2006. There are 13 days of data available.

Day of Year	170-174
Date	June 19-23
Number of Heights	1137
Height Range (km)	9.6-180.0
IPP (s)	0.040

(c) Experiment dates for 2007. There are 4 days of data available.

Day of Year	17-27
Date	January 17-27
Number of Heights	1057
Height Range (km)	9.6-168.0
IPP (s)	0.125

(d) Experiment dates for 2009. There are 10 days of data available.

Day of Year	7-10
Date	January 7-10
Number of Heights	1350
Height Range (km)	0.0-202.5
IPP (s)	0.135

(e) Experiment dates for 2014. There are 4 days of data available.

CHAPTER 4

ESTIMATION OF SPECTRAL PARAMETERS

Chapter 3 discussed the reception, sampling, and coherent integration of MST radar signal, and the procedures that produce collections of coherently integrated time-series at the effective IPP. The absolute square of the discrete Fourier transform (obtained with FFT) of a voltage time-series is a *periodogram*, a poor estimator of the *power spectral density* of the scattered radar signal. Better estimators are obtained by averaging consecutive periodograms, a procedure known as incoherent integration. In this incoherent integration procedure the sum of consecutive periodograms is divided by the product of the periodogram length and the number of periodograms which have been summed. With such a normalization a spectral sum (over its frequency bins) yields the expected value of the scattered signal power. The spectral parameter estimation procedures to be described in this chapter were implemented with one-minute integrated spectra derived from the Jicamarca MST radar data.

4.1 Fitting

Unlike for ISR spectra, no first-principles statistical model exists for the computed MST spectra. Early observers of the MST radar spectra used Gaussian shaped spectral models [*Woodman*, 1985]. To determine the line of sight (LOS) Doppler velocity, spectral

width, and spectral power, the early researchers made use of the relationship of the desired parameters to the first three spectral moments,

$$m_i = \sum_{q=-N/2}^{N/2-1} \omega_q^i S(\omega_q), \tag{4.1}$$

where i is zero, one, or two, N is the number of points in the FFT, ω_q is the Doppler frequency bin, and $S(\omega_q)$ is the radar spectrum. Then the power can be found as

$$P = m_0, \tag{4.2}$$

the mean Doppler frequency as

$$\Omega = \frac{m_1}{m_0}, \tag{4.3}$$

and the mean Doppler velocity is a scaling of this frequency. Finally the spectral width, σ, can be estimated from

$$\sigma^2 = \frac{m_2}{m_0} - \left(\frac{m_1}{m_0}\right)^2. \tag{4.4}$$

These numbers provide good initial estimates and can be reasonably accurate for high SNR cases, but in low SNR cases, fitting a curve to the spectra gives better estimates of these values [*Yamamoto et al.*, 1988].

4.1.1 The Spectrum Model

It is now known that the MST spectra often deviates from the standard Gaussian shape [*Sheth et al.*, 2006] so that the use of a *generalized* Gaussian such as

$$\langle S(\omega) \rangle = A \exp(-|\frac{\omega - \mu}{\sigma}|^p) + B \tag{4.5}$$

— here A is the amplitude, B the noise level, μ the Doppler velocity, and σ is related to the spectral width — improves the accuracy of the fits. In the generalized Gaussian, the exponent, p, is not constrained to be two, which allows for different spectral shapes than the standard Gaussian. As shown in Figure 4.1, setting p as one leads to an exponential shape while setting p to higher values leads to increasingly flatted topped (square-like) spectral shapes.

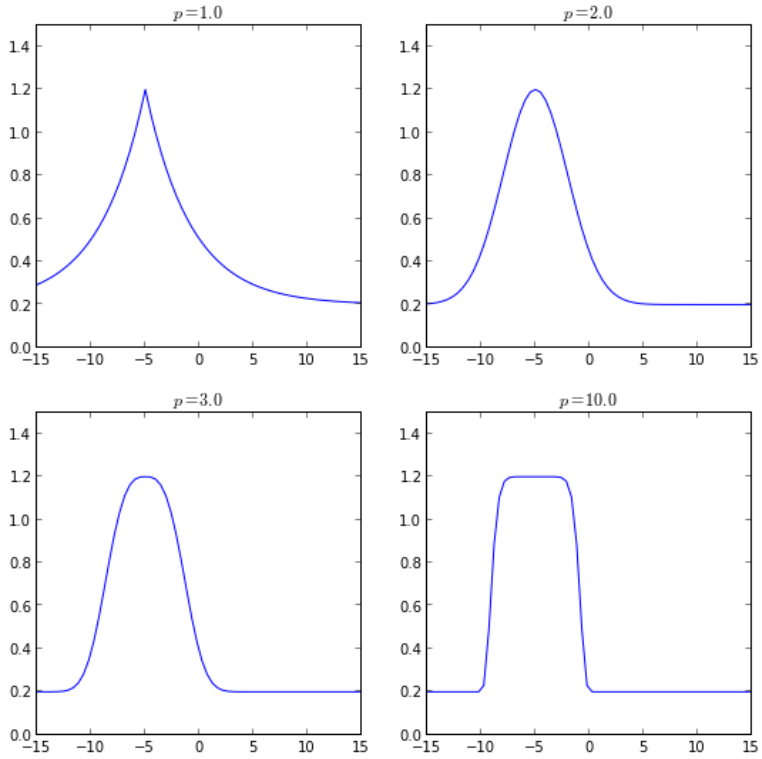


Figure 4.1 – Generalized Gaussian shapes for various p values, with A set as 1.0, μ as -5.0, σ is 4.0, and B set to be 0.2.

As in [*Sheth et al.*, 2006], a generalized Gaussian shape

$$\langle S(\omega)\rangle = B\exp(A\exp(-|\frac{\omega-\mu}{\sigma}|^p)) \tag{4.6}$$

may also be used. It can fit the same shapes as the generalized Gaussian, but with different parameters.

4.1.2 Linear and Logarithmic Misfits

There are two approaches to do the fitting, linear and logarithmic. In linear fitting, the objective is to minimize the misfit, or the square of the difference between the measured spectrum and the model, taking into account variance, as in

$$\chi^2 = \sum_{q=-N/2}^{N/2-1} \frac{(S_q - \langle S_q \rangle)^2}{\langle S_q \rangle^2 / K}, \tag{4.7}$$

where K is the incoherent integration length. In log fitting,

$$\chi^2 = K \sum_{q=-N/2}^{N/2-1} (\ln S_q - \ln \langle S_q \rangle)^2, \tag{4.8}$$

the natural log of the spectrum and model are used.

These misfit equations can be derived as follow [*Kudeki*, 2010]: let S_q be the radar spectrum estimate created by integrating K independent periodograms as described earlier. Note that spectral estimate S_q is in essence a random variable having an expectation $\langle S_q \rangle$ corresponding to the model spectrum, a standard deviation

$$\delta S_{q,rms} = \frac{\langle S_q \rangle}{\sqrt{K}}, \tag{4.9}$$

and a joint Gaussian pdf

$$f(\{S_q\}) = \prod_{\text{all } q} \frac{\sqrt{K}}{\sqrt{2\pi}\langle S_q \rangle} \exp \frac{-(S_q - \langle S_q \rangle)^2}{2\langle S_q \rangle^2 / K}, \tag{4.10}$$

that can be attributed to the central limit theorem. The natural log of the Gaussian pdf can be taken to give

$$-\ln(f(\{S_q\})) = \sum_q \frac{K(S_q - \langle S_q \rangle)^2}{2\langle S_q \rangle^2} + \sum_q \ln(\frac{\sqrt{2\pi}\langle S_q \rangle}{\sqrt{K}}). \tag{4.11}$$

With a large K assumption, this is simplifies to

$$-\ln(f(\{S_q\})) = \frac{K}{2}\sum_{q=-N/2}^{N/2-1}\frac{(S_q - \langle S_q\rangle)^2}{\langle S_q\rangle^2} \equiv \frac{1}{2}\chi^2, \tag{4.12}$$

which is in effect Equation (4.7). Hence, the minimization of χ^2 over the parameters of the model, $\langle S_q\rangle$, amounts to the maximization of the likelihood $f(\{S_q\})$ of all the S_q data over the same set of parameters. This makes the minimization of misfit, Equation (4.7), to provide a maximum likelihood estimate of the input parameters of the spectral model $\langle S_q\rangle$.

To justify the form of the natural log misfit expression, let

$$\delta S_q = S_q - \langle S_q\rangle \tag{4.13}$$

such that

$$\ln(S_q) = \ln(\langle S_q\rangle + \delta S_q) = \ln(\langle S_q\rangle) + \ln(1 + \frac{\delta S_q}{\langle S_q\rangle}), \tag{4.14}$$

which, for a large enough K, will reduce to

$$\ln(S_q) \approx \ln(\langle S_q\rangle) + \frac{\delta S_q}{\langle S_q\rangle} \tag{4.15}$$

because with large K the error δS_q is sufficiently small. Rearranging the terms leads to

$$\ln(S_q) - \ln(\langle S_q\rangle) \approx \frac{\delta S_q}{\langle S_q\rangle} = \frac{S_q}{\langle S_q\rangle} - 1 = \frac{S_q - \langle S_q\rangle}{\langle S_q\rangle}, \tag{4.16}$$

which appears in a squared form in Equation (4.7). It is then obvious that Equation (4.8) follows from Equation (4.7) for a large enough K.

Both of the these χ^2 minimization equations can be solved using nonlinear minimization algorithms. The python code, Appendix B, uses a bounded version of SciPy's optimize.leastsq function. This uses the Levenberg-Marquardt algorithm to solve a nonlinear

least squares problems.

4.1.3 Evaluation of Best Model

To determine which of the models and minimization equations to use, all four possible combinations were tested in a 2500 iteration loop for the nine different spectral shapes (of the type typically observed) specified in Table 4.1. For each case, a spectrum, S, was created. Then inside the loop, a noise-added spectrum was created as

$$S_{na} = S + \frac{S}{\sqrt{K}} N(0,1), \tag{4.17}$$

where $N(0,1)$ is the standard normal distribution, K was set as 20, near the typical number of incoherent integrations. Then each of the four combinations of models and minimizations were used to fit the noise added spectra, as shown in Figure 4.2. Each of the resulting fitted spectra were saved from each iteration. Finally after all the iterations had completed, χ^2 was computed using the true spectra as S_q and each fitted spectra as $\langle S_q \rangle$ and then averaged over all 2500 iterations to determine which model and misfit had the typical best fit. The results of this are shown in Table 4.2. For every case, the first generalized Gaussian model in Equation (4.5) had a lower χ^2 than the second model given in Equation (4.6), making the first model the better choice. The choice between the linear misfit, Equation (4.7), and the log misfit, Equation (4.8), is slightly less obvious (not unexpected since they should be identical in infinite K limit). The linear misfit had a slightly lower χ^2 in two instances, case one, with low p, and case six, with high SNR. However the log misfit had a lower χ^2 in the other seven instances, and significantly lower χ^2 in cases 7 and 8, with low SNR and small σ, respectively. Consequently, the log fit was chosen.

Table 4.1 – The nine test cases, with parameters given for model 1. Parameters for model 2 were calculated to give the same spectral shape for each case.

Case	B	A	μ	σ	p
1	0.2	0.7	-5	4.2	1
2	0.2	0.7	-5	4.2	1.5
3	0.2	0.7	-5	4.2	2
4	0.2	0.7	-5	4.2	2.5
5	0.2	0.7	-5	4.2	3
6	0.2	1.5	-5	4.2	2
7	0.2	0.4	-5	4.2	2
8	0.2	0.7	-5	2.2	2
9	0.2	0.7	-5	8.2	2

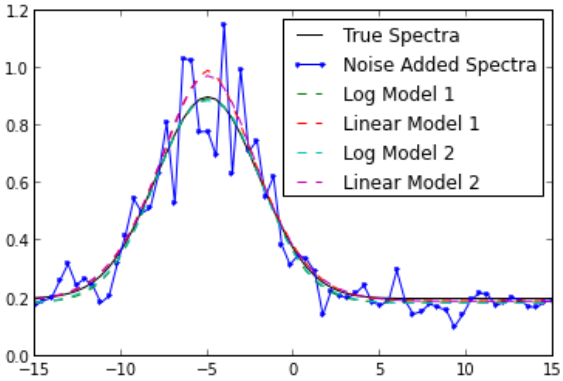


Figure 4.2 – All four combinations of models shown for case 1.

Table 4.2 – Model comparison for all nine cases.

Case1	χ^2	**Case 4**	χ^2	**Case 7**	χ^2
Log Model 1	0.3231	Log Model 1	0.3236	Log Model 1	0.3151
Linear Model 1	0.3141	Linear Model 1	0.3693	Linear Model 1	0.4683
Log Model 2	0.3213	Log Model 2	0.3255	Log Model 2	0.3226
Linear Model 2	0.3213	Linear Model 2	0.3738	Linear Model 2	0.5060
Case 2	χ^2	**Case 5**	χ^2	**Case 8**	χ^2
Log Model 1	0.3213	Log Model 1	0.3294	Log Model 1	0.3283
Linear Model 1	0.3531	Linear Model 1	0.3744	Linear Model 1	0.3659
Log Model 2	0.3118	Log Model 2	0.3307	Log Model 2	0.3392
Linear Model 2	0.3708	Linear Model 2	0.3729	Linear Model 2	0.4283
Case3	χ^2	**Case 6**	χ^2	**Case 9**	χ^2
Log Model 1	0.3396	Log Model 1	0.3399	Log Model 1	0.3268
Linear Model 1	0.3653	Linear Model 1	0.3322	Linear Model 1	0.3341
Log Model 2	0.3414	Log Model 2	0.3398	Log Model 2	0.3265
Linear Model 2	0.3710	Linear Model 2	0.3363	Linear Model 2	0.3212

4.1.4 Constraint on p

It was discovered that the estimated exponent in the generalized Gaussian, here on referred to as p, varied wildly in value between consecutive heights. Values over 100 were sometimes observed, while the value in the height above may be a much more reasonable number, such as 2. Even with bounds applied, values between adjacent heights could still vary greatly. This can be interpreted as instances of "overfitting" of noisy data, an undesirable effect. To rectify this, p was changed from a single value to a second-order linear equation giving the modified spectra model

$$\langle S(\omega)\rangle = A\exp -|\frac{\omega-\mu}{\sigma}|^{(p_0+p_1z+p_2z^2)} + B, \tag{4.18}$$

where z is a height index. Consequently sets of two or more contiguous heights with a detectability of greater than four are grouped and fitted at the same time, so that they all have the same p_0, p_1, and p_2 and the center of the height indexes is rescaled to be zero. This procedure amounts to a practical "regularization" strategy to avoid instances of overfitting.

4.2 Estimation of Measurement Error

The fitting procedure is expected to provide model parameters with estimation errors. To determine the rms error levels, the procedure of *Aster et al.* [2005] is followed: First the

Jacobian of the data model, Equation (4.18), is determined as

$$J = \begin{bmatrix} \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta B} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta B} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta A} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta A} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta \mu} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta \mu} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta \sigma} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta \sigma} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta p_0} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta p_0} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta p_1} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta p_1} \\ \frac{1}{\langle S(\omega_1)\rangle}\frac{\delta\langle S(\omega_1)\rangle}{\delta p_2} & \cdots & \frac{1}{\langle S(\omega_N)\rangle}\frac{\delta\langle S(\omega_N)\rangle}{\delta p_2} \end{bmatrix}. \tag{4.19}$$

The inverse term, $\frac{1}{\langle S(\omega)\rangle}$, in the elements of J is a result of taking the logarithm of the model when fitting. The errors of interest are found from the covariance matrix,

$$\mathrm{Cov}(S) = (JJ^T)^{-1}, \tag{4.20}$$

as

$$\delta\langle S(\omega)\rangle = \sqrt{\mathrm{diag}(\mathrm{Cov}(S))}. \tag{4.21}$$

The result is a vector containing the measurement error pertaining to each of the calculated parameters. Though SciPy's optimize.leastsq can create the same covariance matrix, it is sometimes unreliable when processing large sets of data, particularly because to constrain p, groups of heights are fitted together such that minimization over 50 parameters is possible. If optimize.leastsq encounters a singular matrix, it will not output a covariance matrix. Instead the covariance matrix can be calculated using the fitted parameters for each height. The python code for implementing this is included in Appendix B.

4.3 Fitting and Spectral Parameter Results

All the available data was fitted using the aforementioned procedure, plus fitting the double Gaussians as explained in Section 4.4. Four examples of spectra are shown in Figure 4.3, with the parameters given at the bottom of the figure. Two are of low SNR cases, and the other two are of higher SNR cases for the years 2007 and 2009. Given that 2005-2007 data has the same velocity range, the spectra from these years look very similar. The 2009 data have a narrower velocity range and consequently higher velocity resolution, so the spectra appear wider and show more noise. Figure 4.3 also shows the fitting routine is able to fit most cases with low SNR fairly accurately. The minimum SNR fitted was -15 dB. Additionally, to detect possible presence of signal, the noise variance of each spectra was estimated. Only those spectra with a detectability of four (spectra containing at least one point four times above the estimated noise variance) were fitted.

To view the trends over entire days and data, looking at individual spectra is of limited use. It is more appropriate to look at plots with range on the y-axis, time on the x-axis, and the particular parameter being observed on a color scale. Though there are over 50 days of data available, to show all the parameters across all four beams for all the days in thesis would be of little value. For brevity, only one day is shown here.

First shown in Figure 4.4 is a range-time-velocity plot, also known as a velocity map for all four beams on August 3rd, 2006. Velocity oscillations due to gravity waves are visible in all four beams for most of the day, as the visible repeating vertical bands of color. The gravity waves do not appear to greatly affect σ, shown in Figures 4.5. The map of σ shows that there are patches of smaller σ and patches of larger σ.

The maps of p, shown in Figure 4.6, show how the spectral shapes deviate from the standard Gaussian shape. Most of the time, the middle of the layers have p values near or less than two. The higher values near the edges may be due to the low SNR at those locations. The layers between 11:30 and 14:00 shows interesting nearly vertical striations.

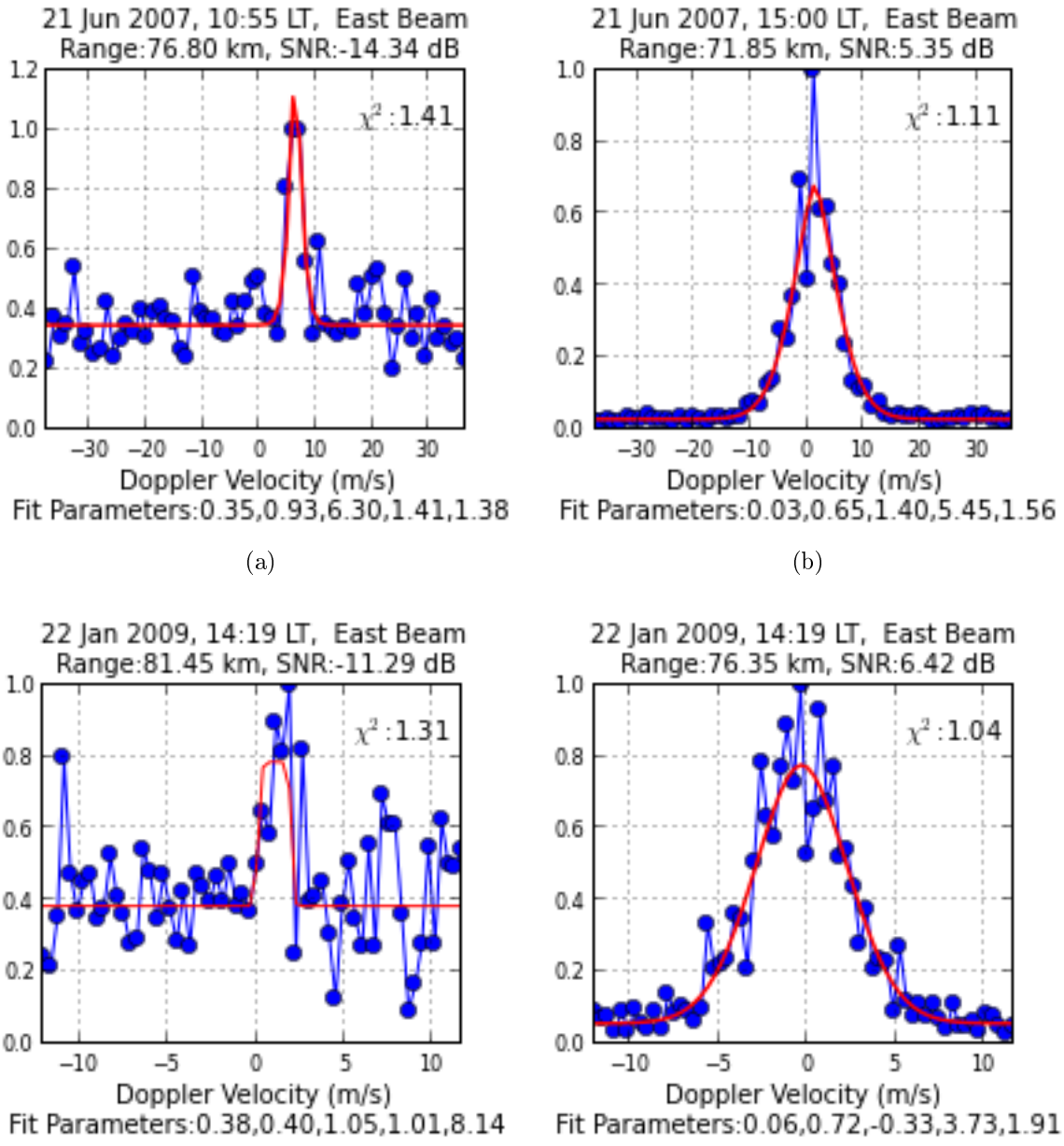


Figure 4.3 – Examples of typical low and high SNR spectra for the years 2007 and 2009. The fit parameters are given as: B, A, μ, σ, p. The normalized misfit and SNR are also shown.

They appear to be nearly on the same frequency as the oscillations in velocity, but it is difficult to tell at first glance if the two are related.

Additional velocity maps for the other days processed may be viewed at http://remote2.csl.illinois.edu/MSTISR/showmaps_2. The data may be downloaded from http://cedar.openmadrigal.org/.

4.4 Double Gaussians

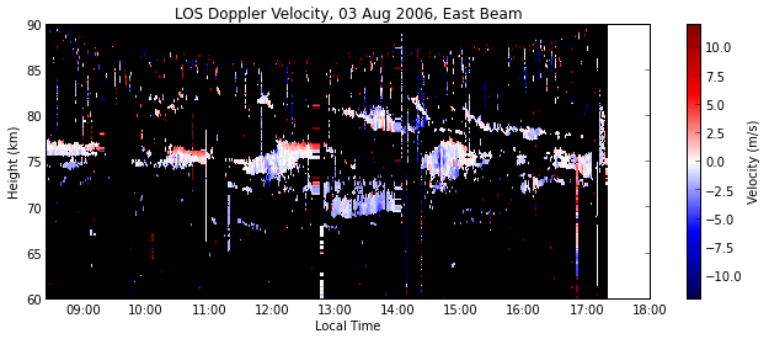
So far only spectra where only one Gaussian peak is observed in the data has been discussed. However, there are times where two separate spectra peaks are observed in the spectral data. In this case, if only one peak is used to try to fit the spectral data, one of two things happen. First, one peak may be much smaller than other and only the large peak is fitted as shown in Figure 4.7a. In the other case, one peak is fitted to both peaks, as shown in Figure 4.7b. In the second case, the LOS Doppler velocity and spectral width estimates are both greatly affected, showing the need to fit two Gaussians to the two peaks.

The model, Equation (4.18), is simply modified to add another generalized Gaussian peak giving

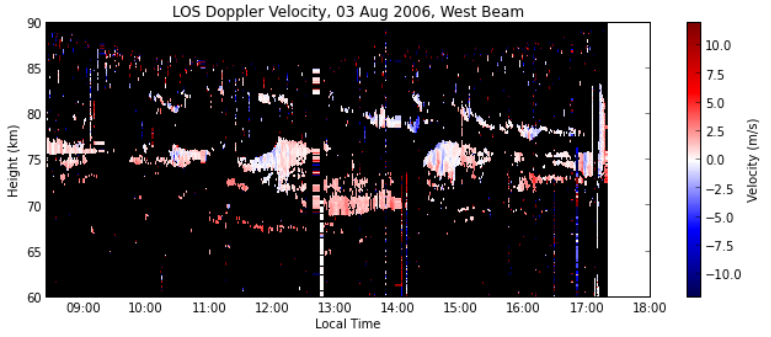
$$\langle S(\omega)\rangle = A_1 \exp -|\frac{\omega-\mu_1}{\sigma_1}|^{(p_0+p_1z+p_2z^2)} + A_2 \exp -|\frac{\omega-\mu_2}{\sigma_2}|^{(p_3+p_4z+p_5z^2)} + B. \qquad (4.22)$$

Only one noise term is needed and both peaks have the linear constraint on p. For the two examples shown before, the new results are shown in Figures 4.7c and 4.7d. Both cases now have two Gaussian peaks for a better fit.

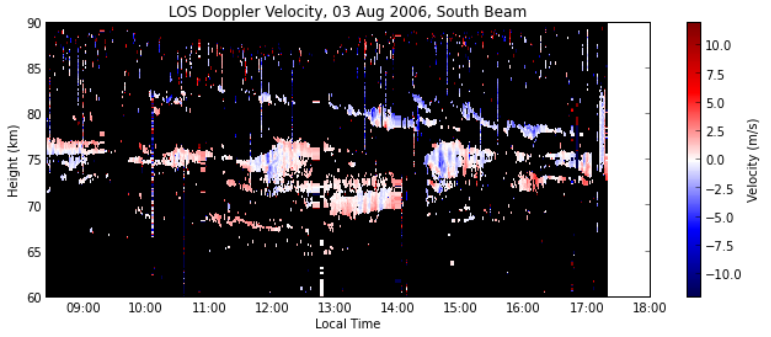
As shown by the spectra in Figures 4.3 not every spectra needs two Gaussians. In fact fitting two Gaussians to every spectra can cause some strange spectra shapes and some bad fits. A detection algorithm is need to determine when/where to fit using the double Gaussian model versus the single Gaussian model. The algorithm developed here searches



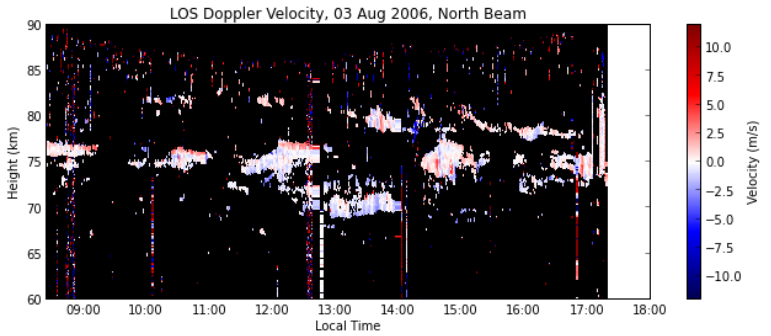
(a)



(b)

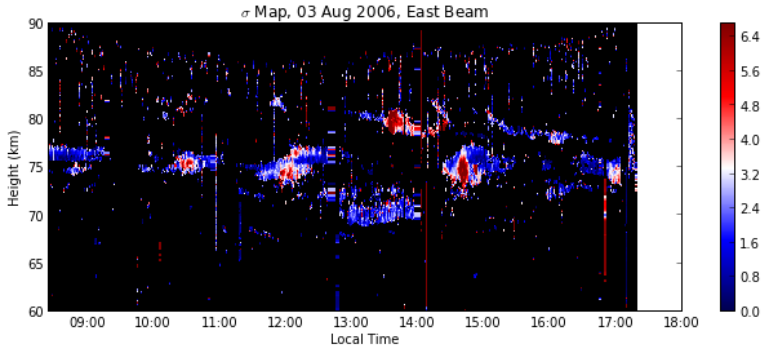


(c)

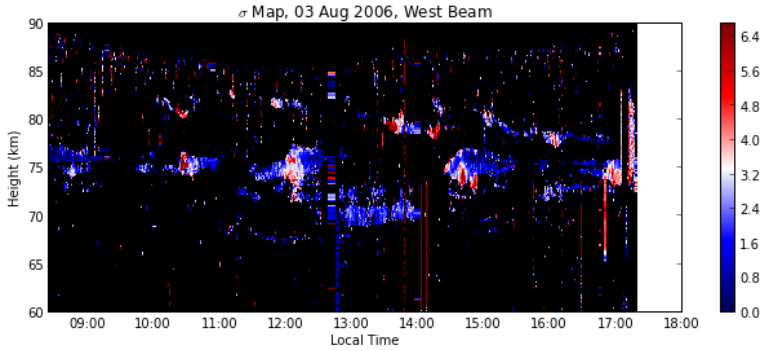


(d)

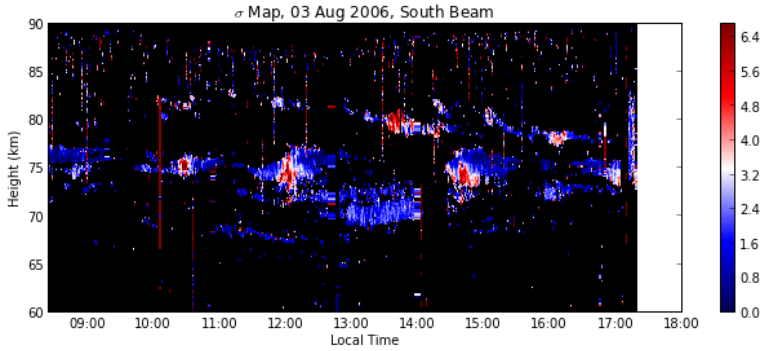
Figure 4.4 – Velocity maps from August 3rd, 2006 for all four beams.



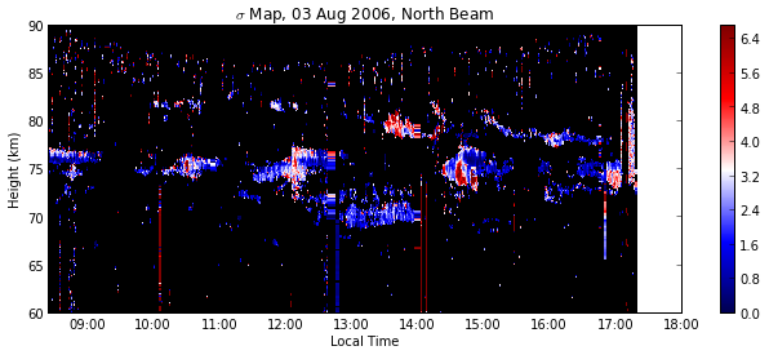
(a)



(b)

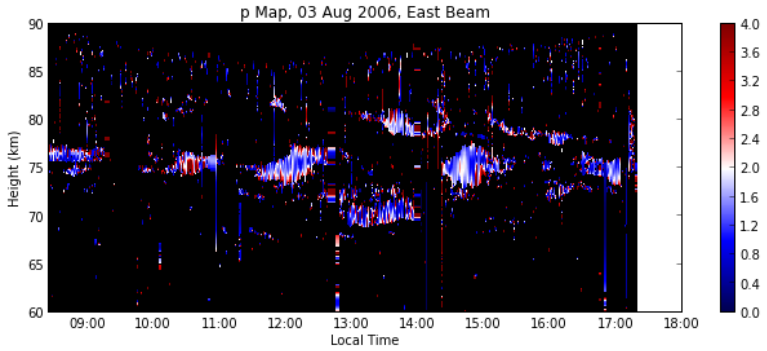


(c)

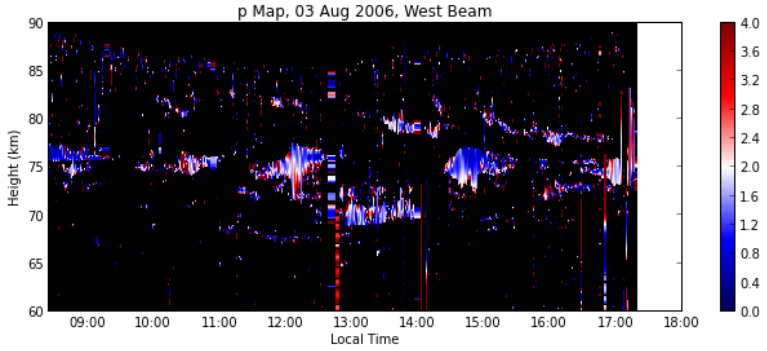


(d)

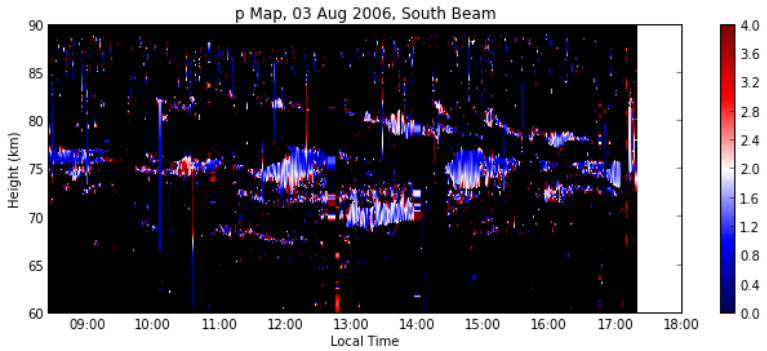
Figure 4.5 – σ maps from August 3rd, 2006 for all four beams.



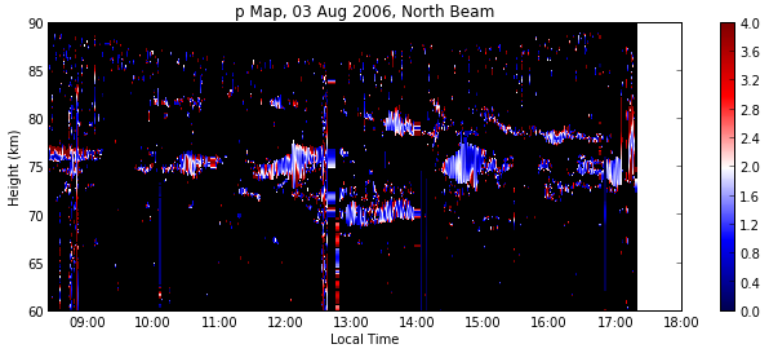
(a)



(b)

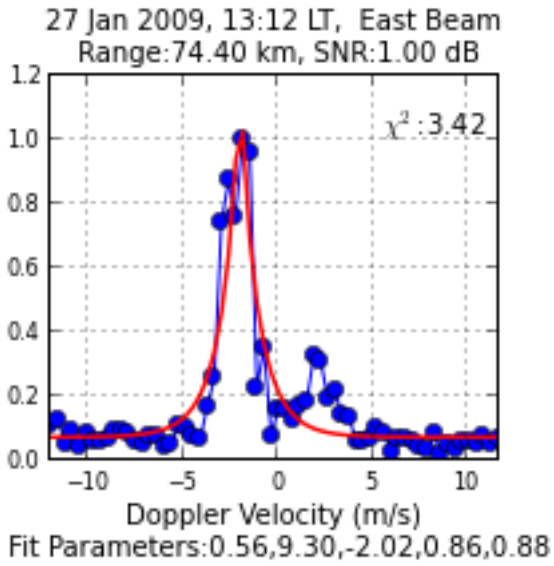


(c)

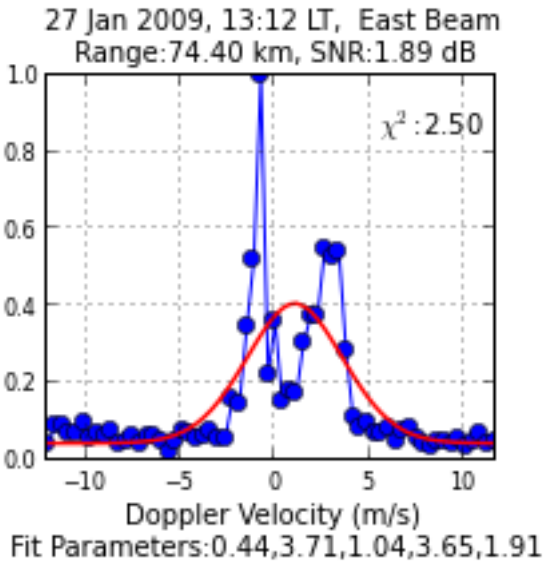


(d)

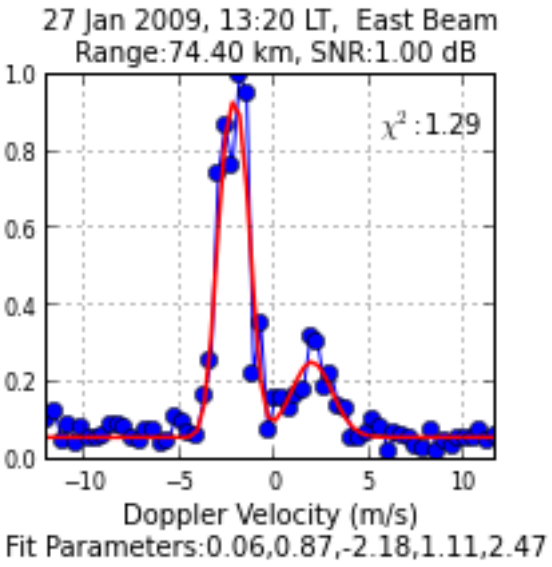
Figure 4.6 – p value maps from August 3rd, 2006 for all four beams.



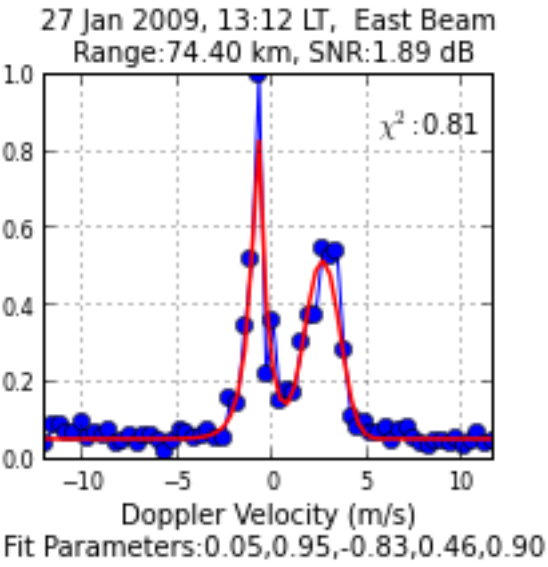
(a) A small peak that is not fitted is shown to the left of main peak.



(b) One peak fitted to two Gaussians.



(c) Small peak fitted with a double Gaussian.



(d) Both peaks are fitted with separate Gaussians.

Figure 4.7 – Examples of fitted spectra with double Gaussians.

for three conditions that may possibly indicate that a double Gaussian is present. The first condition is having a pair of separated peaks, two sets of contiguous points above a threshold, such as a detectability of four. The second condition is having a velocity skew defined as when the estimated velocity from the moments as described in Section 4.1 is significantly different from the velocity of the spectral peak. The third and final condition is having one significant multipoint peak and one single point peak. This is when there is one set of contiguous points above the threshold and one separate single data point above twice the threshold. The flow chart of the algorithm is shown in Figure 4.8, and the python code is located in Appendix B.

The algorithm in general works as follows: First it determines the maximum value and the location (frequency bin) of the spectral peak. Then it creates a new list of spectral samples by removing any samples that are either less than a detectability of four or are lower than one-fifth of the spectral peak value. Next it separates that list into sublists of contiguous points in the spectral data, so that main peak and any other secondary peaks are in separate lists. Under the first and third conditions (separated peaks) discussed above, the list containing the most significant peak is removed, so that only lists containing any potential secondary peaks remain. These lists are then checked to determine if a second Gaussian is present. Under the second condition, the estimated velocity from moments is simply checked against the velocity of the main peak. The second and third conditions are not very robust in the sense that they can frequently lead to erroneously detected double Gaussians. Therefore when a potential double Gaussian is detected under one of these two conditions, it is first tentatively fitted using Equation (4.22) and the resulting fitted spectra (spectral fit line) are again examined against the three conditions. If a double peak is still detected in the fitted spectra, than the spectra are determined to be a double Gaussian.

Additionally, this algorithm is used to determine the initial conditions of the amplitude A, the Doppler velocity μ, and the generalized Gaussian width σ, for fitting. When there is only a single Gaussian detected, the amplitude is estimated as 80 percent of the maximum

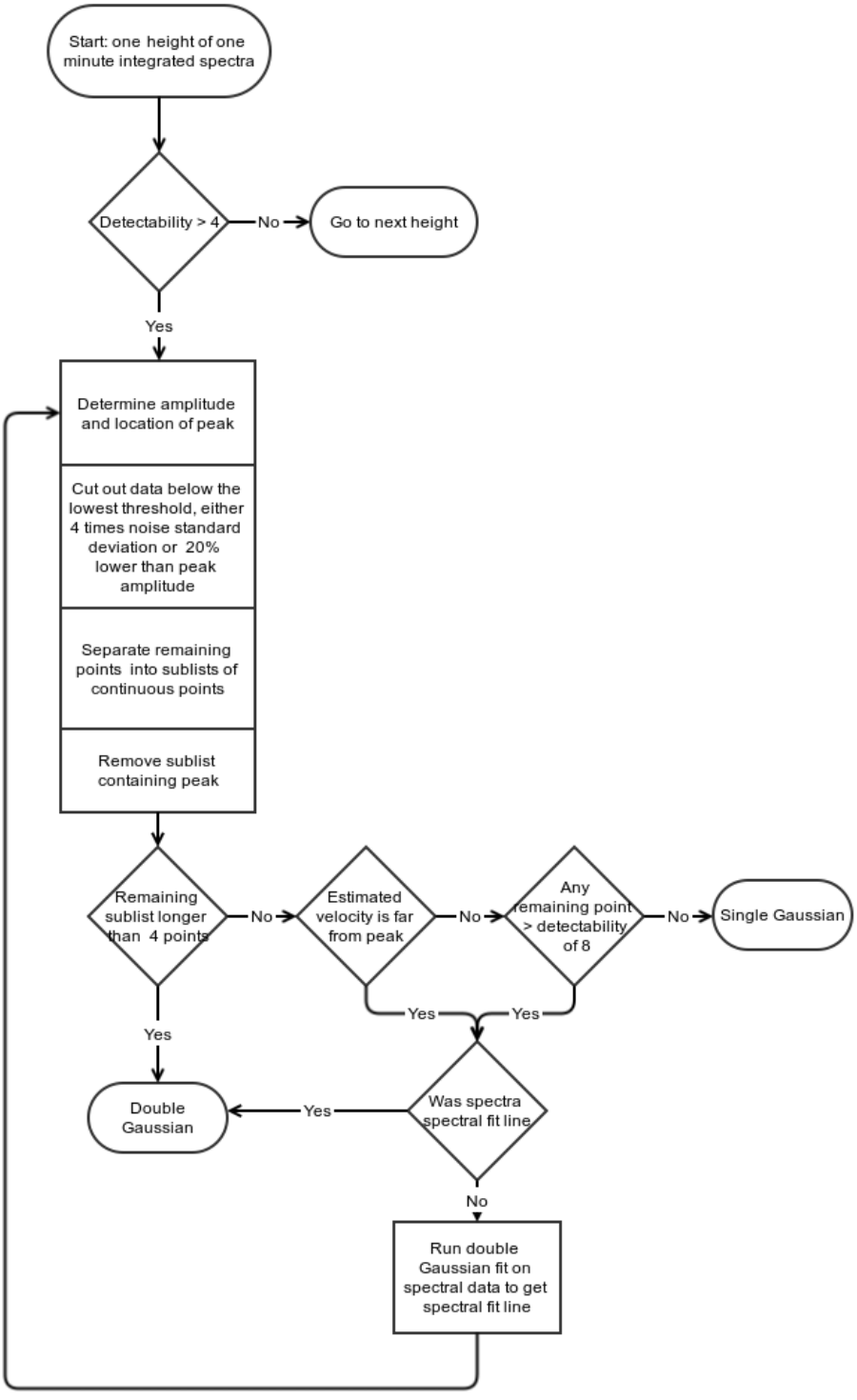


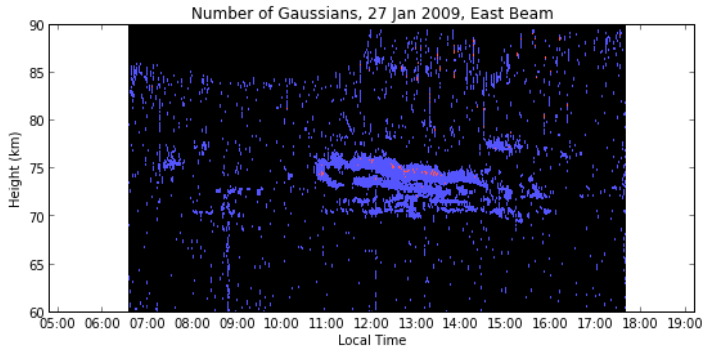
Figure 4.8 – Flow chart for the double Gaussian fitting algorithm, starting after the spectra have been made/loaded.

amplitude; initial amplitude is taken to be lower than the peak value to take into account additive noise. The Doppler velocity is determined using the first moments as explained in Section 4.1, but using a reduced vector containing only the points in the Gaussian above the threshold (the points in the Gaussian peak). The σ is determined by multiplying half the number of points above the threshold by the velocity resolution. When there are two separate Gaussians, the amplitude, Doppler velocity, and width are found similarly to when there is only one Gaussian. The amplitudes are 80 percent of the maximum of each Gaussian peak, and the Doppler velocities are calculated using moments of the velocity values of the sublists for each Gaussian, the σ is calculated using half the points in each peak. In the case of skewed velocity caused by double Gaussians, the initial guess for the amplitude of the main peak is 80 percent of the max, and the initial guess for the velocity of the main peak is the location of the max. For the second Gaussian, the initial amplitude is set as 60 percent of the max, and the initial velocity is set as the velocity estimated by the moments. The initial σ for each is the difference in velocity between the main peak and the calculated velocity. This method is fairly accurate for determining the number of Gaussians and the initial estimates for the amplitude and velocity. However it does not detect every double Gaussian, particularly those that are strongly overlapping and occasionally erroneously determines that there are two Gaussians, when in fact there is only one.

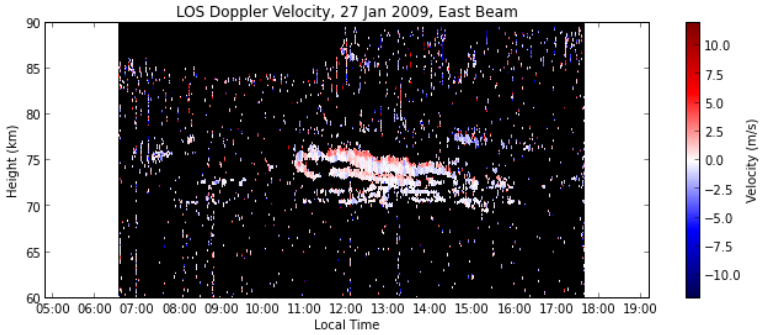
4.4.1 Location and Cause

Figures 4.9 and 4.10 depict range-time maps of the number of Gaussian spectral peaks along with the corresponding LOS velocity maps for June 27, 2009 and June 21, 2007 respectively. When the spectrum is determined to have two peaks, the Doppler velocity shift from the peak with the largest amplitude is used in velocity maps and wind estimations.

Both figures show LOS velocity gradients in range, but the data from January 27, 2009 shows many more double Gaussians near the velocity gradients than 2007 data. Addi-

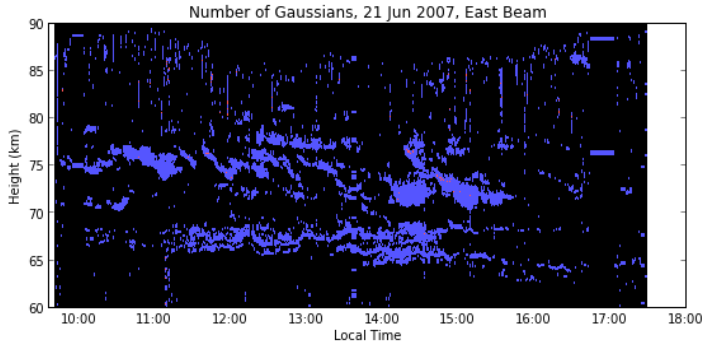


(a) Number of Gaussian peaks, blue designates single Gaussians, while red designated double Gaussians for January 27th, 2009.

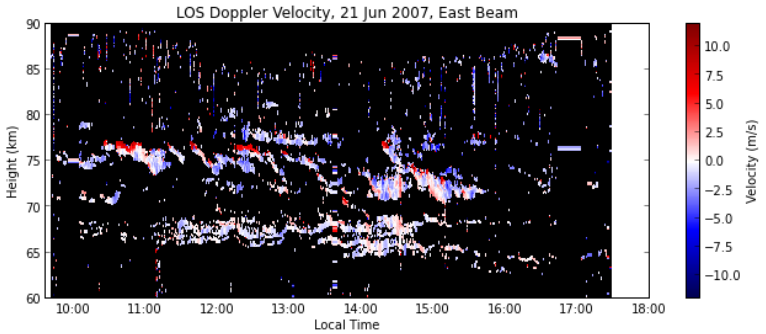


(b) Corresponding LOS doppler velocity map for January 27th, 2009.

Figure 4.9 – The number of Gaussian peaks and corresponding LOS doppler velocity for January 27th, 2009.



(a) Number of Gaussian peaks, blue designates single Gaussians, while red designated double Gaussians for June 21st, 2007.



(b) Corresponding LOS doppler velocity map for June 21, 2007.

Figure 4.10 – The number of Gaussian peaks and corresponding LOS doppler velocity for June 21, 2007.

Table 4.3 – Percentage of fits in a single day that are double Gaussians.

Year	Beam	Mean(%)	Min(%)	Max(%)
2014	East	1.57	0.98	1.96
2014	West	1.32	1.06	1.63
2014	South	1.16	0.64	1.68
2014	North	1.18	0.44	1.84
2009	East	1.33	0.86	2.43
2009	West	1.02	0.47	1.47
2009	South	1.14	0.73	1.88
2009	North	1.23	0.65	1.81
2007	East	0.95	0.51	1.62
2007	West	0.64	0.44	0.90
2007	South	0.53	0.34	0.70
2007	North	0.88	0.43	1.28
2006	East	0.87	0.35	2.15
2006	West	0.57	0.28	1.19
2006	South	0.58	0.17	0.91
2006	North	0.97	0.24	2.40
2005	East	0.83	0.12	1.93
2005	West	0.47	0.13	0.83
2005	South	0.53	0.17	1.20
2005	North	1.00	0.10	2.58

tionally the double Gaussians appear almost exclusively near the velocity gradients. This demonstrates that many of the double Gaussians occur where there are large velocity gradient but not every large velocity gradient causes double Gaussians. An interpretation of these observations will be provided shortly.

In Table 4.3, the average, minimum, maximum percentage of double Gaussians for a single day is compared for each beam for each year. The experiments in 2005-2007 had the same beam pattern, while the 2009 and 2014 experiments both had different beam patterns. The table shows the beam pattern most likely plays a strong role in determining the number of double Gaussians. Additionally in the 2005-2007 experiments, there are significantly more double Gaussians on the east and north beams than on the other two. Additionally, looking at the velocity maps for 2005-2007, only the east and north beams

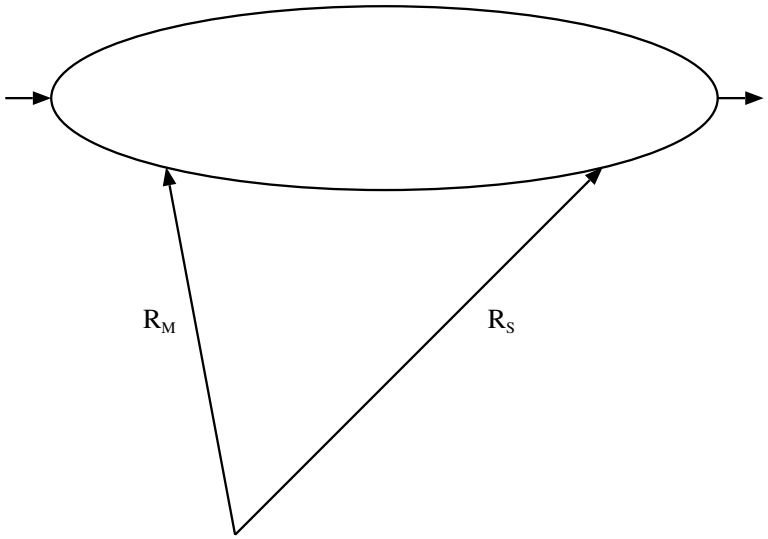


Figure 4.11 – Diagram of the main lobe (R_M) and a sidelobe (R_S) and scattering volume moving parallel to both. The main lobe sees the horizontal wind vector pointing towards it, while the sidelobe sees the horizontal wind vector pointing away. The sidelobe also has a longer distance to travel to the same height, so the backscatter echoes from the volume are delayed from the sidelobe compared to the main lobe.

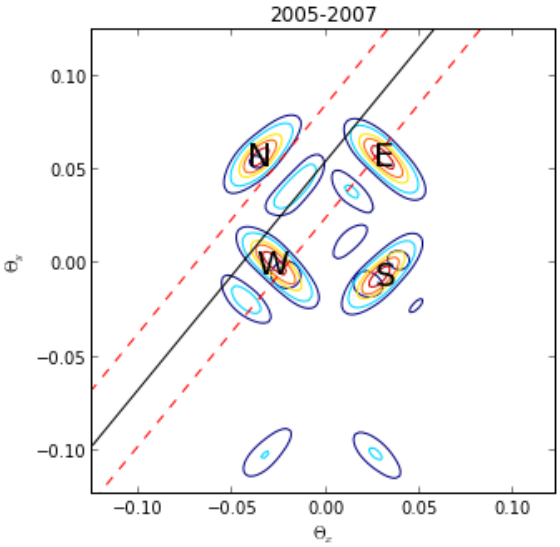
show velocity gradients and only at the tops of layers. This can be explained by the sidelobes in these two beams [*Sheth*, 2004].

Specifically, the north and south beams in the 2005-2007 experiments, as shown in Figures 3.2 and 4.12a have relatively large sidelobes to the southwest of main beam pattern, while the west and south beams do not have such sidelobes. Therefore what appears to be velocity gradients in the east- and north-beam velocity maps may be due to delayed echoes from beam sidelobes that appear at different velocity. The velocity gradients observed are actually artifacts from the sidelobes. A diagram of this is shown in Figure 4.11.

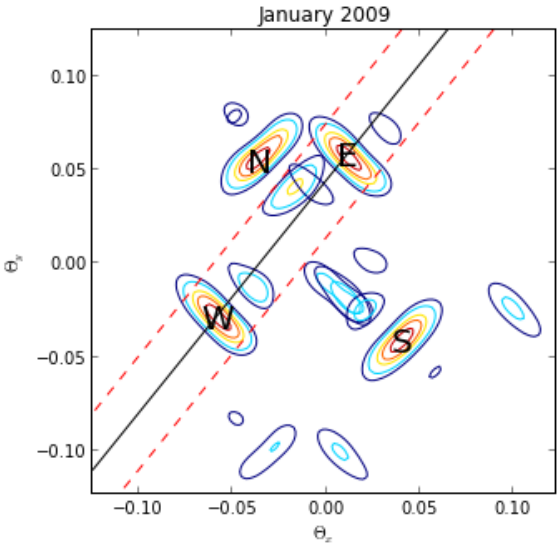
Expanding this conjecture to the 2009 and 2014 experiments, with their beam patterns shown in Figures 4.12b and 4.12c respectively, it can be seen the same theory also applies. Though there are only four days of data for 2014, the east and west beams have more significant sidelobes and have more double Gaussians than the north and south beams and more double Gaussians on average in Table 4.3. In 2009, at first glance it is slightly less clear what is happening. The north, east, and west beams have significant sidelobes, but

the west beam does not show as many double Gaussians as the other two most of the time. This can be explained by the location of their sidelobes relative to the direction of the wind velocity. The north and east beams have large sidelobes southeast of their respective main lobes, while the west beams sidelobe is southwest of its main beam and perpendicular to north and east beams sidelobes. Referring back to Figure 4.11, if the horizontal wind vector is coaligned with the plane defined by the north and east beams main lobes and sidelobes then there may be double Gaussians, but then the wind vector will be perpendicular to the plane of the west beam's main lobe and sidelobe so that the Doppler velocities for the west beam will overlap. In short, if the scattering volume is mainly causing double Gaussians in the north and east beams, then will likely not cause many double Gaussians in the west beam for the 2009 antenna configuration. Using the wind estimates explained in Chapter 5, this is confirmed. For January 27th, whose east beam is shown in Figure 4.9, the wind direction, shown in Figure 5.5, was estimated to be typically south-west to west near 75 km where the double peaked spectra occur. Consequently there are more double Gaussians in the east and north beams, 1.74% and 1.51% respectively, than in the west beam, 0.59%.

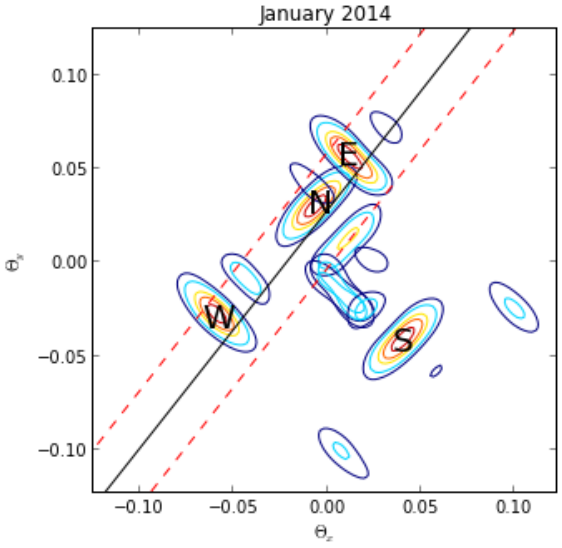
Back in Table 4.3, it is shown the most number of detected double Gaussians is less than 3 percent of the total number of fits for a day, making them fairly rare, despite the sidelobes always being present. There are three main reasons for this: First, the sidelobes are roughly nine dB down from the main lobes, so that many times the signal from the sidelobes is too weak to be separated from the noise. Second, as explained in the preceding paragraphs, the direction of the wind is important. If the wind vector is perpendicular to plane of the main lobe and sidelobe the Doppler velocities from each will appear to be the same and only one Gaussian will be observed. Third, as demonstrated in Figures 4.9 and 4.10, even if there is a Gaussian from the sidelobe at a different Doppler velocity, the Gaussian from the main lobe may disappear before the delayed Gaussian from the sidelobe starts. This is largely dependent on the thickness of the layer.



(a) Two way beam pattern for the 2005-2007 experiments beam.



(b) Two way beam pattern for the 2009 experiment.



(c) Two way beam pattern for the 2014 experiment.

Figure 4.12

CHAPTER 5

WIND ESTIMATION

In previous experiments ([*Fukao et al.*, 1979], [*Hitchman et al.*, 1997], [*Lehmacher et al.*, 2007]) the wind components were estimated purely from the LOS Doppler velocities and the geometry of the beam pattern as

$$U = \frac{\mu_{east} - \mu_{west}}{2\sin(\theta)}, \tag{5.1}$$

and

$$V = \frac{\mu_{north} - \mu_{south}}{2\sin(\theta)}, \tag{5.2}$$

where θ is the zenith angle of the antenna. Two estimates were often created for the vertical wind velocity,

$$W = \frac{\mu_{east} + \mu_{west}}{2\cos(\theta)} \tag{5.3}$$

or

$$W = \frac{\mu_{north} + \mu_{south}}{2\cos(\theta)}. \tag{5.4}$$

In the 2009 and 2014 experiments, the beam pattern has changed so that the four beams are not equally distributed around the zenith in the cardinal directions (see Figure 4.12). Thus, the above equations would not be accurate for these experiments. Therefore new method of estimating the wind components that takes into account the beam directions

and Jicamarca's positional geometry has been developed.

5.1 Determining Jicamarca Unit Vectors

The precise location of the Jicamarca antenna array will be specified in an earth-centered, earth-fixed (ECEF) coordinates, in terms of the World Geodetic System (WGS 84) ellipsoid specifications. As shown in Figure 5.1, the ECEF coordinate frame is centered on earth's center of mass, the X-axis points through the prime meridian at the geographic equator, the Z-axis points north along the earth's average axis of rotation, and the Y-axis also points through the geographic equator, 90^{o} east of the X-axis.

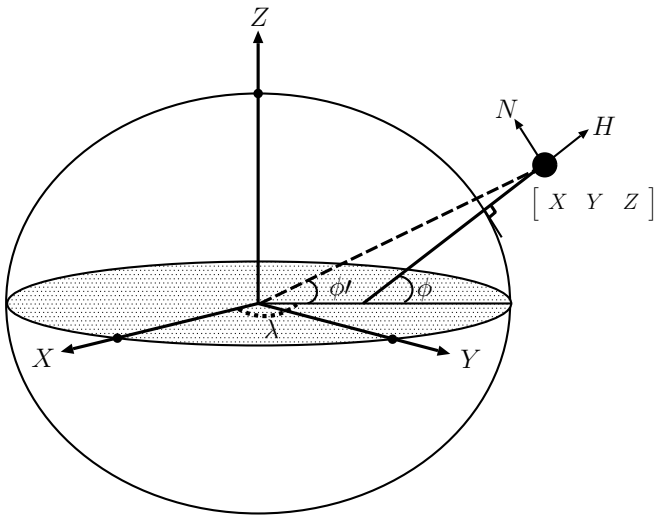


Figure 5.1 – The WGS 84 ellipsoid and corresponding ECEF coordinate system are shown. A point is placed at a general location, with corresponding north (N) and zenith (H) vectors. The east vector (E) is directed into the page. λ is the geodetic longitude. ϕ is the geodetic longitude, which crosses normal to the ellipse. $\phi\prime$ is the geocentric latitude, which is not commonly used. The ECEF coordinates of a location are commonly determined using GPS.

In this coordinate system the Jicamarca main array is located at [1417.539, -6078.202, -1311.862] as measured in kilometers. The local east unit vector can be determined from

the ECEF coordinates as

$$\hat{E} = \frac{\left[-Y, \quad X, \quad 0\right]}{\sqrt{X^2 + Y^2}}, \tag{5.5}$$

since east is tangent to any point on the the globe toward the earth's rotation. The true zenith (local up) unit vector at Jicamarca is determined by

$$\hat{H} = \left[\cos(\phi)\cos(\lambda), \quad \cos(\phi)\sin(\lambda), \quad \sin(\phi)\right], \tag{5.6}$$

where ϕ and λ are the *geodetic* latitude and longitude of the Jicamarca main array, 11.95^{o} S and 76.87^{o} W respectively taking into account the non-spherical shape of the earth. Then the local north unit vector is simply

$$\hat{N} = \hat{H} \times \hat{E}. \tag{5.7}$$

At Jicamarca there is are another set of unit vectors $\hat{x}$ and $\hat{y}$, that are defined by the $x-$ and $y-$axes of the square array. Additionally the the third direction, $\hat{z}$, is along the "on-axis" beam of the antenna. The on axis unit vector is calculated from the declination, δ, measured in degrees, and hour angle, τ, measured in minutes, of the antenna beam along with the longitude as

$$\hat{z} = \left[\cos(\delta)\cos(\tau + \lambda), \cos(\delta)\sin(\tau + \lambda), \cos(\delta)\right]. \tag{5.8}$$

For Jicamarca, the on axis declination is -12.88$^{\text{o}}$, the hour angle is -4.6167 minutes, and λ is 76.87^{o} as indicated above. The x-axis is roughly pointing in the south-east direction and is along the main JRO building, while the y-axis perpendicular to it, away from the building as shown in Figure 5.2. At Jicamarca, the plane of the array is tilted with the

axis of rotation along $\hat{x}$, leading to

$$\hat{x} = \frac{\hat{H} \times \hat{z}}{\left\| \hat{H} \times \hat{z} \right\|}, \tag{5.9}$$

The y unit vector is

$$\hat{y} = \hat{z} \times \hat{x}. \tag{5.10}$$

5.2 Antenna Beam Directions

The vector wind in the mesosphere can be calculated from three or four LOS velocities measured at a particular time interval and height range. Each channel of the MST radar data corresponds to a single beam from the radar experiment. The four beams each have a different pointing direction, which are needed in determining the wind's direction.

Table 5.1 – The antenna beam pointing directions for the 2009 experiment.

Beam	Θ_x	Θ_y
East	0.03229	0.01217
West	-0.03957	0.01267
North	0.01107	0.04833
South	-0.00482	-0.05114

The bean pointing directions given in Table 5.1 are described using "direction cosines"

$$\Theta_x = \frac{x}{\sqrt{x^2 + y^2 + z^2}}, \tag{5.11}$$

$$\Theta_y = \frac{x}{\sqrt{x^2 + y^2 + z^2}}, \tag{5.12}$$

and

$$\Theta_z = \sqrt{1 - \left(\Theta_x^2 + \Theta_y^2\right)}, \tag{5.13}$$

where (x, y, z) denotes the coordinates of beam boresight specified in the local coordinates system depicted in Figure 5.2. The direction cosines in Table 5.1 were determined by the beam pattern of the Jicamarca array, shown in Figure 4.12.

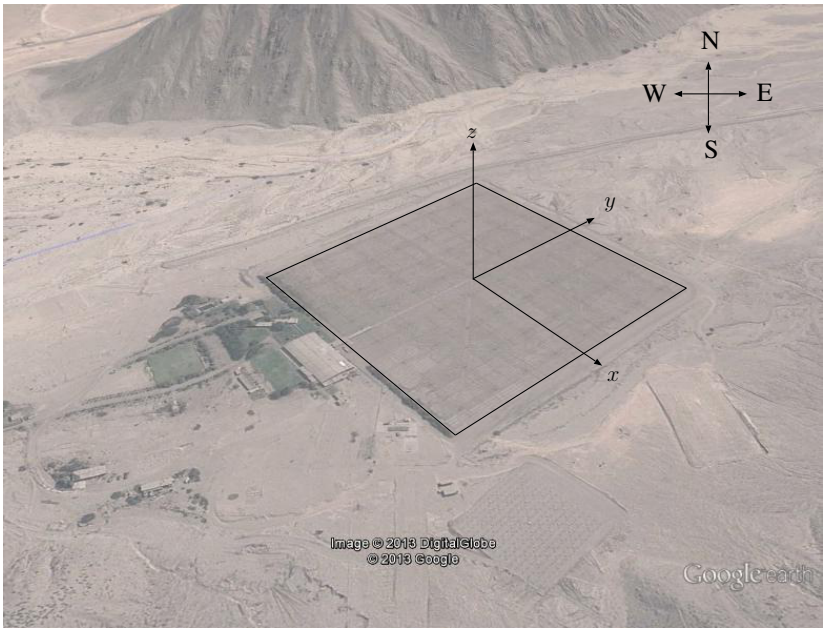


Figure 5.2 – The local JRO coordinate system, the compass directions are approximate.

5.3 Estimation of UVW

Consider a radar scattering volume centered at (x, y, z) in the mesosphere, detected with a radar beam with the direction cosines Θ_x, Θ_y, and Θ_z. Let the wind vector at the same location be denoted as

$$\bar{u} = U\hat{E} + V\hat{N} + W\hat{H}, \tag{5.14}$$

where U is zonal the component of the wind, V is the meridional component, and W is the vertical component while unit vectors $\hat{E}$, $\hat{H}$, and $\hat{N}$ are given by Equations (5.5), (5.6), and (5.7) respectively. Using the direction cosines of the antenna beam boresight and Jicamarca unit vectors, the beam vector

$$\hat{b} = \Theta_x \hat{x} + \Theta_y \hat{y} + \Theta_z \hat{z}, \tag{5.15}$$

gives the direction of a beam in ECEF coordinates. Then, the LOS velocity estimate provided by Doppler spectral analysis is

$$v = \bar{u} \bullet \hat{b} \tag{5.16}$$

for each beam. The k-th beam scalar LOS velocity can be expressed as

$$v_k = \alpha_k U + \beta_k V + \gamma_k W, \tag{5.17}$$

where

$$\alpha_k = \Theta_x \hat{x} \bullet \hat{E} + \Theta_y \hat{y} \bullet \hat{E} + \Theta_z \hat{z} \bullet \hat{E}, \tag{5.18}$$

$$\beta_k = \Theta_x \hat{x} \bullet \hat{N} + \Theta_y \hat{y} \bullet \hat{N} + \Theta_z \hat{z} \bullet \hat{N}, \tag{5.19}$$

and

$$\gamma_k = \Theta_x \hat{x} \bullet \hat{H} + \Theta_y \hat{y} \bullet \hat{H} + \Theta_z \hat{z} \bullet \hat{H}. \tag{5.20}$$

Then the k-th terms can be combined into column vectors to solve a system of equations for the three unknowns using

$$\begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \\ \alpha_4 & \beta_4 & \gamma_4 \end{bmatrix} \begin{bmatrix} U \\ V \\ W \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix}. \tag{5.21}$$

When there are four LOS velocities, as shown, a pseudoinverse or generalized inverse is needed to compute a least-squares solution of the system of equations [*Penrose and Todd*, 1956] to solve for (U, V, W). Sometimes there are only three LOS velocities available, then a simple matrix inverse can be utilized. When only one or two LOS velocities are available (typically due to sidelobe signals at the top of layers — see discussion in Chapter 4) wind estimation is not possible.

The python code for this procedure can be found in Appendix (C).

5.4 Propagation of Measurement Error

The measurement error, determined in Section 4.2, of the LOS Doppler velocity must be carried through for the zonal, meridional, and vertical wind calculations. According to *Wikström and Wedin* [2002], since the matrix in Equation 5.21 is assumed to unperturbed, and only the velocities have some error to be propagated, one only needs to replace the velocity with its measurement error and then carry out the calculation as before. In equation form, this appears as

$$\begin{bmatrix} \alpha_1 & \beta_1 & \gamma_1 \\ \alpha_2 & \beta_2 & \gamma_2 \\ \alpha_3 & \beta_3 & \gamma_3 \\ \alpha_4 & \beta_4 & \gamma_4 \end{bmatrix} \begin{bmatrix} \delta U \\ \delta V \\ \delta W \end{bmatrix} = \begin{bmatrix} \delta v_1 \\ \delta v_2 \\ \delta v_3 \\ \delta v_4 \end{bmatrix}. \tag{5.22}$$

Only if error in the estimation of the beam direction is to be considered, does this process become more complex.

5.5 Results

Figures 5.3 and 5.4 show the zonal, meridional, and vertical wind maps for two different days in 2006. Figure 5.5 shows the same for January 27th, 2009. The zonal and meridional winds are stronger than the vertical winds. In the zonal wind, the lower mesosphere tends to be in opposite direction of the upper mesosphere. This is most clearly shown in Figure 5.4. Unsurprisingly, the oscillations in the Doppler velocity also appears in all the wind directions.

The mean zonal and meridional winds at 71, 75, and 79 km, and mean zonal winds for 60 to 90 km for the data from 2005-2007, 2009, and 2014 are shown in Figures 5.6, 5.7, 5.8, respectively. Each data point is a single day. It is difficult to discern whether or not the semi-annual oscillation (SAO) is visible in the zonal winds for 2005-2007. The mean vertical wind stays rather constant near to zero for most of the data. The mean meridional wind is similar to mean zonal wind in velocity range, while the mean zonal wind stays much closer to zero.

Contamination from the sidelobes in the form of velocity gradients as discussed in Section 4.4, can sometimes affect the wind estimation. For example, the velocity gradients from the north and east beams in Figure 4.4 cause a slight wind shear feature in the zonal wind of Figure 5.3 from 10:00 to 13:00 above 75 km. This is clearly an artifact and not a true

shear. These artifacts in the winds can occur when the scattering from the sidelobes causes double Gaussians. If the sidelobe peak is stronger than the main lobe peak occurring at the same time and received height, it will cause a artifact in the LOS velocity, even though signal from the main lobe is still present. The artifacts could be detected by looking for a change in sign of the wind velocity near the top of layers for a single minute.

However, most such cases of double Gaussians and velocity gradients do not affect the zonal and meridional wind estimation. Typically only two beams have sidelobes that can cause velocity artifacts at a particular time and received height. These velocity artifacts from the sidelobes have a longer radar range and thus appear at an increased height compared to scattering from the main lobe. Since there is no scattering volume at that altitude, the remaining two beams will yield less than the three LOS velocities necessary to estimate wind for the given height.

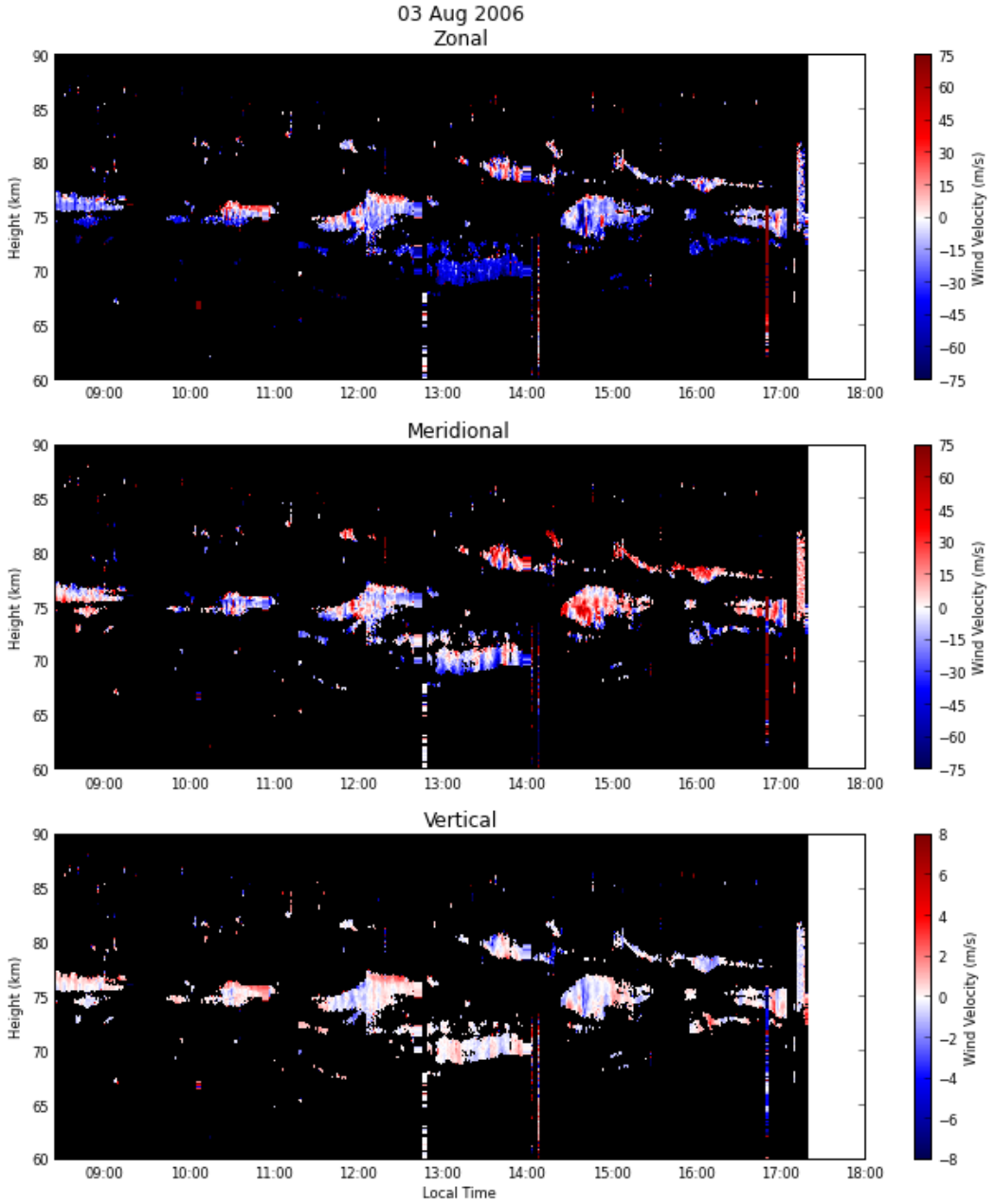


Figure 5.3 – The wind map for August 3, 2006, all three directions are shown.

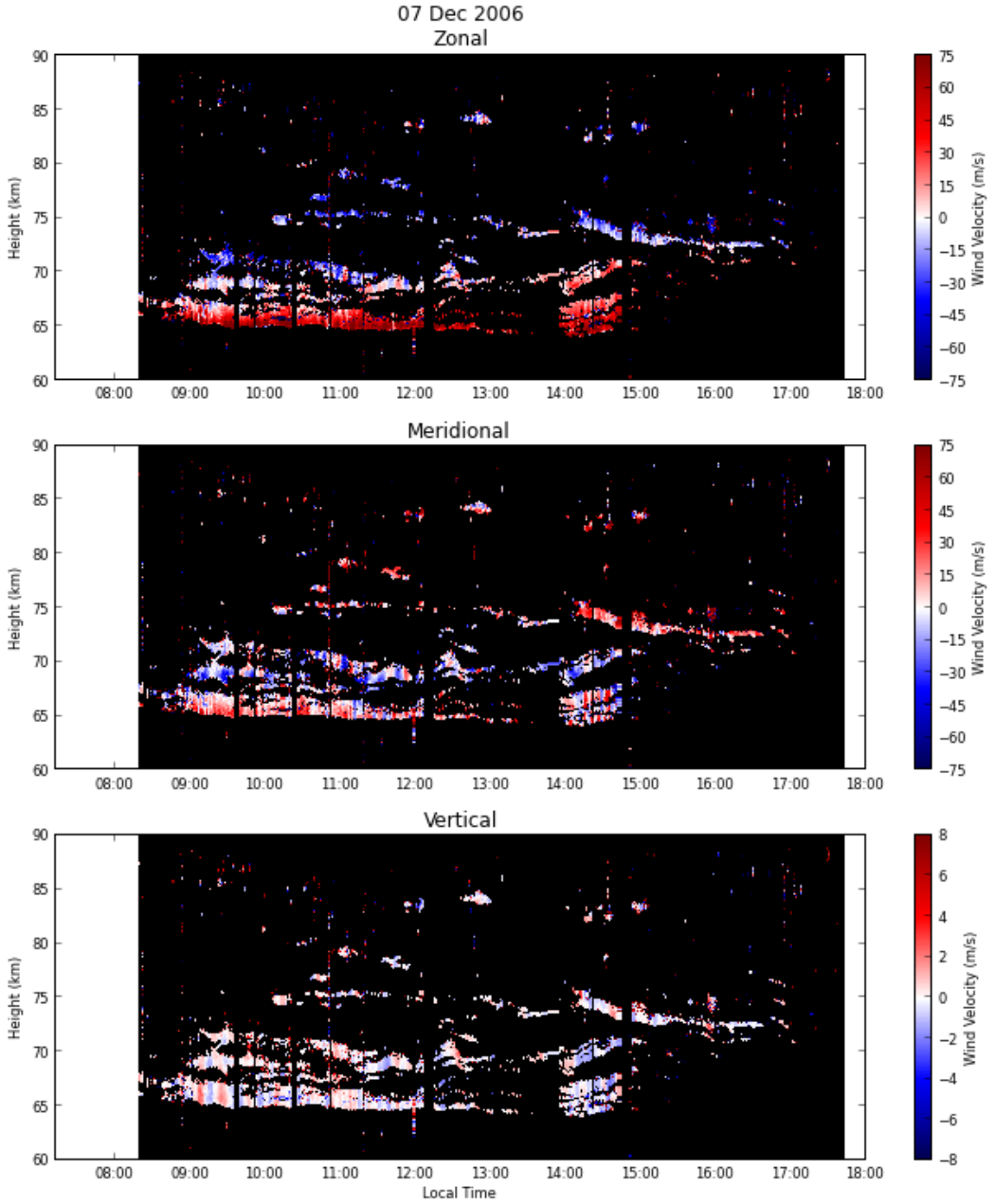


Figure 5.4 – The wind map for December 7, 2006, all three directions are shown.

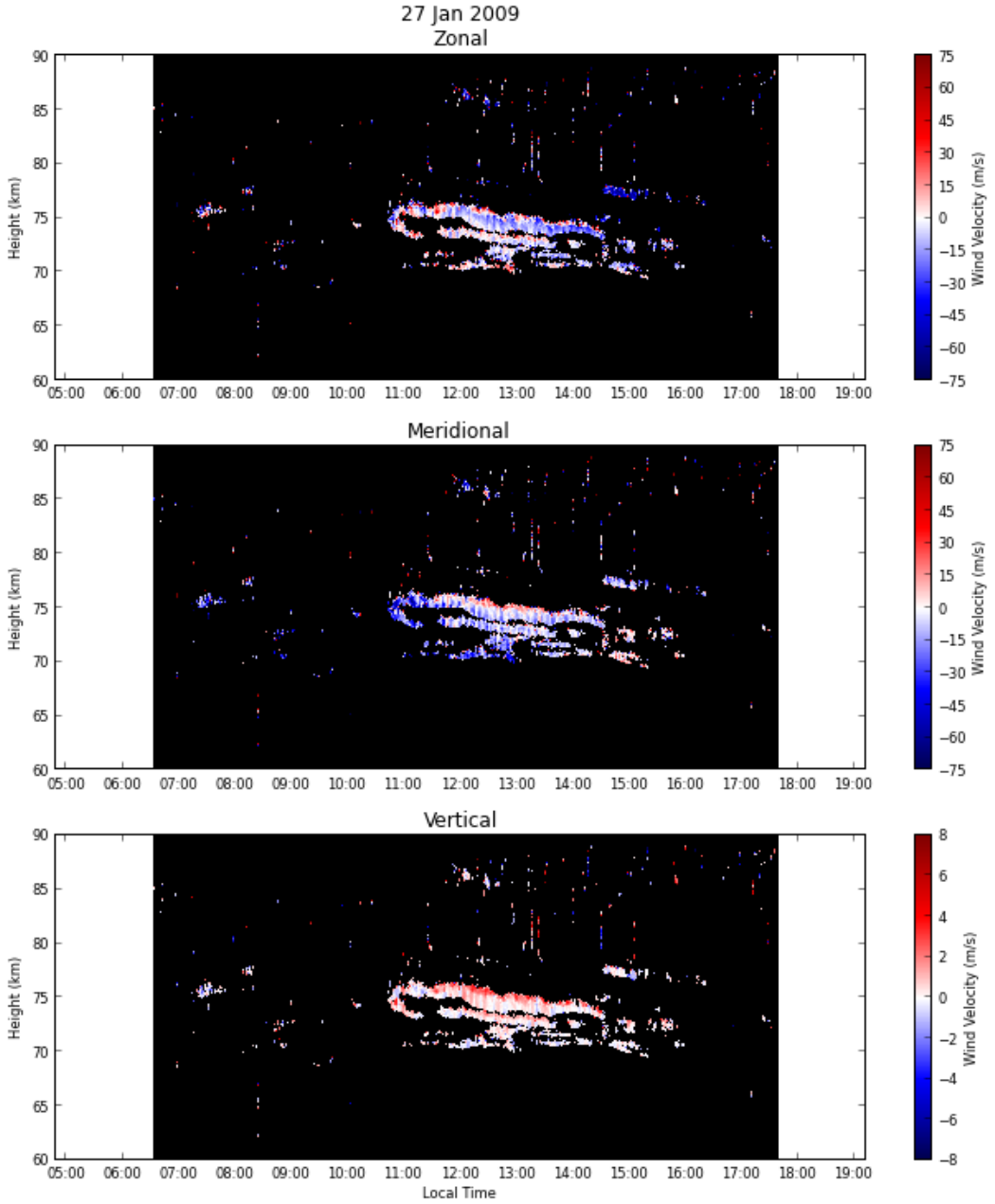


Figure 5.5 – The wind map for January 27, 2009, all three directions are shown.

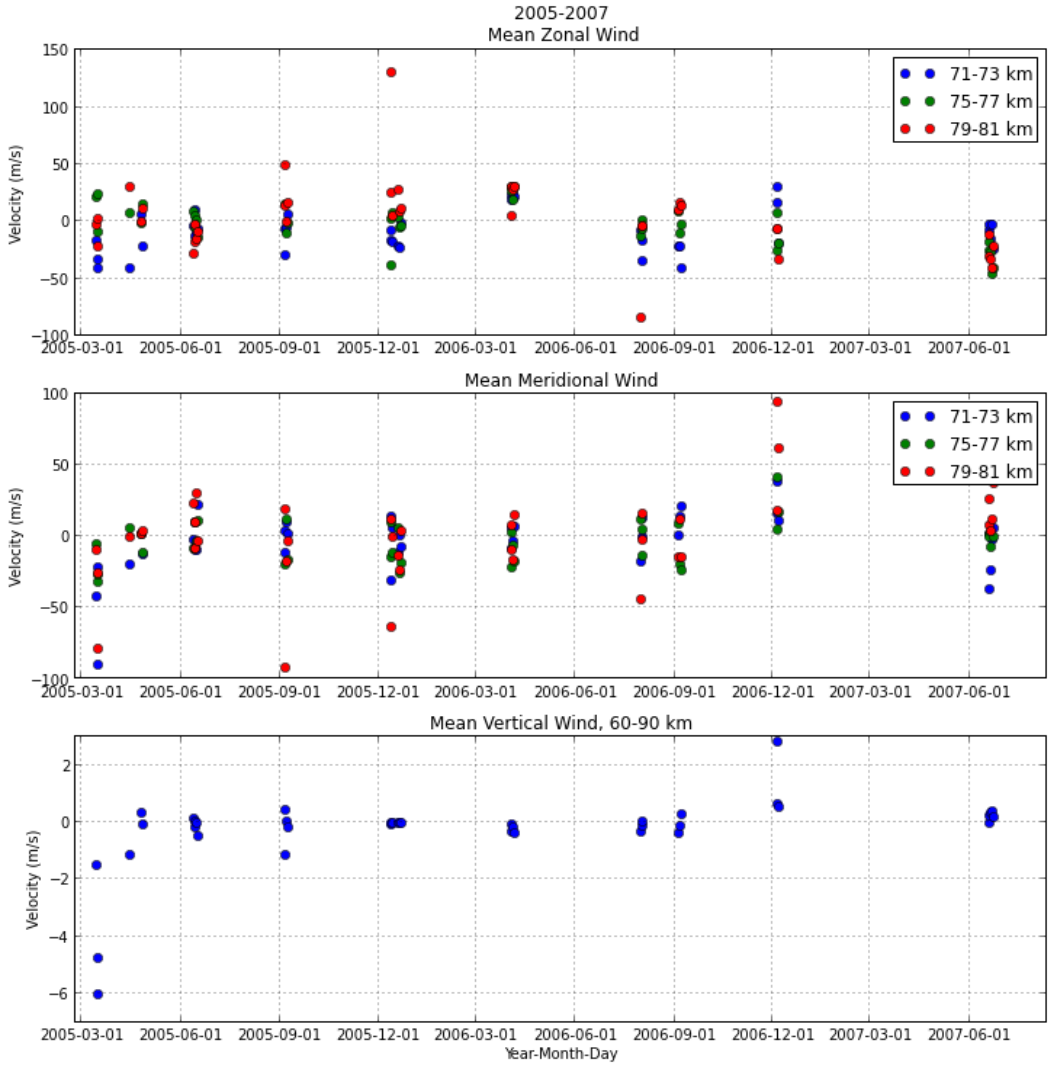


Figure 5.6 – The mean winds for the years 2005-2007, the mean zonal meridional winds are averaged over the specified heights. The mean vertical wind is averaged over all heights in the mesosphere.

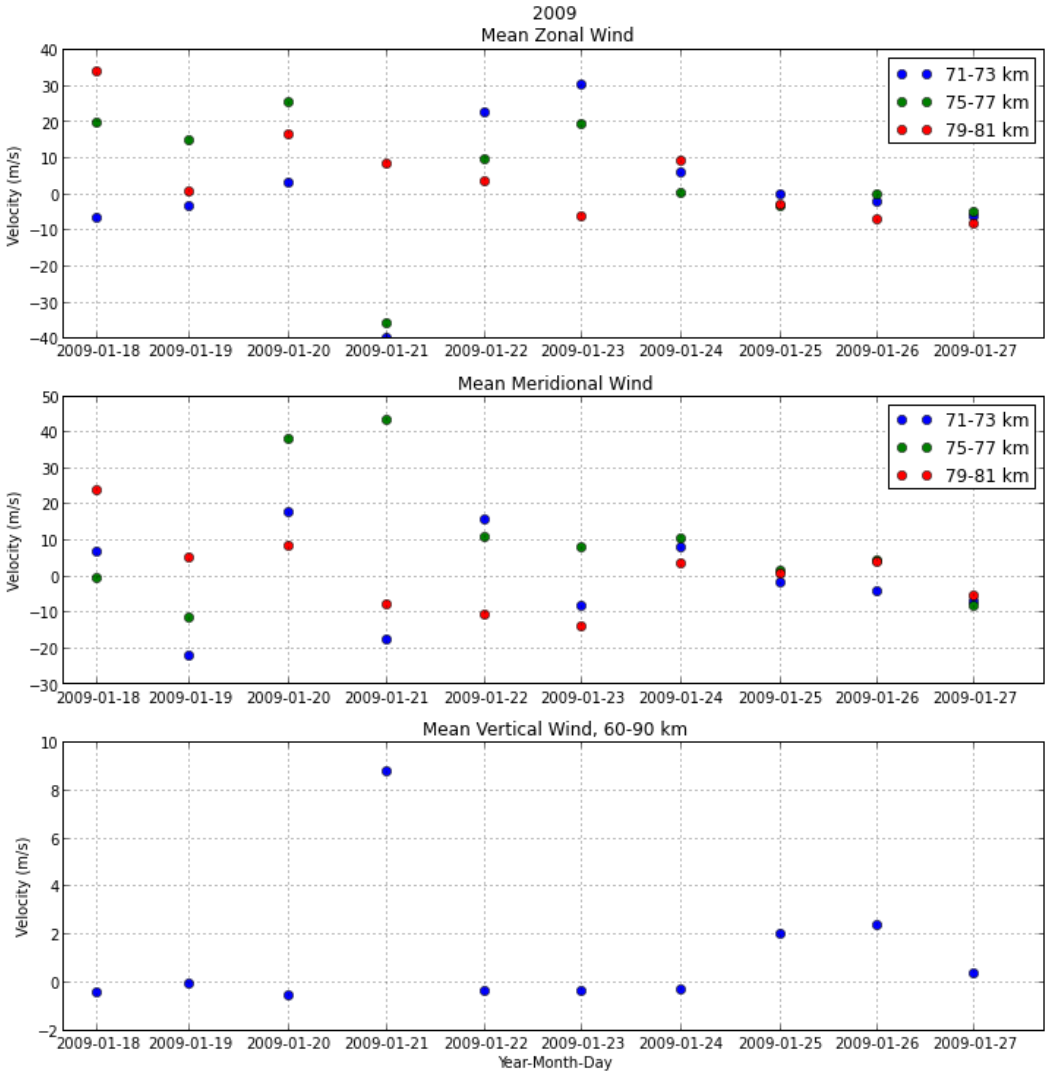


Figure 5.7 – The mean winds for 2009, the mean zonal meridional winds are averaged over the specified heights. The mean vertical wind is averaged over all heights in the mesosphere.

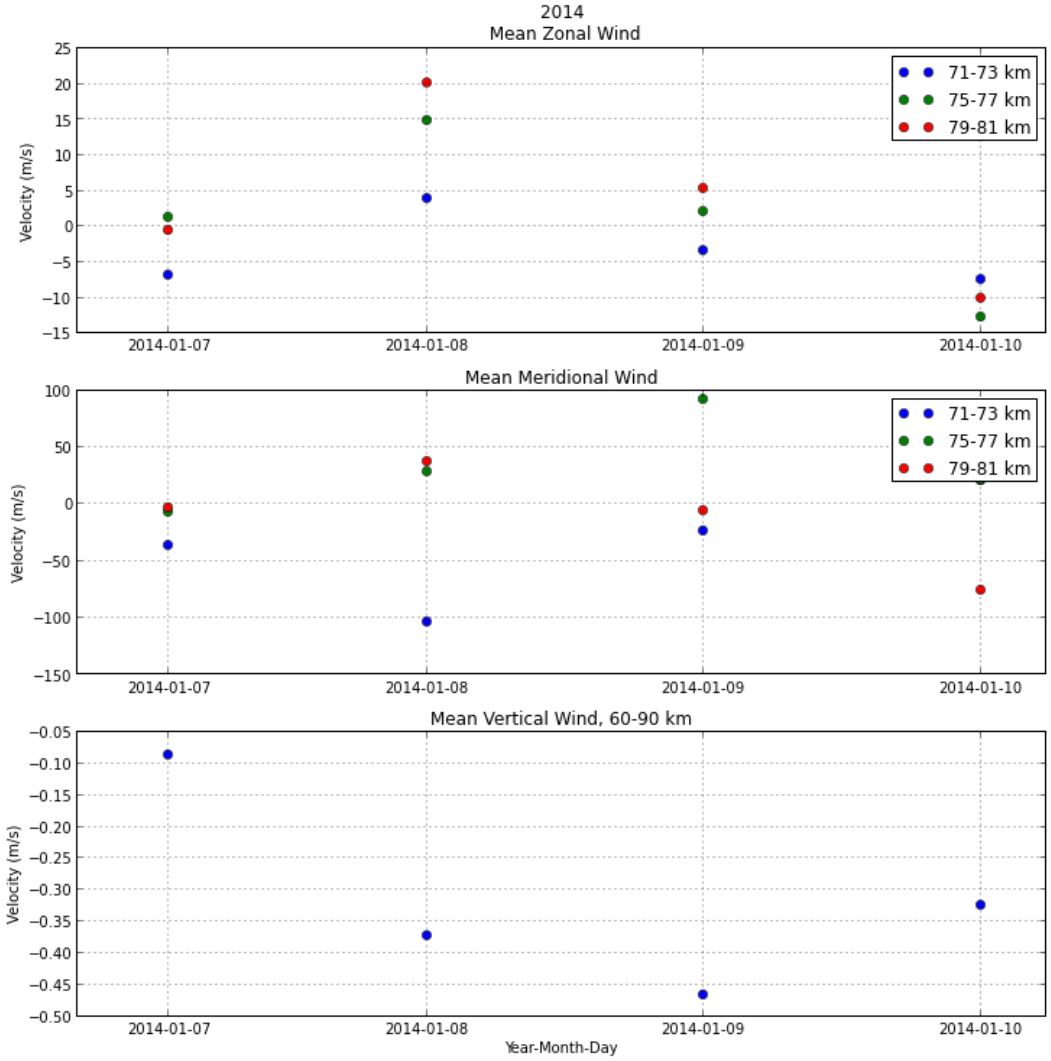


Figure 5.8 – The mean winds for 2014, the mean zonal meridional winds are averaged over the specified heights. The mean vertical wind is averaged over all heights in the mesosphere.

CHAPTER 6

CONCLUSION

To find the spectral parameters of the Jicamarca mesospheric radar data, this thesis built upon the generalized Gaussian spectra model introduced by *Sheth et al.* [2006]. In Chapter 4, a linear constraint on the generalized Gaussian parameter exponent, p, was introduced to limit the large variations observed in the parameter. Also an algorithm was created to find double Gaussians allowing a Gaussian model with two peaks to be used. The new model, including the occasional double Gaussian, was applied to a large data set that spans 5 years and over 50 days of data. By analyzing this data, the causes of the double Gaussians were investigated at the end of Chapter 4. It was found that double Gaussians are caused by contamination from sidelobes when there is a high enough SNR so that the signal is stronger than noise, the wind direction is such that there is a difference in the Doppler velocities in the main lobe and sidelobe, and the layer is thick enough that returns from the sidelobes are received at the same time as those from the main lobe. In Chapter 5, a new method was developed to calculate the zonal, meridional, and vertical wind components that can be applied to different antenna configurations. Using the data from the 2005-2007 experiments, the mean zonal wind was calculated and semi-annual oscillation was shown. Additionally, the double Gaussians from Chapter 4 were shown to sometimes cause winds shear artifacts.

The spectral parameters found in this thesis, in addition to using Doppler velocity to estimate wind, can be used in the analysis of the causes of the radar backscatter. The

spectral width along with returned signal power can be studied to determine whether the radar echo was caused by partial reflected or randomly scattered fields [*Fukao et al.*, 1980; *Sheth et al.*, 2006]. Partial reflections are typically caused by time-invariant density structures leading to a narrow spectral width, while turbulent scattered fields are caused by large deviations in the electron density gradients and lead to broader spectral widths and higher signal power. The resulting spectral parameters and wind estimates obtained with the methods described here will be uploaded to the CEDAR Archival Madrigal Database for the use of other researchers and scientists.

Future work on the mesospheric part of the MST-ISR experiments may include the analysis of spectral width and signal power correlations, as well as analysis of the ISR data. Additionally, as mentioned in Chapter 4, the algorithm to detect the double Gaussians can be improved as it often mistakes two overlapping Gaussians as one single Gaussian. Also, as future MST-ISR campaigns are completed, there will be more data to process and analyze.

APPENDIX A

MADRIGAL DATA STRUCTURE

A.1 Creating a Madrigal File

To upload data to Cedar's Madrigal database, the data needs to be formatted in a specific way. A python API is available and was provided on a CentOS virtual machine.

Table A.1 – Important specifications for creating a Madrigal experiment.

Abbreviation	Specification	
kinst	Instrument Identifier	For Jicamarca ISR: kinst=10
modexp	ID of the mode experiment	Not needed
kindat	ID of the data processing	Code range for Jicamarca: 1001-2000: Algorithms that primarily produces basic parameters 11001- 12000: Algorithms that primarily produce derived parameters
parcode	Parameter Code	Example: Neutral Winds East: 'vne' or 1410 Error parameter codes are the negative of the parameter code: -1410

Cedar supplies an examples and details on creating files at

http://cedar.openmadrigal.org/ad_createFiles.html.

The code below is loads the wind estimation data in a formatted file for uploading to Madrigal, important parameters are referenced in Table A.1

```
import os, os.path
```

```
import time
from time import gmtime
from calendar import timegm
import types
import datetime
import numpy as numpy

from glob import glob1

import madrigal.metadata
import madrigal.cedar

print 'Import Done'



def find_wind_file(yyyy,mm,dd):
    nyear=int(yyyy)
    basefolder = '/mnt/remote2/y%.4d/'%(nyear)
    #basefolder = '/rdata/radar/MSTISR/MST_Maps/y%.4d/'%(nyear)
    fnam=yyyy+'.'+mm+'.'+dd+'.*'
    search_pattern = 'windmap_1min_'+fnam
    flist = glob1(basefolder,search_pattern)
    print flist
    # When testing out of the server:
    flist.sort()
    return basefolder, flist

def read_wind(yyyy,mm,dd):
    mapdir,mapfile = find_wind_file(yyyy,mm,dd)
    rfile = numpy.load(mapdir + mapfile[0])
    finf = rfile['finf'][0]
    UVW = rfile['WindsUVW']
    acqtime_arr=rfile['acqtime_arr']
    d_UVW=rfile['delta_Winds']
    return finf, UVW, d_UVW, acqtime_arr

def Save_Data_Madrigal_Winds(yyyy,mm,dd):
    kinst = 10 # instrument identifier of Millstone Hill ISR
    modexp = None # id of mode of experiment
    kindat = 1602# id of kind of data processing
    #nrow = 6 # all data records have 5 2D rows
    finf,UVW, d_UVW, acqtime_arr= read_wind(yyyy,mm,dd)
    li=numpy.where(finf.hrange==60.)
```

```
ui=numpy.where(finf.hrange==90.)
try:
    rangei=range(li,ui)
except:
    rangei=range(330,540)
    li=330
nrow = len(rangei) # all data records have n 2D rows
print nrow
print numpy.size(acqtime_arr)

startLT=gmtime(acqtime_arr[0])
startTime = datetime.datetime(startLT.tm_year, startLT.tm_mon
   , startLT.tm_mday, startLT.tm_hour,    startLT.tm_min,
   startLT.tm_sec, 0)
recTime = datetime.timedelta(0,60)
endTime = startTime + recTime

newFile = '/home/geodatos/MSTdata/jro'+time.strftime('%Y%m%d'
   ,finf.LTarr0)+'.001'

# create a new Madrigal file
cedarObj = madrigal.cedar.MadrigalCedarFile(newFile, True)

for i in range(numpy.size(acqtime_arr)):
    startLT=gmtime(acqtime_arr[i])
    startTime = datetime.datetime(startLT.tm_year, startLT.
       tm_mon, startLT.tm_mday, startLT.tm_hour, startLT.
       tm_min, startLT.tm_sec, 0)
    recTime = datetime.timedelta(0,60)
    endTime = startTime + recTime

    dataRec = madrigal.cedar.MadrigalDataRecord(kinst,
                                                kindat,
                                                startTime.
                                                   year,
                                                startTime.
                                                   month,
                                                startTime.day
                                                   ,
                                                startTime.
                                                   hour,
                                                startTime.
                                                   minute,
                                                startTime.
```

```
                                                    second,
                                                startTime.
                                                    microsecond
                                                    /10000,
                                                endTime.year,
                                                endTime.month
                                                    ,
                                                endTime.day,
                                                endTime.hour,
                                                endTime.
                                                    minute,
                                                endTime.
                                                    second,
                                                endTime.
                                                    microsecond
                                                    /10000,
                                                ('gdlatr', '
                                                    gdlonr'),
                                                ('range'
                                                    ,1410,
                                                    1420,
                                                    1430,
                                                    -1410,
                                                    -1420,-1430)
                                                    ,
                                                nrow)
        # set 1d values
        dataRec.set1D('gdlatr', -11.95)
        dataRec.set1D('gdlonr', -76.87)
        for h in range(nrow):
            dataRec.set2D('range', h, finf.hrange[h+li])
            try:
                dataRec.set2D(1410, h, UVW[i,0,h+li])
            except:
                dataRec.set2D(1410, h,'missing')
            try:
                dataRec.set2D(1420, h, UVW[i,1,h+li])
            except:
                dataRec.set2D(1420, h, 'missing')
            try:
                dataRec.set2D(1430, h, UVW[i,2,h+li])
            except:
                dataRec.set2D(1430, h, 'missing')
            if not(d_UVW[i,0,h+li]<0.01) and not(numpy.isnan(
```

```
            d_UVW[i,0,h+li])):
                dataRec.set2D(-1410, h, d_UVW[i,0,h+li])
            else:
                dataRec.set2D(-1410, h, 'missing')
            if not(d_UVW[i,1,h+li]<0.01) and not(numpy.isnan(
              d_UVW[i,1,h+li])):
                dataRec.set2D(-1420, h, d_UVW[i,1,h+li])
            else:
                dataRec.set2D(-1420, h, 'missing')
            if not(d_UVW[i,2,h+li]<0.01) and not(numpy.isnan(
              d_UVW[i,2,h+li])):
                dataRec.set2D(-1430, h, d_UVW[i,2,h+li])
            else:
                dataRec.set2D(-1430, h, 'missing')
        # append new data record
        cedarObj.append(dataRec)

# write new file
cedarObj.write()

# next, use the cedar.CatalogHeaderCreator class to add
    catalog and header
catHeadObj = madrigal.cedar.CatalogHeaderCreator(newFile)
catHeadObj.createCatalog(principleInvestigator="Erhan␣Kudeki,
    ␣Gerald␣Lehmacher", sciRemarks="Test␣data␣only␣-␣do␣not␣
    use")
catHeadObj.createHeader(analyst="Jennifer␣Smith", comments="␣
    ")
catHeadObj.createCatalog(expPurpose="Collection␣of␣
    atmospheric␣and␣ionospheric␣backscatter␣with␣the␣Jicamarca
    ␣Radio␣Observatory␣to␣study␣winds,␣turbulence␣in␣
    mesosphere,␣plasma␣instabilities␣in␣the␣electrojet␣region,
    ␣plasma␣drifts␣and␣instabilities␣in␣the␣150-km␣region,␣and
    ␣incoherent␣backscatter␣and␣instabilities␣in␣the␣F␣region␣
    simultaneously␣")
catHeadObj.createCatalog(expMode="Interleaved␣64-baud␣
    complementary␣code␣(MST),␣3-baud␣Barker␣code␣(ISR),␣and␣
    uncoded␣pulses␣(EEJ).␣␣␣For␣details␣see:␣\n␣Akgiray,␣A.␣␣M
    .␣S.␣Thesis,␣2007.␣University␣of␣Illinois,␣Urbana-
    Champaign␣\n␣Smith,␣J.,␣␣M.␣S.␣Thesis,␣2014.␣University␣of
    ␣Illinois,␣Urbana-Champaign")
catHeadObj.createCatalog(correlativeExp="Ionosonde")
catHeadObj.write()
```

```
    print 'done'
    return

Save_Data_Madrigal_Winds('2009','01','26')
```

A.1.1 Creating a New Parameter Code

If an existing parameter code does not properly describe the parameter to be saved, a new parameter code can be added to the parcods.tab file. Existing parameters can be found in http://cedar.openmadrigal.org/cedarFormat.pdf and in the parcods.tab file. Each new parameter will need to have a unique integer identification number and mnemonic.

A.2 Creating an Experiment in Madrigal

From a file or directory of files, create an experiment with only one command from the command line as: `/usr/local/www/cgi-bin/madrigal/createExpWithDir.py --madPath=/home/geodatos/MSTdata --expTitle="MST Winds" --permission=0 --fileDesc=""`.

For more information see: http://cedar.openmadrigal.org/ad_createExp.html.

An experiment will be created at: /usr/local/madrigal/experiments/ .

Then, to view the data on the web, run: UpdateMaster.

A.2.1 Adding a New Experiment

If the experiment had not been created in Madrigal before, the file expPlot.txt will need to be edited to create the new experiment. There is an example on how to do this on the top of the file.

A.3 Adding and Replacing Files an in Experiment

Files can be added to the experiment using addFileToExp.py and files already in the experiment can be changed/replaced using updateFileInExp.py. For more information see: http://cedar.openmadrigal.org/ad_createExp.html#addFileToExp.

APPENDIX B

FITTING AND DOUBLE GAUSSIAN DETECTION CODE

The following code contains definitions that are used by the main fitting function. It uses the following standard python modules: numpy, pylab, sys, os, scipy, time, and calendar. There are two nonstandard modules, leastsqbound, which is available at https://github.com/jjhelmus/leastsqbound-scipy, and the uncertainties package (http://pythonhosted.org/uncertainties/).

```
import time
import calendar
from pylab import *
import pylab as py
import numpy
import sys,os
from scipy.optimize import leastsq
from leastsqbound import leastsqbound
import uncertainties as u
from uncertainties import unumpy

def peval(x,p):#the Generalized Gaussian model
    return p[1]*py.exp(-abs((x-p[2])/p[3])**p[4])+p[0]

def doub_peval(x,p):# the Generalized Gaussian model for two
   Gaussian
    return p[1]*py.exp(-abs((x-p[3])/(p[5]))**(p[7]))+p[2]*py.
       exp(-abs((x-p[4])/(p[6]))**(p[8]))+p[0]

def both_combined(p,x,n,nGauss):#combines models for group
   fitting
    l=0
```

```
    pev=py.zeros([n,64])
    for j in range(0,n):#each height to be fitted
        gpow=p[-3]+p[-2]*(j-n/2.0)+p[-1]*(j-n/2.0)**2
        if nGauss[j]==1:#one peak
            if py.any(py.greater_equal(nGauss,2)):
                gpow2=p[-6]+p[-5]*(j-n/2.0)+p[-4]*(j-n/2.0)**2
            v=p[l+2]
            if 'vl' in locals():
                if abs(v-vl)>=abs(v-vr):
                    gpow_s=gpow
                else:
                    gpow_s=gpow2
            else:
                gpow_s=gpow
            pg=[p[l],p[l+1],v,p[l+3],gpow_s]#initial guess
            pev[j,:]=peval(x,pg)
            l+=4
        if nGauss[j]==2:#two peaks
            gpow2=p[-6]+p[-5]*(j-n/2.0)+p[-4]*(j-n/2.0)**2
            vl=p[l+3]
            vr=p[l+4]
            if 'v' in locals():
                if abs(v-vl)>=abs(v-vr): #preceding gaussian on
                   the left
                    gpowl=gpow
                    gpowr=gpow2
                else:
                    gpowl=gpow2
                    gpowr=gpow
            else:
                gpowl=gpow
                gpowr=gpow2

            pg=[p[l],p[l+1],p[l+2],vl,vr,p[l+5],p[l+6],gpowl,
               gpowr]#initaial guess
            pev[j,:]=doub_peval(x,pg)#peval
            l+=7
    pev=py.reshape(pev,n*64)
    return pev

def residuals_log(p,y,x,n,nGauss): #for groups of single and
   double Gaussians
    err=py.log(py.array(y))-py.log(py.array(both_combined(p,x,n,
       nGauss)))
```

```
    return err

def residuals_log_DGs(p,y,x): #for single double Gaussian
    err=py.log(py.array(y))-py.log(py.array(doub_peval(x,p)))
    return err

def makeJacobian(p,x,l,n):#creates the Jacobian
    gpow=p[-3]+p[-2]*(l-n/2.0)+p[-1]*(l-n/2.0)**2
    JN=py.ones(py.shape(x))
    JA=py.exp(-abs((x-p[2])/p[3])**gpow)
    Jv=p[1]*JA*gpow*abs((x-p[2])/p[3])**gpow*1/(x-p[2])
    Jw=p[1]*JA*gpow*(abs(x-p[2])/p[3])**gpow*(1/p[3])
    Jp=-p[1]*JA*(abs(x-p[2])/p[3])**gpow*log(abs(x-p[2])/p[3])
    Jp1=-(l-n/2.0)*p[1]*JA*(abs(x-p[2])/p[3])**gpow*log(abs(x-p
        [2])/p[3])
    Jp2=-(l-n/2.0)**2.*p[1]*JA*(abs(x-p[2])/p[3])**gpow*log(abs(x
        -p[2])/p[3])
    J=py.array([JN,JA,Jv,Jw,Jp,Jp1,Jp2])*1/(p[1]*JA+p[0])
    return J

def CalcError(p,x,h,n):#calculates error using Jacobian
    Jac=makeJacobian(p,x,h,n)
    JT=Jac.transpose()
    if not(any(isnan(JT))):
        cov=pinv(dot(Jac,JT))#calculate covariance matrix
    else:
        cov=(empty([7,7]))*NaN
    return py.sqrt(py.diag(cov))

def determineInitialGuess(x,y,noisef,N):#determine initial sigma
    for single peak
    vt=sum(x*(y))/sum(y)
    std_=py.sqrt(abs(sum(x**2*(y-noisef))/sum(y-noisef)-vt**2))
    stdn=EstNoiseStd(y-noisef,N)
    if py.sum(py.greater(y-noisef,4.0*stdn))<5:#for narrow peaks
        std_=3.0
    if std_<0.8:#check lower bound
        std_=0.81
    if std_>x.max():#check upper bound
        std_=x.max()-3
    return std_

def CalcSpectralWidth(x,fp,Err):#calculate spectral width and
    error propagation
```

```
    A=u.ufloat(fp[1],Err[1])
    V=u.ufloat(fp[2],Err[2])
    S=u.ufloat(fp[3],Err[3])
    P=u.ufloat(fp[4],Err[4])
    xu=unumpy.uarray(x,ones(size(x)))
    fit_err=A*unumpy.exp(-abs((xu-V)/S)**P)
    Pow=sum(fit_err)
    sw2=1/Pow*sum((xu-V)**2*fit_err)
    sw=sum(unumpy.sqrt(sw2))
    return sw.nominal_value, sw.std_dev

def sub_detectGaussians(x,y,noisef,N):#detects
    sig=y-noisef #subtract noise
    peak=sig.max() #peak level
    peak_index=sig.argmax() #peak location
    A,A2=py.NaN,py.NaN
    v,v2=py.NaN,py.NaN
    std,std2=py.NaN,py.NaN
    stdn=EstNoiseStd(sig,N) #estimate noise standard deviation
    if stdn!=0:
        threshold1=max(4.5*stdn,0.2*peak)
    else:
        threshold1=0.1*peak
    nGauss=1
    SNR = 10*py.log10(py.mean(y)/noisef-1)
    spcn=sig/peak
    b=sum(py.absolute([q - spcn[c - 1] for c, q in enumerate(
       spcn)][1:]))
    flag=None
    A=0.8*peak
    v=sum(x*y)/sum(y)
    if peak>4*stdn and b<10:
        list1 = py.nonzero(range(0,N)*py.rint(sig>threshold1))[0]
           #cut out data below threshold
        if peak_index==0:
            list1=py.insert(list1,0,0)
        group_1=py.array_split(list1,py.where(py.diff(list1,)> 2)
           [0]+1)#split in sublist
        len_g=[]
        if py.shape(group_1)[0]>1 and SNR>-4:
            for i in range(0,py.shape(group_1)[0]):
                group_1[i]=list(group_1[i])
                len_g+=[py.size(group_1[i])]
            q = [q for q in group_1 if peak_index in q][0]
```

```
        main_gaussian=group_1[group_1.index(q)]
        v=sum(x[main_gaussian]*(sig[main_gaussian]))/sum(sig[
            main_gaussian])
        group_1.remove(main_gaussian)#remove main peak
        len_g.remove(py.size(main_gaussian))
        std=py.size(main_gaussian)/2.0*x.max()/(N/2.0)
        temp=sig
        temp[main_gaussian]=0
        if py.any(py.greater_equal(len_g,4)):
            nGauss=2
            A2=0.8*temp.max()
            ind=py.argmax(len_g)
            v2=sum(x[group_1[ind]]*(sig[group_1[ind]]))/sum(
                sig[group_1[ind]])
            std2=py.size(sig[group_1[ind]])/2.0*x.max()/(N
                /2.0)
        elif abs(x[peak_index]-v)>=x.max()/5.0 and py.size(
            main_gaussian)>=5:#skew-
            nGauss=2
            v2=v
            v=x[peak_index]
            A2=0.6*A
            flag=1
            std2=(x[peak_index]-v)
            std=std2
        elif temp.max()>8*stdn:
            nGauss=2
            A2=0.8*temp.max()
            ind=py.argmax(len_g)
            v2=sum(x[group_1[ind]]*(sig[group_1[ind]]))/sum(
                sig[group_1[ind]])
            std2=2
            flag=1
        else:
            nGauss=1.0
    elif py.size(group_1[0])>=2: #one peak
        nGauss=1
        group_1[0]=list(group_1[0])
        v=sum(x[group_1[0]]*(sig[group_1[0]]))/sum(sig[
            group_1[0]])
        std=py.size(group_1[0])/2*x.max()/(N/2.0)
        if abs(x[peak_index]-v)>=x.max()/5.0 and py.size(
            group_1[0])>=5:
            nGuass=2
```

```
                v2=v
                v=x[peak_index]
                A2=0.6*A
                std2=(x[peak_index]-v)
                std=std2
                flag=1
        else: #one skinny large peak
            nGauss=1
            std=2
            v=x[peak_index]
    else:
        nGauss=1
    return nGauss,A,v,A2,v2,std,std2,flag

def detectGaussians(x,y,noisef,N):
    nGuass=1
    A,Ar,A2=py.NaN,py.NaN,py.NaN
    v,vr,v2=py.NaN,py.NaN,py.NaN
    std,stdr,std2=py.NaN,py.NaN,py.NaN
    nGauss,A,v,A2,v2,std,std2,flag=sub_detectGaussians(x,y,noisef
        ,N)
    if not(py.isnan(v2)):#when there are two gaussians
        if v<=v2:
            vr=v2
            Ar=A2
            stdr=std2
        else:
            vr=v
            v=v2
            Ar=A
            A=A2
            stdr=std
            std=std2
    if nGauss==2 and flag!=None:
        #stdl,stdr=DoublnitialGuess(x,y,noisef,N)
        p0=[noisef,A,Ar,v,vr,std,stdr,1.8,1.8]
        p_fit=leastsq(residuals_log_DGs,p0,args=(y,x))
        nGauss,A2,v2,Ar2,vr2,std2,stdr2,flag=sub_detectGaussians(
            x,doub_peval(x,p_fit[0]),p_fit[0][0],N)
    return nGauss,A,v,Ar,vr,std,stdr

def EstNoiseStd(spc_noise,N):#Estimate noise standard deviation
    c=[]
    ln=min(spc_noise)
```

```
    for k in range(0,N):
        if spc_noise[k]>abs(ln):
            c+=[k]
    spc_noise_s=py.delete(spc_noise,c)
    stdd=py.std(spc_noise_s)
    return stdd

def CheckNoiseLevel(noise,spc):#checks the noise level is
    accurate
    noise_c=mean(spc)
    if noise_c>(noise+100):
        return noise_c
    else:
        return noise
```

The following is the main fitting function that calls all the preceding definitions, some of the code for loading and saving data is not shown. Comments are placed to show where data is loaded and saved.

```
def save_fits_and_vel_map(yyyy,mm,dd):#main fitting code

#Not Shown: loading 1-min integrated spectra file (finf,spc,noise
    )

    x = finf.vel_arr
    plsqs=array(zeros([finf.num_chan,finf.num_hei,finf.nFFT]))
    #plsqs1=array(zeros([finf.num_chan,finf.num_hei,finf.nFFT]))
    fitparams=(empty([finf.num_chan,finf.num_hei,5]))*NaN
    fitparams_2ndG=py.array(py.zeros([finf.num_chan,finf.num_hei
        ,5]))*NaN
    dp=(empty([finf.num_chan,finf.num_hei,7]))*NaN

    chisq=array(zeros([finf.num_chan,finf.num_hei]))*NaN
    swidth=array(zeros([finf.num_chan,finf.num_hei])*NaN)
    del_swidth=array(zeros([finf.num_chan,finf.num_hei])*NaN)
    fit_noise=array(zeros([finf.num_chan,finf.num_hei])*NaN)

    npeaks=py.array(py.zeros([finf.num_chan,finf.num_hei]))*NaN
    A=py.array(py.zeros([finf.num_chan,finf.num_hei]))
    v=py.array(py.zeros([finf.num_chan,finf.num_hei]))
    std=py.array(py.zeros([finf.num_chan,finf.num_hei]))
    Ar=py.array(py.zeros([finf.num_chan,finf.num_hei]))
```

```
vr=py.array(py.zeros([finf.num_chan,finf.num_hei]))
stdr=py.array(py.zeros([finf.num_chan,finf.num_hei]))
li=py.find(finf.hrange==60.)
ui=py.find(finf.hrange==90.)
try:
    rangei=range(li,ui)
except:
    rangei=range(335,538)
    ui=538
warnings.filterwarnings("ignore")

for ch in range(finf.num_chan):
    fit_heights=[]
    argument=spc.mean(2)[ch]/noise[ch]-1
    if py.any(argument<=0):
            b=py.find(argument<=0)
            argument[b]=py.NaN
    SNR = py.around(10*py.log10(argument),decimals =3)
    noise1=CheckNoiseLevel(noise[ch],spc[ch,335:337,:])
    noisef=noise[ch]/(finf.nFFT*finf.num_intg)
    for j in rangei:
        spc_noise=spc[ch,j,:]-noise1
        stdd=EstNoiseStd(spc_noise,finf.nFFT)
        if (spc_noise.max()>4*stdd and SNR[j]>-15): #do
            fittings
             fit_heights+=[j]
             #npeaks[ch,j],A[ch,j],v[ch,j]=detectGaussians(x,
                 spc[ch,j,:],noise1,3,finf.nFFT)

    group_heights = py.array_split(py.array(fit_heights),py.
       where(py.diff(py.array(fit_heights),)!=1)[0]+1)#
       seperates array into sequential groups
    num_g=py.size(group_heights)
    nb=(noisef-0.3, noisef+0.4)
    wb=(0.1,x.max())
    vb=(x.min(),x.max())
    pb=(0,8)
    if num_g<100 and num_g != 0:
        if py.size(py.shape(group_heights)) != 1:
            num_g=1
        for k in range(0,num_g+100):  # for each group
            if k>=num_g:
                break
            n=py.size(group_heights[k])  #number of
```

```
    consecative heights
if n>20 and n<70 and not(py.any(py.
    greater_equal(group_heights[k],ui-5))):
     num_g+=1
     group_heights.extend(array([group_heights[k
         ][14:]]))
     group_heights[k]=group_heights[k][0:14]
     n=py.size(group_heights[k])
if n != 1 and n<70 and not(py.any(py.
    greater_equal(group_heights[k],ui-5))): #check
     that it is longer than one
     p0=[]
     bounds=[]
     y_datac=zeros([n,finf.nFFT])
     for h in range(0,n): #for each height
          hi=group_heights[k][h]
          y_data=spc[ch,hi,:]/(finf.nFFT*finf.
              num_intg)
          y_datac[h,:]=y_data
          npeaks[ch,hi],A[ch,hi],v[ch,hi],Ar[ch,hi
              ],vr[ch,hi],std[ch,hi],stdr[ch,hi]=
              detectGaussians(x,y_data,noisef,finf.
              nFFT)
          if npeaks[ch,hi]==1:
               if isnan(std[ch,hi]):
                    std_=determineInitialGuess(x,
                        y_data,noisef,finf.nFFT)
               else:
                    std_=std[ch,hi]
               p0.append(noisef)
               p0.append(A[ch,hi])
               p0.append(v[ch,hi])
               p0.append(std_)
               Ab=(0.01, 10.0*A[ch,hi])
               bounds.append(nb)
               bounds.append(Ab)
               bounds.append(vb)
               bounds.append(wb)
          if npeaks[ch,hi]==2:
               bounds.append(nb)
               p0.append(noisef)
               p0.append(A[ch,hi])
               p0.append(Ar[ch,hi])
               p0.append(v[ch,hi])
```

```
            p0.append(vr[ch,hi])
            Abl=(0.01, 10.0*A[ch,hi])
            Abr=(0.01, 10.0*Ar[ch,hi])
            bounds.append(Abl)
            bounds.append(Abr)
            bounds.append(vb)
            bounds.append(vb)
            #stdl,stdr=DoubInitialGuess(x,y_data,
                noisef,finf.nFFT)
            p0.append(std[ch,hi])
            p0.append(stdr[ch,hi])
            bounds.append(wb)
            bounds.append(wb)
    if py.any(npeaks[ch,list(group_heights[k])
       ]>=2):
        p0.append(2)
        bounds.append(pb)
        p0.append(0.1)
        bounds.append((-5.0,10))
        p0.append(0.01)
        bounds.append((-2.0,10))
    p0.append(2)
    bounds.append(pb)
    p0.append(0.01)
    bounds.append((-5.0,10))
    p0.append(0.01)
    bounds.append((-2.0,10))
    ydatac=py.reshape(y_datac,n*finf.nFFT)

    plsq= leastsqbound(residuals_log,p0,args=(
       ydatac,x,n,npeaks[ch,list(group_heights[k
       ])]),bounds=bounds)

    fit_lines=both_combined(plsq[0],x,n,npeaks[ch
       ,list(group_heights[k])])
    fit_lines=py.reshape(fit_lines,[n,finf.nFFT])
    res_err=residuals_log(plsq[0],ydatac,x,n,
       npeaks[ch,list(group_heights[k])])
    res_err=py.reshape(res_err,[n,finf.nFFT])
    l=0
    for h in range(0,n):
        a=group_heights[k][h]
        gpow=plsq[0][-3]+plsq[0][-2]*(h-n/2.0)+
            plsq[0][-1]*(h-n/2.0)**2.0
```

```
try:
    gpow2=plsq[0][-6]+plsq[0][-5]*(h-n
        /2.0)+plsq[0][-4]*(h-n/2.0)**2.0
except:
    gpow2=NaN
if npeaks[ch,a]==1:
    v_s=plsq[0][l+2]
    if 'v_dl' in locals():
        if abs(v_s-v_dl)>=abs(v_s-v_dr):
            gpow_s=gpow
            p_fit=[plsq[0][l],plsq[0][l
                +1],v_s,plsq[0][l+3],plsq
                [0][-3],plsq[0][-2],plsq
                [0][-1]]
        else:
            gpow_s=gpow2
            p_fit=[plsq[0][l],plsq[0][l
                +1],v_s,plsq[0][l+3],plsq
                [0][-6],plsq[0][-5],plsq
                [0][-4]]
    else:
        gpow_s=gpow
        p_fit=[plsq[0][l],plsq[0][l+1],
            v_s,plsq[0][l+3],plsq[0][-3],
            plsq[0][-2],plsq[0][-1]]
    fitparams[ch,a,:]=[plsq[0][l],plsq
        [0][l+1],v_s,plsq[0][l+3],gpow_s]

    l+=4
if npeaks[ch,a]==2:
    v_dl=plsq[0][l+3]
    v_dr=plsq[0][l+4]
    A_dl=plsq[0][l+1]
    A_dr=plsq[0][l+2]
    if 'v_s' in locals():
        if abs(v_s-v_dl)>=abs(v_s-v_dr):
            #preceding gaussian on the
            left
            gpowl=gpow
            pl1,pl2,pl3=plsq[0][-3],plsq
                [0][-2],plsq[0][-1]
            gpowr=gpow2
            pr1,pr2,pr3=plsq[0][-6],plsq
                [0][-5],plsq[0][-4]
```

```
                else:
                    gpowl=gpow2
                    pl1,pl2,pl3=plsq[0][-6],plsq
                        [0][-5],plsq[0][-4]
                    gpowr=gpow
                    pr1,pr2,pr3=plsq[0][-3],plsq
                        [0][-2],plsq[0][-1]
            else:
                gpowl=gpow
                pl1,pl2,pl3=plsq[0][-3],plsq
                    [0][-2],plsq[0][-1]
                gpowr=gpow2
                pr1,pr2,pr3=plsq[0][-6],plsq
                    [0][-5],plsq[0][-4]
            if A_dl>=A_dr:
                fitparams[ch,a,:]=[plsq[0][l],
                    A_dl,v_dl,plsq[0][l+5],gpowl]
                p_fit=[plsq[0][l],A_dl,v_dl,plsq
                    [0][l+5],pl1,pl2,pl3]
                fitparams_2ndG[ch,a,:]=[plsq[0][l
                    ],A_dr,v_dr,plsq[0][l+6],gpowr
                    ]
            else:
                fitparams[ch,a,:]=[plsq[0][l],
                    A_dr,v_dr,plsq[0][l+6],gpowr]
                p_fit=[plsq[0][l],A_dr,v_dr,plsq
                    [0][l+5],pr1,pr2,pr3]
                fitparams_2ndG[ch,a,:]=[plsq[0][l
                    ],A_dl,v_dl,plsq[0][l+5],gpowl
                    ]
            l+=7
        if not(py.any(py.isnan(p_fit))):
            dp[ch,a,:]=CalcError(p_fit,x,h,n)
            #delv[ch,a]=dp[a,:][2]
        plsqs[ch,a,:]=fit_lines[h]*(finf.nFFT*
            finf.num_intg)
        if not(py.any(py.isnan(dp[ch,a,:]))):
            swidth[ch,a],del_swidth[ch,a]=
                CalcSpectralWidth(x,p_fit,dp[ch,a
                ,:])
        chisq[ch,a]=py.around(finf.num_intg/(finf
            .nFFT-py.size(fitparams[ch,a,:])-1.0)*
            py.norm(res_err[h])**2, decimals=3)
    fit_noise[ch,group_heights[k][0:n]]=mean(
```

```
                fitparams[ch,group_heights[k][0:n],0])*(
                finf.nFFT*finf.num_intg)
            if 'v_dl' in locals():
                del v_dl

#Not Shown: formatting and saving parameters to file
return
```

All of the previous python code could easily be included into a module and called using: save_fits_and_vel_map('yyyy','mm','dd').

APPENDIX C

WIND ESTIMATION CODE

The following code contains the wind estimation code and associated definitions. It uses the following standard python modules: numpy, pylab, and time. There is only one nonstandard module, radarpack, which is used to load the Jicamarca unit vectors. This code could easily be included into a module and called using:

Calc_Winds('yyyy','mm','dd').

```
import matplotlib
import pylab as py
import time
from radarpack import radarbeam as rb

def beam_directions(): #load the beam directions
    #Direction Cosines
    #North--channel 3
    Tx_N=-0.0078125
    Ty_N=0.029296875
    #East--channel 0
    Tx_E=0.01171875
    Ty_E=0.056640625
    #West--channel 1
    Tx_W=-0.05859375
    Ty_W=-0.02734375
    #South--channel 2
    Tx_S=0.041015625
    Ty_S=-0.041015625

    Tx=py.array([Tx_E,Tx_W,Tx_S,Tx_N])#Theta X
    Ty=py.array([Ty_E,Ty_W,Ty_S,Ty_N])#Theta Y
    Tz=py.sqrt(1-(Tx**2+Ty**2))# Theta Z

    jro=rb.radarspecs(location="JRO")
    A=py.array([jro.east0,jro.north0,jro.zenith0])# JRO ENU unit
```

```
    vectors

    B=zeros([4,3])
    for k in range(0,4):
        B[k,:]=dot(A,(Tx[k]*jro.ux+Ty[k]*jro.uy+Tz[k]*jro.uo))#
            ENU dot beam vector
    return B


def Calc_Winds(yyyy,mm,dd): #main function to calculate the wind

    B=beam_directions()

    finf,acqtime_arr, vel, error= read_vel(yyyy,mm,dd)#read the
        velocity from a file
    del_vel=error[:,:,:,2] #velocity measurement error
    nm,nch,nh=shape(vel) #number of minutes, number of channels,
        number of heights
    UVW=array(empty([nm,3,nh])*NaN) #initialize winds array
    delta_UVW=array(empty([nm,3,nh])*NaN)#initialize measurement
        error array

    for m in range(0,nm): #for each minute
        for h in range(335,538): #for each height
            num_of_vel=nansum(vel[m,:,h]/vel[m,:,h]) #determine
                number of velocities
            if num_of_vel==3:
                v_ch=find((vel[m,:,h]/vel[m,:,h])==1) #find the 3
                    directions
                UVW[m,:,h]=py.dot(py.inv(B[v_ch]),vel[m,v_ch,h])
                delta_UVW[m,:,h]=py.dot(py.inv(B[v_ch]),del_vel[m
                    ,v_ch,h])
            if num_of_vel==4:
                UVW[m,:,h]=py.dot(py.pinv(B),vel[m,:,h])
                delta_UVW[m,:,h]=py.dot(py.pinv(B),del_vel[m,:,h
                    ])


    basefolder = '/rdata/radar/MSTISR/MST_Maps/y%.4d/'%(int(yyyy)
        )
    outfname = 'windmap_1min_'+time.strftime('%Y.%m.%d.%H.%M.%S',
        finf.LTarr0)
    print 'saving: '+outfname
    numpy.savez_compressed(basefolder+outfname,finf=[finf],
```

```
    WindsUVW=UVW, delta_Winds=delta_UVW, acqtime_arr=acqtime_arr
    )
return
```

APPENDIX D

FILE LOCATION

The files in Figure D.1 are located on the server, remote2, in research folder of the radar account (radar@remote2:~/research).

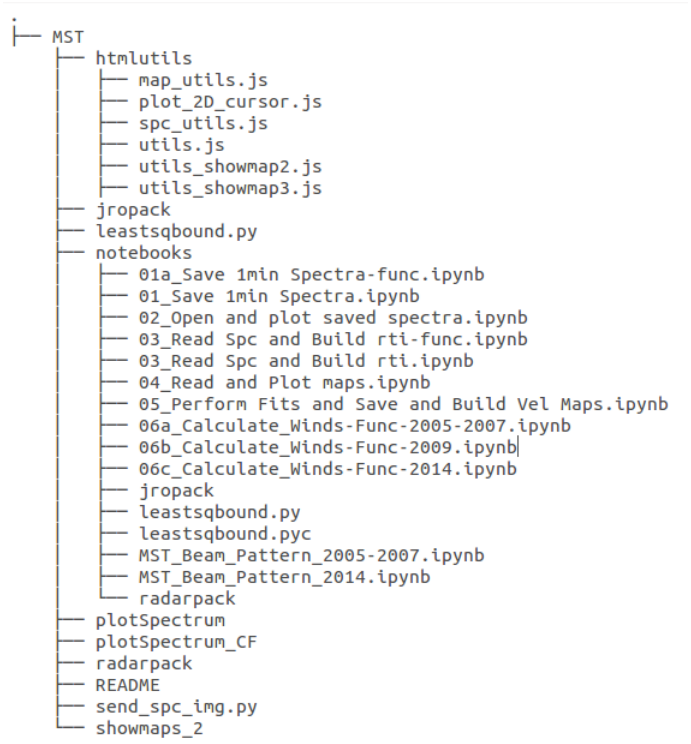


Figure D.1 – The file tree showing the files and their location.

The files for processing are located in the notebooks folder and the files for the web page are located in the MST folder with javascript files located in the htmlutils folder.

REFERENCES

Akgiray, A. H. (2007), Calibration of Jicamarca Radar Using F-Region Inchorent Scatter for Measurement of D-region Backscatter RCS, Master's thesis, University of Illinois at Urbana-Champaign.

Aster, R. C., B. Borchers, and C. H. Thurber (2005), *Parameter Estimation and Inverse Problems*, Elsevier Academic Press.

Bilitza, D., L.-A. McKinnell, B. Reinisch, and T. Fuller-Rowell (2011), The international reference ionosphere today and in the future, *Journal of Geodesy*, *85*(12), 909–920, doi: 10.1007/s00190-010-0427-x.

Buriti, R. A., W. K. Hocking, P. P. Batista, A. F. Medeiros, and B. R. Clemesha (2008), Observations of equatorial mesospheric winds over Cariri (7.4°S) by a meteor radar and comparison with existing models, *Annales Geophysicae*, *26*(3), 485–497, doi:10.5194/ angeo-26-485-2008.

Chandra, H., H. S. S. Sinha, A. K. Patra, U. Das, D. Selvaraj, R. N. Misra, and J. Datta (2012), Low-latitude mesospheric turbulence investigated using coordinated MST radar and rocket-borne observations from India, *Journal of Geophysical Research: Atmospheres*, *117*(D22), doi:10.1029/2011JD016946.

Fritts, D. C., and P. K. Rastogi (1985), Convective and dynamical instabilities due to gravity wave motions in the lower and middle atmosphere: Theory and observations, *Radio Science*, *20*(6), 1247–1277, doi:10.1029/RS020i006p01247.

Fukao, S., R. M. Sato, T. Harper, R. F. Woodman, and W. E. Gordon (1979), Mesospheric winds and waves over Jicamarca on May 23-24, *Journal of Geophysical Research*, *84*, 4379–4386, doi:10.1029/JA084iA08p04379.

Fukao, S., T. Sato, R. M. Harper, and S. Kato (1980), Radio wave scattering from the tropical mesosphere observed with the Jicamarca radar, *Radio Science*, *15*(2), 447–457, doi:10.1029/RS015i002p00447.

Gage, K. S., and J. L. Green (1978), Evidence for specular reflection from monostatic VHF radar observations of the stratosphere, *Radio Science*, *13*(6), 991–1001, doi:10.1029/ RS013i006p00991.

Hargreaves, J. K. (1992), *The solar-terrestrial environment: An introduction to geospace - the science of the terrestrial, upper atmosphere, ionosphere and magnetosphere*, Cambridge Atmospheric and Space Science Series (5), 1 ed., Cambridge University Press.

Hedin, A. E. (1991), Extension of the MSIS Thermosphere Model into the Middle and Lower Atmosphere, *Journal of Geophysical Research*, *96*(A2), 1159–1172, doi:10.1029/90JA02125.

Hitchman, M. H., et al. (1997), Mean winds in the tropical stratosphere and mesosphere during January 1993, March 1994, and August 1994, *Journal of Geophysical Research: Atmospheres*, *102*(D22), 26033–26052, doi:10.1029/97JD01784.

Kelley, M. (2009), *The Earth's Ionosphere: Plasma Physics & Electrodynamics*, International Geophysics, Elsevier Science.

Kudeki, E. (1988), Radar interferometer observations of mesospheric echoing layers at Jicamarca, *Journal of Geophysical Research: Atmospheres*, *93*(D5), 5413–5421, doi:10.1029/JD093iD05p05413.

Kudeki, E. (2010), Applications of Radiowave Propagation, ECE 458 Lecture notes, University of Illinois at Urbana-Champaign.

Kudeki, E. (2013), EK01: Coherent and Incoherent Scatter-Power, Spectrum, Spectral Analysis and Fits, hCOPAR Training Course TCR2013.

Kudeki, E., and M. A. Milla (2011), Incoherent Scatter Spectral Theories–Part I: A General Framework and Results for Small Magnetic Aspect Angles, *IEEE Transactions on Geoscience and Remote Sensing*, *49*(1), 315–328, doi:10.1109/TGRS.2010.2057252.

Lehmacher, G., and E. Kudeki (2003), Variability of equatorial mesospheric echoes, *Advances in Space Research*, *32*(5), 747–752, doi:10.1016/S0273-1177(03)00410-1.

Lehmacher, G., L. Guo, E. Kudeki, P. Reyes, A. Akgiray, and J. Chau (2007), High-resolution observations of mesospheric layers with the Jicamarca {VHF} radar, *Advances in Space Research*, *40*(6), 734–743, doi:http://dx.doi.org/10.1016/j.asr.2007.05.059.

Li, T., A. Z. Liu, X. Lu, Z. Li, S. J. Franke, G. R. Swenson, and X. Dou (2012), Meteor-radar observed mesospheric semi-annual oscillation (SAO) and quasi-biennial oscillation (QBO) over Maui, Hawaii, *Journal of Geophysical Research: Atmospheres*, *117*(D5), n/a–n/a, doi:10.1029/2011JD016123.

Lieberman, R., et al. (1993), Zonal mean winds in the equatorial mesosphere and lower thermosphere observed by the High Resolution Doppler Imager, *Geophysical Research Letters*, *20*(24), 2849–2852.

Penrose, R., and J. A. Todd (1956), On best approximate solutions of linear matrix equations, *Mathematical Proceedings of the Cambridge Philosophical Society*, *52*, 17–19, doi:10.1017/S0305004100030929.

Röttger, J., P. K. Rastogi, and R. F. Woodman (1979), High-resolution VHF radar observations of turbulence structures in the mesosphere, *Geophysical Research Letters*, *6*(7), 617–620, doi:10.1029/GL006i007p00617.

Sheth, R. (2004), High-resolution correlation studies of mesosphic VHF backscatter at the Jicamarca Radio Observatory, Master's thesis, University of Illinois at Urbana-Champaign.

Sheth, R., E. Kudeki, G. Lehmacher, M. Sarango, R. Woodman, J. Chau, L. Guo, and P. Reyes (2006), A high-resolution study of mesospheric fine structure with the Jicamarca MST radar, *Annales Geophysicae*, *24*(5), 1281–1293, doi:10.5194/angeo-24-1281-2006.

Venkateswara Rao, N., T. Tsuda, and Y. Kawatani (2012), A remarkable correlation between short period gravity waves and semiannual oscillation of the zonal wind in the equatorial mesopause region, *Annales Geophysicae*, *30*(4), 703–710, doi:10.5194/angeo-30-703-2012.

Wikström, G., and P.-Å. Wedin (2002), Interpretation and Practical Use of Error Propagation Matrices, *Technical Report UMINF*, *2*.

Woodman, R. F. (1985), Spectral moment estimation in MST radars, *Radio Science*, *20*(6), 1185–1195, doi:10.1029/RS020i006p01185.

Woodman, R. F., and A. Guillen (1974), Radar observations of winds and turbulence in the stratosphere and mesosphere, *Journal of Atmospheric Sciences*, *31*, 493–505, doi:10.1175/1520-0469(1974)031<0493:ROOWAT>2.0.CO;2.

Yamamoto, M., T. Sato, P. T. May, T. Tsuda, S. Fukao, and S. Kato (1988), Estimation error of spectral parameters of mesosphere-stratosphere-troposphere radars obtained by least squares fitting method and its lower bound, *Radio Science*, *23*(6), 1013–1021, doi:10.1029/RS023i006p01013.