INVERSION OF ARECIBO INCOHERENT SCATTER RADAR CODED LONG PULSE BACKSCATTER SPECTRA

BY

YULUN WU

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, 2020

Urbana, Illinois

Adviser:

Professor Erhan Kudeki

ABSTRACT

Incoherent scatter radar (ISR) at Arecibo Observatory measures the scattering of electromagnetic waves from random density fluctuations of ionospheric plasma particles (electrons and ions). Information about particle temperatures, ion concentrations, and Doppler shifts caused by particle motions can be estimated by inverting the power spectra of received scatter signals to the numerically implemented ISR forward spectral model in the frequency domain. Power spectrum estimates are derived by taking FFT of signal samples and averaging the magnitude square of the FFTs. Power spectra include both statistical estimation errors due to the use of finite length data sets and a characteristic shape that depends on ionospheric parameters via a known non-linear relationship that is exploited during the inversion process. This thesis first describes the numerical implementation of the complete collisional ISR spectral model using the chirp-z algorithm, and mainly focuses on Arecibo coded long pulse (CLP) data analysis, including spectrum generation and inversion of raw voltage data from two receivers of Arecibo Observatory. Regular FFT method and the multi-level chirp-z algorithm for speeding up spectrum computation are presented. Also discussed is the weighted least-square spectrum inversion of the spectral estimates to double-humped spectral model of ionospheric incoherent scatter signals using various inversion techniques including the inversion of ion drift velocity using measured spectra ACF.

TABLE OF CONTENTS

CHAPTER 1 INTRODUCTION 1

CHAPTER 2 INCOHERENT SCATTER SPECTRAL THEORY AND NUMERICAL IMPLEMENTATION 4
2.1 Earth's Ionosphere 4
2.2 Incoherent Scatter Spectral Theory 6
2.3 Numerical Implementation of Forward Model 8

CHAPTER 3 RADAR CONFIGURATION AT ARECIBO OBSERVATORY 14
3.1 Arecibo ISR - System Description 14
3.2 Arecibo ISR Data Acquisition Modes 15
3.3 Experiment Dates and Notes 17

CHAPTER 4 COMPUTATION OF CODED LONG PULSE ION-LINE SPECTROGRAM 18
4.1 ISR Signal Processing and Spectral Estimation 18
4.2 Computation of Radar Interface Receiver Power Spectrogram . 20
4.3 Multi-level Chirp-z Transform 24
4.4 Point Spread Function 27

CHAPTER 5 INVERSION OF ARECIBO ISR CODED LONG PULSE POWER SPECTRA 35
5.1 Incoherent Scatter Spectral Model and Ionospheric Plasma Parameters 35
5.2 Ion Velocity from Spectra ACF Inversion 37
5.3 CLP Ion-line Power Spectra Least-square Inversion 38
5.4 Program Implementation and September 2016 Campaign Results 42

CHAPTER 6 CONCLUSION AND FUTURE WORK 49

REFERENCES 50

APPENDIX A SPECTRAL MODEL COMPUTATION CODE . . . 52

APPENDIX B ION-LINE INVERSION CODE 65

CHAPTER 1

INTRODUCTION

Large power VHF/UHF radar systems used with very large gain antennas in ionospheric research are known as incoherent scatter radars (ISRs). The highest sensitivity ISR in operation in the world today is located at the Arecibo Observatory in Puerto Rico. This thesis describes the analysis of Arecibo ISR data collected using the coded long pulse (CLP) technique for Doppler spectral measurements at ionospheric altitudes above 75 km.

The mechanism underlying incoherent scattering in ISR operations is the "dipole radiation" of each free electron in the ionosphere made to oscillate by the transmitted radar pulse – this is known as the Thomson scattering process. The number density of Thomson scattering free electrons in the ionosphere fluctuates as described by a superposition of randomly phased electron density waves propagating in all directions across a broad spectrum of wavelengths having propagation velocities governed by the Langmuir wave and ion-acoustic wave dispersion relations. An ISR will only detect the superposition of Thomson scattered fields from electrons "belonging to" density waves whose wavefronts are perpendicular to the radar beam because scattering from electrons of waves propagating in other directions will be self-cancelling due to the destructive interference. Furthermore, only the scattering from electron density waves with a wavelength equal to one half of the wavelength of the transmitted radar pulse will not self-cancel – this wave component solely responsible for the backscattered radar signal is called the "Bragg wave".

The operation frequency of the Arecibo ISR is 430 MHz, which corresponds to about 70 cm wavelength, and 35 cm Bragg wavelength, meaning that Arecibo ISR will only detect signals returned from 35 cm wavelength electron density waves which are propagating parallel or anti-parallel to the direction of radar beam. The Arecibo ISR signal spectrum will then exhibit a pair of peaks up-shifted from 430 MHz, each one caused by 35 cm Bragg

waves propagating toward the radar at the ion-acoustic velocity C_s as well as the plasma-wave phase shift speed ω_p/k_B, respectively, where ω_p is the plasma frequency and k_B is the Bragg wavenumber, in addition to a pair of down-shifted peaks caused by the same waves propagating in the opposite direction. The slower phase velocity peaks in the ISR spectrum relate to ion-acoustic waves in the ionospheric plasma while the fast phase velocity peaks represent electron plasma (Langmuir) waves. Landau damping and collisional damping of the scattering density waves will contribute to the broadening of each of these spectral peaks. The broadened low-frequency ion-acoustic peaks will tend to merge together to form the "double humped" ion-line feature of the ISR spectrum. The shape of this "double-humped" ion-line components has been derived by *Kudeki and Milla* [2011], and is essentially a superposition of electron and ion velocity distribution functions scaled by k_B and frequency-dependent weighting coefficients describing the collective interactions of ionospheric charged particles (electrons and ions) via polarization electric fields that they cause.

In ISR power spectrum measurements, different types radar pulsing techniques can be chosen to establish some type of a compromise between height and frequency resolutions. The coded long pulse (CLP) method developed by *Sulzer* [1986] is one such technique used at Arecibo to probe the F-region ionosphere. The idea in CLP is to use random binary phase coding to add either a 0° or 180° random phase to each baud of the transmitted pulse, and multiply the received signal samples with the samples of the transmitted pulse to achieve a height resolution specified by the baud length rather than the pulse length.

This thesis describes the procedures used to compute the scattered power and ion-line spectra of Arecibo CLP data returns and the method for the inversion of the CLP power spectra.

There are five additional chapters in this thesis:

- Chapter 2 introduces the basics of the ionosphere and presents the derivation of the model equations of the spectrum of electron density fluctuations causing the incoherent radar backscatter – a convolutionally distorted version of the electron density spectrum also models the backscattered radar signal spectrum that can be estimated using the sampled radar data.

- Chapter 3 describes the configuration of the Arecibo ISR system and the technologies utilized in the experiment using the CLP data collection mode.

- Chapter 4 describes the estimation of ISR ion-line power spectrum from CLP raw voltage data obtained from the Arecibo Radar Interface receiver as well as a broadband USRP receiver using fast Fourier transform (FFT) operations, as well as the point spread function (PSF) describing the height ambiguities and distortions included in the measured power spectra.

- Chapter 5 presents the inversion of electron and ion temperatures from CLP ion-line power spectra using weighted least square fitting techniques. The inversion of ion velocity from auto-correlation function (ACF) of power spectra is discussed.

- Chapter 6 presents the conclusions of this study and ideas for future work for improving our currently available spectral inversion method.

CHAPTER 2

INCOHERENT SCATTER SPECTRAL THEORY AND NUMERICAL IMPLEMENTATION

2.1 Earth's Ionosphere

The earth's ionosphere caused by partial ionization of the upper atmosphere by solar radiation forms an interface between the atmosphere and space in the 60 km to $\sim$ 1000 km altitude range. The ionosphere is divided into three regions: D-region that starts at about 60 km altitude, and extends upwards to about 90 km where collisions of the electrons and ions dominate over their response to Earth's magnetic field; E-region that starts at about 90 km, and extends up to about 150 km where the electrons are magnetically controlled while ions are collision dominated; and finally the topmost F-region that starts at about 150 km, and extends up to about 1000 km where all charged particles are magnetized [*Kelley*, 1989]. In different ionosphere regions, ion composition varies as heavier ions tend to concentrate at lower altitudes. The main ion compositions for different ionosphere regions are summarized in Table 2.1. This thesis will focus on the 100-600 km altitudes of the ionosphere, which is the top part of the E-region and most of the F-region

Table 2.1: Classification of the ionosphere based on ion composition.

Name	Range	Main Ion Compositions
D-region	$60 \sim 90$ km	NO_3^-
E-region	$90 \sim 150$ km	O_2^+, NO^+
F-region	$150 \sim 1000$ km	O^+,H^+

Temperatures profiles and plasma density profiles are used to describe the properties of the ionosphere. Figure 2.1 shows the typical temperatures and electron densities during daytime and nighttime.

Figure 2.1 left shows the temperatures profile up to 700 km. As altitude

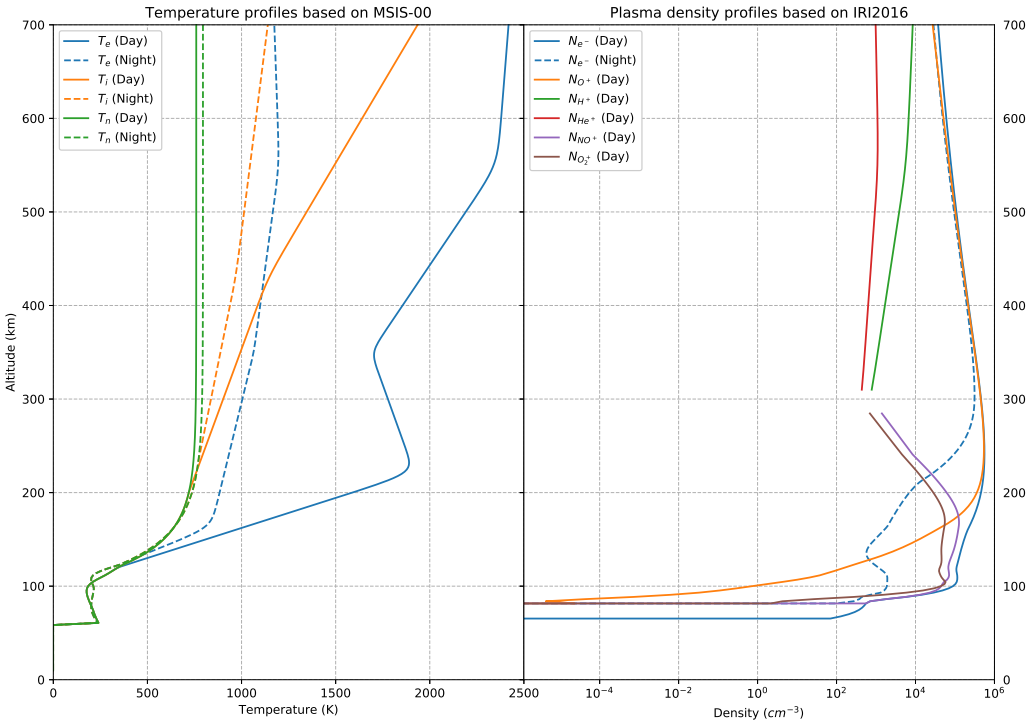

Figure 2.1: Typical neutral and plasma temperatures and electron and ion densities at 0-700 km altitude. These plots are generated from "mass spectrometer & incoherent scatter" (MSIS-00) model and "international reference ionosphere" (IRI2016) model using the latitude and longitude of Arecibo Observatory, and date September 24, 2016, and the times 12 and 0 local time for daytime and nighttime data, respectively.

increases, The neutral temperature increases until about 200 km then remains constant, while the electron and ion temperatures continue increases and go beyond 2000 K.

Figure 2.1 right shows the plasma density profile during daytime. With increase of altitude, the electron density increases, reaches the peak at $\approx$ 250 km altitude, and then decreases. For altitudes below the electron density peak, the radiation from the sun is weak, such that the electron density decreases with altitude decrease. For higher altitudes above the electron density peak where the sun radiation is stronger, the neutral particle density is smaller, such that the electron density decreases with altitude increase. Moreover, during this photo-ionization process, equal numbers of free ions and electrons are generated to maintain the charge neutrality of the ionosphere. Therefore, the sum of all ion densities must nearly equal the electron density at the same altitude.

2.2 Incoherent Scatter Spectral Theory

Each oscillating free electron radiates like a Hertzian dipole and backscatters an electric field — this phenomenon is called Thomson scattering. Ionospheric incoherent scatter consists of Thomson scattering from a collection of ionospheric free electrons made to oscillate at radio frequencies of the probing radars. The incoherent scatter signal spectrum depends not only on the motion of electrons but also on the interaction between electrons and ions. Therefore information about both ionospheric electrons and ions can be extracted from the measured signal spectra of ISR.

According to the incoherent scatter spectral theory by *Kudeki and Milla* [2011], the general framework of ISR power spectrum is formulated as

$$\begin{aligned}\left\langle |n_e(\mathbf{k},\omega)|^2 \right\rangle = &\frac{|j\omega\epsilon_0 + \sigma_i|^2 \left\langle |n_{te}(\mathbf{k},\omega)|^2 \right\rangle}{|j\omega\epsilon_0 + \sigma_e + \sigma_i|^2} \\ &+ \sum_i \frac{|\sigma_e|^2 \left\langle |n_{ti}(\mathbf{k},\omega)|^2 \right\rangle}{|j\omega\epsilon_0 + \sigma_e + \sigma_i|^2},\end{aligned} \tag{2.1}$$

where

$$\left\langle |n_{ts}(\mathbf{k},\omega)|^2 \right\rangle = N_s \cdot 2\mathrm{Re}\{J_s(\omega_s)\} \tag{2.2}$$

is the fluctuation spectrum for a single particle s with density N_s in absence of interaction with other particles, in which the Gordeyev integral

$$J_s(\omega) \equiv \int_0^\infty d\tau e^{-j\omega\tau} \left\langle e^{\mathbf{jk\Delta r_s}} \right\rangle \tag{2.3}$$

is the one-sided Fourier transform of the single particle ACF $\left\langle e^{\mathbf{jk\Delta r_s}} \right\rangle$ for particle species s that can be computed with the Dawson integral in collision-less and non-magnetized case or with the chirp-z transform in the more general case. Above σ_s is the conductivity for particle species s that depends on J_s as

$$\frac{\sigma_s(\mathbf{k},\,\omega)}{j\omega\epsilon_0} = \frac{1 - j\omega_s J_s(\omega_s)}{k^2 h_s^2}, \tag{2.4}$$

in which $\omega_s \equiv \omega - \mathbf{k}\cdot\mathbf{V_s}$ is the Doppler-shift frequency due to line-of-sight bulk velocity $\mathbf{V_s}$ of species s, and $h_s \equiv \sqrt{\epsilon_0 K T_s / N_s e^2}$ is the corresponding

Debye length.

Single particle ACF $\left\langle e^{\mathbf{jk\Delta r_s}} \right\rangle$ is $e^{-\frac{1}{2}k^2C^2\tau^2}$ in a non-magnetized and collisionless plasma but takes a complicated form in the presence of charged particle collisions — in that case it can be shown that (see *Milla and Kudeki* [2009])

$$J_s(\omega) = \frac{G(\omega)}{1 - \nu G(\omega)} \tag{2.5}$$

and

$$G(\omega) = \int_0^\infty d\tau e^{-j\omega\tau} e^{-\nu\tau} e^{\frac{-k^2C^2}{\beta^2}(\beta\tau - 1 + e^{\beta\tau})}, \tag{2.6}$$

where ν denotes the ion-neutral collision frequency and β denotes the Coulomb collision frequency between plasma particle species. Furthermore, for any magnetic aspect angle θ, the wave vector $\mathbf{k}$ can be decomposed to $\mathbf{k} = \hat{b}k_\parallel + \hat{p}k_\perp$ where $\hat{b}$ and $\hat{p}$ are unit orthogonal vectors parallel and perpendicular to $\mathbf{B}$. It can then be shown as in *Kudeki and Milla* [2011] that the Gordeyev integral (2.5) can be expressed in terms of

$$G(\omega) = \int_0^\infty d\tau e^{-j\omega\tau} e^{-\nu\tau} e^{-k^2 \sin^2\theta \langle \Delta l^2 \rangle - k^2 \cos^2\theta \langle \Delta p^2 \rangle}, \tag{2.7}$$

where

$$\langle \Delta l^2 \rangle = \frac{C^2}{\beta^2}(\beta\tau - 1 + e^{-\beta\tau}) \tag{2.8}$$

and

$$\begin{aligned} \langle \Delta p^2 \rangle = \frac{C^2}{\beta^2 + \Omega^2} \cos\left(2\tan^{-1}\frac{\beta}{\Omega}\right) \\ + \beta\tau - e^{-\beta\tau}\cos\left(\Omega\tau - 2\tan^{-1}\frac{\beta}{\Omega}\right) \end{aligned} \tag{2.9}$$

are the variances of displacements Δl and Δp along unit vectors $\hat{b}$ and $\hat{p}$ with particle gyro-frequency $\Omega = qB/m$.

2.3 Numerical Implementation of Forward Model

The collisional and magnetized incoherent scatter forward spectral model described in Section 2.2 is numerically implemented using Python. The numerical model is composed of three layers. The first layer computes the neutral atmosphere constants based on some external atmospheric model, and it computes the ionosphere constants which are required in Gordeyev integral computation based on the user input ionosphere parameters: electron and ion temperatures, electron densities, ion compositions, and ion velocities. The second layer computes the single particle ACF for each ion species and evaluates Gordeyev integral (the single-sided Fourier transform) in (2.7) using numerical chirp-z transform algorithm based on the ionosphere and neutral atmosphere constants from the first layer. The third layer computes the ion-lines and modeled power spectrum (2.1) based on the single-sided Fourier transform results from the second layer, and functions as an interface between the numerical model and user for debugging purposes.

The ionosphere parameters that are required for forward model computation include thermal speed C_s, gyro frequency Ω_s, Debye length h_s, Coulomb collision frequency β_s for each particle species $s \in \{e^-, O^+, H^+, He^+, O_2^+, NO^+\}$, and plasma Debye length h_p that depends on h_s. The equations that are used in implementation and required constants are summarized in Table 2.2 and Table 2.3 (the equations for Coulomb collision frequency is not shown due to its complexity; the complete implementation is shown in Appendix A).

Table 2.2: Ionosphere parameters that are used in forward model implementation.

Ionosphere parameter	Description
$C_s = \sqrt{k_b T_s / m_s}$	Thermal speed
$\Omega_s = q_e B / m_s$	Gyro frequency
$h_s = \sqrt{\epsilon_0 k_b T_s / (N_s q_e^2)}$	Debye length
$h_p = 1/\sqrt{\sum_s 1/h_s^2}$	Plasma Debye length

Neutral atmosphere parameters are also included during the implementation for ion-neutral and electron-neutral collision frequency ν_i/ν_e, geomagnetic field strength B, and magnetic aspect angle θ. Ion-neutral and electron-neutral collision frequencies are empirical models that depend on neutral particle density N_n, neutral temperature T_n, and neutral atomic density A_n,

Table 2.3: Ionosphere constants that are used in forward model implementation.

Ionosphere constant	Description
$m_e = 9.1093826\text{e-}31$	Electron mass (kg)
$m_O = 16 * 1836.152 m_e$	O^+ Ion mass
$m_H = 1 * 1836.152 m_e$	H^+ Ion mass
$m_{He} = 4 * 1836.152 m_e$	He^+ Ion mass
$m_{NO} = (14 + 16) * 1836.152 m_e$	NO^+ Ion mass
$m_{O_2} = (16 * 2) * 1836.152 m_e$	O_2^+ Ion mass
$q_e = 1.60217653\text{e-}19$	Electron charge (C)
$k_b = 1.3806505\text{e-}23$	Boltzmann constant ($m^2kg/(s^2K)$)
$\epsilon_0 = 8.854187817\text{e-}12$	Free space permittivity (F/m)
$c = 299.792458\text{e}6$	Speed of light (m/s)
$r_e = 2.817940325\text{e-}15$	Electron radius (m)

where A_n is calculated using N_n and neutral particle compositions including O, N_2, O_2, He, Ar, H, N. The empirical model equations used for calculating electron-neutral and ion-neutral collision frequencies in MKS units are shown as follows (see *Rishbeth and Garriott* [1969]):

$$\nu_e = N_n \cdot 5.4 \times 10^{-16} \cdot \sqrt{T_n}, \tag{2.10}$$

$$\nu_i = N_n \cdot 2.6 \times 10^{-15} \cdot \frac{1}{\sqrt{A_n}}. \tag{2.11}$$

The N_n and T_n information is obtained using Mass Spectrometer & Incoherent Scatter Model of the Upper Atmosphere (MSIS-00) with `pyglow` package[1] in Python for offline usage availability. The geomagnetic field strength B and magnetic aspect angle θ information are obtained from International Geomagnetic Reference Field (IGRF-12) model. Figure 2.2 shows the plot for ν_e, ν_i, B, and θ with respect to altitude for a given time instant.

With ionosphere and neutral atmosphere constants computed above, we compute the single particle ACFs (the integrand in (2.7)) for all six ionospheric particle species, then the one-sided Fourier transform is evaluated numerically using chirp-z algorithm. Unlike regular FFT algorithm in which sampling rate dt and frequency resolution df are coupled through $df = 1/(dt \cdot N)$ relation for input time samples with length N, chirp-z algorithm allows the user to select arbitrary frequency resolution df and starting fre-

[1]Credit to https://github.com/timduly4/pyglow

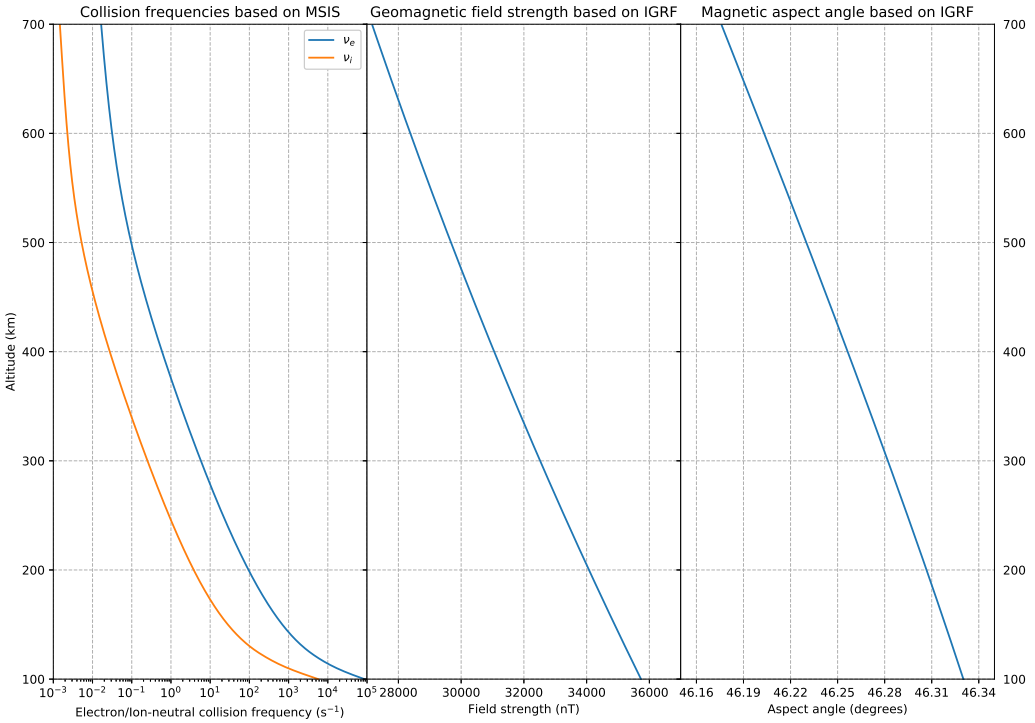

Figure 2.2: Electron/ion-neutral collision frequency, geomagnetic field strength, and magnetic aspect angle at Arecibo Observatory (18.3464° N, 66.7528° W). The data shown are using date September 24, 2016, at 12 PM local time.

quency f_0 given time samples with length N and sampling rate dt, such that dt and df are decoupled. In our application, the modeled frequency spectrum is chosen to have the same frequency resolution and bandwidth as the measured spectrum (will be discussed in Chapter 4). However, the single particle ACFs (the time domain samples) for different particles decay at different rates, which makes the chirp-z algorithm a more favorable choice than regular FFT for its flexibility to use different time ranges for different particle species and thereby reduce numerical computation error during numerical Fourier transform.

Furthermore, time window and time resolution are carefully chosen when computing ACF samples to further reduce numerical error. The integration range of the Gordeyev integral (2.7) is $[0, \infty)$, meaning that the time window should cover the ACF from $t_{start} = 0$ to some t_{end} where ACF decays to zero as much as possible. While the time window should be large enough to cover the entire non-zero section of ACF, time resolution dt should be

as small as possible to better approximate the theoretical ACF. In our application, the numerically evaluated forward model using some appropriate ACF time window T_{max} and $N = 220$ (in order to match the baud length of transmitted pulse) results large numerical error. Such numerical error is reduced with larger N, which equivalently reduces dt, when computing particle ACFs. Choosing larger N results higher frequency resolution during chirp-z transformation when frequency range is fixed, hence the output frequency spectrum is resampled to match the frequency resolution of measured power spectrum.

Figure 2.3 shows the single particle ACF for electron and all five ion species included in the forward model. The time ranges for ACFs are chosen differently for electron and ions such that all ACFs decay to zero as close as possible within the time window we choose. In Figure 2.3 the time windows we choose for electron ACF and ion ACFs are $T_{emax} = 4$ μs and $T_{imax} = 440$ μs, respectively. The electron ACF and ion ACFs are computed with length $N = 2200$, which is 10 times the baud length of the transmitted pulse used in our experiments. Therefore, the ion-lines are down-sampled by 10 times after chirp-z transform to match the measured power spectrum.

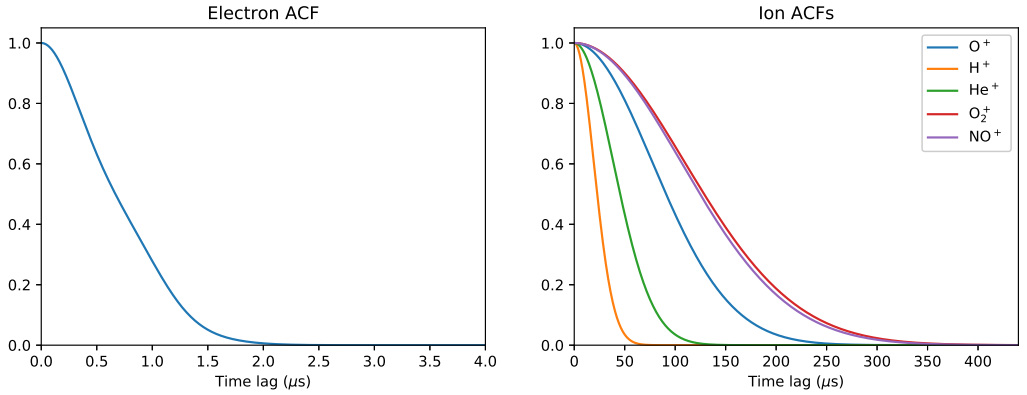

Figure 2.3: Single particle ACFs for electron and five ion species. The time ranges for electron and ions are separately chosen to accommodate different decay rates of ACFs due to particle mass difference. The ACFs are computed using atmosphere parameters from September 24, 2016, at 12 PM local time, and at 300 km altitude. The ionosphere parameters are chosen to be $T_e = T_i = 1000$ K, $N_e = N_i = 10^{12}$ m^{-3}, and $V_i = 0$ m/s.

The forward model power spectrum (2.1) is computed after the numerical chirp-z Fourier transform. Figure 2.4 shows the numerically computed electron ion-line (first term in equation 2.1) and ion-lines (the second term

in equation 2.1) using chirp-z algorithm based on the single particle ACFs shown in Figure 2.3. Figure 2.5 shows the reconstructed ion-line model from 75 km to 700 km altitude with temperature and ion density profiles retrieved from MSIS-00 database on September 24, 2016, at 12 PM local time.

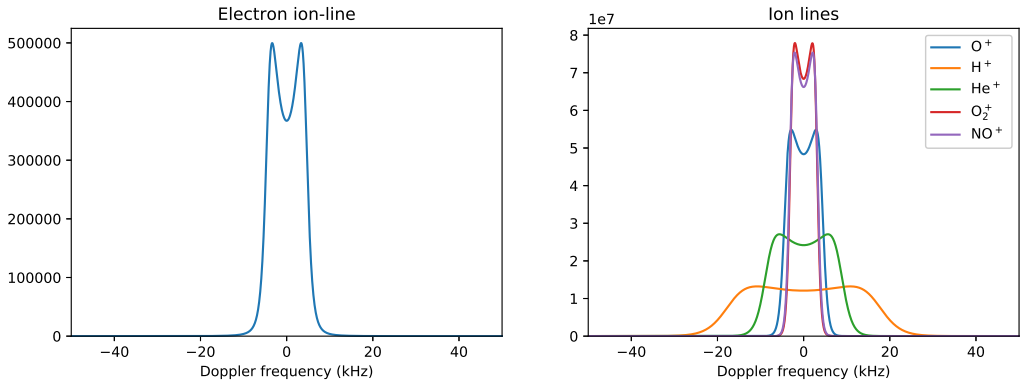

Figure 2.4: Electron ion-line and single particle ion-lines five ion species using chirp-z algorithm and single particle ACFs shown in Figure 2.4. The ACFs are computed using atmosphere parameters from September 24, 2016, at 12 PM local time, and at 300 km altitude. The ionosphere parameters are chosen to be $T_e = T_i = 1000$ K, $N_e = N_i = 10^{12}$ m^{-3}, and $V_i = 0$ m/s.

In the real ionosphere, only two or three ion species dominate the ionosphere simultaneously at certain altitudes, while other ions are ignored because of their small densities and therefore cannot be detected correctly. Based on the altitude of measured data, different power spectrum models (2.1) are established for faster computation speed during numerical chirp-z transform and lower complexity of numerical model for better inversion stability. Table 2.4 shows the ion-lines that are included in the forward model computation based on the measured spectrum altitude during inversion (to be discussed in Chapter 5).

Table 2.4: The ion-lines that are included in the forward model based on the altitude information provided during inversion.

Altitude range	Ion-lines included
$h \lessapprox 200$ km	O^+, NO^+, O_2^+
$h \gtrapprox 200$ km	O^+, He^+, H^+

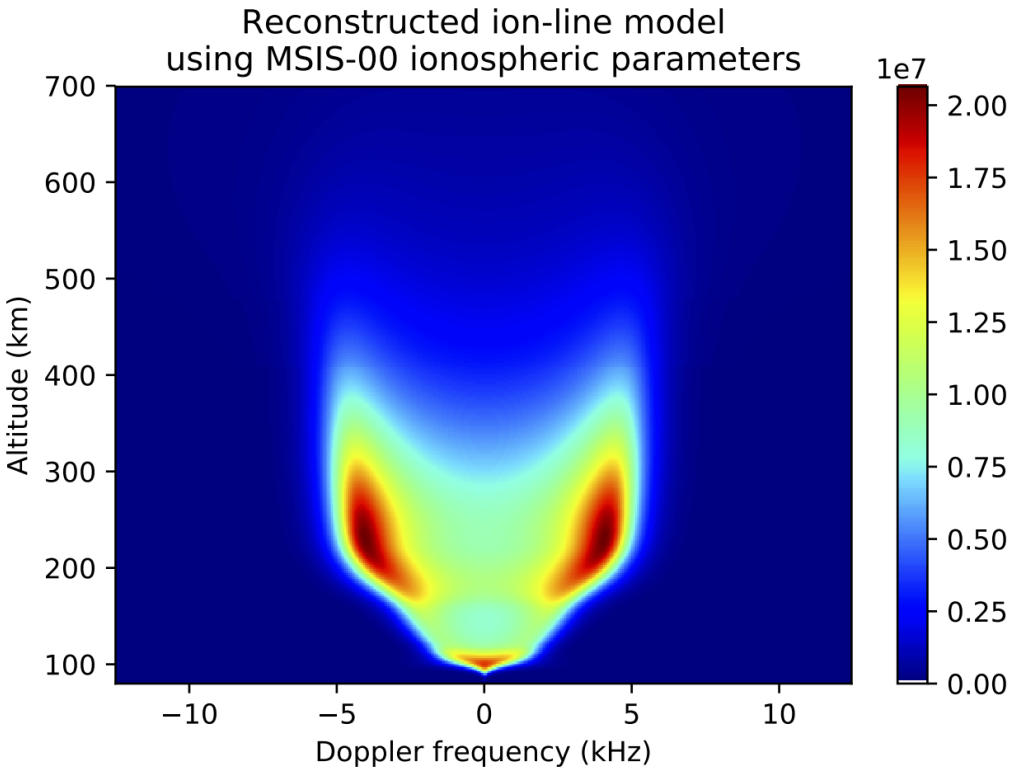

Figure 2.5: Reconstructed ion-line model from 75 km to 700 km using ionospheric parameters from MSIS-00 database. The temperature and ion density profiles are obtained from September 24, 2016, at 12 PM local time.

CHAPTER 3

RADAR CONFIGURATION AT ARECIBO OBSERVATORY

The Arecibo Observatory is located near the northern coastal town of Arecibo on the island of Puerto Rico in a region populated by natural sinkholes in its terrain. One such sinkhole about 15 km inland from Arecibo houses the largest single dish spherical reflector antenna in the world used for space research. The Arecibo reflector antenna is part of the Arecibo ISR system that was designed and built by William E. Gordon of Cornell University in the mid-1960s and maintained by Cornell University until 2011. Since 2011 the Arecibo Observatory and its ISR system have been operated under cooperative agreements with the National Science Foundation. The Arecibo facility supports three major areas of research: radio astronomy, atmospheric science, and radar astronomy. The observatory has radar transmitters with effective isotropic radiated power of 1 MW at 2380 MHz (S-band system) and 2.5 MW at 430 MHz (the ionospheric ISR system). This thesis is focused on the use of Arecibo ISR system for ionospheric measurements and research, specifically in coded long pulse data mode of the system.

3.1 Arecibo ISR - System Description

The Arecibo ISR is a 430 MHz backscatter radar system using a 305 m diameter spherical dish antenna as shown in Figure 3.1. The radar transmitter generates pulses with peak power of 2.5 MW at the 430 MHz operating frequency [*Isham et al.*, 2000]. Given its very large antenna aperture and transmitted power operated within the UHF band, Arecibo ISR achieves an overall sensitivity about 100 times larger than that of other ISR systems currently in existence. The 430 MHz line feed makes an efficient use of the main dish when pointed vertically, as its radiation pattern fills the available aperture. The new Gregorian feed that was added to the system during the 2000

Figure 3.1: Arecibo Observatory 305 m diameter dish antenna.

upgrades enables dual beam operations for more efficient determinations of ionospheric plasma drifts [*Isham et al.*, 2000].

3.2 Arecibo ISR Data Acquisition Modes

Isham et al. [2000] describe six data acquisition modes for Arecibo ISR operations. Two of them, uncoded long pulse (ULP), and coded long pulse (CLP), are of importance in F-region and topside ionospheric studies. The use of CLP and ULP data modes for narrow-band ion-line measurements is described below.

3.2.1 Coded long pulse data mode

In soft target radar measurements, if the correlation times of the scattering density waves are short compared to the inter-pulse period (IPP), then pulse-to-pulse correlation methods cannot be used and it becomes necessary to utilize "within pulse" correlation methods with multiple samples taken from the superposed echoes of transmitted pulses whose lengths need to exceed the sampling interval by some substantial margin. Such "within pulse" operations are generally referred to as long pulse techniques. The disadvantage of using long pulses is accepting relatively poor radar range resolution as backscattered signals from multiple heights are entangled during reception.

The coded long pulse (CLP) mode utilized at Arecibo Observatory alleviates the height ambiguity issue, and is used to probe the lower F-region altitudes of the ionosphere where the electron density is relatively large and the corresponding scatterer SNR is sufficient. The mode was developed by *Sulzer* [1986] for high-resolution ion-line measurements.

Arecibo CLP mode for F-region ionosphere measurement utilizes 10 ms IPP with 440 μs rectangular envelope for transmitted long pulses. The rectangle envelope is divided into 220 bauds with 2 μs per baud, which corresponds to 300 m range resolution. Each baud is applied with a random binary phase of either $+90^\circ$ or -90°. The transmitted signal is sampled with backscattered signal for decoding purpose: the received samples are multiplied with the complex conjugate of the transmitted coding such that only the envelope of backscattered signal from target altitude is recovered to rectangular shape whose impulse response is delta function; while the envelope for other neighboring altitudes within the detection range are randomed out, meaning that their impulse responses are no longer a delta function but spread out over the frequency band of received power spectrogram. A more detailed discussion about CLP spectrogram processing and its characteristics is in Chapter 4.

3.2.2 Uncoded long pulse data mode

The uncoded long pulse (ULP) mode is similar to CLP but no random phase coding is applied. Consequently, the backscattered signals from the illumination range of one transmitted long pulse are not distinguishable, such that the range resolution in this mode is determined by the pulse length instead of the baud length. ULP mode accompanies CLP mode to extend the ISR detection range to up to 1200 km altitude where CLP can only probe up to 600 km due to its coding. The coding in CLP causes the spread of power for untargeted altitude over a wide frequency range, while the lack of coding in ULP reinforces the ion-line spectrogram within a narrow frequency band and hence has stronger SNR at higher altitudes. On the other side, the lack of coding causes the backscattered ion-line spectrogram from different altitudes to mix together, and a 2-dimensional deconvolution with respect to range and frequency needs to be performed during the data inversion stage

to mitigate the mixing.

Arecibo ULP mode utilize 10 ms IPP with 500 μs rectangular envelope and 250 bauds for transmitted pulse. The pulses are transmitted in pulse pairs with $\pm$62.5 kHz deviation from 430 MHz frequency because ULP mode pulses have a very large detection range — this frequency switching procedure helps with avoiding spectrogram contaminations from the echoes coming from the "previously transmitted" pulses. The received baseband samples are further demodulated by $\mp$62.5 kHz before performing spectral analysis.

Furthermore, Arecibo also runs Longer ULP (LULP) mode where the 1000 μs rectangular envelope is transmitted using 500 baud length and 20 ms IPP. The LULP mode has higher detection range compared to ULP mode because of its longer transmitted pulse and hence larger transmitted power.

3.3 Experiment Dates and Notes

The data used for ion-line spectrogram inversion in this thesis were acquired during a data campaign conducted in September 2016 at the Arecibo Observatory. The pulses were transmitted in half-minute cycles of 10 seconds CLP, followed by 10 seconds of ULP, followed by 10 seconds of LULP. Table 3.1 shows the details of the campaign measurements including data acquisition mode, campaign duration, as well as the technical details for each mode.

Table 3.1: Radar configuration during September 23-26, 2016, campaign in Arecibo Observatory.

Campaign date	September 23-26, 2016		
Pulse modes	CLP	ULP	LULP
IPP (ms)	10	20	20
Pulse length (μs)	440	500	1000
Baud length	220	250	250
Detection range (km)	75-1200	105-1400	105-1400

CHAPTER 4

COMPUTATION OF CODED LONG PULSE ION-LINE SPECTROGRAM

The transmitted pulse and received backscattered signal are recorded as demodulated (baseband) complex voltage data from which power spectra and point spread function (PSF) (or radar ambiguity function (AF) in some context) can be obtained using FFT based analysis. This chapter describes the computation procedure of power spectra from the Arecibo Radar Interface (RI) receiver and the USRP receiver used at the receiving of the Arecibo ISR. These two receivers have 2 μs and 40 ns sampling rates, respectively, and different processing techniques are applied to each of the receiver outputs. The computation of point spread function (PSF) (ambiguity function (AF) in come context) is also presented for both receivers to facilitate the ion-line inversion procedure described in Chapter 5.

4.1 ISR Signal Processing and Spectral Estimation

Consider an N-point voltage sequence with sampling period T

$$v_n = v(nT), \quad n = 0, \cdots, N-1 \tag{4.1}$$

whose discrete Fourier transform pair is given by

$$V_k = \sum_{n=0}^{N-1} v_n e^{-j\frac{2\pi}{N}kn} \tag{4.2}$$

$$v_n = \frac{1}{N}\sum_{k=0}^{N-1} V_k e^{j\frac{2\pi}{N}kn} \tag{4.3}$$

for $k = 0, \cdots, N-1$. The average power of v_n using Parseval's theorem can be found as

$$\sum_{n=0}^{N-1} |v_n|^2 = \sum_{k=0}^{N-1} \frac{|V_k|^2}{N}, \tag{4.4}$$

where $\frac{|V_k|^2}{N}$ is the signal periodogram and its expected value $\frac{\langle |V_k|^2 \rangle}{N}$ is the power spectrum of the backscattered signal.

Let the measured voltage signal sequence v_n be a weakly stationary (WSS) random process whose expected value does not change with time and whose autocorrelation function (ACF) only depends on the time delay $p = m - n$. Then

$$\begin{aligned} \langle |V_k|^2 \rangle &= \sum_{n=0}^{N-1} \sum_{m=0}^{N-1} \langle v_n v_m^* \rangle e^{-j\frac{2\pi}{N}k(m-n)} \\ &= \sum_{p=-N}^{N-1} (N - |p|) \langle v_n v_{n+p}^* \rangle e^{-j\frac{2\pi}{N}kp} \\ &= N \sum_{p=-N}^{N-1} \left(1 - \frac{|p|}{N}\right) \langle v_n v_{n+p}^* \rangle e^{-j\frac{2\pi}{N}kp} \end{aligned} \tag{4.5}$$

can be expressed in terms of the voltage signal ACF $\langle v_n v_{n+p}^* \rangle$, such that power spectrum is related to the triangular weighted signal ACF through

$$\frac{\langle |V_k|^2 \rangle}{N} = \sum_{p=-N}^{N-1} \left(1 - \frac{|p|}{N}\right) \langle v_n v_{n+p}^* \rangle e^{-j\frac{2\pi}{N}kp} \tag{4.6}$$

and

$$\left(1 - \frac{|p|}{N}\right) \langle v_n v_{n+p}^* \rangle = \sum_{k=-N}^{N-1} \frac{\langle |V_k|^2 \rangle}{N} e^{j\frac{2\pi}{N}kp}. \tag{4.7}$$

The WSS assumption cannot be applied to the voltage signals obtained from CLP and ULP transmitted pulses because of the height dependence of the ionosphere where the signals are backscattered; hence, the received power spectrum $\frac{\langle |V_k|^2 \rangle}{N}$ will be the discrete Fourier transform of the mixed ACF $\langle v_n v_{n+p}^* \rangle$ from a range of altitudes governed by some AF that will be discussed in later sections.

4.2 Computation of Radar Interface Receiver Power Spectrogram

One receiver in Arecibo Observatory used for ISR measurements is the Radar Interface (RI) receiver that has 2 μs sampling rate. The CLP pulse transmitted uses 10 ms inter-pulse period (IPP) and 2 μs radar transmitter bauds and matched receive filters. Transmitted pulses for CLP have 220 baud length. The transmission and reception scheme of one data record is shown in Figure 4.1. Each CLP data record (transmission and sampling process within one

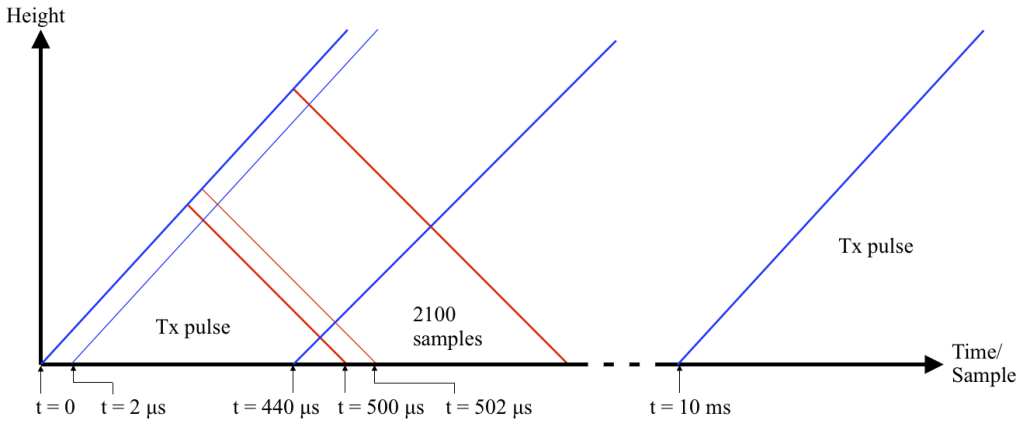

Figure 4.1: Transmission and reception scheme of one data record.

IPP) contains 2740 samples, the first 220 samples of which are transmitted pulse samples obtained from a directional coupler located near the transmitter output. The receiver samples start from the 250th sample in the data record. Figure 4.2 shows an example of CLP transmit pulses received by RI receiver, and Figure 4.3 shows an example of one received signal for some target altitude by RI receiver.

In each data record, the first received signal sample is taken at 500 μs after the pulse is transmitted, and the corresponding scattering height is $h = ct/2 = 3 \times 10^8 \cdot 500 \times 10^{-6}/2 = 75$ km. By taking samples from point 0 to point 219 and doing Fourier transform of the data multiplied by the complex conjugate of the transmitted pulse samples (called decodes), we can obtain a periodogram for a target range of 75 km altitude. Then the samples are right-shifted by one point for the next altitude, and therefore the height resolution is 300 m for CLP campaigns. Repeating this process until the end of one record, we can find periodograms for 1881 heights in total, and a 3D

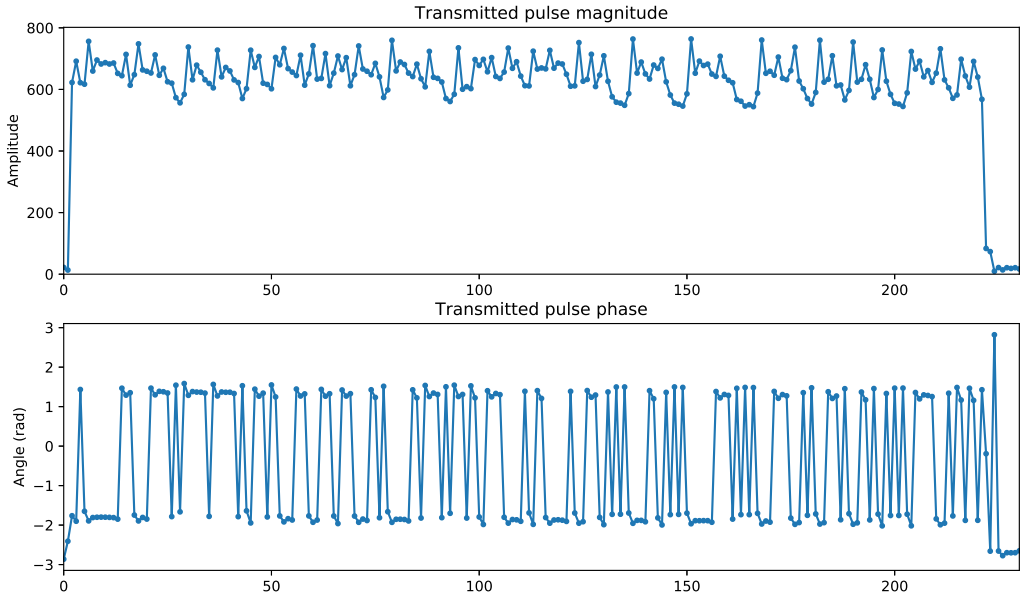

Figure 4.2: Example of transmitted CLP pulse with 220 baud length. The transmitted pulse is obtained from September 25, 2016 data at 2 PM local time.

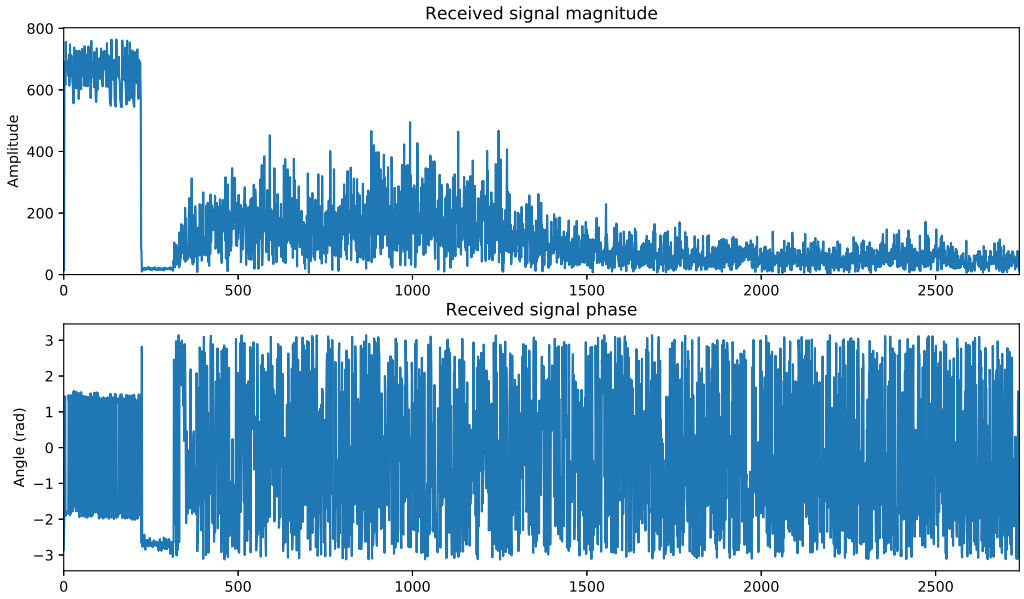

Figure 4.3: Example of received signal in one data record. The data record has 2740 samples that correspond to 822 km detection range. The beginning 220 samples are transmitted pulse shown in Figure 4.2. The ending 500 samples are injected noise signals for calibration. This data record is obtained from September 25, 2016 data, at 2 PM local time.

plot from 75 km to 640 km can be generated.

Periodograms are poor estimators of the spectrum of radar signals because of large statistical errors. Better estimates can be obtained by averaging many periodograms obtained from successive time intervals. We add 2000 such periodograms together to generate a spectrum for 1 minute (only 20 seconds in every minute are used for CLP pulses and the other 40 seconds are used for uncoded long pulses (ULP) which is not discussed here). Here we assume the ionosphere is stationary inside 1 minute range; i.e., particle temperatures and ion compositions in ionosphere will not change greatly in 1 minute. The spectral results with a clear shape can be obtained. Figure 4.4 shows a processed spectrogram example using 2000 integration length. The example spectrogram is computed from September 25, 2016, data at 2 PM local time. Figure 4.5 shows an example of power spectrum at different altitudes; the example is obtained by taking horizontal cuts from Figure 4.4.

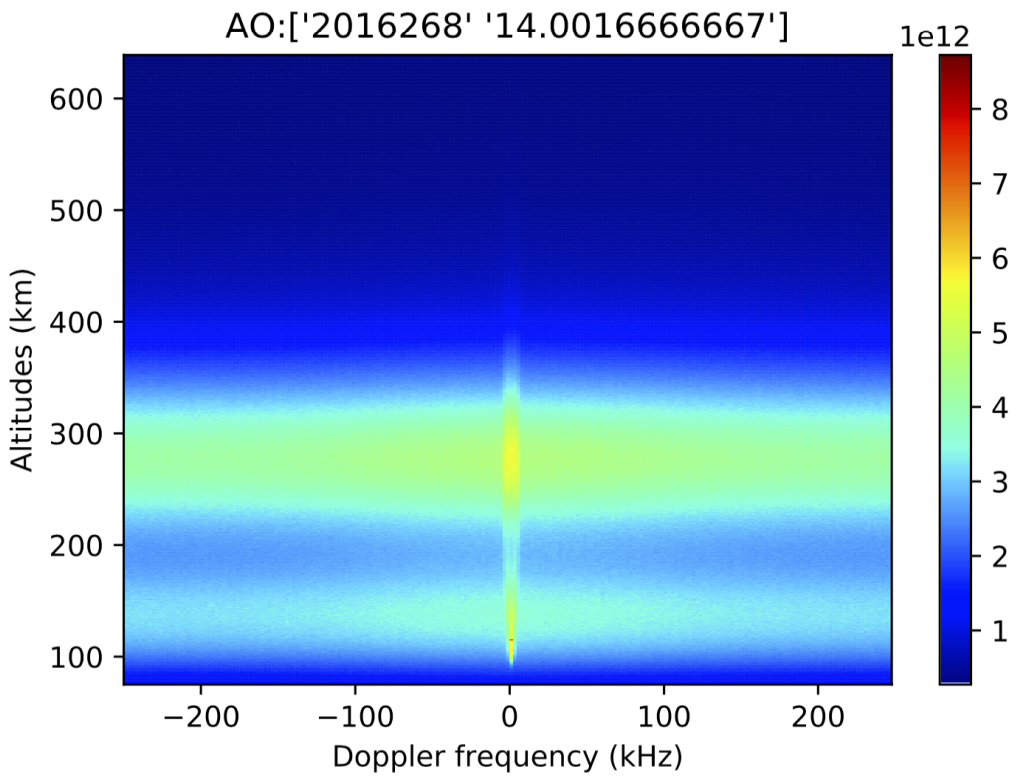

Figure 4.4: Power spectrogram using 1 minute integration from 75 km to 640 km altitude, and has 500 kHz bandwidth. The spectrogram is obtained from September 25, 2016, data at 2 PM local time.

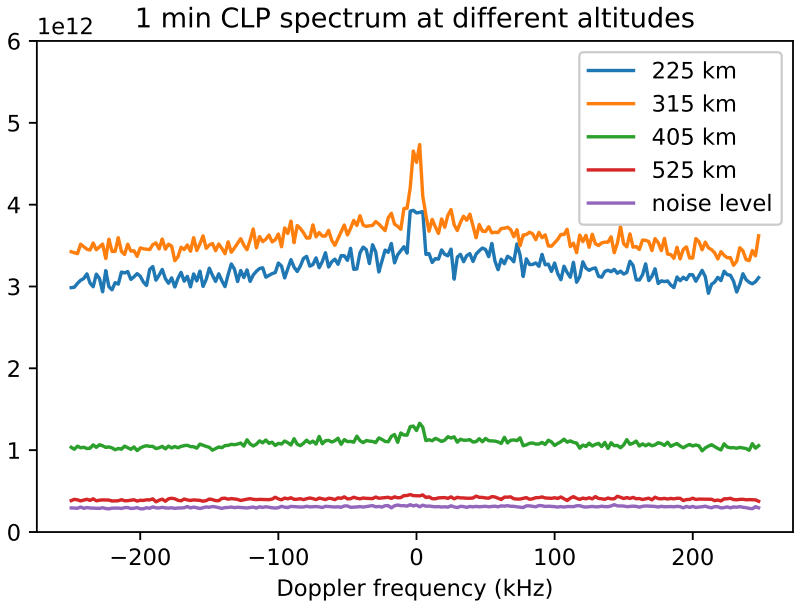

Figure 4.5: Power spectrum at different altitudes by taking horizontal cuts from Figure 4.4. The background noise level is obtained by taking the cut from topmost altitude when ion-line signal totally disappears.

4.2.1 Computation of USRP receiver power spectrogram

One major hardware upgrade performed in Arecibo Observatory in 2017 enabled the USRP receiver that has sampling rate of 40 ns [*Vierinen et al.*, 2017]. Each USRP data record contains 250000 samples that correspond to 10 ms IPP and 40 ns sampling rate. The beginning 11000 samples are transmitted pulse that is same as RI receiver data but sampled 50 times faster. The received signal starts from 12500 samples that correspond to 75 km scattering altitude. One sampled transmitted pulse from USRP receiver is shown in Figure 4.6, and the entire data record is shown in Figure 4.7.

For one target altitude, 11000 consecutive samples are decoded with transmitted pulse; by doing Fourier transform to the decoded samples we obtain one 25 MHz bandwidth power periodogram. Integrating the periodograms over a period of time, power spectrograms are obtained, one example of USRP receiver power spectrum is shown in Figure 4.8.

The center peak in the broadband spectrum is the ion-line that we desire, and the peaks at two sides are plasma-line features from which information of electron density can be obtained (will not be discussed here). Our interest is only the center 500 kHz bandwidth; therefore, the direct Fourier transform is

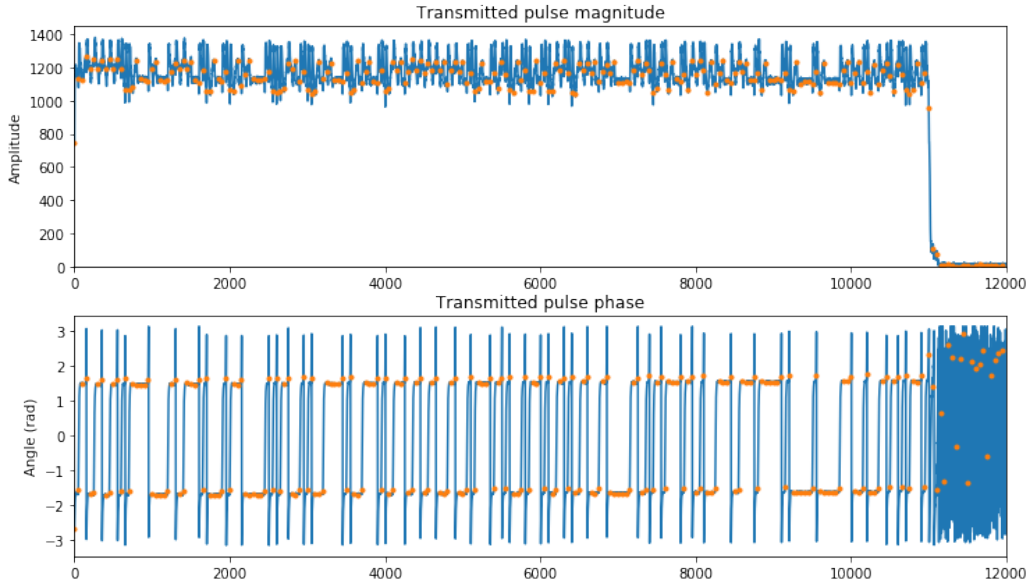

Figure 4.6: Example of transmitted CLP pulse with 220 baud length sampled by USRP receiver. The sampled pulse has 11000 points. The orange dots show the original 220 baud length by down-sampling the data record by a factor of 50. The example data is obtained from June 7, 2018, data at 11:38 PM local time.

inefficient as the information beyond 500 kHz Doppler frequency is discarded. Instead, a refined Fourier transform algorithm presented in *Li et al.* [1991] is combined with chirp-z transform to compute only the center 500 kHz bandwidth power spectrum and reduce the computation time.

4.3 Multi-level Chirp-z Transform

In regular FFT programs, the time sampling rate and frequency resolution are coupled through relation $\Delta f = 1/(N\Delta t)$ where N is number of samples in both time domain and frequency domain. The chirp-z algorithm has the capability to decouple the time sampling rate and frequency resolution, i.e., we can choose the frequency resolution Δf and starting frequency f_0 of spectrum regardless the original time sampling rate Δt of data. However, the chirp-z algorithm still requires same number of samples in both time domain and frequency domain. In situations where fewer points are desired in frequency domain, a multi-level refine algorithm can be applied. In *Li et al.* [1991], single level chirp-z transform for signal $F(n\Delta t)$ with sampling

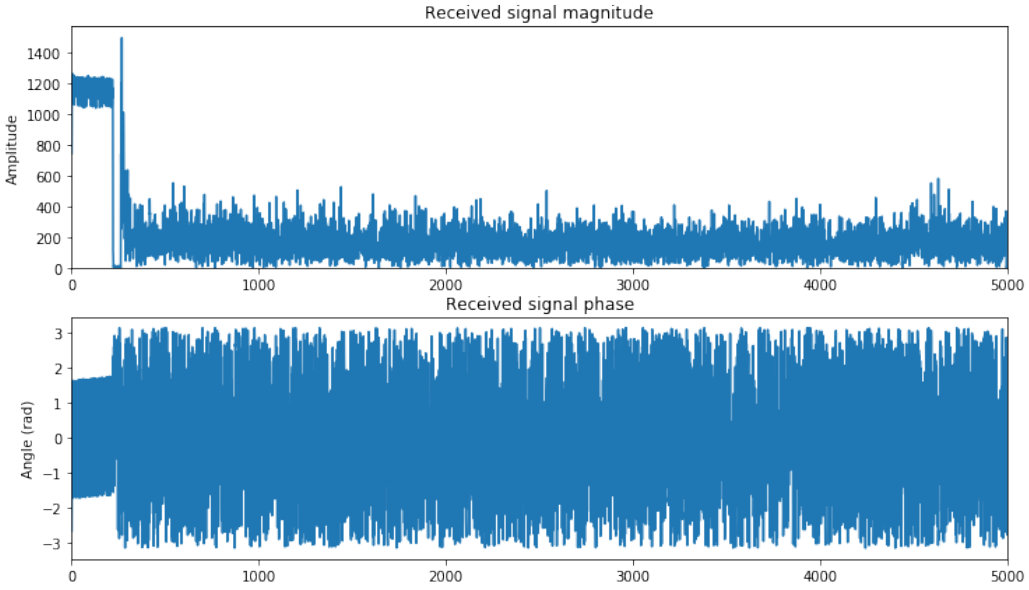

Figure 4.7: Example of received USRP samples in one data record. The data record has 250000 samples (down-sampled by a factor of 50 for display purpose) that correspond to 1500 km detection range. The beginning 11000 samples are transmitted pulse shown in Figure 4.6. This data record is obtained from June 7, 2018, data at 11:38 PM local time.

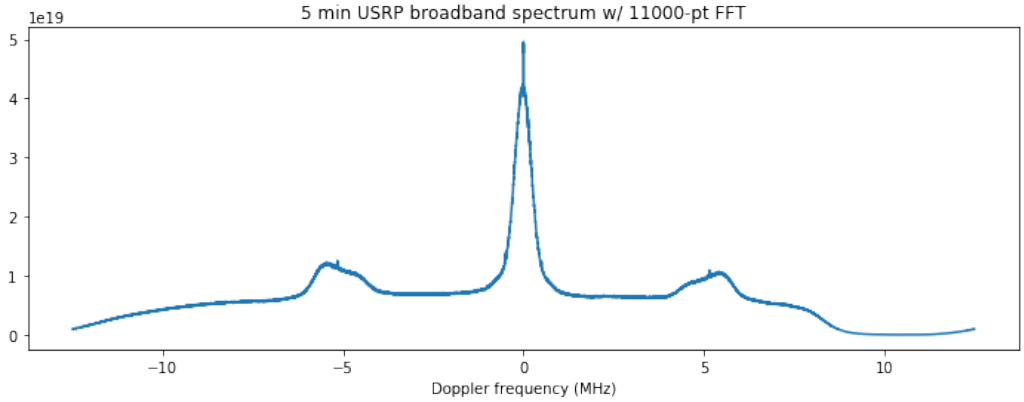

Figure 4.8: Example broadband spectrum of 5 minutes USRP receiver data.

rate Δt, length N, desired frequency resolution $\Delta\omega = 2\pi\Delta f$, and starting frequency $\omega_0 = 2\pi f_0$ can be calculated as follows:

$$S_{1N}(F, n, \Delta t, \Delta\omega, \omega_0) = \Delta t \sum_{n=0}^{N-1} F(n\Delta t) \exp(-jn\Delta t\omega_0) W_N^{nm}. \tag{4.8}$$

In Equation (4.8),

$$W_N^{nm} = \exp\left(\frac{-j2 \cdot (\frac{1}{2}n^2 + m^2 - (m-n)^2) \cdot \pi}{N}\right), \tag{4.9}$$

where n, m are sample indices for input and output domains, respectively.

In the situation we desire to reduce the bandwidth by X-fold, a multi-level chirp-z transform can be applied, which is outlined as

$$S_{XN} = \frac{1}{X}\sum_{x=0}^{X-1} \exp\left(\frac{-jm2x\pi}{XN}\right) S_{1N}(F, n + \frac{x}{X}, \Delta t, \Delta\omega, \omega_0). \tag{4.10}$$

The multi-level chirp-z algorithm in Equation (4.10) states that the reduced bandwidth spectrum can be computed using down-sampled signals with a phase correction factor based on the input signal length XN and output frequency domain indices m. To better illustrate the idea, one simplified example is shown in Figure 4.9.

Figure 4.9: Multi-level chirp-z transform illustration.

In the example shown in Figure 4.9, the input samples have length 12, and the desired output length is 4. Therefore the input sequence is folded by three times, i.e., we perform chirp-z transform for every three samples in the input sequence, then multiply by the corresponding phase correction term $\exp\left(\frac{-jm2x\pi}{XN}\right)$. In our application the decoded signals have 11000 samples, so we apply the 50 level refine algorithm with 220 samples in each level. Comparing the 11000-point FFT and S_{50N} algorithm described in (4.10) using our application, we find that the latter algorithm runs nearly twice as fast as the primary FFT, as $11000\log_2(11000)/(50 \cdot 220\log_2(220)) = 1.73$, with FFT computation cost $O(N\log_2 N)$.

Examples shown in Figure 4.10 of spectra calculated using S_{50N} and direct N=11000 point FFT are identical as expected. Figure 4.11 shows an additional comparison of spectra obtained with USRP and RI receiver data — the higher wings of the RI spectrum must be a consequence of frequency aliasing from which the chirp-z based estimation method of USRP data sets is immune.

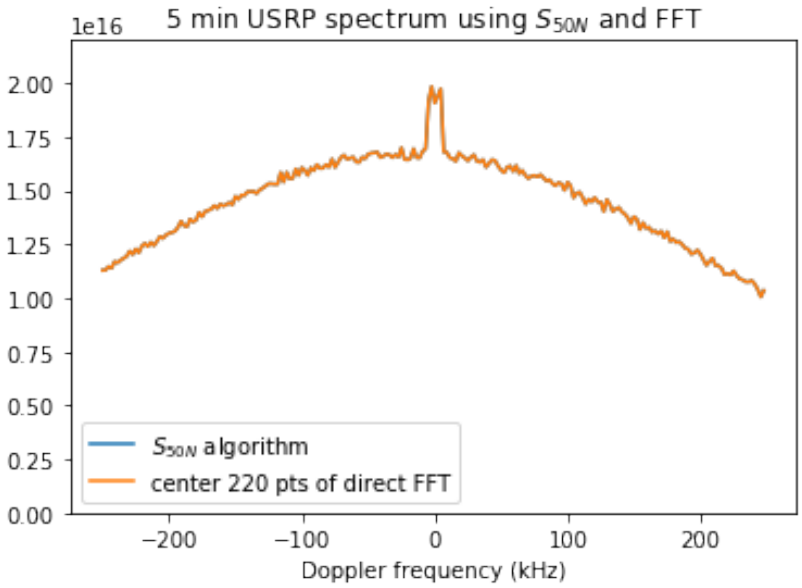

Figure 4.10: Example of multi-level chirp-z algorithm and comparison with direct 11000-pt FFT.

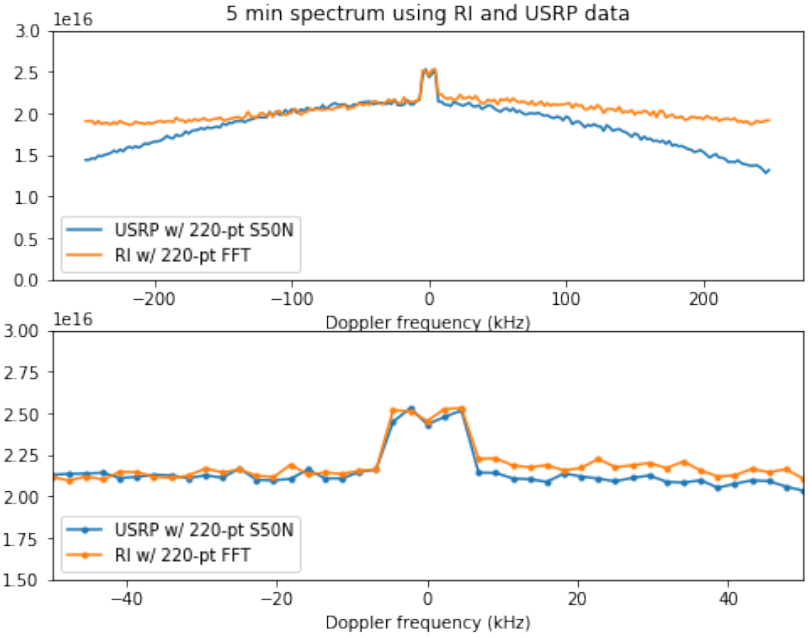

Figure 4.11: Comparison between spectra of USRP and RI receiver at same time.

4.4 Point Spread Function

The point spread function is an indicator of how an intrinsic delta function spectrum would be smeared over Doppler frequencies as a consequence of

the pulse coding and decoding step. It can be computed using the encoded transmitted signal captured by the receiver. The PSFs for RI receiver and USRP receiver are different due to their different sampling rates. However, the computation procedures for both PSFs are the same. First, the computation steps are described and the PSF is shown in RI receiver case. The USRP receiver PSF is shown next.

Figure 4.12 shows a blow-up detail of the transmission and sampling process in Figure 4.1 for some target altitude for RI receiver. In Figure 4.12 up-going lines represent transmitted signal while the down-going lines represent received signal from backscattering of the ionosphere. The total power that can be received for a transmitted long pulse is the summation of powers that come from all sub-volumes. At center cut labelled as the target height, the pulse can be fully recovered by conjugate multiplying the samples with the original randomized binary coding. At this height the Fourier transform of the received signal, which is a rectangle pulse after conjugate multiplication, has the shape of δ function (a sinc function that is sampled at its zero crossings except at zero Doppler frequency point).

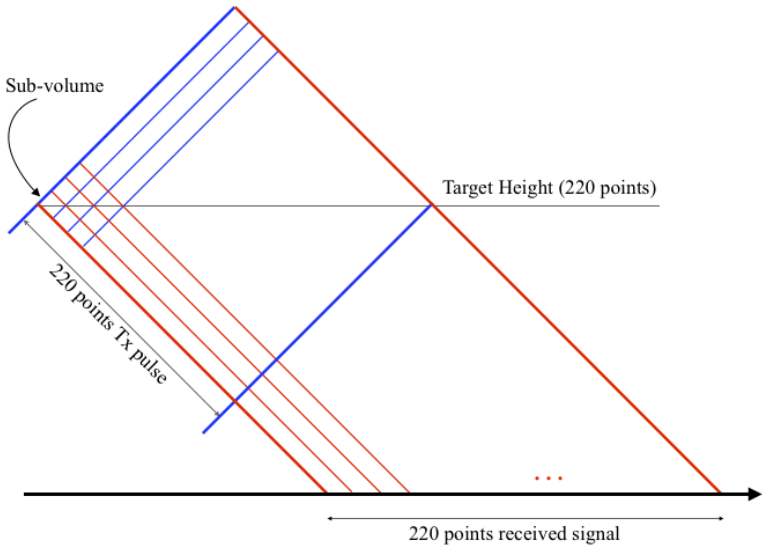

Figure 4.12: Reception of one 220-points data record. Blue up-going lines represents the 220-baud transmission long pulse. Red down-going lines represents the received signal for some target altitude.

However, for one gate lower and one gate higher than the target altitude, the received powers are conjugate multiplication of the first 219 points and

last 219 points of the original coded pulse and therefore are randomized by the binary coding. This mismatch causes a spread of power at unwanted altitudes across the Doppler frequency points. We keep doing this for all neighboring altitudes within one data record to obtain a power distribution over its Doppler frequency and altitude ranges. Integrating such power distributions of many data records over a period of time we obtain the point spread function of RI receiver shown in Figure 4.13. The PSF in Figure 4.13 has shape `(439, 220)` which comes from 1 target height plus 438 neighboring heights (219 for both above and below). Figure 4.14 shows a few horizontal cuts from the 2-dimensional PSF that represents the signal from different neighbors of the target altitude. The center plot shows the cut from target altitude that has the shape of δ function. One plot above and below show the cuts from 0.3 km above and below the target altitude, from which the signal at zero Doppler frequency peak is weaker by two orders of magnitude, and the total power is spread over entire 500 kHz bandwidth due to mismatch of coding. The plots from two gates above and below show stronger power spread over receiver bandwidth, and from the plots from 10 and 100 gates above and below the zero Doppler frequency peak can hardly be observed.

The procedure for computing the PSF for USRP receiver is similar to that for RI receiver described above except the sampled transmitted pulse length is 11000 instead of 220. At target altitude, the received signal is fully recovered by taking the conjugate multiplication of 11000 transmitted samples with itself, and the Fourier transform of the recovered rectangle function has the shape of a δ function. However, instead of shifting the signal by one gate each time for neighboring altitudes, we decide to shift 50 samples each time, followed by conjugate multiplication of the shifted signal with the original, such that the height resolution is 300 m, the same as that of the RI receiver. Figure 4.15 shows the 25 MHz bandwidth 2D PSF of USRP receiver, and Figure 4.16 shows the center 500 kHz portion for better comparison with the PSF of the RI receiver. Figure 4.17 shows the horizontal cuts at target altitude and different neighboring gates. Similar to Figure 4.14, the cut at target altitude has the shape of the δ function, and the power at neighboring gates of the target altitude spreads over the receiver bandwidth.

The PSF represents the frequency response of the system when the intrinsic spectrum to be measured is a δ function in frequency, i.e., it shows how a point source spreads over range and frequency. In our spectral inversions

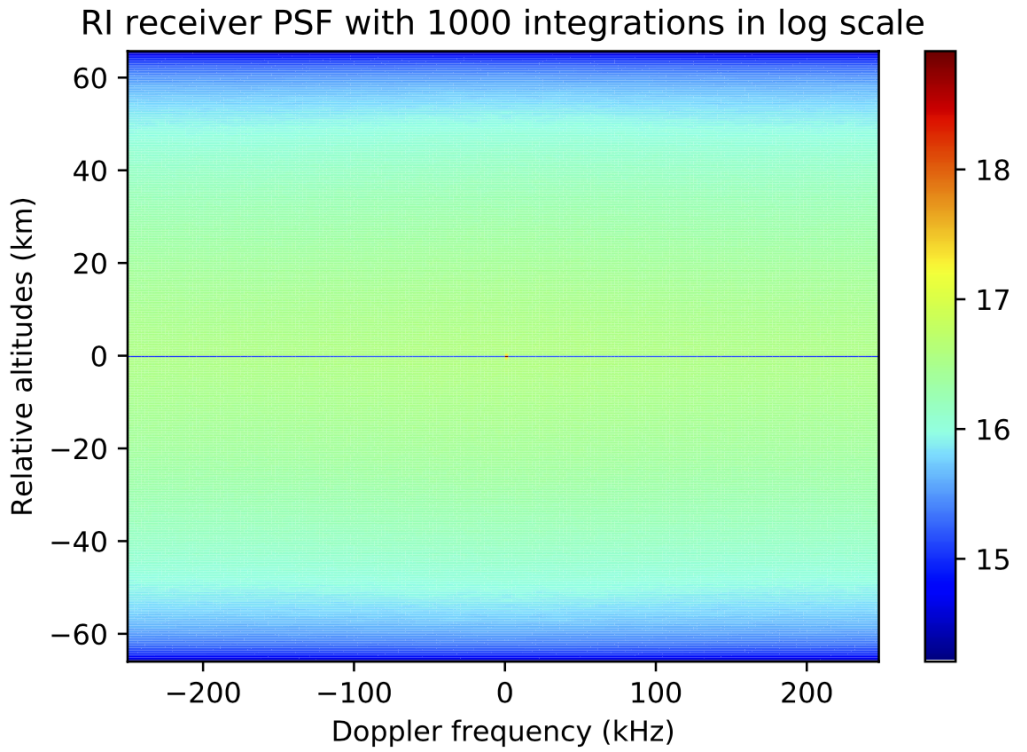

Figure 4.13: $\log_{10}$(PSF) of RI receiver by averaging over 1000 data records. The 2D spread function in relative altitudes and Doppler frequency is shown.

the PSF can be convolved with theoretical model spectra to form modeled measured spectra using a two-dimensional convolution operation. Alternatively, for CLP data, the 2D PSF can be approximated by a one-dimensional PSF over only the Doppler frequency by integrating the 2D PSF over the range axis in order to reduce computation complexity. As shown in Figure 4.14 and 4.17, only the signal at target altitude is recovered while the signal power from other neighboring altitudes is spread over the receiver bandwidth that forms the "pedestal" that elevates the entire received signal.

Figure 4.18 right and Figure 4.19 top show the 1D collapsed PSF by integrating over range axis of 2D PSF, from which we observe that the 1D PSF is composed of two parts: the center zero Doppler frequency peak that represents the signal at target altitude, and the pedestal that spreads over the bandwidth of the receiver from neighboring altitudes of the target. In practice, the 1D PSF is first convolved with the ion-line model to form a modeled spectrum, then the pedestal is fitted together with the ion-line feature (the

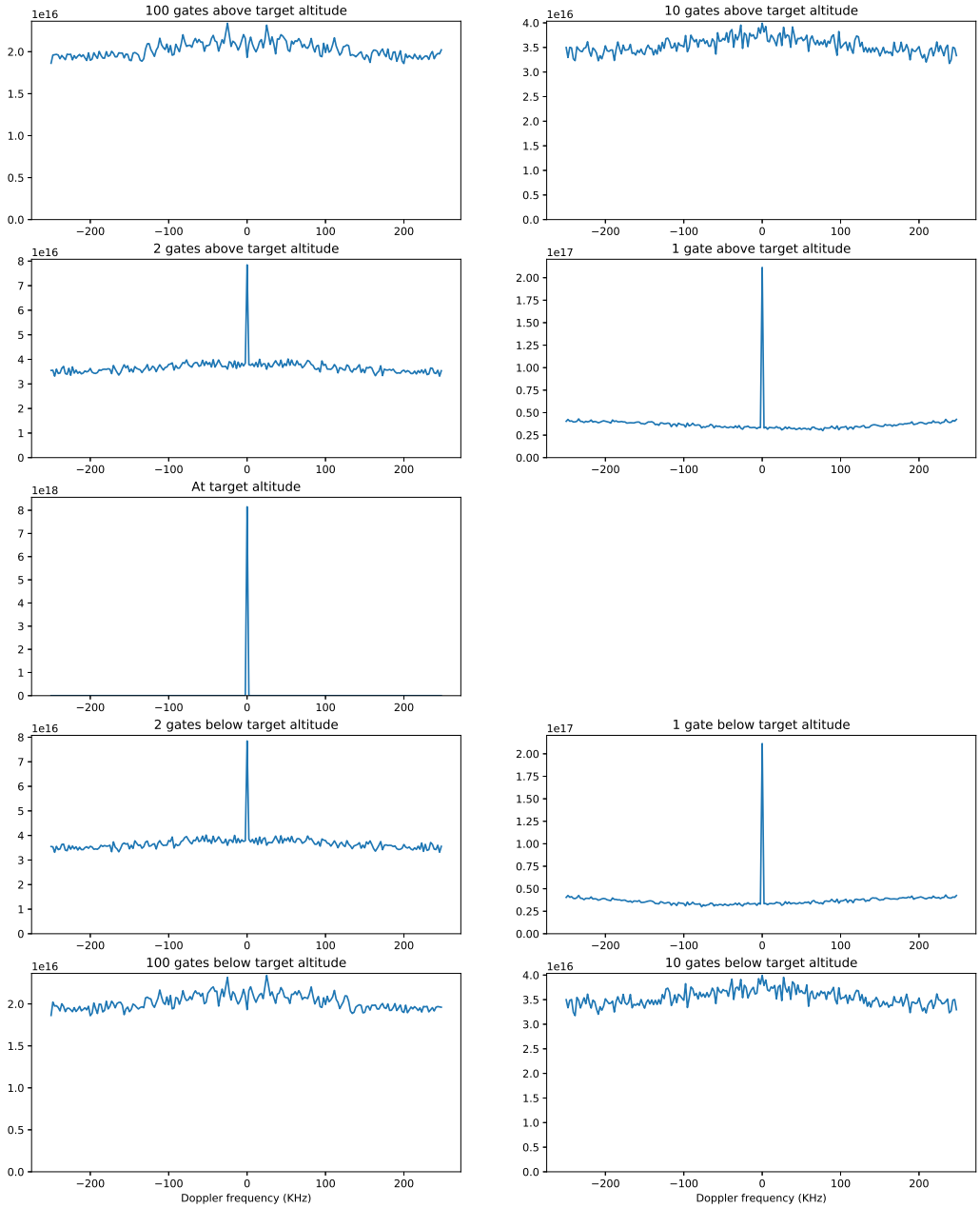

Figure 4.14: Horizontal cuts from 2D point spread function in Figure 4.13. The center plot shows the cut from target altitude that has a δ function shape. The remaining plots show the cut from 1, 2, 10, 100 gates above and below the target altitude.

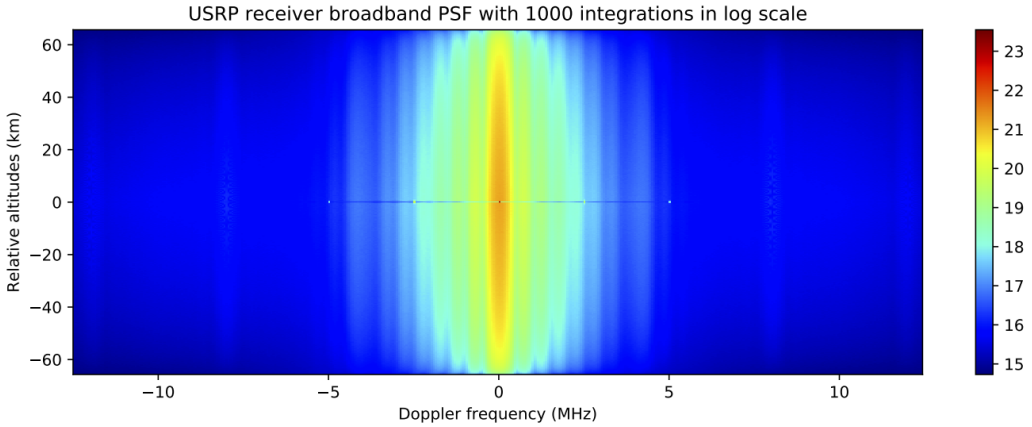

Figure 4.15: 25 MHz broadband USRP receiver point spread function.

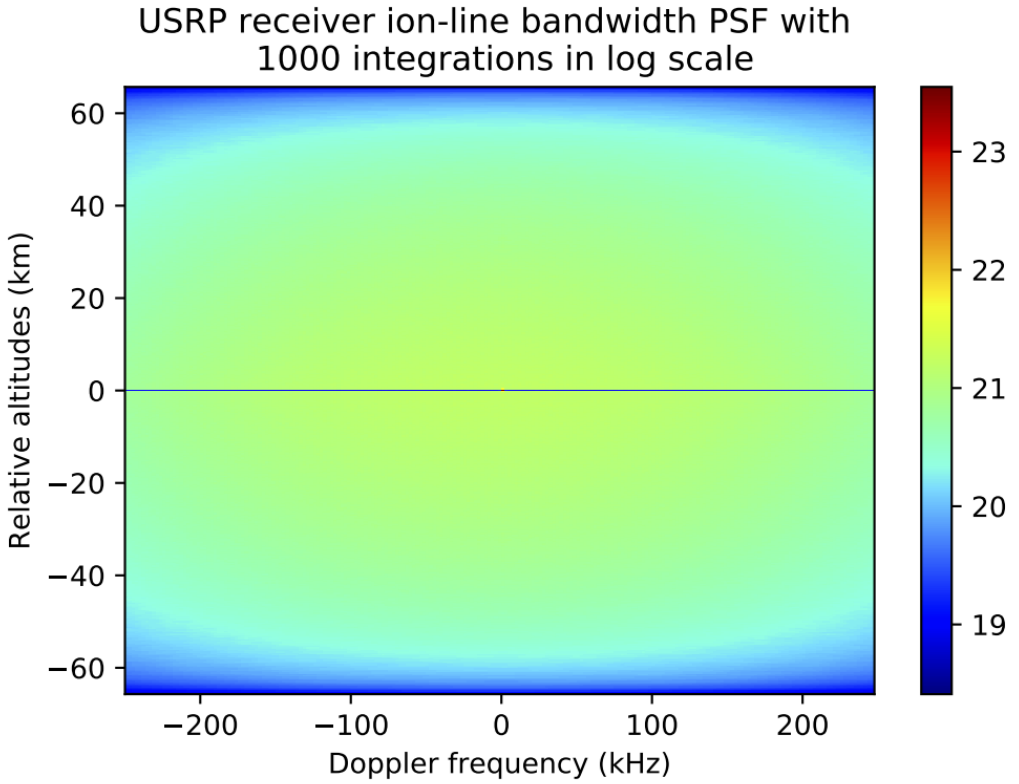

Figure 4.16: Center 500 kHz narrow-band point spread function of USRP receiver.

details of ion-line inversion are shown in Chapter 5). Figure 4.18 left and Figure 4.19 lower left show the power distribution over the illumination range of one transmitted pulse (±65.7 km of the target altitude) by integrating over the Doppler frequency axis. The triangularly shaped power distribution

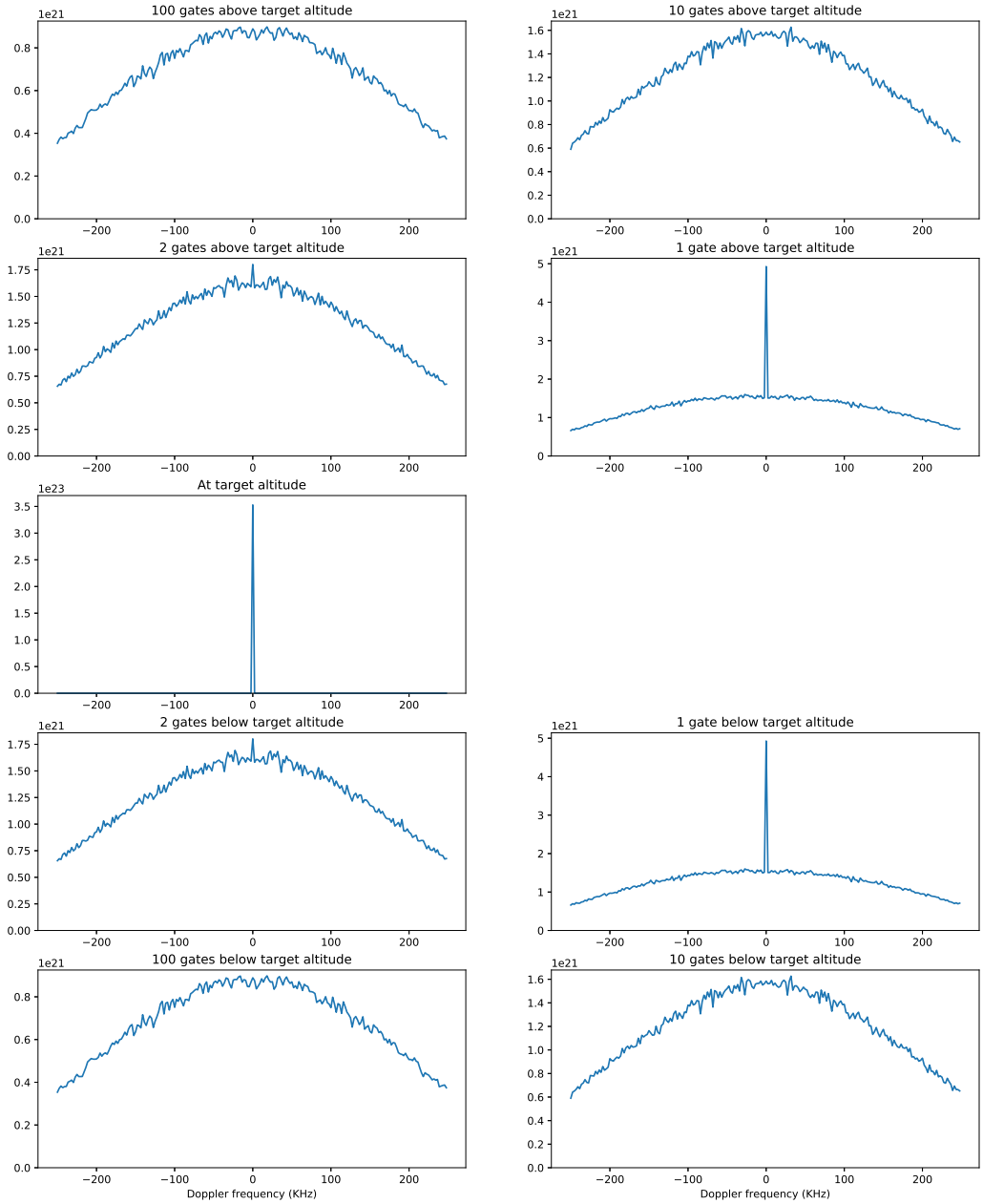

Figure 4.17: Horizontal cuts from USRP 2D point spread function in Figure 4.16. The center plot shows the cut from target altitude that has a δ function shape. The remaining plots show the cut from 1, 2, 10, 100 gates above and below the target altitude.

is consistent with the transmission and reception scheme discussed at the beginning of Section 4.4.

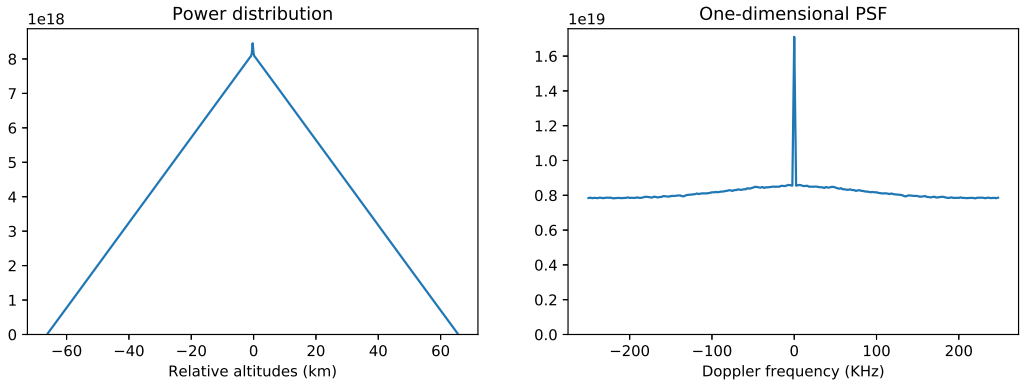

Figure 4.18: Left: Power distribution of all altitudes within the detection range for one target altitude; the power distribution has a triangular shape that agrees with the transmission scheme in Figure 4.1. Right: One-dimensional approximation to 2D PSF by integrating over altitude range.

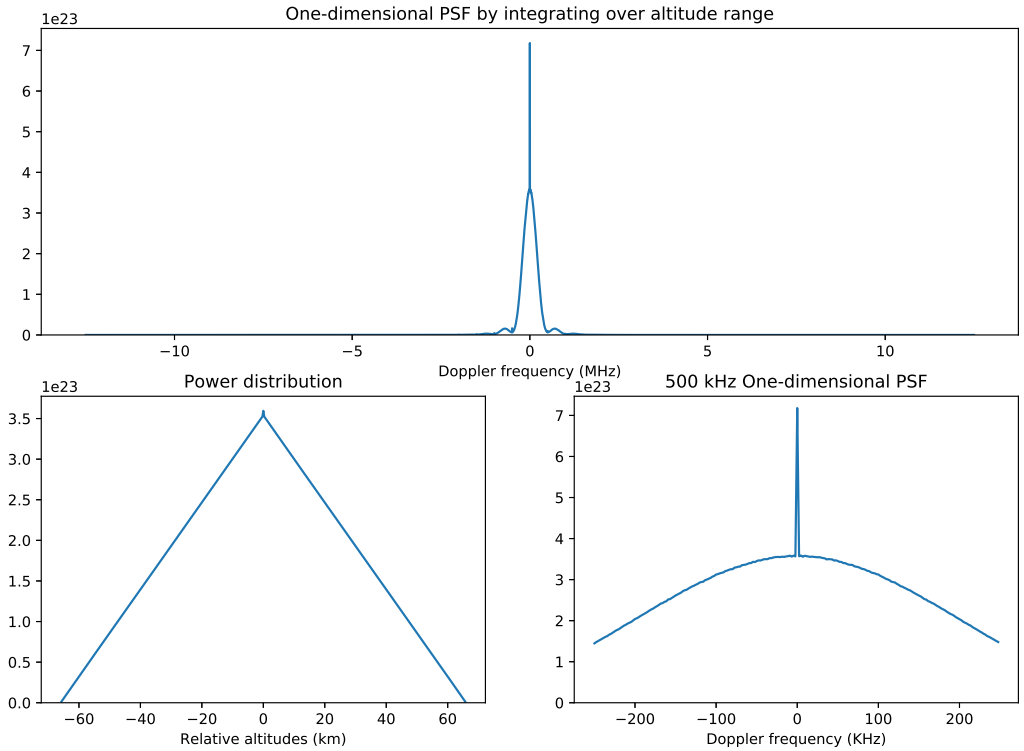

Figure 4.19: Top: Broadband one-dimensional approximation to 2D PSF by integrating over altitude range. Lower left: Power distribution of all altitudes within the detection range for one target altitude. Lower right: 1D PSF containing only 500 kHz Doppler frequency range; this frequency range is chosen to be same with RI receiver bandwidth for comparison purpose.

CHAPTER 5

INVERSION OF ARECIBO ISR CODED LONG PULSE POWER SPECTRA

In this chapter we first investigate the sensitivity of the incoherent scatter ion-line spectrum to different ionospheric state parameters that we need to invert for. With the information about how different parameters control different ion-line features, the inversion of measured ion-line power spectra can be achieved with additional assistance received from the plasma line profile regarding the plasma density distribution with height. The inversion of line-of-sight ion velocity using measured spectra ACF is also discussed to assist the measured spectrum inversion.

5.1 Incoherent Scatter Spectral Model and Ionospheric Plasma Parameters

The complete incoherent scatter (IS) forward spectral model depends on both plasma particle parameters and neutral particle parameters — the objective of ISR experiments is to invert the measured ISR spectra to extract as many as possible of these state parameters of the coupled ionosphere thermosphere system. Such efforts combined with other experimental methods form the basis of standard atmospheric and ionospheric models that we rely on in upper atmosphere science, including MSIS-00 [*Picone et al.*, 2002] and IGRF [*Thébault et al.*, 2015] from which neutral temperature, neutral particle densities, and geomagnetic field strength can be obtained for altitudes that cover the full range of CLP measured spectra. In the present work we assume that for a given time and altitude, the shape of IS spectral model depends only on the plasma parameters, i.e., electron temperature T_e, ion temperature T_i, electron density N_e, ion composition, and ion drift velocity V_i, while we rely on MSIS-00 and IGRF to specify the required neutral atmospheric and magnetic inputs.

ISR double-humped ion-line spectrum shape is found to be sensitive to the following changes of ionospheric plasma parameters:

- T_e/T_i influences peak-to-valley ratio of double-humped shape; this is the most clear feature that can be extracted from measured spectrum. Figure 5.1 shows the ion-line with different combinations of T_e and T_i.

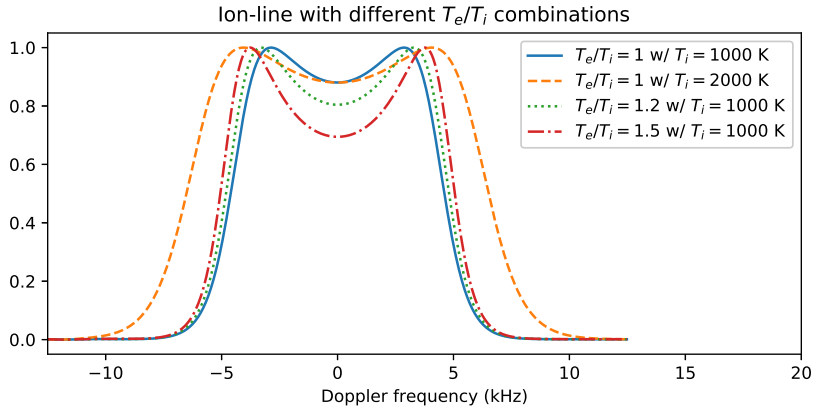

Figure 5.1: Ion-line model with different T_e T_i combinations while keeping other parameters fixed. We observe that the peak-to-valley ratio of double-humped shape depends only on T_e/T_i ratio.

- N_e greater than 10^{11} m^{-3} (a typical value during daytime) does not influence the shape of the ion-line. Therefore, information about N_e from plasma line is required. Figure 5.2 shows the change of ion-line with change of N_e.

- Ion composition influences the width of ion-line because different atomic masses of each particle cause the different decay rates of corresponding ACF, such that ion-line for each ion species exhibits different width. If multiple ion species present in one ionosphere region, the measured spectrum is a superposition of multiple ion-lines. Figure 5.3 shows the ion-line for different ions.

- V_i ion velocity influences the frequency shifts of the measured spectra. It can be extracted by fitting the phase of the auto-correlation function obtained from the measured spectrum. Figure 5.4 shows the ion-line with different V_is.

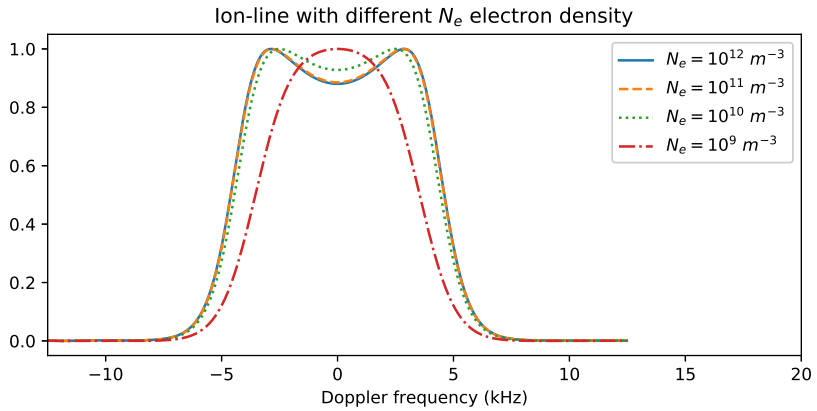

Figure 5.2: Ion-line model with different N_e values while keeping other parameters fixed. We observe that the double-humped shape saturates when $N_e \gtrapprox 10^{11}$ m^{-3}.

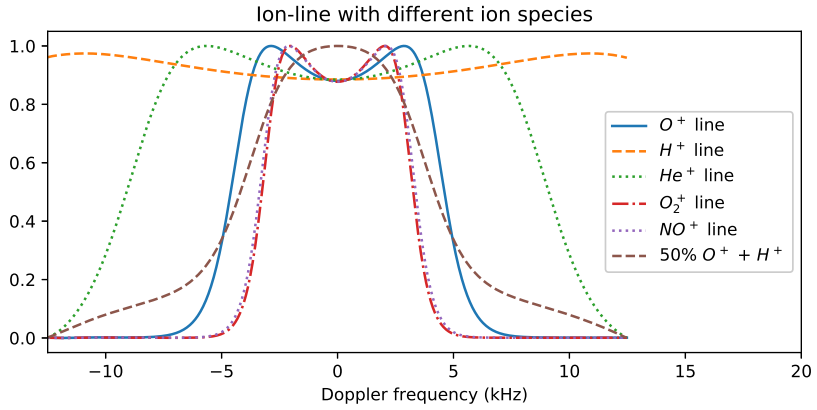

Figure 5.3: Single ion-line model with different ion species while keeping other parameters fixed. We observe that the width of double-humped shape is related to atomic mass of ion: the ion-line is narrower for heavier ion species.

5.2 Ion Velocity from Spectra ACF Inversion

Ion velocity V_i that causes the shifts of power spectrum (as shown in Figure 5.4) is difficult to invert directly because V_i is close to 0 for most of the day except during sunrise and sunset when the V_i may go beyond ± 100 m/s. Therefore, such asymmetry of spectrum due to V_i can hardly be fitted together with the rest of the ionospheric parameters. However, we circumvent such difficulty by examining the phase of the auto-correlation function (ACF) of power spectra.

Fourier transform frequency shift property states that shifts in the fre-

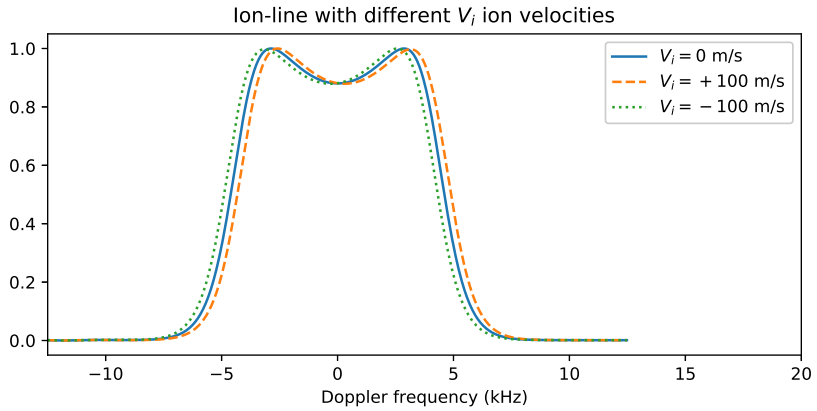

Figure 5.4: Ion-line model with different ion velocity V_i while keeping other parameters fixed. We observe that V_i causes the spectrum to shift within its Doppler frequency frame.

quency domain correspond to linear phases in time domain as follows:

$$\mathcal{F}\{x(\tau)\exp(j\omega_0\tau)\} = X(\omega - \omega_0). \tag{5.1}$$

The Doppler frequency shift is given by $\omega_0 = \mathbf{k} \cdot \mathbf{V}_i$ where $\mathbf{k}$ is Bragg wave vector and $\mathbf{V}_i$ is the ion drift velocity along $\mathbf{k}$. Theoretically, when all ion species share the same ion velocity, the phase exhibits a constant slope, while when multiple ion species have different velocities, the phase is no longer linear with respect to τ [*Vickrey et al.*, 1976]. In this thesis, we assume single ion velocity because we expect most of the ions are O^+ in the detection range of CLP spectra ($\sim 200 - 600$ km), such that the ion drift velocity can be extracted by measuring the phase slope of the inverse Fourier transform of measured spectrum. In practice, the phase slope is fitted to a linear function $\phi = kV_i\tau$ with $kV_i = |\mathbf{k}||\mathbf{V}_i|$ being the slope of the linear function; then V_i can be approximated by dividing the fitted slope by Bragg wavenumber k. This approximation of V_i provides an initial guess that can be used for power spectra inversion.

5.3 CLP Ion-line Power Spectra Least-square Inversion

Given the measured spectrum, temperatures T_s for particles of species s, plasma composition, and ion drift velocity V_i can be found by doing non-

linear least square inversion of the ISR ion-line model described in Chapter 2 to the measured spectrum.

The measured spectrum should be viewed as a convolution of the theoretical spectra for a given set of ionospheric parameters with the point spread function that is integrated over all altitudes (shown in lower left of Figure 4.13) related to the transmitted pulse shape and spectral estimation method. PSF is an indicator of how a delta function spectrum would be smeared over Doppler frequencies as a consequence pulse coding and decoding step. When doing fitting, the PSF is convolved with ion-line model to form a complete modeled spectrum.

The measured spectrum $\mathbf{S}$ is composed of the underlying true spectrum $\langle \mathbf{S} \rangle = f(\mathbf{m})$ and the zero mean random measurement error $\delta\mathbf{S}$ with variance σ^2, where $f(\mathbf{m})$ denotes the theoretical model that depends on a set of parameters $\mathbf{m}$ that includes ion and electron temperatures, ion compositions, and ion drift velocities. In the inversion problem in the thesis, we would like to find the optimized parameters $\mathbf{m}^*$ given measured spectrum $\mathbf{S}$. From maximum likelihood perspective, the probability of any set of parameters $\mathbf{m}$ is given by $p(\mathbf{m}|\mathbf{S})$. Using Bayes theorem, this *a posteriori* probability for parameters $\mathbf{m}$ can be written as

$$p(\mathbf{m}|\mathbf{S}) \propto p(\mathbf{S}|\mathbf{m})p(\mathbf{m}), \tag{5.2}$$

where $p(\mathbf{S}|\mathbf{m})$ is the probability of the measurements given the parameters $\mathbf{m}$, and $p(\mathbf{m})$ is the *a priori* probability for $\mathbf{m}$. To get started with, we assume that we have no *a priori* information about parameters $\mathbf{m}$ such that $p(\mathbf{m})$ is ignored. Then $p(\mathbf{m}|\mathbf{S})$ can be simplified to

$$p(\mathbf{m}|\mathbf{S}) \propto p(\mathbf{S}|\mathbf{m}). \tag{5.3}$$

Measured spectrum $\mathbf{S} = f(\mathbf{m}) + \delta\mathbf{S}$, $p(\mathbf{S}|\mathbf{m})$ can be considered as a Gaussian probability model with expectation $f(\mathbf{m})$ and variance σ^2, whose probability density function (pdf) is given by

$$p(\mathbf{S}|\mathbf{m}) = \frac{1}{\sqrt{(2\pi)^N |\mathbf{C}|}} \exp\left(-\frac{1}{2}[\mathbf{S} - f(\mathbf{m})]^T \mathbf{C}^{-1} [\mathbf{S} - f(\mathbf{m})]\right), \tag{5.4}$$

where $\mathbf{C}$ denotes the covariance matrix of $\delta\mathbf{S}$, and N is the length of data

vector of measured spectrum. In order to maximize $p(\mathbf{S}|\mathbf{m})$, it is equivalent to minimize $\chi^2 = [\mathbf{S}-f(\mathbf{m})]^T\mathbf{C}^{-1}[\mathbf{S}-f(\mathbf{m})]$, from which maximum likelihood (ML) solution of parameters $\mathbf{m}^*$ can be obtained.

Covariance matrix $\mathbf{C}$ for measured power spectrum is a diagonal matrix, i.e., the measurement error for every frequency bin is independent. Therefore, the residue function χ^2 can be further simplified to

$$\chi^2 = \sum_{i=0}^{N-1} \frac{|S_i - [f(\mathbf{m})]_i|^2}{\sigma_i^2}, \tag{5.5}$$

where σ_i^2 is the variance of measurement error at each frequency bin i. For the noisy signal $\mathbf{S} = f(\mathbf{m}) + \delta\mathbf{S}$, the variance σ_i^2 can be computed as

$$\begin{aligned} \sigma_i^2 = \langle \delta S_i^2 \rangle &= \langle (S_i - [f(\mathbf{m})]_i)^2 \rangle \\ &= \langle S_i^2 - 2S_i[f(\mathbf{m})]_i + [f(\mathbf{m})]_i^2 \rangle \\ &= \langle S_i^2 \rangle - 2[f(\mathbf{m})]_i^2 + [f(\mathbf{m})]_i^2 \\ &= \langle S_i^2 \rangle - [f(\mathbf{m})]_i^2. \end{aligned} \tag{5.6}$$

For measured spectrum $\mathbf{S}$ with k integrations,

$$\langle S_i^2 \rangle = \left\langle \frac{1}{k^2} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} S_{i,m} S_{i,n} \right\rangle = \frac{1}{k^2} \sum_{m=0}^{k-1} \sum_{n=0}^{k-1} \langle S_{i,m} \rangle \langle S_{i,n} \rangle, \tag{5.7}$$

where $\langle S_{i,m} \rangle \langle S_{i,n} \rangle = [f(\mathbf{m})]_i^2$ if $m \neq n$, and $\langle S_{i,m} \rangle \langle S_{i,n} \rangle = \langle S_{i,m}^2 \rangle = 2[f(\mathbf{m})]_i^2$ if $m = n$. Therefore,

$$\begin{aligned} \langle S_i^2 \rangle &= \frac{1}{k^2} \big(k(k-1)[f(\mathbf{m})]_i^2 + 2k[f(\mathbf{m})]_i^2 \big) \\ &= \frac{k-1}{k}[f(\mathbf{m})]_i^2 + \frac{2}{k}[f(\mathbf{m})]_i^2, \end{aligned} \tag{5.8}$$

from which we obtain

$$\begin{aligned} \sigma_i^2 &= \langle S_i^2 \rangle - [f(\mathbf{m})]_i^2 \\ &= \frac{k-1}{k}[f(\mathbf{m})]_i^2 + \frac{2}{k}[f(\mathbf{m})]_i^2 - [f(\mathbf{m})]_i^2 \\ &= \frac{1}{k}[f(\mathbf{m})]_i^2. \end{aligned} \tag{5.9}$$

The σ_i^2 result above shows that the variance of measurement error is propor-

tional to the magnitude square of underlying true spectrum (or theoretical model) and the integration time. Plugging the σ_i^2 result (5.9) into the residue function χ^2 (5.5), we obtain

$$\begin{aligned}\chi^2 &= \sum_{i=0}^{N-1} \frac{|S_i - [f(\mathbf{m})]_i|^2}{\sigma_i^2} \\ &= k \sum_{i=0}^{N-1} \left(\frac{S_i}{[f(\mathbf{m})]_i} - 1 \right)^2\end{aligned} \tag{5.10}$$

that can be implemented numerically.

In cases when *a priori* information for $\mathbf{m}$ is available, the probability $p(\mathbf{m}|\mathbf{S})$ in (5.3) is modified to

$$p(\mathbf{m}|\mathbf{S}) \propto p(\delta\mathbf{S}|\mathbf{m})p(\mathbf{m}). \tag{5.11}$$

Assuming the set of parameters $\mathbf{m}$ with length M has Gaussian distribution with expected value $\mu_\mathbf{m}$ and variance $\sigma_\mathbf{m}^2$, we obtain

$$\begin{aligned}p(\mathbf{m}|\mathbf{S}) \propto \exp &\left(-\frac{1}{2} [\mathbf{S} - f(\mathbf{m})]^T \mathbf{C}^{-1} [\mathbf{S} - f(\mathbf{m})] \right) \\ \cdot \exp &\left(-\frac{1}{2} [\mathbf{m} - \mu_\mathbf{m}]^T \mathbf{C}_\mathbf{m}^{-1} [\mathbf{m} - \mu_\mathbf{m}] \right),\end{aligned} \tag{5.12}$$

where the covariance matrix $\mathbf{C}_\mathbf{m}$ is a diagonal matrix $\mathrm{diag}(\sigma_\mathbf{m})$ assuming all parameters in $\mathbf{m}$ are independent of each other. In cases when covariance matrices are known, the residue function (5.5) becomes

$$\chi^2 = \sum_{i=0}^{N-1} \frac{|S_i - [f(\mathbf{m})]_i|^2}{\sigma_i^2} + \sum_{j=0}^{M-1} \frac{|m_j - \mu_{\mathbf{m},j}|^2}{\sigma_{\mathbf{m},j}^2}. \tag{5.13}$$

The computations were coded in Python, which is a computation platform that includes a repertoire of optimization packages that can be used for performing such inversion tasks. The package routine we use is `scipy.optimization.least_squares` which allows us to set empirical bounds on inversion ionospheric parameters (which can be considered as another kind of *a priori* information when $p(\mathbf{m})$ is uniformly distributed within some physical ranges) to improve convergence.

5.4 Program Implementation and September 2016 Campaign Results

In practice, the inversion program is composed of several parts. Figure 5.5 shows the complete steps during the inversion process. In this setup, we use the two-ion model including O^+ and H^+. The ion-line model is formed with some initial values and empirical bounds summarized in Table 5.1.

Table 5.1: Initial guesses and empirical bounds for inversion parameters.

Parameter	Initial guess	Empirical bound
T_i	1000 K	$[0\text{K}, +\infty]$
T_e	1000 K	$[T_i, +\infty]$
$\%O^+$	100%	$[0\%, 100\%]$
V_i	from spectrum ACF	initial ± 10 m/s
n	0.5	$[-1, 1]$

After the ion-line model is convolved with the 1D PSF, both modeled and measured spectra are normalized with respect to their peak value as we only consider the shape change of ion-line without the effect of magnitude change. In addition to normalization, the modeled spectrum is added with a "pseudo noise level" parameter because the measured spectrum contains some background cosmic noise that elevates the spectrum. The "noise level" parameter n is added using the following formula:

$$S_{\text{model w/ noise}} = S_{\text{model}} \times (1 - n) + n \text{ for } n \in [-1, 1]. \tag{5.14}$$

The effect of parameter n is to compress or stretch the modeled spectrum based on its value while maintaining the magnitude of the normalized spectrum. Note that this n is set to bound between $[-1, 1]$ because in practice we find that the n can sometimes be negative, meaning the modeled spectrum is stretched. One possible reason for this negative n value is that the 1D PSF contains some noise level from the receiver (the PSF is computed from sampled transmitted pulse from receiver). After the convolution, such receiver noise is added to the modeled spectrum, causing the noise level of modeled spectrum to be higher than that of the measured spectrum. Therefore, the parameter n can also be interpreted as the scaling parameter as it scales the modeled spectrum, and the inversion program finds the best value that matches the pedestals of modeled spectrum and the measured spectrum

during the inversion.

The scaling factor n also provides the SNR information of measured spectrum. For n values in the range $[-1, 1]$, smaller n values imply higher SNR, while for $n \to 1$, SNR$\to 0$.

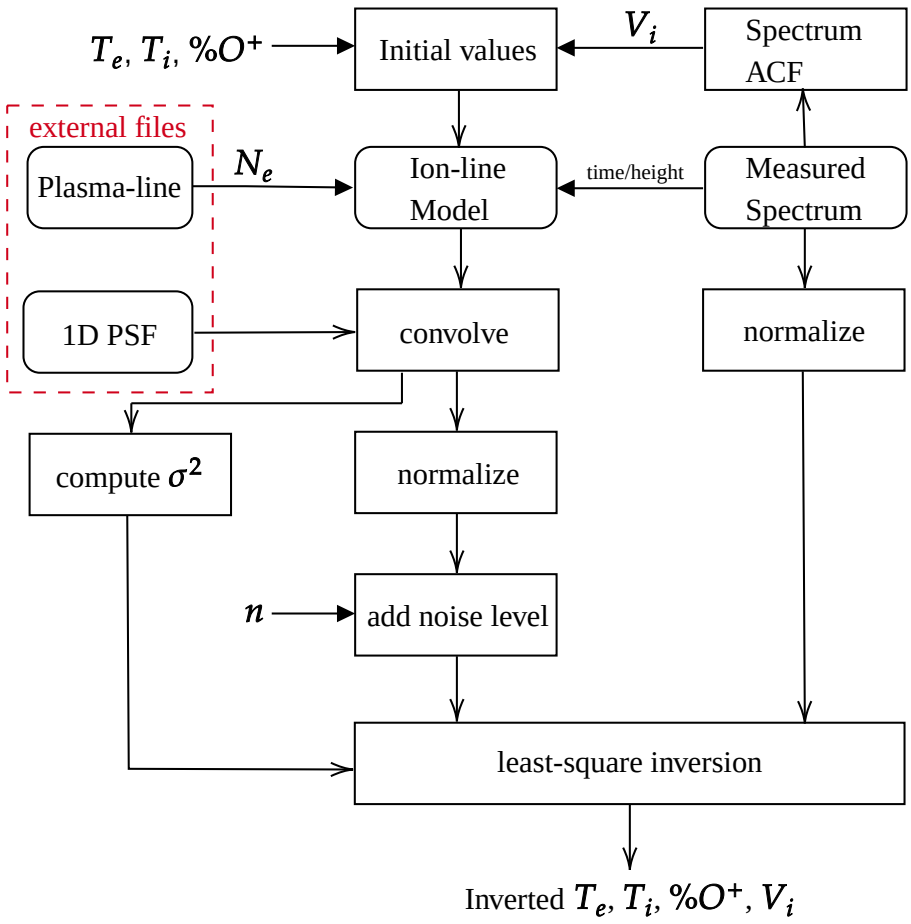

Figure 5.5: Complete inversion steps used in implementation. The rounded rectangles stand for the data or model, and the regular rectangles stand for operations. The solid arrows stand for data input, and the regular arrows stand for the computation order. The plasma-line data and PSF data are pre-computed and stored in files before inversion.

The September 2016 campaign CLP spectra are inverted using the inversion program described above. To illustrate the inversion performance, we first show the inversion results from a single altitude spectrum. The measured spectrum is obtained from September 24, 2016, at 2 PM local time, at 225 km altitude. The inverted spectrum and measured spectrum are shown

in Figure 5.6, and the inversion results are tabulated in Table 5.2.

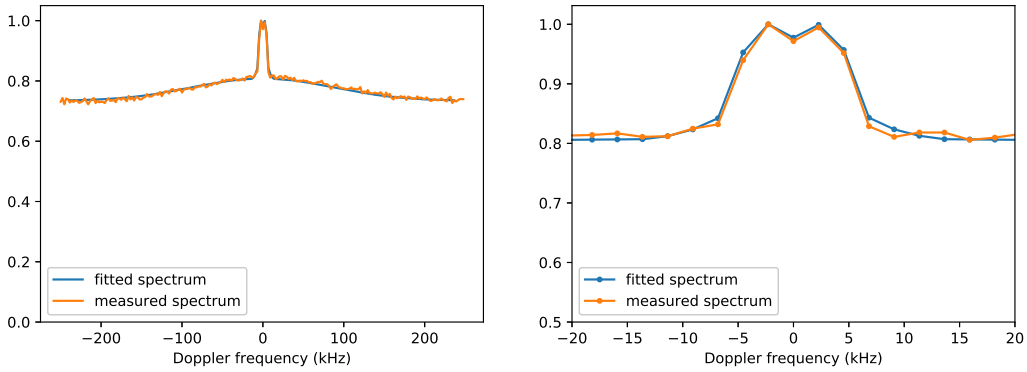

Figure 5.6: Measured spectrum and fitted spectrum from September 24, 2016, at 2 PM local time, at 225 km altitude. Left plot shows the inversion for 500 kHz bandwidth spectrum, right plot shows the center 40 kHz bandwidth spectrum that contains the ion-line feature.

Table 5.2: Inversion results for measured spectrum from September 24, 2016, at 2 PM local time, at 225 km altitude.

Parameters	Inversion results
T_e	1684.50 K
T_i	977.58 K
$\%O^+$	78%
V_i	9.62 m/s
n	-0.141

Next the inversion program is run on the entire dataset of the September 2016 campaign. Figure 5.7 top shows the electron density map used in the inversion program. The map is composed of the N_e obtained from plasma-line measurements between $\sim$ 100 and 400 km during the daytime of the September 2016 campaign and the deconvolved ion-line power profile for the rest of the altitudes and nighttimes on the map (see *Wang* [2019]). Figure 5.7 bottom shows the inverted scaling factor n that serves as the SNR map for the measured spectra.

Figures 5.8 and 5.9 show the inverted T_i, T_e, $\%O^+$, and V_i parameters when the SNR is large. The results are plotted for the altitudes when $n < 0.5$, which corresponds to SNR $\gtrapprox 1$. In all four plots, the inverted parameters are found to have regular periodic structures over 24 hours periods. During the daytime when electron density is high, the valid inversion results with high

SNR extend above 500 km altitude, while during the nighttime when electron density is low, the backscattered signal power is weaker, such that the detection range drops below 400 km. In Figure 5.8, inverted T_i and T_e between $\sim$ 6 AM and $\sim$ 6 PM have greater T_e/T_i ratio on all September 23, 24, and 25 results, while after sunset, the results show $T_e/T_i \approx 1$ that matches our expectation. Figure 5.9 top shows the inverted O^+ ion percentage. In most regions the percentage is $70-80\%$. However, we expect most ions below 400 km altitude ionosphere are O^+. The contradiction between the inversion and our expectation may be caused by the ambiguous ion-line shape between different T_e, T_i combinations and multi-ion density combinations. Further study will be conducted regarding this ambiguity among different combinations of ionosphere fitting parameters including the verification of inversion results with uncoded long pulse (ULP) data. Figure 5.9 bottom shows the inverted ion drift velocity V_i. The positive velocities shown in red represent the upward ion drifting direction while the blue velocities represent the opposite. Since the initial V_i are obtained from measured spectrum ACF and this parameter is heavily bounded during the spectrum inversion, the inverted V_i is trustworthy as long as the initial values are well approximated from ACF. From another perspective, inverted V_i matches the vertical movements of the ionosphere region: When the entire ionosphere is moving downwards during nighttime (from $\sim$ 2 AM to $\sim$ 6 AM), the V_i plot shows blue negative values, meaning the ions drift downward, and the more intense blue color stands for faster downward movement of the ionosphere. A similar pattern can be obtained for upward movements of the ionosphere during the sunrise (from $\sim$ 7 AM to $\sim$ 10 AM), showing the inverted V_i is trustworthy.

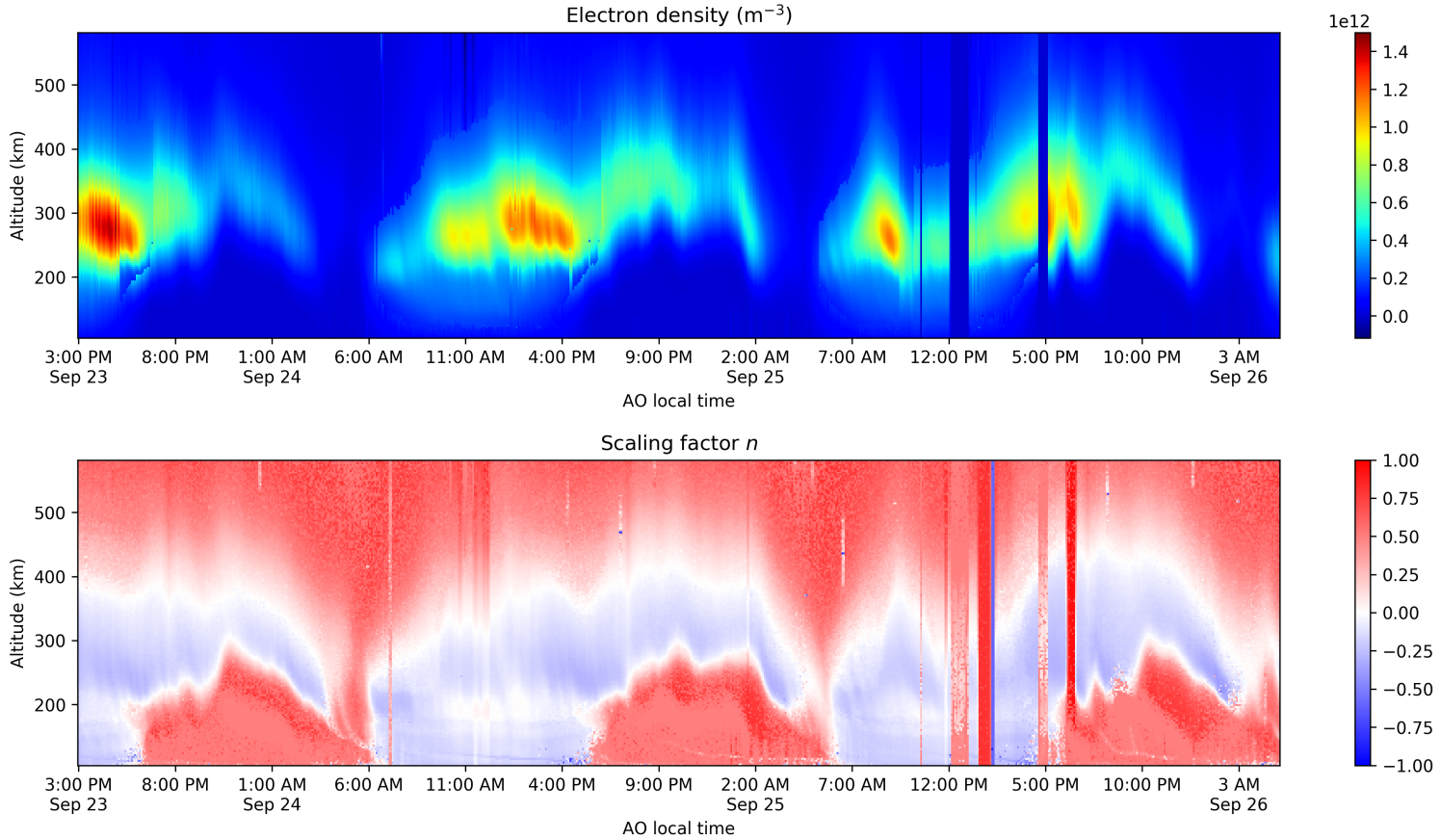

Figure 5.7: Top: The electron density N_e map used for September 2016 campaign inversion. Bottom: The inverted scaling factor n that serves as the SNR map for September 2016 campaign; higher value of n stands for higher background noise level, which implies lower SNR.

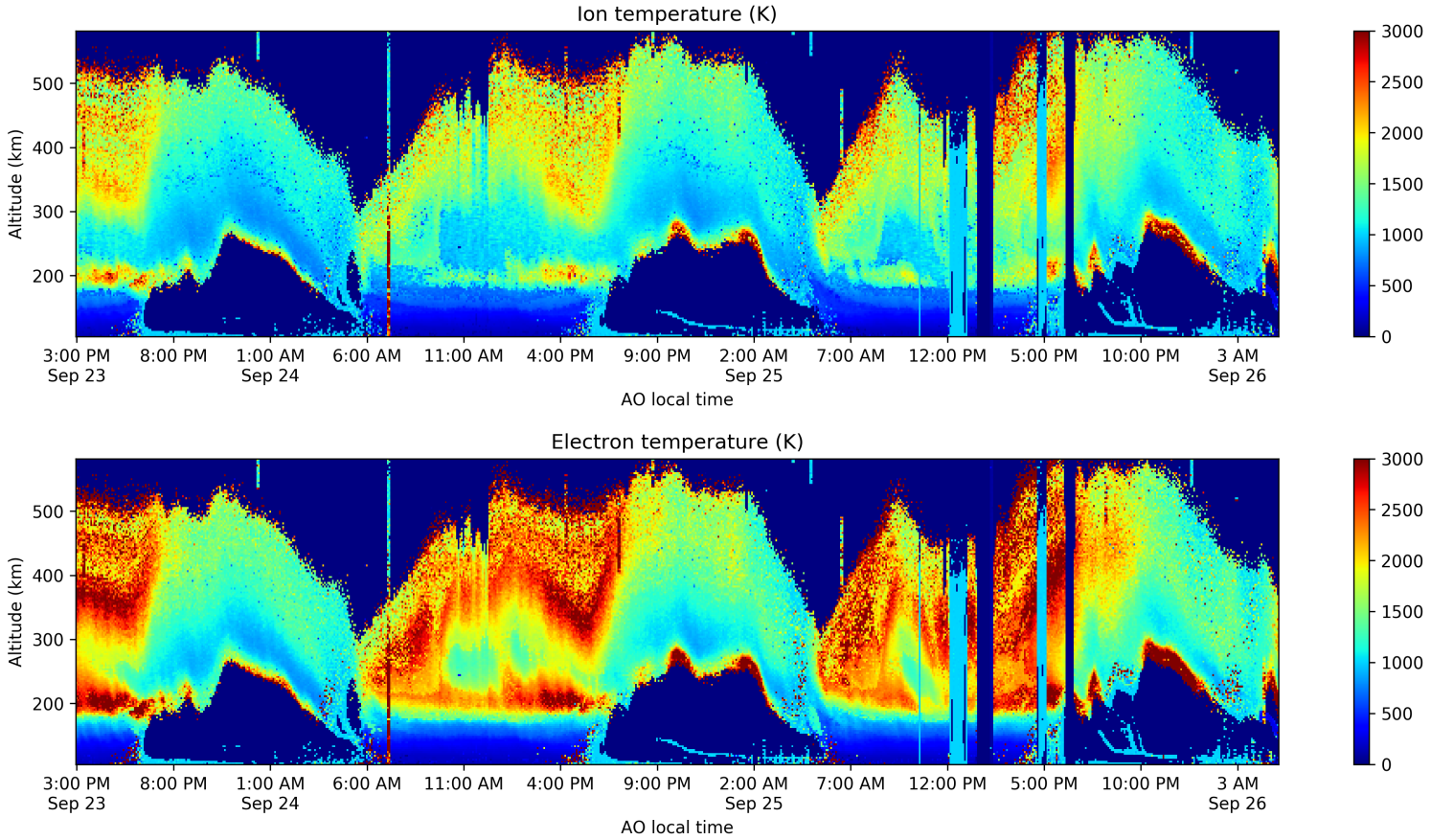

Figure 5.8: Top: Inverted ion temperatures for September 2016 campaign, from 3 PM on September 23 to 5 AM on September 26. Bottom: Inverted electron temperatures for September 2016 campaign, from 3 PM on September 23 to 5 AM on September 26. The inverted results are shown only for $n < 0.5$ cases (when SNR $\gtrapprox 1$).

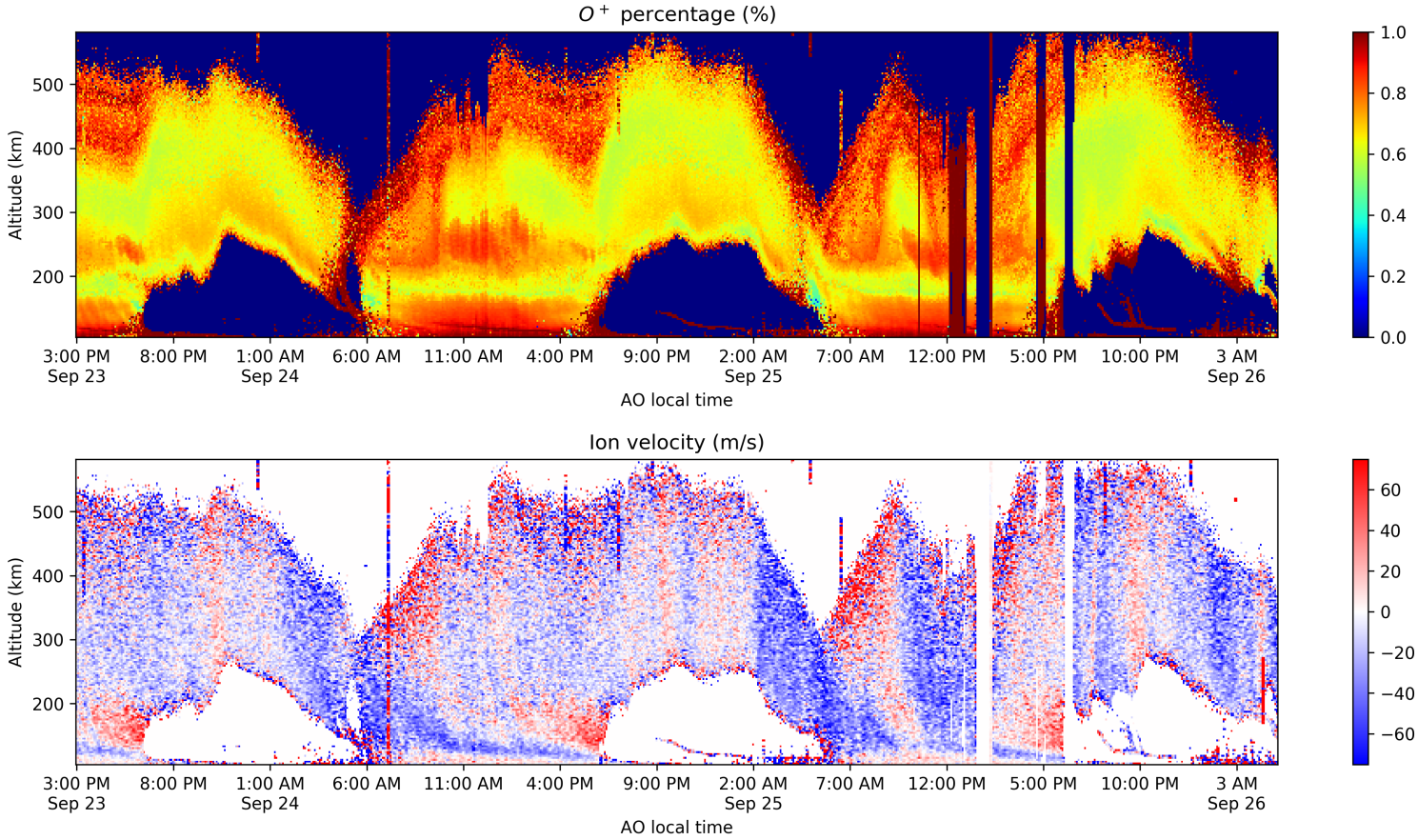

Figure 5.9: Top: Inverted oxygen ion percentage for September 2016 campaign, from 3 PM on September 23 to 5 AM on September 26. Bottom: Inverted ion velocity for September 2016 campaign, from 3 PM on September 23 to 5 AM on September 26. The inverted results are shown only for $n < 0.5$ cases (when SNR $\gtrapprox 1$).

CHAPTER 6

CONCLUSION AND FUTURE WORK

Incoherent scatter consists of Thomson scattering from collections of free electrons in the ionosphere. The information about ionospheric charge carriers including electron and ion temperatures, bulk velocities, and plasma compositions can be obtained by inverting the frequency spectrum of incoherent backscattered radar signals from the ionosphere. In this study, the complete incoherent scatter forward model including the effects of Coulomb collision, ion-neutral collision, ambient geomagnetic field, and aspect angle was computed and examined using the chirp-z transform method. The ion-line located at the center of the full incoherent spectral model has a double-humped shape that is controlled by ionospheric parameters T_e, T_i, V_i, and N_e, assuming a single ion species model. Investigating the characteristics of the ion-line model with changing ionospheric parameters allows us to use a flexible inversion scheme for different parameters. The weighted least square program is applied to invert the normalized ion-line model and measured spectrum to improve convergence. The ion velocity V_i is inverted using ACF of measured spectrum and is used as an initial input to the ion-line inversion.

This study is one part of the research that the Arecibo ISR group conducts including inversion of CLP, uncoded long pulse (ULP), and plasma line spectra. Plasma line spectra provide the electron density profile that is used for CLP and USP spectra inversion. While CLP and ULP spectra inversion cross-validates both inversion results, the electron density profile from plasma line is also validated from ion-line spectra using the method presented in this thesis. Even though three inversions are working independently, ultimately we will combine them into a single program as a part of future work.

REFERENCES

Isham, B., C. Tepley, M. Sulzer, Q. Zhou, M. Kelley, J. Friedman, and S. González, Upper atmospheric observations at the Arecibo Observatory: Examples obtained using new capabilities, *Journal of Geophysical Research: Space Physics*, *105*(A8), 18,609–18,637, 2000.

Kelley, M. C., *The Earth's Ionosphere*, International Geophysics Series, 487 pp., Academic Press, 1989.

Kudeki, E., and M. A. Milla, 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 PART 2), 315–328, doi:10.1109/TGRS.2010.2057252, 2011.

Li, Y. L., S. Franke, and C. H. Liu, Numerical implementation of an adaptive fast-field program for sound propagation in layered media using the chirp z transform, *The Journal of the Acoustical Society of America*, *89*(5), 2068–2075, doi:10.1121/1.400896, 1991.

Milla, M., and E. Kudeki, Particle dynamics description of "BGK collisions" as a Poisson process, *Journal of Geophysical Research: Space Physics*, *114*(7), doi:10.1029/2009JA014200, 2009.

Picone, J. M., A. E. Hedin, D. P. Drob, and A. C. Aikin, NRLMSISE-00 empirical model of the atmosphere: Statistical comparisons and scientific issues, *Journal of Geophysical Research: Space Physics*, *107*(A12), SIA 15–1–SIA 15–16, doi:10.1029/2002JA009430, 2002.

Rishbeth, H., and O. K. Garriott, *Introduction to Ionospheric Physics*, Academic Press, 1969.

Sulzer, M. P., A radar technique for high range resolution incoherent scatter autocorrelation function measurements utilizing the full average power of klystron radars, *Radio Science*, *21*(06), 1033–1040, 1986.

Thébault, E., et al., International Geomagnetic Reference Field: The 12th generation, *Earth, Planets and Space*, *67*(1), 79, doi:10.1186/s40623-015-0228-9, 2015.

Vickrey, J. F., W. E. Swartz, and D. T. Farley, Incoherent scatter measurements of ion counterstreaming, *Geophysical Research Letters*, *3*(4), 217–220, doi:10.1029/GL003i004p00217, 1976.

Vierinen, J., B. Gustavsson, D. L. Hysell, M. P. Sulzer, P. Perillat, and E. Kudeki, Radar observations of thermal plasma oscillations in the ionosphere, *Geophysical Research Letters*, *44*(11), 5301–5307, doi:10.1002/2017GL073141, 2017.

Wang, B., Full-profile inversion of ionospheric radar data from Arecibo Observatory, Master's thesis, University of Illinois at Urbana-Champaign, 2019.

APPENDIX A

SPECTRAL MODEL COMPUTATION CODE

The following codes complete the implementation of spectral forward model used in the ion-line inversion. The code is composed of three files: `modfit.py`, `cztspec.py`, and `modfit.py`, shown below respectively.

```
from pylab import *
import h5py
import isrpy.igrf.igrf12 as igrf
from datetime import datetime, timedelta
import isrpy.beampack.AObeam as AObeam
import pyglow
import os

# al = [list of desired altitudes]
# dt = [year, month, day, hour]
# ll = [latitude, longitude]

def iri(al,dt,ll):
    lat = ll[0]
    lon = ll[1]
    alts = al
    dn = datetime(int(dt[0]), int(dt[1]), int(dt[2]),
                  int(dt[3]), int((dt[3]-int(dt[3]))*60))

    Tn = []   #Neutral_temp, K
    Ti = []   #Ion_temp, K
    Te = []   #Electron_temp, K
    Ne = []   #Electron_density_Ne, m-3
    NiO = []  #O_ions, %
    NiH = []  #H_ions, %
    NiHe = [] #He_ions, %
    NiO2 = [] #O2_ions, %
    NiNO = [] #NO_ions, %

    for h in alts:
```

```
        pt = pyglow.Point(dn, lat, lon, h)
        pt.run_iri()
        Tn.append(pt.Tn_iri)
        Ti.append(pt.Ti)
        Te.append(pt.Te)
        Ne.append(pt.ne)
        NiO.append(pt.ni['O+'])
        NiH.append(pt.ni['H+'])
        NiHe.append(pt.ni['HE+'])
        NiO2.append(pt.ni['O2+'])
        NiNO.append(pt.ni['NO+'])

    T = [Tn,Ti,Te]
    N = [Ne,NiO,NiH,NiHe,NiO2,NiNO]
    return T,N

def msis(al,dt,ll):
    lat = ll[0]
    lon = ll[1]
    alts = al
    dn = datetime(int(dt[0]), int(dt[1]), int(dt[2]),
                  int(dt[3]), int((dt[3]-int(dt[3]))*60))

    Tn = [] # neutral temperature
    Nn = [] # Neutral density
    An = [] # Automic density
    automic_mass = array([16,28,32,4,40,1,14],dtype=float)# O,N2,O2,He,Ar,H,N

    for h in alts:
        pt = pyglow.Point(dn, lat, lon, h)
        pt.run_msis()
        nn = array([pt.nn['O'],
                    pt.nn['N2'],
                    pt.nn['O2'],
                    pt.nn['HE'],
                    pt.nn['AR'],
                    pt.nn['H'],
                    pt.nn['N']])
        Tn.append(pt.Tn_msis)
        An.append(sum(nn*automic_mass))
        Nn.append(sum(nn))
    Tn = array(Tn); An = array(An); Nn = array(Nn)
    An = An/Nn

```

```
    nu_en = Nn*1.0e6*5.4e-16*sqrt(Tn)
    nu_in = Nn*1.0e6*2.6e-15*An**(-1./2)
    nu_neutral = [Nn,nu_en,nu_in]
    return nu_neutral

def mag(al,dt,ll):
    B = []; aspect = []
    for h in al:
        B = append(B,igrf.igrf_B(dt[0],h,ll[1],ll[0])[3]*1.0e-9) # [T] unit

        # the aspect angle in igrf is defined to be the angle between
        #the vector perp to B and k vector (k points to ground)
        # we use pi-aspect to use the angle when k points to sky
        aspect = append(aspect,pi-AObeam.aspect_elaz(dt[0],h,pi/2.,0.)[5])
        #[rad] unit
    return B,aspect
```

```
from pylab import *
import warnings
# ignore division by zero warning
warnings.filterwarnings("ignore")

# Physical Paramters (MKS):
me=9.1093826e-31 # Electron mass in kg
mO=1836.152*16.*me # Ion mass
mH=1836.152*1*me # Ion mass
mHe=1836.152*4*me # Ion mass
mO2=1836.152*16.*2*me # Ion mass
mNO=1836.152*(16.+14.)*me # Ion mass

qe=1.60217653e-19 # C (Electron charge)
K=1.3806505e-23 # Boltzmann cobstant m^2*kg/(s^2*K);
eps0=8.854187817e-12 # F/m (Free-space permittivity)
c=299.792458e6 # m/s (Speed of light)
re=2.817940325e-15 # Electron radius

fradar=430.0e6 # Radar Frequency (Hz)
lam=c/fradar/2.
kB=2*pi/lam # Bragg wavenumber kB = 2*ko

def chirpz(g,n,dt,dw,wo):
    """transforms g(t) into G(w)
    g(t) is n-point array and output G(w) is (n/2)-points starting at wo
```

```
    dt and dw, sampling intervals of g(t) and G(w), and wo are
    prescribed externally in an idependent manner
    --- see Li, Franke, Liu [1991]"""
    g[0]=0.5*g[0] # first interval is over dt/2, and hence ...
    W = exp(-1j*dw*dt*arange(n)**2/2.)
    S = exp(-1j*wo*dt*arange(n)) # frequency shift by wo
    x = g*W*S; y = conj(W)
    x[n/2:] = 0.; y[n/2:] = y[0:n/2][::-1] # treat 2nd half of x and y specially
    xi = fft(x); yi = fft(y); G = dt*W*ifft(xi*yi) #in MATLAB use ifft then fft (EK)
    return G[0:n/2]

def ChirpZ(f,n,dt,dw,wo):
    """transforms f(t) into F(w)
    f(t) is n-point array and output F(w) is n-points starting at wo.
    dt and dw, sampling intervals of f(t) and F(w), and wo are
    prescribed externally in an idependent manner
    --- see Li, Franke, Liu [1991]"""
    #f[0]=0.5*f[0] # first interval is over dt/2, and hence ...
    W = exp(-1j*dw*dt*arange(n)**2/2.)
    S = exp(-1j*wo*dt*arange(n)) # frequency shift by wo
    x = f*W*S;
    xi = fft.fft(x,2*n); # zero padded fft into 2n-points
    y = exp(1j*dw*dt*arange(2*n)**2/2.)
    y[n:] = y[0:n][::-1] # treat 2nd half of y specially
    yi = fft.fft(y);  # 2n point fft
    F = dt*W*fft.ifft(xi*yi)[0:n] #in MATLAB use ifft then fft (EK)
    return F

def cc_sp(Ns,Np,Cs,Cp,ms,mp,qs,qp,deb_plasma):

    # calcaulte Coulomb collision between particle species s and p

    # INPUT:
    #    Ns, Np: concentrations of particle s and p in m^-3 unit
    #
    #    Cs, Cp: thermal speeds sqrt(K*T/m) in m/s unit
    #
    #    ms, mp: particle mass in kg unit
    #
    #    qs, qp: particle charge, should equals 1.60217e-19 Coulumb
    #
    #    deb_plasma: Debye length of plasma
    #
    # OUTPUT:
```

```
71      #    nusp: Coulumb collision frequency between two species
72
73      vTsp=sqrt(2*(Cs**2+Cp**2))
74      msp=ms*mp/(ms+mp)
75      bm_sp=qs*qp/(4*pi*eps0)/msp/vTsp**2
76      log_sp=log(deb_plasma/bm_sp)
77      nusp=Np*qs**2*qp**2*log_sp/(3*pi**(3/2)*eps0**2*ms*msp*vTsp**3)
78      return nusp
79
80  def Coulomb_collision(N,C,debp):
81
82      # calcualte Coubomb collision for all charge carrier species
83
84      # INPUT:
85      #    N: electron and ion concentrations in m^-3 unit
86      #       should be in tuple format as shown below
87      #
88      #    C: thermal speeds for electrons and ions in m/s
89      #       should be in tuple format as shown below
90      #
91      #    debp: Plasma Debye length
92      #
93      # OUTPUT:
94      #    nue,nuO,nuH,nuHe,nuO2,nuNO: Coulumb collision frequency for each species
95
96      Ne,NiO,NiH,NiHe,NiO2,NiNO = N
97      Ce,CiO,CiH,CiHe,CiO2,CiNO = C
98      # electron
99      nu_ee  = cc_sp(Ne,Ne  ,Ce,Ce  ,me,me ,qe,qe,debp)
100     nu_eO  = cc_sp(Ne,NiO ,Ce,CiO ,me,mO ,qe,qe,debp)
101     nu_eH  = cc_sp(Ne,NiH ,Ce,CiH ,me,mH ,qe,qe,debp)
102     nu_eHe = cc_sp(Ne,NiHe,Ce,CiHe,me,mHe,qe,qe,debp)
103     nu_eO2 = cc_sp(Ne,NiO2,Ce,CiO2,me,mO2,qe,qe,debp)
104     nu_eNO = cc_sp(Ne,NiNO,Ce,CiNO,me,mNO,qe,qe,debp)
105     nue=nu_ee+nu_eO+nu_eH+nu_eHe+nu_eO2+nu_eNO
106     nuel=nu_eO+nu_eH+nu_eHe+nu_eO2+nu_eNO
107     nuep=nuel+nu_ee
108
109     # oxygen
110     nu_Oe  = cc_sp(NiO,Ne  ,CiO,Ce  ,mO,me ,qe,qe,debp)
111     nu_OO  = cc_sp(NiO,NiO ,CiO,CiO ,mO,mO ,qe,qe,debp)
112     nu_OH  = cc_sp(NiO,NiH ,CiO,CiH ,mO,mH ,qe,qe,debp)
113     nu_OHe = cc_sp(NiO,NiHe,CiO,CiHe,mO,mHe,qe,qe,debp)
114     nu_OO2 = cc_sp(NiO,NiO2,CiO,CiO2,mO,mO2,qe,qe,debp)
```

```
115     nu_ONO = cc_sp(NiO,NiNO,CiO,CiNO,mO,mNO,qe,qe,debp)
116     nuO=nu_Oe+nu_OO+nu_OH+nu_OHe+nu_OO2+nu_ONO
117
118     # hydrogen
119     nu_He  = cc_sp(NiH,Ne  ,CiH,Ce  ,mH,me ,qe,qe,debp)
120     nu_HO  = cc_sp(NiH,NiO ,CiH,CiO ,mH,mO ,qe,qe,debp)
121     nu_HH  = cc_sp(NiH,NiH ,CiH,CiH ,mH,mH ,qe,qe,debp)
122     nu_HHe = cc_sp(NiH,NiHe,CiH,CiHe,mH,mHe,qe,qe,debp)
123     nu_HO2 = cc_sp(NiH,NiO2,CiH,CiO2,mH,mO2,qe,qe,debp)
124     nu_HNO = cc_sp(NiH,NiNO,CiH,CiNO,mH,mNO,qe,qe,debp)
125     nuH=nu_He+nu_HO+nu_HH+nu_HHe+nu_HO2+nu_HNO
126
127     # helium
128     nu_Hee  = cc_sp(NiHe,Ne  ,CiHe,Ce  ,mHe,me ,qe,qe,debp)
129     nu_HeO  = cc_sp(NiHe,NiO ,CiHe,CiO ,mHe,mO ,qe,qe,debp)
130     nu_HeH  = cc_sp(NiHe,NiH ,CiHe,CiH ,mHe,mH ,qe,qe,debp)
131     nu_HeHe = cc_sp(NiHe,NiHe,CiHe,CiHe,mHe,mHe,qe,qe,debp)
132     nu_HeO2 = cc_sp(NiHe,NiO2,CiHe,CiO2,mHe,mO2,qe,qe,debp)
133     nu_HeNO = cc_sp(NiHe,NiNO,CiHe,CiNO,mHe,mNO,qe,qe,debp)
134     nuHe=nu_Hee+nu_HeO+nu_HeH+nu_HeHe+nu_HeO2+nu_HeNO
135
136     # oxygen2
137     nu_O2e  = cc_sp(NiO2,Ne  ,CiO2,Ce  ,mO2,me ,qe,qe,debp)
138     nu_O2O  = cc_sp(NiO2,NiO ,CiO2,CiO ,mO2,mO ,qe,qe,debp)
139     nu_O2H  = cc_sp(NiO2,NiH ,CiO2,CiH ,mO2,mH ,qe,qe,debp)
140     nu_O2He = cc_sp(NiO2,NiHe,CiO2,CiHe,mO2,mHe,qe,qe,debp)
141     nu_O2O2 = cc_sp(NiO2,NiO2,CiO2,CiO2,mO2,mO2,qe,qe,debp)
142     nu_O2NO = cc_sp(NiO2,NiNO,CiO2,CiNO,mO2,mNO,qe,qe,debp)
143     nuO2=nu_O2e+nu_O2O+nu_O2H+nu_O2He+nu_O2O2+nu_O2NO
144
145     # NO
146     nu_NOe  = cc_sp(NiNO,Ne  ,CiNO,Ce  ,mNO,me ,qe,qe,debp)
147     nu_NOO  = cc_sp(NiNO,NiO ,CiNO,CiO ,mNO,mO ,qe,qe,debp)
148     nu_NOH  = cc_sp(NiNO,NiH ,CiNO,CiH ,mNO,mH ,qe,qe,debp)
149     nu_NOHe = cc_sp(NiNO,NiHe,CiNO,CiHe,mNO,mHe,qe,qe,debp)
150     nu_NOO2 = cc_sp(NiNO,NiO2,CiNO,CiO2,mNO,mO2,qe,qe,debp)
151     nu_NONO = cc_sp(NiNO,NiNO,CiNO,CiNO,mNO,mNO,qe,qe,debp)
152     nuNO=nu_NOe+nu_NOO+nu_NOH+nu_NOHe+nu_NOO2+nu_NONO
153
154     return nue,nuO,nuH,nuHe,nuO2,nuNO
155
156
157 def parameters(N,T,B):
158
```

```
159     # Calculate thermal speed,
160     #           gyro frequency,
161     #           particle Debye length,
162     #           plasma Debye length,
163     #           Coulumb collision frequency
164     # for e-, O+, H+, He+, O2+, NO+
165
166     # INPUT:
167     #    N: electron and ion concentrations in m^-3 unit
168     #       should be in tuple format [e-,O+,H+,He+,O2+,NO+]
169     #
170     #    T: electron temperature and ion temperature in Kelvin unit
171     #       should be in tuple format [T_ion, T_electron]
172     #
173     #    B: Geomagnetic field in Tesla unit
174     #       this data is read from igrf_12 package
175     #
176     # OUTPUT:
177     #    C,Gfreq,deb,debp,nu: explained as shown in function description above
178
179     Ti,Te = T[1],T[2]
180     Ne,NiO,NiH,NiHe,NiO2,NiNO = N[0],N[1],N[2],N[3],N[4],N[5]
181
182     # Thermal speeds (m/s)
183     Ce,CiO,CiH,CiHe,CiO2,CiNO = sqrt(K*Te/me),sqrt(K*Ti/mO),sqrt(K*Ti/mH), \
184                                 sqrt(K*Ti/mHe),sqrt(K*Ti/mO2),sqrt(K*Ti/mNO)
185     C = array([Ce,CiO,CiH,CiHe,CiO2,CiNO])
186
187     # Gyro-frequencies (plus doppler shift frequencies)
188     Omge,OmgO,OmgH,OmgHe,OmgO2,OmgNO = qe*B/me,qe*B/mO,qe*B/mH, \
189                                         qe*B/mHe,qe*B/mO2,qe*B/mNO
190     Gfreq = array([Omge,OmgO,OmgH,OmgHe,OmgO2,OmgNO])
191
192     # Debye Lengths
193     debe,debO,debH,debHe,debO2,debNO = sqrt(eps0*K*Te/(Ne*qe**2)), \
194                                        sqrt(eps0*K*Ti/(NiO*qe**2)), \
195                                        sqrt(eps0*K*Ti/(NiH*qe**2)), \
196                                        sqrt(eps0*K*Ti/(NiHe*qe**2)), \
197                                        sqrt(eps0*K*Ti/(NiO2*qe**2)), \
198                                        sqrt(eps0*K*Ti/(NiNO*qe**2))
199     deb = array([debe,debO,debH,debHe,debO2,debNO])
200     debp = 1./sqrt(1./debe**2+1./debO**2+1./debH**2
201                    +1./debHe**2+1./debO2**2+1./debNO**2) # Plasma Debye Length
202
```

```
    nue,nuO,nuH,nuHe,nuO2,nuNO = Coulomb_collision(N,C,debp)
    nu = array([nue,nuO,nuH,nuHe,nuO2,nuNO])

    return C,Gfreq,deb,debp,nu


def Gordeyev(par,dt,dw,N,om):

    # calculate Gordeyev integral for one particle species

    # INPUT:
    #    par: input parameters for on particle, formatted as ion the code below
    #         C,deb,gfreq,nu,nu_n,B,aspect should be float numbers
    #         (should not input tuples for each variable)
    #         calucaltion for input tuples are not tested
    #
    # OUTPUT:
    #    dt, dw, N, om: parameters used for chirpz() function
    #                   these numbers are tuned specifically
    #                   in order to have good performance
    #                   they can be found in model_utils.model_fit file

    ts=arange(N)*dt
    C,deb,gfreq,nu,nu_n,B,aspect = par

    varil=((2.*C**2)/nu**2)*(nu*ts-1+exp(-nu*ts)) # collisional
    gam=arctan(nu/gfreq)
    varip=((2.*C**2)/(nu**2+gfreq**2))*(cos(2*gam) \
          +nu*ts-exp(-nu*ts)*cos(gfreq*ts-2*gam))
    acfs=exp(-((kB*sin(aspect))**2)*varil/2.) \
        *exp(-((kB*cos(aspect))**2)*varip/2.) \
        *exp(-nu_n*ts)
    acfs1 = copy(acfs)
    Gs = chirpz(acfs1,N,dt,dw,om[0])
    Js = Gs/(1-nu_n*Gs)

    sigS = (1-1j*om*Js)/(kB**2 * deb**2)

    return acfs,Js,sigS

def target_info(target,database_input):

    # take out information at desired height

```

```
    h,T,N,nu_neutral,B,aspect = database_input

    idx = int((target-75)/(h[1]-h[0]))
    N = array(N).T[idx]
    T = array(T).T[idx]
    nu_en = nu_neutral[1][idx]
    nu_in = nu_neutral[2][idx]
    B = B[idx]
    aspect = aspect[idx]

    C,Gfreq,deb,debp,nu = parameters(N,T,B)

    return T,N,nu_en,nu_in,B,aspect,C,Gfreq,deb,debp,nu
```

```
from pylab import *
import isrpy
import os
from scipy import signal
import model_utils.stdmod as ip
import model_utils.cztspec as cs
import isrpy.igrf.igrf12 as igrf
import isrpy.beampack.AObeam as AObeam

c=299.792458e6
fradar=430.0e6
lam=c/fradar/2.
kB=2*pi/lam
ll = [18.3464,360-66.7528] # latitude longitude of AO


Tmax = 440*1.0e-6
factor=10
N_length = 220*factor
dt = Tmax/N_length
df=(1./(N_length*dt))/10/factor
f=df*(arange(N_length)-N_length/2)
om = 2*pi*f
dw = 2*pi*df
om = om[N_length/4:N_length/4+110*factor]

Temax=440/80*1.0e-6
dte=Temax/N_length
te=arange(N_length)*dte
```

```
30
31  Timax=440*4.*1.0e-6
32  dti=Timax/N_length
33  ti=arange(N_length)*dti
34
35  spread_path=os.path.dirname(__file__)+'/spread2d.npz'
36  print spread_path
37  spread_data = load(spread_path)
38  spread_orig = spread_data['spread']
39  spread_orig = sum(spread_orig,0)/1000
40  spread = spread_orig - min(spread_orig) # normalized
41  spread = spread/max(spread) # normalized
42  spread_ex = spread[110-5:110+6] # original amplitude
43  spread_ex_orig = spread_orig[110-5:110+6] # original amplitude
44
45
46  # calculate model for a single height
47  # output model without convoluting PSF
48  def model_no_conv(env,T,N,V,ex=False,norm=True):
49      # input has following structure
50      #time = [2014,9,23,12]  # year, month, day, hour
51      #T = [TN, Ti, Te]
52      #N = [Ne, NO, NH, NHe, NO2, NNO]
53      #V = [dope, dopO, dopH, dopHe, dopO2, dopNO]
54
55      h,t = env
56      dop = array(V)*kB
57
58
59      # updated 01/23/2019
60      B,aspect = ip.mag([h],t,ll); B = B[0]; aspect = aspect[0]
61
62      # replace this msis by another local file,
63      # otherwise hard to retrieve files from online lib
64      N_neutral,nu_en,nu_in = ip.msis([h],t,ll);
65      #N_neutral,nu_en,nu_in = 0,0,0 # set to 0 for now
66
67      # this part check if the neutral collision data is exist for current t and h
68      # if exist use the stored data in dictionary
69      # else retrieve data from msis and rewrite the dictionary file
70
71      #a = np.load("temp_nc_dict.npz")
72      #if altitude[0]==a['h'] and all(array(a['t'])==t):
73      #    N_neutral, nu_en, nu_in = a['nn']
```

```
74      #else:
75      #    N_neutral, nu_en, nu_in = ip.msis(altitude,t,ll,T[0])
76      #    savez("temp_nc_dict", t=t, h=altitude[0], nn = [N_neutral, nu_en, nu_in])
77
78      C,Gfreq,deb,debp,nu = cs.parameters(N,T,B)
79
80      pare = C[0],deb[0],Gfreq[0],nu[0],nu_en,B,aspect
81      acfe,Je,sige = cs.Gordeyev(pare,dte,dw,N_length,om-dop[0])
82
83      if N[1] > 0.01*N[0]:
84          pariO = C[1],deb[1],Gfreq[1],nu[1],nu_in,B,aspect
85          acfiO,JiO,sigiO = cs.Gordeyev(pariO,dti,dw,N_length,om-dop[1])
86      else:
87          acfiO,JiO,sigiO = 0,0,0
88
89      if N[2] > 0.01*N[0]:
90          pariH = C[2],deb[2],Gfreq[2],nu[2],nu_in,B,aspect
91          acfiH,JiH,sigiH = cs.Gordeyev(pariH,dti,dw,N_length,om-dop[2])
92      else:
93          acfiH,JiH,sigiH = 0,0,0
94
95      if N[3] > 0.01*N[0]:
96          pariHe = C[3],deb[3],Gfreq[3],nu[3],nu_in,B,aspect
97          acfiHe,JiHe,sigiHe = cs.Gordeyev(pariHe,dti,dw,N_length,om-dop[3])
98      else:
99          acfiHe,JiHe,sigiHe = 0,0,0
100
101     if N[4] > 0.01*N[0]:
102         pariO2 = C[4],deb[4],Gfreq[4],nu[4],nu_in,B,aspect
103         acfiO2,JiO2,sigiO2 = cs.Gordeyev(pariO2,dti,dw,N_length,om-dop[4])
104     else:
105         acfiO2,JiO2,sigiO2 = 0,0,0
106
107     if N[5] > 0.01*N[0]:
108         pariNO = C[5],deb[5],Gfreq[5],nu[5],nu_in,B,aspect
109         acfiNO,JiNO,sigiNO = cs.Gordeyev(pariNO,dti,dw,N_length,om-dop[5])
110     else:
111         acfiNO,JiNO,sigiNO = 0,0,0
112
113     dispersion = 1.+sige+sigiO+sigiH+sigiHe+sigiO2+sigiNO
114
115     e_line = abs(1.+sigiO+sigiH+sigiHe+sigiO2+sigiNO)**2 \
116             *2.*N[0] *real(Je)/abs(dispersion)**2
117     iO_line = abs(sige)**2  *2.*N[1]*real(JiO) /abs(dispersion)**2
```

```
118     iH_line = abs(sige)**2  *2.*N[2]*real(JiH) /abs(dispersion)**2
119     iHe_line = abs(sige)**2 *2.*N[3]*real(JiHe)/abs(dispersion)**2
120     iO2_line = abs(sige)**2 *2.*N[4]*real(JiO2)/abs(dispersion)**2
121     iNO_line = abs(sige)**2 *2.*N[5]*real(JiNO)/abs(dispersion)**2
122 
123     spec  = e_line+iO_line+iH_line+iHe_line+iO2_line+iNO_line
124     if norm == True:
125         spec = spec - min(spec)
126         spec = spec / max(spec)
127 
128     spec_extra = [spec[5*factor:105*factor:10]
129     if ex==True:
130         spec=spec_extra
131 
132     return spec
133 
134 def model_fit(m0,var,ex=True,norm=True):
135     # m0: fitting parameters, should have following structure
136     # [[T(2)],[N(5)],[V(6)]]
137     # var: other variables needed in the fitting, should have following structure
138     # [TN,Ne,time,h]
139 
140     T = zeros(3); N = zeros(6); V = zeros(6)
141 
142     T[0] = var[0]; N[0] = var[1]
143     time = var[2:6]; h = var[6]
144 
145     T[1] = m0[0]*1000; T[2] = m0[1]*1000
146     N[1] = m0[2]*N[0]; N[2] = m0[3]*N[0]; N[3] = m0[4]*N[0]
147     N[4] = m0[5]*N[0]; N[5] = m0[6]*N[0];
148     V[0] = m0[7]; V[1] = m0[8]; V[2] = m0[9]; V[3] = m0[10]
149     V[4] = m0[11]; V[5] = m0[12];
150     #Noise = m0[13]; Scaling = m0[14]
151 
152     spec = model_no_conv([h,time],T,N,V,ex=ex,norm=norm)
153 
154     if norm == False:
155         spread_ex = spread_ex_orig
156         spread = spread_orig
157 
158     if ex == True:
159         model_conv = convolve(spec,spread_ex,'same')
160     else:
161         model_conv = convolve(spec,spread,'same')
```

```
162
163     model_conv = model_conv - min(model_conv)
164     model_conv = model_conv / max(model_conv)
165     #model_conv = model_conv*Scaling+Noise
166
167     return model_conv
```

APPENDIX B

ION-LINE INVERSION CODE

The following codes are used in the implementation of ion-line spectral inversion. The codes for single spectrum inversion and entire data inversion are shown, respectively.

```
%pylab inline
import os
import isrpy
import spec_utils as su
import spec_utils.schspec as suss
import spec_utils.intspec as suis
import spec_utils.necomp as necomp
import spec_utils.extphs as extphs


import model_utils.old_modfit as mf
import model_utils.stdmod as ip
import scipy.optimize as so
from scipy.interpolate import UnivariateSpline as spline
from IPython.display import clear_output

# load PSF file
data = load('/home/yulunwu3/yulunwu3/lib/model_utils/spread2d.npz')
note=data['note']; print note
spread=data['spread']
spread_h = sum(spread,1)
spread_f = sum(spread,0)

# load spectrum
time = [2016,9,24,14]
fpath,fname = suss.find_file_tuple(time); print fpath,fname
s1,s2,ang,tt = suis.integration(fpath,fname,10)

# compute vi from acf
v_init = extphs.ion_velocity(s1)
```

```
h_spln = arange(75+0.3*0,75+0.3*1881,0.3)

v_spln = spline(arange(75,75+0.3*1881,0.3),v_init)
v_spln.set_smoothing_factor(7e7)
v_spln = v_spln(h_spln)

# residue function
def residue_spec(m0,var):
    # m0 = [Ti,Te/Ti,NO+,V,NO]
    # var = [TN,Ne,time,h,spec]

    # input has following structure
    #time = [2014,9,23,12]  # year, month, day, hour
    #T = [TN, Ti, Te]
    #N = [Ne, NO, NH]
    #V = [Ve, VO, VH]

    env = [var[6], var[2:6]]
    spec = var[-220:]

    T = [var[0], m0[0]*1000, m0[1]*m0[0]*1000]
    N = [var[1], var[1]*m0[2], var[1]*(1-m0[2])]
    V = [0, m0[3], m0[3]]

    model = mf.model_2ions(env,T,N,V,ex=True)

    model_conv = convolve(model,spread_f,'same')

    spec_norm = spec/max(spec)
    model_norm = model_conv/max(model_conv)
    model_norm = model_norm*(1-m0[-1]) + m0[-1]

    spec_mag_order = 10**round(sum(log10(spec))/size(spec))
    sigma = spec/spec_mag_order

    chi = (spec_norm-model_norm)/sigma

    weighting = ones(size(chi))
    #weighting[:110-10] = 0.5*weighting[:110-10]
    #weighting[110+10:] = 0.5*weighting[110+10:]

    chi = chi * weighting

    return chi[5:220-5]
```

```
75
76  # inversion
77  spec_h_idx = 600
78  #spec_target = sum(s1[spec_h_idx-5:spec_h_idx+6],0)
79  spec_target = s1[spec_h_idx]
80
81  h = 75+0.3*spec_h_idx # km
82  T = [1.,1.] # [Ti,Te/Ti], Ti is scaled by 1/1000
83  N = [1.] # this N is the percentage of each species corresponds to Ne
84  V = [v_spln[spec_h_idx]] # doppler velocity, this remains unchanged
85  NO = 0.5
86
87  m0 = append(T,append(N,append(V,NO)))
88
89  print m0
90
91  TN = 790
92  Ne = Ne_profile_target[spec_h_idx-21] # there's 21 offsets on Ne profile
93  var = append(TN,Ne)
94  var = append(var,time)
95  var = append(var,h)
96  #var = append(var,elevation)
97  var = append(var,spec_target)
98
99  # m0 = [Ti,Te,percO+,V]
100 # var = [TN,Ne,time,h,spec]
101
102
103 upper_bound = [inf,inf, 1, v_spln[spec_h_idx]+2,1]
104 lower_bound = [0,  1,   0, v_spln[spec_h_idx]-2,-1]
105 fit = so.least_squares(residue_spec,m0,args=([var]),\
106                        bounds=(lower_bound,upper_bound))
107
108 clear_output(wait=True)
109 print h,fit.x
110 sys.stdout.flush()
```

```
1  import logging
2  logging.basicConfig(filename='errorlog.log', level=logging.DEBUG,
3                      format='%(asctime)s %(levelname)s %(name)s %(message)s')
4  logger=logging.getLogger(__name__)
5
6  results = []
```

```
7
8  # for all specified files in the file array
9  for f_index in range(size(fnarr)):
10     #if f_index < 47:
11     #    continue
12     # retrieve 10 min integrated spectra
13     s1,s2,tt,ang = suis.integration(fparr[f_index],fnarr[f_index],10)
14     time = [2016,9,23+(int(tt[0])-2016267),float(tt[1])]
15
16     # calculate the doppler velocity
17     v_init = extphs.ion_velocity(s1)
18     h_spln = arange(75+0.3*0,75+0.3*1881,0.3)
19
20     v_spln = spline(arange(75,75+0.3*1881,0.3),v_init)
21     v_spln.set_smoothing_factor(7e7)
22     v_spln = v_spln(h_spln)
23
24     # for every altitude in spectrum do the fit
25
26     for spec_h_idx in range(100,1700,10):
27
28         spec_target = sum(s1[spec_h_idx-5:spec_h_idx+6],0)
29
30         h = 75+0.3*spec_h_idx # km
31         T = [1.,1.] # [Ti,Te/Ti], Ti is scaled by 1/1000
32         N = [1.] # this N is the percentage of each species corresponds to Ne
33         V = [v_spln[spec_h_idx]] # doppler velocity, this remains unchanged
34         N0 = 0.5
35
36         m0 = append(T,append(N,append(V,N0)))
37
38         TN = 790
39         Ne = Ne_profile_target[f_index,spec_h_idx]
40         var = append(TN,Ne)
41         var = append(var,time)
42         var = append(var,h)
43         #var = append(var,elevation)
44         var = append(var,spec_target)
45
46         # m0 = [Ti,Te,percO+,V]
47         # var = [TN,Ne,time,h,spec]
48
49
50         upper_bound = [inf,inf, 1, v_spln[spec_h_idx]+2,1]
```

```
        lower_bound = [0,  1,   0, v_spln[spec_h_idx]-2,-1]
        try:
            fit = so.least_squares(residue_spec,m0,args=([var]),\
                                   bounds=(lower_bound,upper_bound))

            clear_output(wait=True)
            print spec_h_idx,fit.x
            print fnarr[f_index]
            sys.stdout.flush()

            results = append(results,fit.x)
        except Exception as e:
            results = append(results,zeros(5))
            logger.error(e)
            continue
```