UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

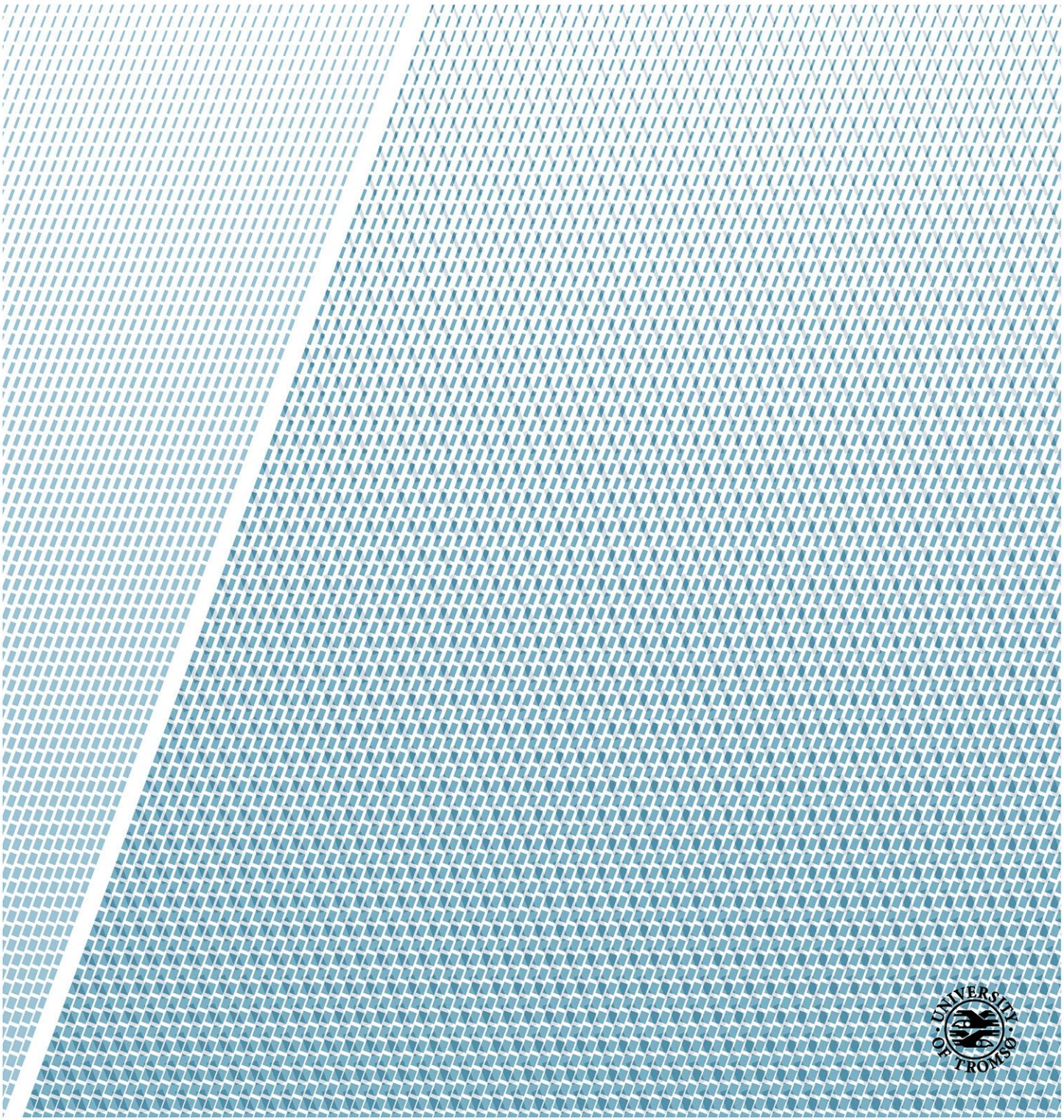
Department of Physics and Technology

Estimation of Neutral Densities in the Thermosphere

—

Will Stock

FYS-3900 Master thesis in physics - May 2017



ABSTRACT

Ionospheric energy balance is studied with incoherent scatter radar measurements that allow the estimation of neutral density and temperature from a statistical inversion problem. This method has been successfully studied at low latitudes using data over several years to make a robust model. This has not been done within the high latitude auroral regions and there are many factors that can complicate energy balance studies in this area, such as effects of Joule heating of the plasma. To study the feasibility of this type of study at high latitudes, a comparison was made between a parameterisation by Nicolls et al. (2006), a parameterisation of the MSIS model atmosphere included in the MATLAB aerospace toolbox and the estimations directly from the MSIS model atmosphere. Measurements were taken with the EISCAT tristatic system in hopes to determine Joule heating, however this failed due to outside radio interference causing an insufficient number of tristatic observations and making the results too imprecise to be conclusive. An in-depth analysis of the parameterisations was still carried out to find errors and determine ways to improve the method and model for future.

ACKNOWLEDGEMENTS

This thesis would have been impossible for not the support and friendship of those around me and even those afar such as my friends and family below latitude 66.3°. I would like to greatly thank my supervisor Professor Björn Gustavsson for all the help, guidance and occasional laugh that got me through my thesis and even my entire degree; in short, best supervisor you could ask for.

While we are on the subject, I would also like to give credit to the entire Space Physics Dept. of the University of Tromsø whose friendliness, feeling of community and 'Space Lunches' were a constant motivation.

A mention would also like to be made for the studentforening Imladris for providing an ever-present place for relaxation and fridge snacks.

Also, I guess I should thank my so-called friend, Sarah as she is watching me write this and made a tea for me like… one time.

It would be unremittable to also not thank those within the citations as although we will likely never meet or that you will even read this, you played the vital role of "giant" to whom I climbed up onto the shoulders of.

Many people have been a huge part in this and I apologise for those that I have missed out.

CONTENTS

1 INTRODUCTION

Thoughts on the atmosphere and what it is composed of have drastically changed the over the course of human history. Starting off as mere colours and lights above and seen in a more religious nature than scientific and later being seen as its own type of matter and fifth element, quintessence, today it is known that it is the same materials that we live and breathe in, simply rarefied and in case of the upper reaches, ionised from solar radiation. There is however, still more information to be gained about the atmosphere; information that would be extremely useful for applications involving long distance communication, operation of space technology and terrestrial astronomy. However, it is also important to reaffirm what is already known, to ensure that any information gathered is reliable.

Knowledge of the upper atmosphere is still relatively new as the concept of an electrically charged layer in the atmosphere was only proven in the early 1900s. Since then it has become a point of major research fractalized into many different areas of expertise. Determining the components of the atmosphere has been done numerous times by inferring the data from energy transfer rates [e.g., Bauer et al., 1970; Swartz and Nishet, 1971; Burnside et al., 1988; Oliver and Glotfelty, 1996; Nicolls et al., 2006; and many others]; a crucial development for this is the introduction of radar to scientific research.

Radar studies such as the ones listed above have over time provided the information and grounding to create global models that can determine the atmospheric constituents with just the location, time, F10.7 index (a number representing the amount of radio waves at 10.7cm wavelength the sun is emitting) for the previous day as well an eighty-one-day average, and the A_p index (a factor representing the amount of geomagnetic activity for a given day). The two models most commonly used are the Mass Spectrometer Incoherent Scatter (MSIS) and the International Reference Ionosphere (IRI) which provide neutral densities, and electron and ion densities respectively. These models use global data gathered from radar, satellite drag, and mass spectrometry, which are then used for interpolating data to get a good model that can be extrapolated across all longitudes, latitudes and altitude. This provides the best estimate of the atmosphere that can currently be determined as unlike previous tabulated atmospheres, the model can account for additional conditions such as latitude varying gravity, F10.7 and A_p index

(Hedin, 1991). This approach does however have the drawback that in areas of high ionospheric activity, such as the two magnetic poles, the atmospheric densities can be far from the average.

The ionosphere is defined as the region in the atmosphere where chemical reaction occurs through ionisation from solar extreme-ultra-violet radiation (EUV) at 105nm (~2mHz) and subsequent recombination. Due to this loose definition, there is no exact transition point between the ionosphere and it overlaps the neutral mesosphere below it and the exosphere above; it roughly begins around 65km and extend to an altitude of 1000km. Much like its boundaries, the ionosphere is inherently variable; it is constantly affected by solar EUV and the earth's magnetosphere. The shape of the magnetosphere and the cusp it has at both magnetic poles causes direct solar precipitation on the dayside. In addition to this, the entire auroral oval contains ring currents and the potential for aurora; caused by field line recombination in the magneto-tail pulling coupled, charged particles back into the atmosphere. These latitudes are often far from the average "expected" atmosphere due to auroral particles and ring currents causing sudden peaks of ionisation and higher electron density. This in turn leads to an incredibly varying atmospheric composition.

Due to inclination of the Earth to the sun, season is a significant factor as the summer hemisphere has greatly increased conductance compared to the winter hemisphere while retaining a similar potential; this infers a higher field-aligned current (FAC) and therefore more electrons and a higher temperature in the atmosphere (Ridley, 2007). Since the EUV is the main initiator of all these reactions, the Sun's eleven-year cycle (as shown in Figure 1) is the largest factor; during solar maximums, there can be up to one hundred times more EUV than on solar minimums. It is measured either as number of sun spots or amount of radiance at wavelength 10.7cm, hence the name F10.7 index. In addition to this, as the sun shines more directly onto the summer hemisphere, the upper reaches of the ionosphere (~200-350km) have greater ion production. In mid-latitudes and especially within the northern hemisphere, seasonal changes to the molecular-to-atomic

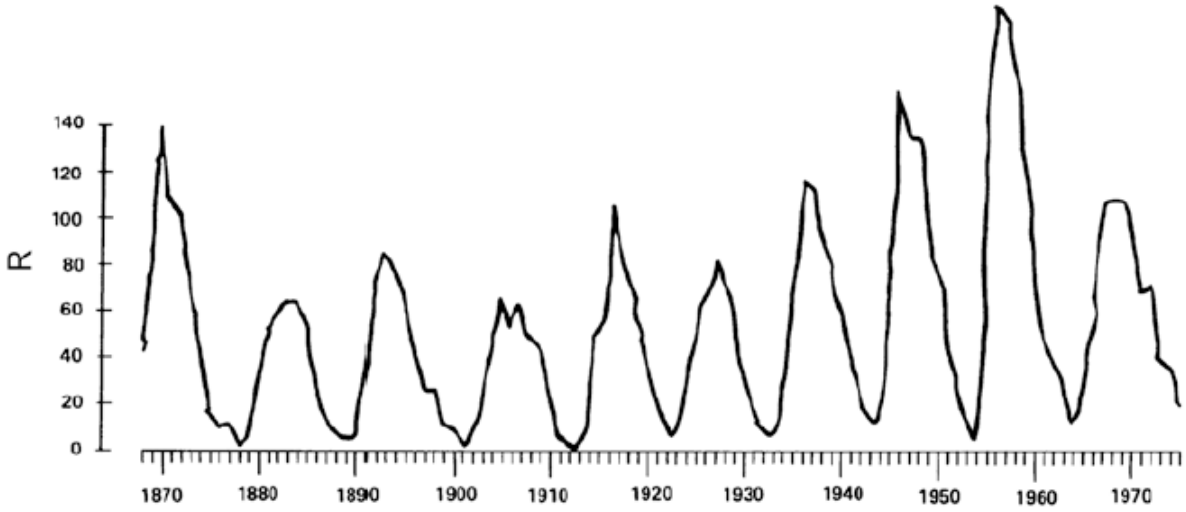


Figure 1 - Solar Sunspot Numbers Over years *(Brekke, 2013)*

ratio of neutral particles causes a drastic increase in ion loss that ultimately leads to a lower ionisation rate than in winter (Beynon & Williams, 1970).

Due to the shape of the magnetosphere at high latitudes and the angle of the field line to the incoming solar wind, the magnetosphere reshapes. On the day-side of earth, the magnetosphere is compressed and enveloped by the solar wind and so the field lines over the polar caps are parallel to those from the solar wind, leading to an influx of energy and particles. The influx causes an increase in current within the day and night sides, forming a closed ring on either side of the pole and centred around the dusk and dawn sides but separated by discontinuities at the midday and midnight part of the globe (Brekke, 2013). The coupling of the ionosphere and magnetosphere leads to a lot of instability from energy deposition and allowing auroral substorms, causing particle precipitation and affecting the composition. During daytime there are more stable conditions as there is a constant supply of EUV from the sun, providing a more stable number of electrons and ions compared to the aurora which causes high variability.

The following thesis will be focusing on the use of radar for atmospheric study. Although a lot of research technologies including radar was originally made for other purposes, (such as radar being used for detecting large solid objects like missiles and aircraft in the 1940's) scientists often explore areas with potential for scientific research and in the 1950's (e.g. Gordon, 1958). It was during this time the phenomena *Incoherent Scattering* was first discovered, making it one of the more recent methods to be used. Incoherent

Scatter (IS) is the term given to the radio waves back-scattered from free electrons showing the amount, and temperature of electrons in a volume (Gordon, 1958). Radio waves scatter incoherently and propagate in all directions and due to the long wavelength allowing for travel through clouds and other sources of interference, radio waves can therefore be detected and precisely measured from large distances away. Radar installations such as the Ramsfjord EISCAT (European Incoherent SCATter radar) can be used in conjunction with other EISCAT sites (Kiruna and Sodankylä) so that all receivers can measure back-scatter from the same transmission. When three or more radar are aligned in this way, it allows additional information to be gathered on particle motion (direction and speed) and is referred to as a multistatic system. Each receiver can detect the electron density and ratios of ion temperature/mass and electron/ion temperature that can then be used to calculate the temperatures and ion velocity.

The following thesis will detail an experiment examining two analysis techniques for neutral densities and temperature. This will use the Ramsfjord EISCAT site, to evaluate the feasibility of attaining accurate results from such high latitudes with a multistatic system. The method used is adapted from "*Daytime F region ion energy balance at Arecibo for moderate to high solar flux conditions"* (Nicolls, et al., 2006). This article shows that there are large deviations between neutral atmosphere estimations from Arecibo, Puerto Rico and the MSIS neutral atmosphere model. These properties are derived from evaluating the energy transfer rates from electrons to ions and then from ions to neutrals (aka energy balance) in a non-linear least square fitting problem, fitting profiles for the data. This works on the principle that the high temperature electrons that have measured density and temperature from the scan primarily heat the ions that are only cooled by the neutrals. Joule heating should also be added to the energy balance due to the presence of high latitude auroral electrojet currents; this should be determined using the multistatic system allowing the benefits of this configuration to be exploited. How well the results gathered fit the MSIS model will be analysed and should provide a clear indication on whether this experiment can be used for furthering the MSIS and IRI models' accuracy for high latitudes.

1.1 APPLICATIONS FOR RESEARCH

The main objective of this analysis is to determine an experiment to improve the MSIS and IRI models, the foremost application would be incorporating this experiment's data and results into the models for further running and refinement of the models at high latitudes. As these models are used in numerous projects, each with their own unique aims, this research would become part of a much greater whole and indirectly applied to any mid-high latitude study involving atmospheric chemistry. Such a change would further validate the models, even when in areas with abnormally high solar input and potentially allow for new facets of these areas to be discovered.

1.2 OTHER EXISTING METHODS

Energy balance is not the only way to estimate neutral densities and the Burnside Factor, as collisions lead to the coupling of ions and therefore directly impact the population of neutrals, the parameters around this can also be used to infer the neutral densities. Vickers et al. (2013) detail a method using the simplified version of the ion momentum balance equation from Schunk (1975) and IS measurements where ion to neutral collision frequency can be determined. The estimates of Vickers, et al., (2013) concur with drag measurements from the CHAMP satellite when in quiet conditions ($A_p < 2$) and in the altitude 300-400km, with work on going to expand these. This method seems equally as valid for the purposes of neutral estimation but a year of scans was not available for the study and therefore less applicable for this thesis.

2 IONOSPHERE

2.1 COMPOSISION AND PROPERTIES OF THE UPPER ATMOSPHERE

The term ionosphere refers to the electrons and ions that overlap the three highest layers of the atmosphere and that are exposed to EUV and cosmic ray radiation that made it past the magnetosphere. The ionosphere is comprised of various species of charged particles as well as free electrons from ionisation, caused by interaction with photons of wavelength shorter than 105nm. Ionisation occurs when an electron orbiting a neutral particle absorbs a photon with energy higher than the ionisation threshold; this causes the electron to break away from its orbit, leaving a positively charged particle (ion) behind. Electrons will also recombine with ions to create neutrals, the two types of recombination are radiative and dissociative; the former being when an electron interacts with an atomic ion and neutralises it, with excess energy being emitted as a photon, and the latter being when an electron encounters a molecular ion, neutralising it but with the excess energy braking the atomic bond and creating two neutrals. The production rate of radiative recombination is in the order of $10^{-18}m^3/s$ while dissociative recombination is on the order of $10^{-13}m^3/s$ (Brekke, 2013). This means that in areas of high ionisation, there will be high populations of atomic particles made from dissociative recombination that when ionised have 'difficulty' in neutralising again. These factors give the ionosphere unique properties; such as a peak temperature approximately three times greater than at sea level and the ability to reflect radio waves, caused by direct heating from EUV and exothermic chemical reactions, and high densities of plasma respectively.

Radio waves interact with plasma differently depending on frequency. Low frequencies (<10MHz) will be reflected and refracted, while higher frequencies will be able to propagate; assuming there are no density irregularities meeting the Bragg condition which cause the scattering of the wave. The wave may also be absorbed by the plasma but only when at an exact frequency set by the number of electrons, electric charge and mass of the ion. These properties are used for practical applications, with reflections being used for long-range communications around the world, propagating frequencies being used for satellite communication and space observation, and even absorption can be used for inducing artificial conditions within the upper atmosphere for study.

In addition to the ionosphere having unique properties across altitude, it also varies with latitude and time of day; as the shape of the magnetosphere causes the plasma coupled to it to precipitate from either the magneto-tail or from the conjugate foot-point, if it is in daylight, chemical processes are not just dependent on the solar. These lead to higher electron, plasma density and ionisation rates at night causing aurora. The lower temperatures and higher amounts of electrons and protons from the magnetosphere causes a slightly lower altitude for the peak of plasma density, making the auroral oval often act with larger dynamic variability compared to lower more 'stable' parts of the planet.

At ionospheric heights, the atmosphere is still dominated by the same species as at sea level but there are, however, much higher proportions of other elements such as Argon, helium and hydrogen as well as atomic oxygen and nitrogen from photo-dissociation and ion chemistry. As photons pass through the thermosphere, they interact with the atmosphere in several ways depending on wavelength and temperature, varying from absorption to transmission with little to no interaction. Each element has different collision cross section and frequencies required for absorption so populations of ion variants of these vary drastically. As altitude increases, more ions are present as less photons have already been absorbed or scattered.

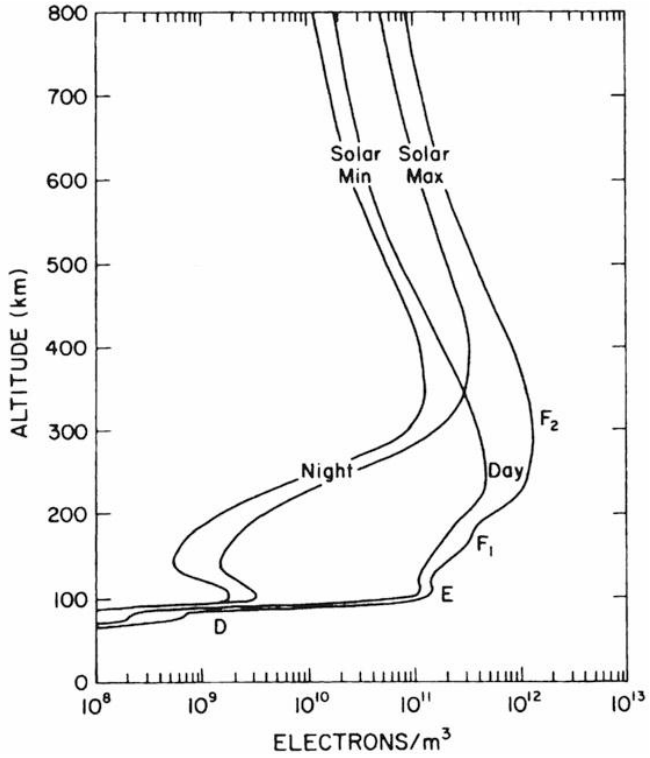


Figure 2 - Electron Density Profile for Night and Day *(Brekke, 2013)*

Even though the ionosphere is highly effected by direct sunlight, there is still ionisation occurring on the night-side of the planet; this is due to diffusion of dayside particles and geomagnetic sub-storms pulling large amounts of charged particles into the polar ionosphere after field lines in the earth's magnetosphere-tail have recombined. Without the sun's input, there is a drop in ionisation rates and ions with shorter lifetimes such as N_2^+, NO^+ and O_2^+ are highly diminished in lower regions. This absence of ionisation completely reshapes the electron density distribution across the ionosphere; under normal daytime conditions the ionosphere is described as three layers (See Figure 2) that each change during night-time.

2.2 IONOSPHERIC STRUCTURE

2.2.1 D Region

Stretching between approximately 60-90km, the D Region is the lowest layer in the ionosphere. The D Region is mostly populated by NO^+, O_2^+ which overlaps with the mesosphere which contains N_2, O_2, O; other, lighter particles can reach higher in the atmosphere. Negative ions are also present in the lower half of this region as the densities are high enough for three-body reactions that can produce large cluster ions such as $X^-(HNO_3)_n$, $X^-(H_2SO_4)_n$, and $X^-(HCl)_n$ where X is generally O^- or O_2^- but in some cases can also be other cluster ions (Kazil, et al., 2003). The main source of ionisation is the highly penetrative solar x-ray photons, as photons of longer wavelengths will have already been absorbed. Excluding sporadic cases of ionisation occurring through cosmic rays or interacting with meteors, almost all ionisation is caused by solar photons. When in daylight electron density is 10^8–$10^9 m^{-3}$, however during night-time the presence of negative ions and absence of ionisation causes the D Region to disappear (Schunk & Nagy, 2009).

2.2.2 E Region

Above the D Region, at about 90-150km altitude is the E Region. Its ions are primarily NO^+ and O_2^+ with NO^+ dominating but the ratio between the two changes a lot with altitude, while neutral O, O_2 and N_2 are all equally prevalent. The neutral temperature starts to sharply increase over this region due to it being a highly energetic and ionising layer. Like the D Region, the primary source of ionisation is solar photons but at this altitude EUV is the ioniser; unlike the D Region, the E Region does not disappear during the night-time

hours. Despite rapid recombination throughout the night, there is some ionisation from cosmic rays, as well as diffusion of charged particles from the higher density F Region above. This will still cause the E Region to weaken and rise such that there is just 1% of the electron density of the day side that doesn't start to increase again until outside the E region.

The E Region is also susceptible to sporadic events caused by areas or "clouds" of unusually ionised gas that have a higher plasma frequency than normal and thus increase the limit of radio frequencies reflected; these events are either auroral precipitation or a phenomenon known as "sporadic E-layer". When not undergoing sporadic events and in normal conditions, low frequency radio waves will be reflected at all times; during night-time however, the altitude of reflection will rise allowing for angled transmissions to be reflected to further ranges.

2.2.3 F Region

The highest and largest region of the ionosphere is the F Region, stretching from 150km and upwards to roughly 1000km altitude. At these altitudes the neutral atmosphere is dominated by atomic oxygen that is ionised by EUV, the primary ion is O^+ but with H^+ and He^+ becoming more prominent until H^+ is the dominant species; this transition occurs at ~600-1000km at low-mid latitudes during night and day respectively, and even higher at high latitudes. He^+ may also be the dominant species, but only during incidents of high solar activity. It is in this region that the electron density the and temperature are at their highest. As this has the electron peak in the region $10^{12}m^{-3}$, only high frequency radio waves will reach this altitude and still be reflected. Once a wave has propagated through the F region, it will continue indefinitely (Schunk & Nagy, 2009). The height of this peak is determined by competing processes of photo-ionisation, recombination and plasma transport.

As this region is the highest, it is highly affected by the large amounts of solar EUV meaning there are significant changes between day and night. During summer daylight hours at low-mid latitudes, it is split into two regions (F1 and F2) as there is a drastic change in profile between around 200km to 250km. When the sun sets and the primary form of ionisation is lost, the electron density drops as recombination occurs and particles diffuse downward causing the highest concentrations to vary between 250-400km.

3 EISCAT INCOHERENT SCATTER RADARS

3.1 INCOHERENT SCATTER THEORY

The basic description of incoherent scatter is when a radio wave of frequency far above the plasma frequency at the F2 peak (so that the radio waves travel the entirety of the atmosphere) scatters off any density structures that fit the Bragg condition ($2K_R=K_{ne}$). Charged particles within the atmosphere meet this condition and cause radio waves to scatter in all directions. Measurements can be made from these interactions by examining the returning wave and observational estimates of autocorrelation functions.

The scattering process occurs when the wave scatters off all electrons within the transmitted beam; due to the random motion and distribution of the electrons the wave is scattered in all directions including back along the beam path. As this occurs from all structures matching the Bragg condition within the beam, many signals are scattered in the same direction and superimpose on one another. The density structures in the scattering volume can be described as ion acoustic waves; this is due to them being longitudinal oscillations of Coulomb collisions and velocity of these waves is close to the thermal velocity of the major ions (Evans, 1969). These scattered waves can only maintain a coherent structure when the incident wave is larger than the Debye length (>1cm at altitudes below 1000km (Evans, 1969)). For wavelengths shorter than this, it would fail to enact equal effect upon all the electrons, causing only single particle scattering and a return signal too weak to distinguish from noise.

It is possible to measure the returned power from back scattered waves, but it is typically better to measure the auto-correlation function of the signal with a tailored altitude and spectral resolution. This will produce a matrix based on the lag for each function. A zero-lag profile is just the 'power profile' and only provides electron density. Other profiles with higher lag can have their auto-correlation function put through a matched filtering Fourier transform to get a spectrum that allows other parameters to be estimated, as shown in Figure 2. The spectrum of the back-scattered beam shows two different profiles, as the back scattered beam is from a mixture of ions and plasma there will be both an ion and an electron lines. For this to be determined accurately the transmitted pulse must be

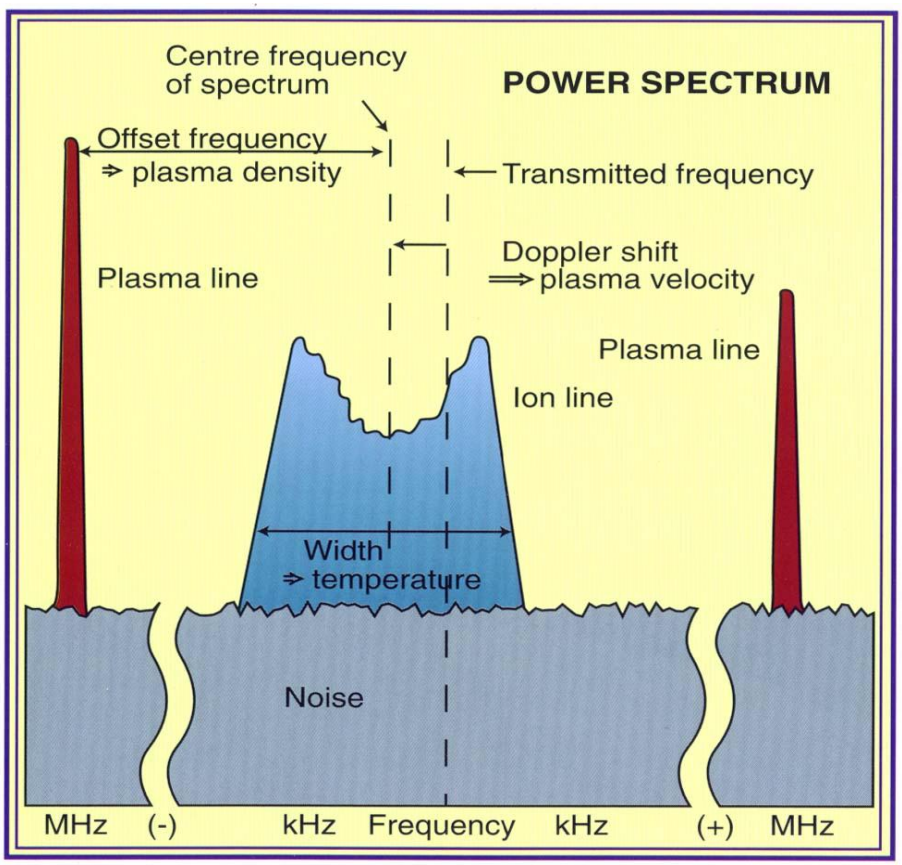


Figure 2 - Measured spectrum from a scattered beam (Strømme, 2005)

longer than 3.5 times the correlation time (Vallinkoski, 1988) while still being short enough to achieve a good resolution. Both species can be assumed to be Maxwellian with clearly defined different temperature and mass. The warmer, and therefore faster, moving charged particles have a broadening effect on the spectrum, while a higher mass narrows as phase velocity of the ion acoustic wave is slower. Narrower spectra originate from lower altitude where the average ion mass is larger and higher collisions frequencies cools ions. Unfortunately, as the ion composition is estimated through the species' mass ratio and O^+/O_2^+ is comparatively small, there is at present no simple way to determine exactly what the ion proportions within the atmosphere are when using IS radar.

3.2 EISCAT RADAR SYSTEMS

The EISCAT radar installations are located at high latitudes with Tromsø being 69.58° and are the only system that allows for tri-static measurements to be taken. The reason for them being in this location is that they can observe the auroral oval and observe the effects of ionosphere and magnetosphere coupling. The VHF is four 30x40m parabolic cylinders and the UHF is a 32m diameter parabolic dish; these are far smaller than many

others such as the 100m diameter antenna in Kharkiv, Ukraine or the 305m diameter system in Arecibo, Puerto Rico but this allows extra mobility and wide array of coverage and applications.

EISCAT also has several standard experiments, which are variations on what pulse transmission schemes it can provide, each with a different focus. The scan used within this experiment was Beata due to it having maximum efficiency between ~175-650km, covering the key altitude ranges of the study. The main difference between the schemes is the ranges covered and the inherent efficiency across that range.

PULSE TRANSMISSION SCHEME	SUPPORTING RADARS
ARC_DLAYER	UHF & VHF
ARC1	VHF
BEATA	UHF & VHF
BELLA	UHF & VHF
MANDA	UHF & VHF
TAU1	UHF & VHF
TAU7	VHF
TAU8	VHF

Table 1 - EISCAT Tromsø experiment capabilities for more information see *(Tjulin, 2016)*

This thesis will focus on the observation of Earth's atmosphere for which IS radar provides accurate temperatures and electron density of the ionosphere at short integration times, as well as velocities that can determine electric fields and flows. This has many different applications from inferring energy balance, to investigating the effects of space weather. It is commonly used for investigating aurora but can also be used for extra-terrestrial research; observing the moon, meteors and even as far out as interplanetary scintillation from stars and outside of our solar system. EISCAT formats received data as time integrated autocorrelation functions. The ionospheric parameters are then estimated, typically using the 'Grand Unified Incoherent Scatter Design and Analysis Package' v8.7 (GUISDAP) that carries out the processes described throughout section 3.1 to provide the following:

PARAMETER	
Altitudes (km)	standard deviation of electron temperatures
Time (seconds of day)	standard deviation of ion temperatures
Electron density (#/m3)	standard deviation of ion velocities
Electron temperature (k)	azimuth angle of radar (degrees)
Ion temperature (k)	elevation angle of radar (degrees)
'Line of sight' ion velocities (m/s)	date and time (DD/MM/YYYY)
Standard deviation of electron densities	Ranges (km)

Table 2 - GUISDAP outputs

3.2.1 Tromsø VHF & UHF

The Tromsø EISCAT site is located at Ramfjordmoen (69.58, 19.23); this location allows the radar to scan the auroral oval and with highly mobile radar such as the UHF, which can scan parallel to the magnetic field. VHF only has rotation in one axis and thus faces zenith or northward while the Tromsø UHF and the remote sites (Kiruna (KIR) and Sodankylä (SOD)) utilise steerable dishes that can be orientated to almost any direction (Tjulin, 2016). This enables for tailored experiments with desired combination of radar pointing, pulse transmission scheme and received bandwidth. This allows the measurement of data that cannot be normally gathered such as "beam swing" experiments that allow approximations of velocity not normally available for a monostatic system, (e.g. Aikio, et al., 2012).

3.2.2 Remote Receiving Sites

The remote sites, Kiruna and Sodankylä are near identical copies of each other, located at (67.87, 20.45) and (67.37, 26.63) respectively but both receiving VHF signals and having fully steerable dishes. Both are receive-only but as they have such great mobility they are extremely suited to creating tristatic system with the Tromsø site.

	VHF	UHF	KIR	SOD
LATITUDE	69.58°	69.58°	67.87°	67.37°
LONGITUDE	19.23°	19.23°	20.43°	26.63°
TRANSMITTER FREQUENCIES	222.8-225.4 MHz	926.6-930.5 MHz	N.A.	N.A.
RECEIVER FREQUENCIES	214.3-234.7 MHz	921.0-933.5 MHz	224.0-230.5 MHz	224.0-230.5 MHz
ANTENNA TYPE	Four 30x40m steerable parabolic cylinders	32m steerable parabolic dish	32m steerable parabolic dish	32m steerable parabolic dish
POLARISATION	Circular	Circular	Any	Any

Table 3 - Locations and technical specifications of relevant EISCAT radar

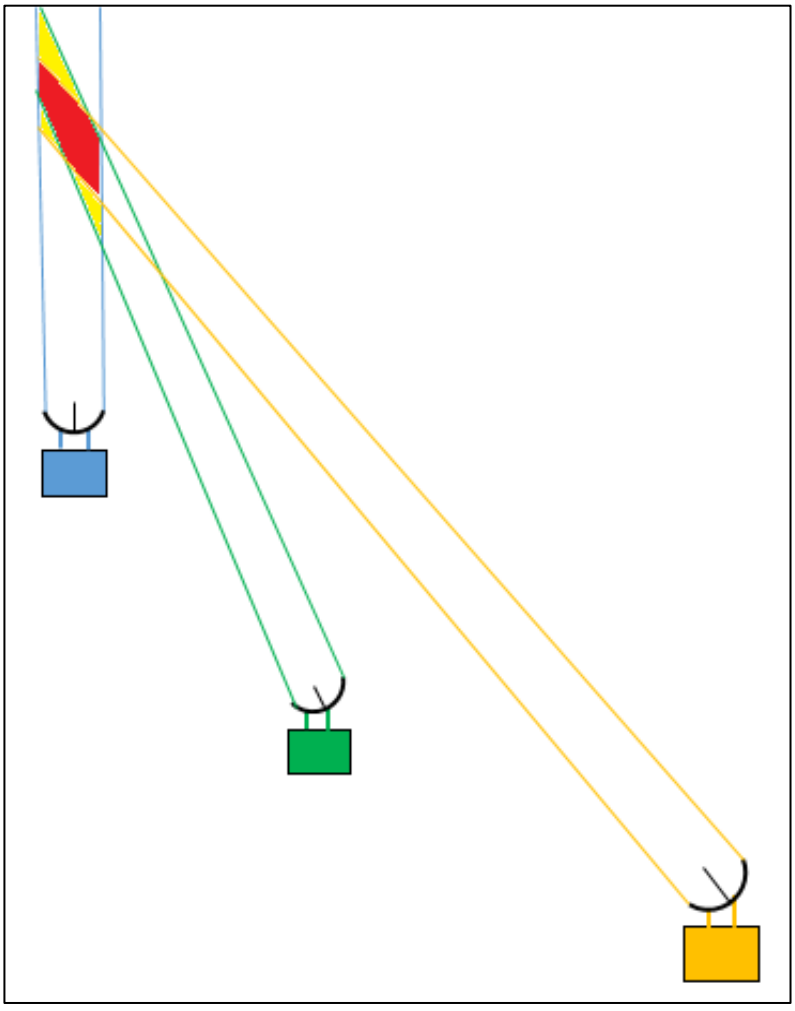


Figure 3 - Sketch of tristatic observations. Red patch is area of tristatic scan, yellow patch is area of bistatic scan

3.3 THE TRISTATIC SYSTEM

Due to the locations of the Tromsø and remote sites, tristatic scans are a very good option to apply. Even though all radar measure the same atmospheric characteristics as a monostatic experiment, the geometry of the experiment allows for additional data to be attained. Each radar receives the same spectrum such as in Figure 2, the Doppler shift can then be measured and used to determine the resultant ion velocity at the bisecting angle between the incidence and reflected beam as shown in Figure 4. At the Tromsø site, the measured velocity towards the radar is the resultant, as the wave propagates upwards before being reflected down the same line, the remote sites however, have scattered beams travelling at an angle and the resultant will be tilted accordingly.

With the additional radar receivers observing three non-collinear components of velocity, it can be determined what the north-east and height axis components are. This works best at lower altitudes as higher altitude measurements have a higher signal to noise ratio that can blur the smaller difference in direction of the beams. This can mean that the components can no longer be reliably determined.

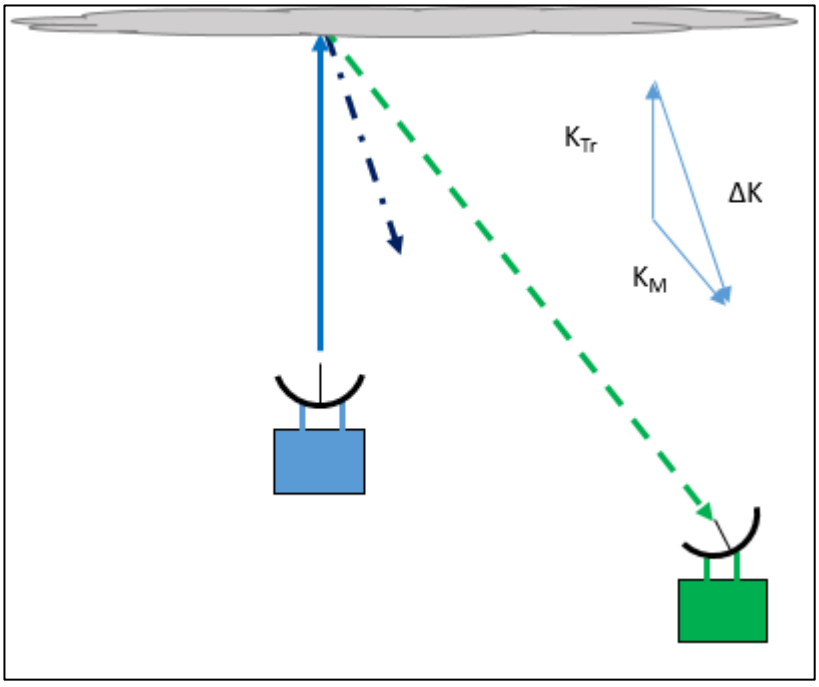


Figure 4 - Diagram of Bistatic wind measurement, where the full line is the transmitted beam of wavenumber K_{Tr}, the dashed line is the measured scatter of the transmitted beam, with wavenumber K_M and the dash-dot line is the resultant between the two with wave number $\Delta K = (K_{Tr}^2 + K_M^2)^{0.5}$.

4 HEAT TRANSFER AT HIGH ALTITUDES

4.1 THERMODYNAMICS WITHIN UPPER ATMOSPHERE

Temperature within the ionosphere is governed by the chemical reactions following solar EUV absorption that happens across the thermosphere. These ionising reactions produce both an ion and a free electron with considerable energy ~1-100eV. As electrons have high conductivity between other electrons, high energy electrons quickly pass on their energy to the bulk of the electrons leading to a large and quick increase in electron temperature compared to other particles. The bulk electrons will interact and collide with ions to pass on this energy but at higher altitudes, collisions are so infrequent energy transfer is slow and they retain their temperature. The ions will in turn pass heat to neutrals through elastic and inelastic collisions. This in turn means ions only achieve significantly higher temperature than neutrals above ~250km (See Figure 4). The large difference in particle mass causes the wide temperature differentials between particles as the small electrons transfer no energy in elastic conditions while the approximately

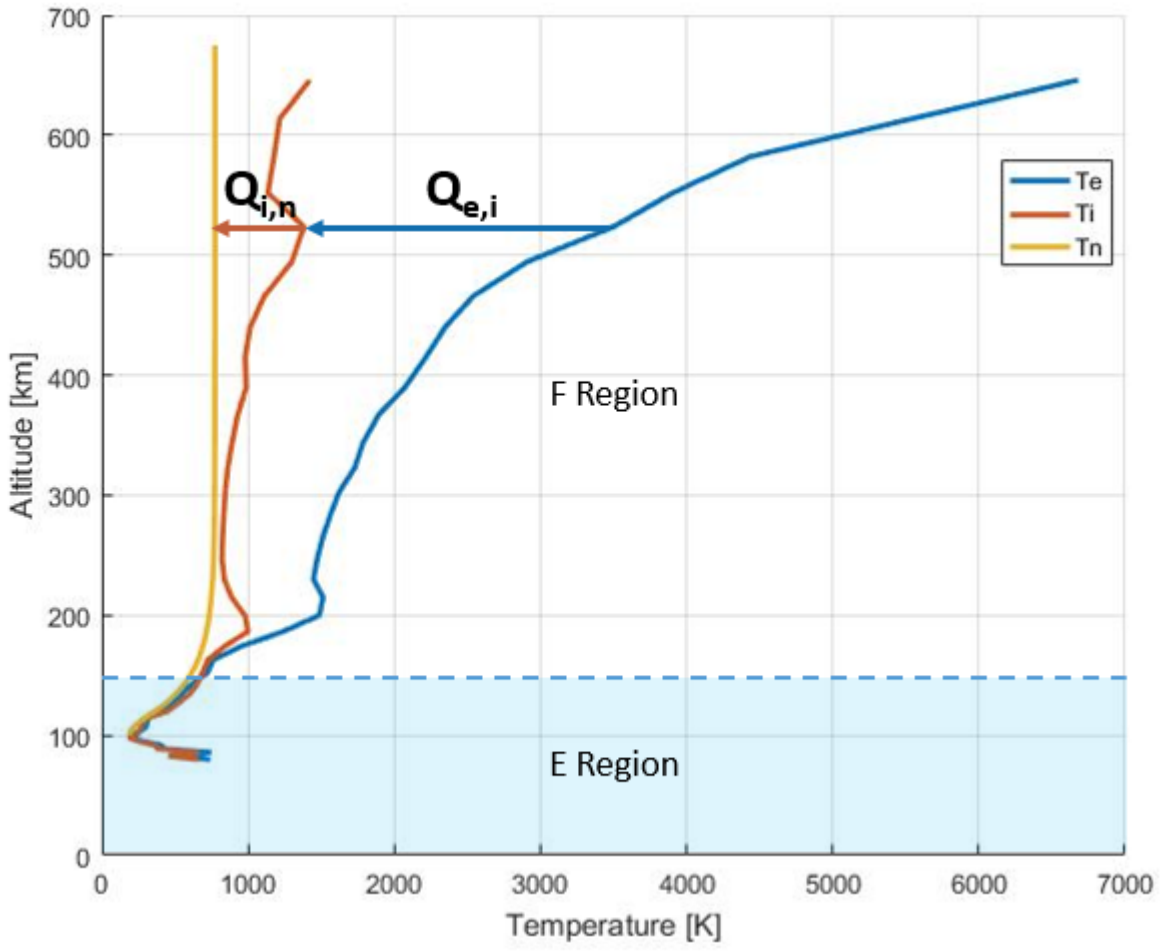


Figure 4 - Altitude Profiles of the Ion, Electron and Neutral Temperatures Also Showing the Flow of Energy

equally sized ions and neutrals transfer energy very inefficiently. Near the F2 peak, atomic oxygen ions only encounter electrons and their neutral counterpart; the temperatures of the particles mean that the electrons will carry excess energy from photo-ionisation to the ions through Coulomb collisions that then heat the neutrals.

Due to the diversity of the ionosphere and its constituents, estimating heat transfer is difficult, especially as oxygen (a significant factor of heat transfer throughout the atmosphere) cannot have its heat transfer tested in laboratory conditions due to its reactivity. To simplify the process, assumptions can be made such as all ion species have the same temperature. This is not exactly true for the topside ionosphere where different temperatures for H^+ and O^+ have been measured (Sulzer & Gonzalez, 1996) however as the focus thesis in on ~300-600km, it is a good approximation. It can also be assumed that the only ions present are O^+ and H^+, IS radar analysis can account for the presence of He^+ in the topside but this was deemed negligible (Nicolls, et al., 2006).

The heat transfer of the atmosphere is best described by the ion energy balance equation for species *i* (in this case, O^+) [e.g., (Nicolls, et al., 2006)]

$$\frac{3}{2}k_B n_{O^+}\left(\frac{\partial T_{O^+}}{\partial t} + v_{O^+}\cdot\nabla T_{O^+}\right) = Q_{O^+} - k_B n_{O^+} T_{O^+}\nabla\cdot v_{O^+} + \nabla\cdot\left(k_{O^+}\nabla T_{O^+}\right) + Q_J \qquad (1)$$

As this experiment is done during daytime conditions, heat transfer rates are dominated by solar EUV that varies very slowly therefore, the rate of change for T_{O^+} is small over the observation time-frame. The gradient of T_{O^+} is small compared to the beam size and divergence of plasma velocity, and thermal conductivity is negligible as well as Q_i must be 0 therefore, allowing the energy equation to be simplified. As the experiment takes place in high latitudes, Joule heating (Q_J) is likely to be a significant factor and cannot safely be assumed to be zero (see Section 4.2) and the net heating rate is given by:

$$Q_{O^+} = Q_{e,O^+} - \sum_k \left(Q^{el}_{O^+,k} + Q^{inel}_{O^+,k}\right) + Q_J = Q_{e,O^+} - Q_{O^+,n} + Q_J \qquad (2)$$

Where Q_{e,O^+} is the electron - O^+ heat transfer rate, $Q^{el}_{i,k}$ represents the heat transfer from ion to neutral *k* via elastic collisions and $Q^{inel}_{i,k}$ to neutral *k* via inelastic collisions. $Q_{O^+,n}$ is now defined as:

$$Q_{O^+,n} = \sum_k (Q^{el}_{O^+,k} + Q^{inel}_{O^+,k}) + Q_J \tag{3}$$

And as

$$Q_{O^+} = 0 \tag{4}$$

And net heating rate is consequently;

$$Q_{e,O^+} - Q_{O^+,n} + Q_J = 0 \tag{5}$$

The electron-ion heat transfer rate, Q_{e,O^+} can be defined as (Banks, 1966a):

$$Q_{e,O^+} = A_{e,O^+} n_e^2 (1 - P_{H^+})(T_e - T_i) T_e^{-\frac{3}{2}} \qquad eV/cm^3 s \tag{6}$$

Where T_e is the electron temperature in kelvins, P_{H^+} is the H^+ ion fraction n_e is the electron density in cm^{-3}, ϵ_0 is the permittivity of free space in standard units, e is the charge of an electron in Coulombs and:

$$A_{e,O^+} = C_{e,O^+} \ln(\Lambda) \qquad eVK^{\frac{1}{2}}cm^3/s \tag{7}$$

$$C_{e,O^+} = 3.1970 \times 10^{-8} \qquad eVK^{\frac{1}{2}}cm^3/s \tag{8}$$

$$Coulomb\ Logarithm\ \ln(\Lambda) = \ln\left(\frac{16\pi n_e \lambda_d^3}{\gamma^2}\right) \quad * \tag{9}$$

$$Euler's\ Constant\ \ \gamma = 0.57722 \tag{10}$$

$$Debye\ length\ \lambda_D = \frac{1}{10}\left(\frac{\epsilon_0 k_B T_e}{n_e e^2}\right)^{\frac{1}{2}} \qquad \text{cm} \tag{11}$$

**Shown by (Itikawa, 1975)*

The remaining $Q_{O^+,n}$ term is dominated by resonant pseudo-elastic collisions with atomic oxygen (Schunk & Nagy, 2009). Other studies have neglected the other terms however to extend analysis to the lower topside this thesis follows the method of (Nicolls, et al., 2006)

and includes small corrections for elastic collisions with N_2, O_2, He and H in addition to inelastic collisions with H atoms. This creates the heat transfer equation:

$$Q_{O^+,n} = \sum_k Q^{el}_{O^+,k} + Q^{inel}_{O^+,H} \tag{12}$$

The inelastic term is given by (Baily, 1983) as:

$$Q^{inel}_{O^+,H} = C^{inel}_{O^+,H} n_e \left\{ [H](1 - P_{H^+}) T_i \sqrt{T_n} - \frac{8}{9}[O] P_{H^+} T_n \sqrt{T_i} \right\} \tag{13}$$

Where $C^{inel}_{O^+,H} = 2.1 \times 10^{15}$. The elastic terms are given in (Banks, 1966b) & (Schunk & Nagy, 2009)

$$Q^{el}_{O^+,n} = Q^{el}_{O^+,O} + Q^{el}_{O^+,N_2} + Q^{el}_{O^+,O_2} + Q^{el}_{O^+,H} + Q^{el}_{O^+,He} \tag{14}$$

$$Q^{el}_{O^+,O} = C^{el}_{O^+,O} F [O] n_e (1 - P_{H^+}) \sqrt{T_i + T_n} (T_e - T_i) \tag{15}$$

$$Q^{el}_{O^+,M} = C^{el}_{O^+,M} [M] n_e (1 - P_{H^+}) (T_e - T_i) \tag{16}$$

In which $C^{el}_{O^+,O} = 2.1 \times 10^{-15}$, $C^{el}_{O^+,N_2} + 6.6 \times 10^{-14}$, $C^{el}_{O^+,O_2} = 5.8 \times 10^{-14}$, $C^{el}_{O^+,H} = 3.3 \times 10^{-14}$ and $C^{el}_{O^+,He} = 2.8 \times 10^{-14}$; with all heat transfer rates in units of eV/cm^3s, densities in cm^{-3} and temperatures in Kelvins. $C^{el}_{O^+,O}$ is in $eVcm^3/sK^{3/2}$ while the other coefficients are in the units of $eVcm^3/sK$. Using this method omits the weak temperature dependence on O^+-O collision cross section, giving the potential for an error of ±2% for $C^{el}_{O^+,O}$ (Banks, 1966b).

4.2 EFFECTS OF WIND/ATMOSPHERIC MOVEMENT

The movement of the neutral particles within the ionosphere is referred to as the neutral wind as they are removed from electromagnetic forces and are simply effected by the same conditions as the wind at sea level. They are dependent on a mixture of inputs such as pressure differentials from uneven heating and cooling of the atmosphere, as well as the presence of charged particles to collide and interact with; making the Earth's magnetic field a factor (King & Kohl, 1965). The more specific components of these will also vary across the height of the ionosphere as proportions of particles change.

The movement of the atmosphere both neutral and charged can be summarised as a superposition of the two forces, ion-drag; the slowing effect on neutral wind from the

presence of ions, and electromagnetic drift; the motion of charged particles that follow field lines and can cause larger atmospheric movements. These two effects can be observed by examining wind speeds, night time can see speeds in the range of 200-300m/s while the day time and its higher plasma density only reaches 50-100m/s on average (Akasofu & Champan, 1972). At higher altitudes, the magnetic effects are severely dominant and the effects of neutral wind are negligible. This allows the velocity of the particles to be used to determine the electromagnetic drift and thus the electrical fields present which can also be applied to the neutral winds within the lower F and E regions. By understanding how the winds are, an estimation of Joule heating can be made.

As detailed in (Aikio, et al., 2012), Joule heating can be written as:

$$Q_J(z) = \sigma_P(z)\{E + (u(z)\times B)\}^2 \tag{17}$$

Where σ_P is the Pederson Conductivity, E is the electric field, u is the average bulk velocity and B is the magnetic field. Due to the altitudes examined in this method, based off the works of Aikio, et al. (2012), E and B are constant. This was not seen to be the only factor for electromagnetic work applied to the neutrals. As mentioned in (Aikio, et al., 2012), (Brekke & Rino, 1978), (Thayer & Semeter, 2004), (Fuji, et al., 1999), (Thayer & Semeter, 2004), defines a mechanical energy transfer rate term that acts in addition to Q_J, creating the terms:

$$Q_m(z) = u(z) \cdot (j\times B) \tag{18}$$

$$Q_{EM}(z) = Q_J(z) + Q_m(z) \tag{19}$$

It is however, often assumed that for u(z) = 0 across the E region due to the difficulty in measuring wind speed over that altitude. As the majority of this investigation focuses on the F region and that Tristatic observations can be taken from with EISCAT, this will not be assumed. In which j is the current and is defined as:

$$j = \sigma_P \cdot (E + u\times B) \tag{20}$$

Q_J will always be positive but Q_m can be either positive or negative depending on whether the neutral wind is applying a kinetic force and doing work on the ions or vice versa. This

means that by evaluating Q_{EM}, it can be determined whether the ionosphere is a sink or source of energy depending on the sign.

4.3 CALCULATION OF NEUTRAL PARAMETERS

Equations 15 and 16 come from Banks (1966b), but Eq. 15 has been adapted to the method of Nicolls, et al. (2006), and including the Burnside scaling factor; as with this method, it will first be set F=1 and examined later through comparison to MSIS values. By using definitions detailed earlier in this section, a least squares problem examining the the energy balance (Eq. 3) and itteratively test different inputs (neutral parameter values) to infer the optimum conditions. This requires the assumption of model functions for atomic oxygen density ([O]) and neutral temperature (T_n) and fit for the density at the chosen altitudes along with the exospheric temperature.

The model to be used is the MSIS model, based on the analytical method given by (Bates, 1959) and (Walker, 1965). This neutral temperature model assumes diffusive equilibrium thus neglecting thermal diffusion and therefore is a Bates-Walker profile:

$$T_n(\zeta) = T_\infty - (T_\infty - T_{n0})e^{-s\zeta} \tag{21}$$

Where T_{n0} is the neutral temperature at a reference altitude Z_0 and s is an inverse scale height given by:

$$s = \frac{1}{T_\infty - T_{n0}} \frac{\partial T_n}{\partial z}\bigg|_{z_0} \equiv \frac{ks}{T_\infty - T_{n0}} \tag{22}$$

Therefore definig ks as the gradient of neutral temperature at z_0. ζ is the geopotential height:

$$s = \int_{z_0}^{z} \frac{g(z')}{g(z_0)} dz' = \frac{z - z_0}{1 + (z - z_0)/R_E} \tag{23}$$

In this, g is gravity and Re is the Earth's radius.

Atomic oxygen denisty is given by

$$[O](z) = [O]_0 \frac{T_{n0}}{T_n(z)} \exp\left[- \int_{z_o}^{z} \frac{dz'}{H(z')} \right] \tag{24}$$

Where $[O]_0$ is atomic oxygen density at z_0, the scale height is as below with m_0 as the mass of atomic oxygen.

$$H(z) = \frac{k_b T_n(z)}{m_o g(z)} \tag{25}$$

When the scale height integral in Eq 21 has ζ substituted in place of z, the integral can be computed analytically and inserted back into Eq 21 so that it may be written in the form of (Nicolls, et al., 2006):

$$[O](z) = [O]_0 \left[\frac{T_{n0}}{(e^{s\zeta} - 1)T_\infty + T_{n0}} \right]^{1+\gamma} e^{s\zeta} \tag{26}$$

Where $\gamma = m_0 g_0 / k_B T_\infty s$.

All values with the X_0 subscript are taken at the reference height which for this experiment shall be 100km. All values at the reference height will be determined from the MSIS-E-90 model (Hedin, 1991) with the exception of $[O]_0$ that will only use the MSIS value to generate an intial guess for our free parameters; T_∞ will also be treated this way. This leaves just the one problem of minimisng the squared residual of the weighted function in a least squares sense by adjusting the two estimated values of $[O]_0$ and T_∞ (Nicolls, et al., 2006).

$$\chi^2 = \sum_{k-1}^{N_{alt}} \frac{\left(Q_{e,O^+}([O]_0, T_\infty) - Q_{O^+,n}([O]_0, T_\infty) \right)^2}{\sigma_k^2} \tag{27}$$

This question will be applied to measurements at N_{alt} altitudes with $1/\sigma_k^2$ being an altitude dependant weight, calculated from the error from intitial solutions found.

5 ANALYSIS METHODS

Using the data gathering methods detailed in Chapter 3, the properties of the ionosphere explained in section 4.1 can be measured with a weighted least squares problem. The method by Nicolls et al. (2006) (from here on out will be referred to as the Nicolls parameterisation) describes the process of using data gathered from IS radar (T_e, T_i, n_e and V_i) and estimations of the neutral atmosphere to determine heat transfer. This can then be applied as an iterative system first using a placeholder weighting (σ_k=1) and determine the best estimated inputs for the inverse problem. So that an estimation of the weighting can be made, the standard deviations from GUISDSAP and the estimated parameters are used to run eight extra cases, applying the deviations to T_e and T_i as shown in Table 4 to get eight extra results; the standard deviation of the varying results produce $\sigma_{Q_{e^-,O^+}}$ and $\sigma_{Q_{O^+,n}}$. The total of these standard deviations is then used as the best estimated weighting and used with the best estimated inputs to best fit the results, for details on the implementation of this, see Appendix A, Section 9.5. In addition to the Nicolls parameterisation, this analysis also examines parameterisation of a MATLAB function (Now referred to as MATLAB parameterisation) that uses the same procedure as the MSIS model atmosphere; this function requires the F10.7 daily and monthly averages as well as the averaged A_p indexes as opposed to the Nicolls parameterisation that required and fit for $[O^+]$ and T_∞. The comparison of the two parameterisations would determine how the analysis method handles inverse problems and the accuracy of the results.

Case 1 (T_e+dT_e, T_i-dT_i)	Case 2 (T_e+dT_e, T_i+0)	Case 3 (T_e+dT_e, T_i+dT_i)
Case 4 (T_e+0, T_i-dT_i)	Case 5 (T_e+0, T_i+0)	Case 6 (T_e+0, T_i+dT_i)
Case 7 (T_e-dT_e, T_i-dT_i)	Case 8 (T_e-dT_e, T_i+0)	Case 9 (T_e-dT_e, T_i+dT_i)

Table 4 - Difference in input parameters for main fitting analysis

The analysis in this paper works with the assumptions that O^+ is the dominating factor for heat transfer and any minor ions contribute negligible heat transfer and that the temperatures are such that $T_e > T_i > T_n$; thus setting a heat transfer system of electrons heating ions than in turn heat neutrals. It is also assumed that the neutral atmosphere is well described by the parameterised models of Nicolls et al. (using $[O^+]$ & T_∞) and MATLAB (using $F_{10.7(\text{daily})}$, $F_{10.7(90\text{ days})}$ & A_P index) such that any values are correct. The experiment in this thesis has a consistent altitude range (300km – 600km) due to it only

containing data from one day, should this experiment be carried out over multiple days, it is likely that an altitude range set by the range where Te>Ti>Tn as when this condition was not met, extremely anomalous results were produced. Past methods such as the one by Nicolls, et al., (2006) have a varying altitude window to ensure they have a range with conditions fitting for their assumptions, this is due to their data being from world day data ranging from 1988 to 1994.

Measurements from the entire four-hour time-frame were averaged and analysed with the Nicolls and MATLAB parameterisation, and the analysis focuses on testing the models' abilities to estimate accurate data applicable for any time of the day. As well as this, the total time-frame was broken into six parts, four parts were averaged over one hour portions of the full-time-frame and remaining two each averaged over a two-hour time-frame. The comparison of residuals from each data-set will determine the versatility of the full data-set average and can also give some indication on if there were any errors or anomalies across the time-frame.

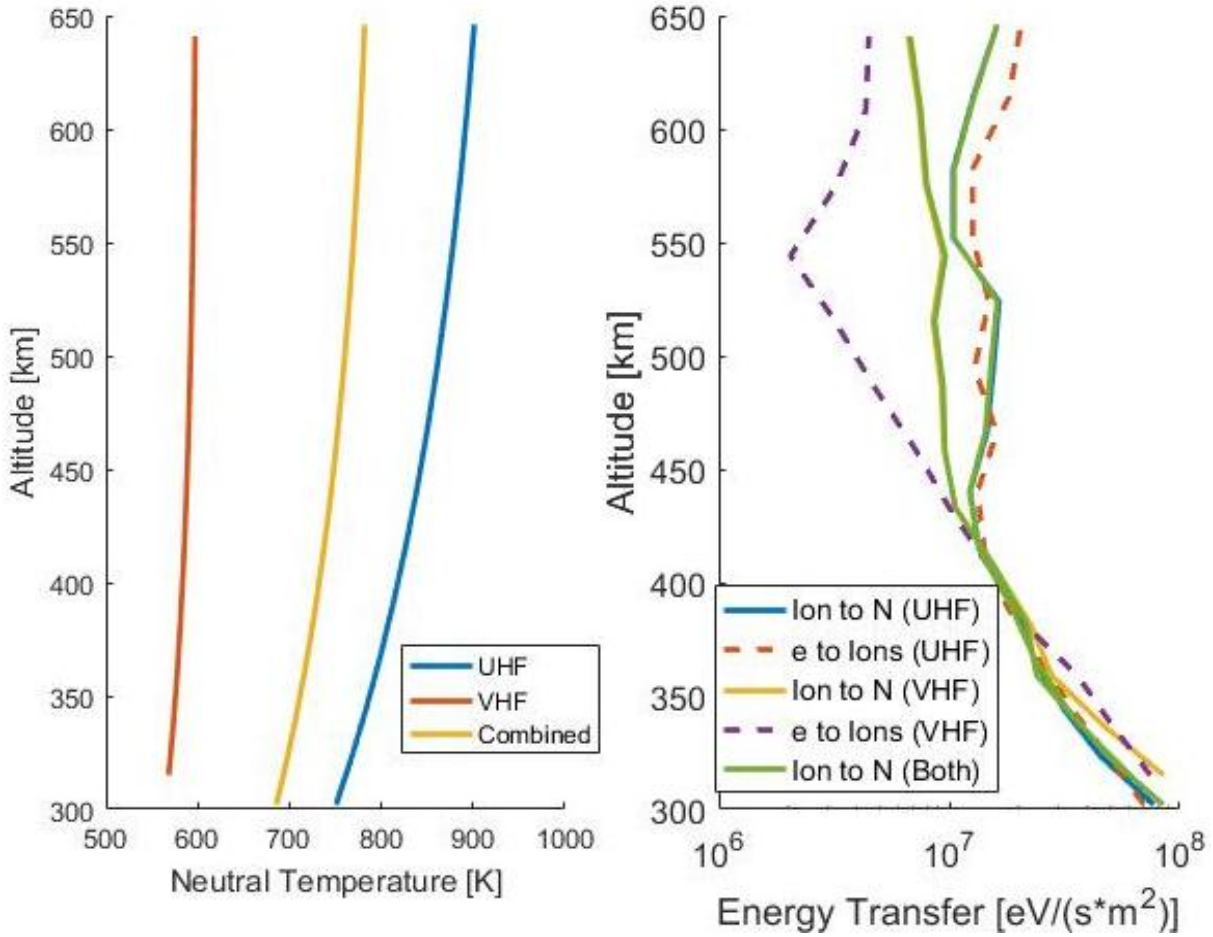


Figure 5 - Example fitted results. Left: Neutral Temperature, right: Heat Transfer Rates

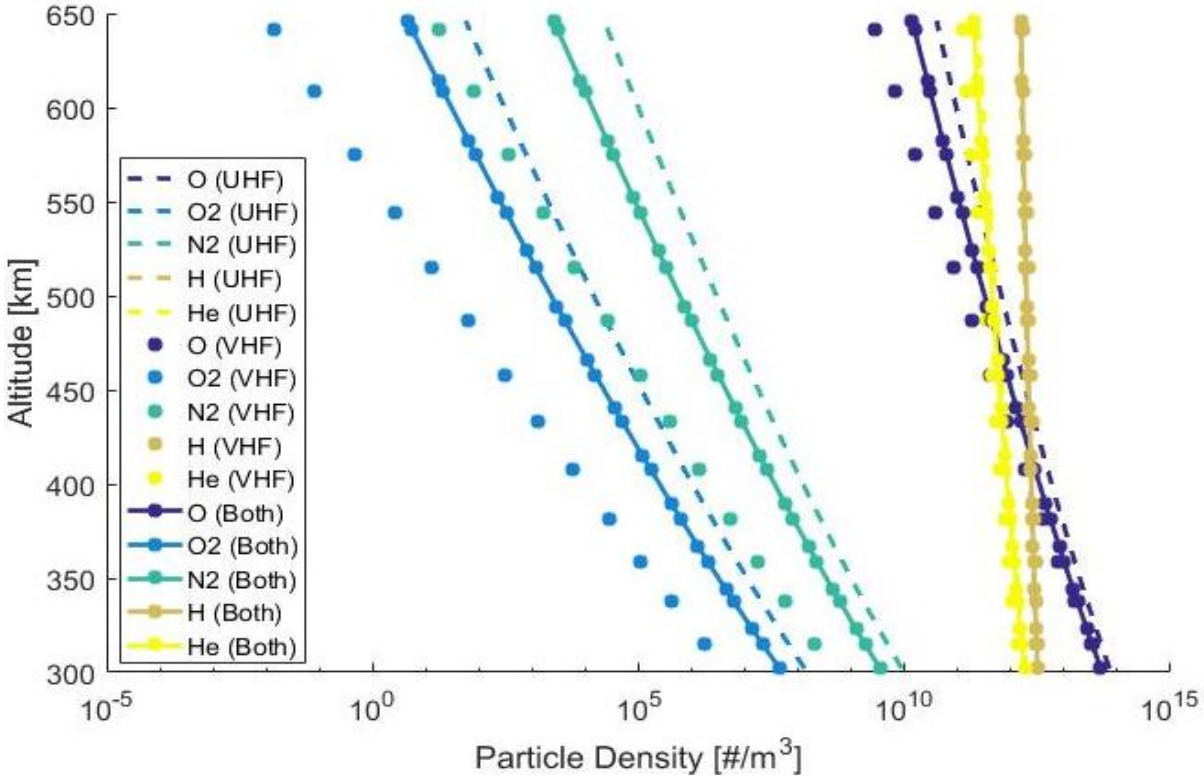


Figure 6 - Example of fitted neutral atmosphere

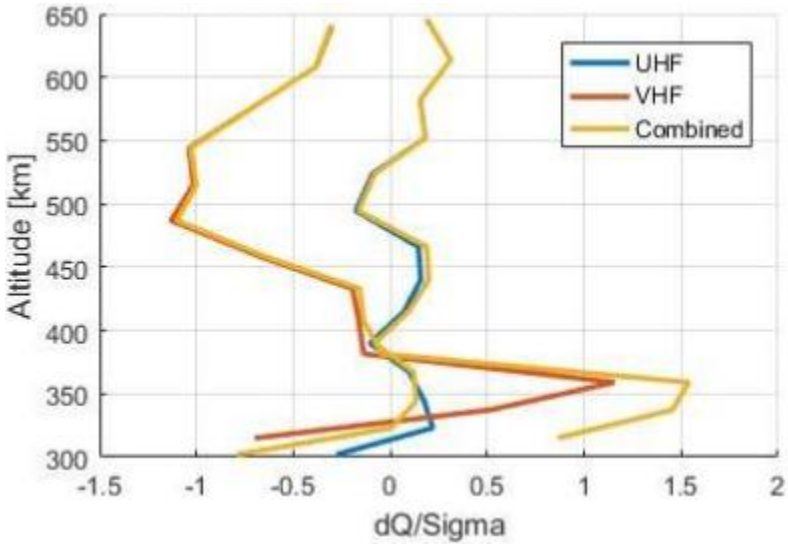


Figure 7 - Residuals for the lest squares function

Figure 5 shows example estimations giving neutral temperatures and heat transfer rates that are intermediate results in the fitting process. The overall aim of the analysis is to produce a graph such as Figure 6 that shows the atmospheric densities that have been determined as solutions to the inverse problem made by the neutral temperature and heat transfer values. As these are just estimates, the residuals of the least squares problem are also determined (Figure 7) and used for investigating how good the model is and if

the results generated are precise. The residuals are the determined by the equation $\Delta Q/\sigma$ where ΔQ is the difference between the absolute values of the ion to neutral temperature and the electron to ion temperature and σ is the combined standard deviation of the two heat transfers.

5.1 EXPERIMENT OBSERVATIONS

As shown in the example graphs, Figure 5, Figure 6 and Figure 7, combining the data-sets of the two radar does show some effect within the estimations for neutral temperature and neutral atmosphere, being a near average between the UHF and VHF results, it however, not show any effect in the heat transfers. In addition to this, it does not seem to provide any improvement to the residuals and its only noticeable effect is inflating the residuals at lower altitudes that already are quite high. Due to this, the data-sets of other time-frames will not be combined and analysed.

At approximately 10:30, it became apparent from the results that SOD was not showing reliable measurements. It was taken offline while the cause of the issue was determined in hopes that it could resume receiving for the experiment with reliable results after the issue was resolved. After a thorough investigation by the onsite team, the causes of the issue was determined as outside interference somewhere between Sodankylä and Tromsø. With no way to resolve the issues SOD was taken offline and the tristatic configuration abandoned, removing the possibility to determine the ion drift vector. The potential for interference can be a major obstacle of tristatic operations, especially when using radar so far apart as the long elevation of the receivers can cause other radio signals to enter side lobes of higher gain. It is however unlikely that this was the cause for interference for this experiment as it would have shown improvement at higher altitudes, as this was not the case, there was an unknown source radio frequency interference (RFI). This could likely be solved by a repeat of the experiment, either still using SOD or by using a different installation such as the Kilpisjärvi Atmospheric Imaging Receiver Array (KAIRA), Finland. KAIRA is much closer to Tromsø than SOD and removes the required assumption that the ionosphere does not change significantly between scan patterns. This is due to KAIRA being an array antenna that gets simultaneous data from all altitudes along the transmitted beam, providing components of the ion drift velocity and allowing the Ramsfjord-KAIRA velocity vector to have a time resolution determined by integration time, rather than scan period.

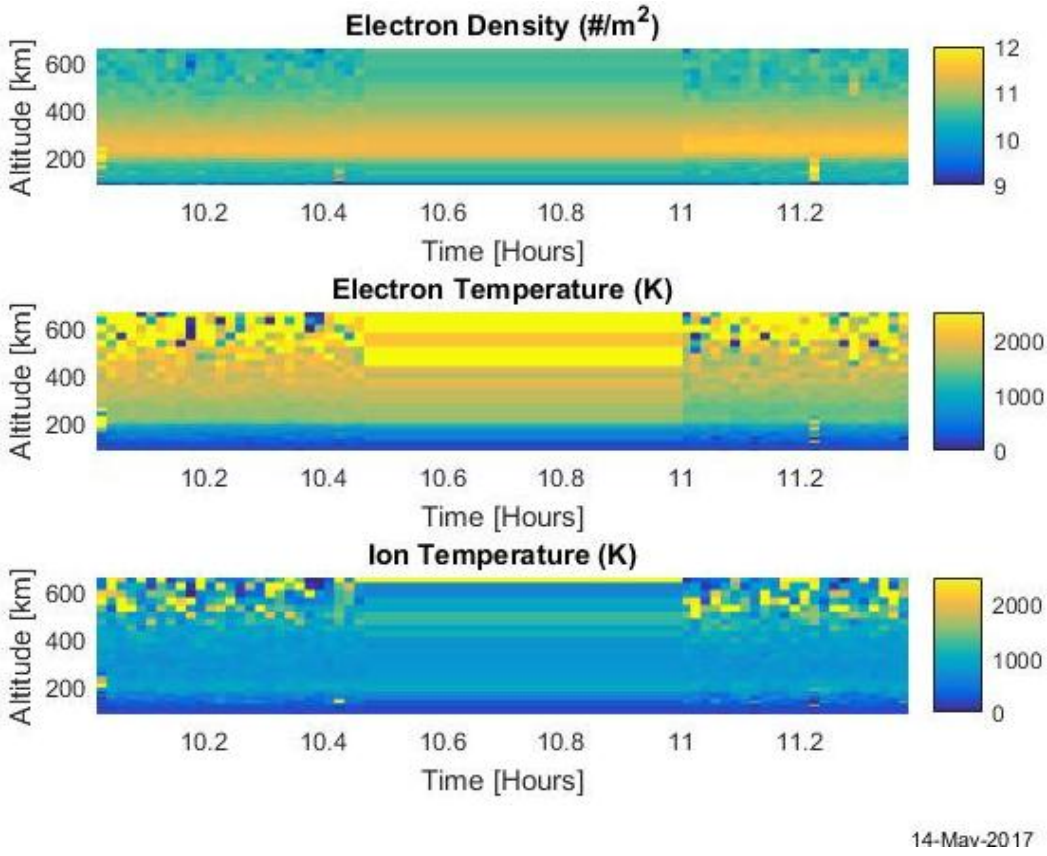


Figure 8 - 1st Hour Measured Data-set from VHF

As the problems with SOD were discovered, investigations at the Tromsø main site lead to the accidental deactivation of the VHF recorder, causing no data to be collected between approximately 10:30 and 11:00 UST (See Figure 8). 60 minutes into the experiment this was corrected and SOD stopped observations. Data was still recorded at the Tromsø site, but were only being stored in a temporary file and constantly over written, meaning that the time-frame shows the values at 11:00. Measurements from the UHF data covering that time show no large variations so although it is unknown if this significantly changed the results, they could still be used. All of this happened within the first quarter of the experiment; as there is no clear difference between the 1st one-hour data-set residuals and other plots, as well as none of the graphs comparing the results from the different data-sets show it as an outlier, it can be assumed to be insignificant.

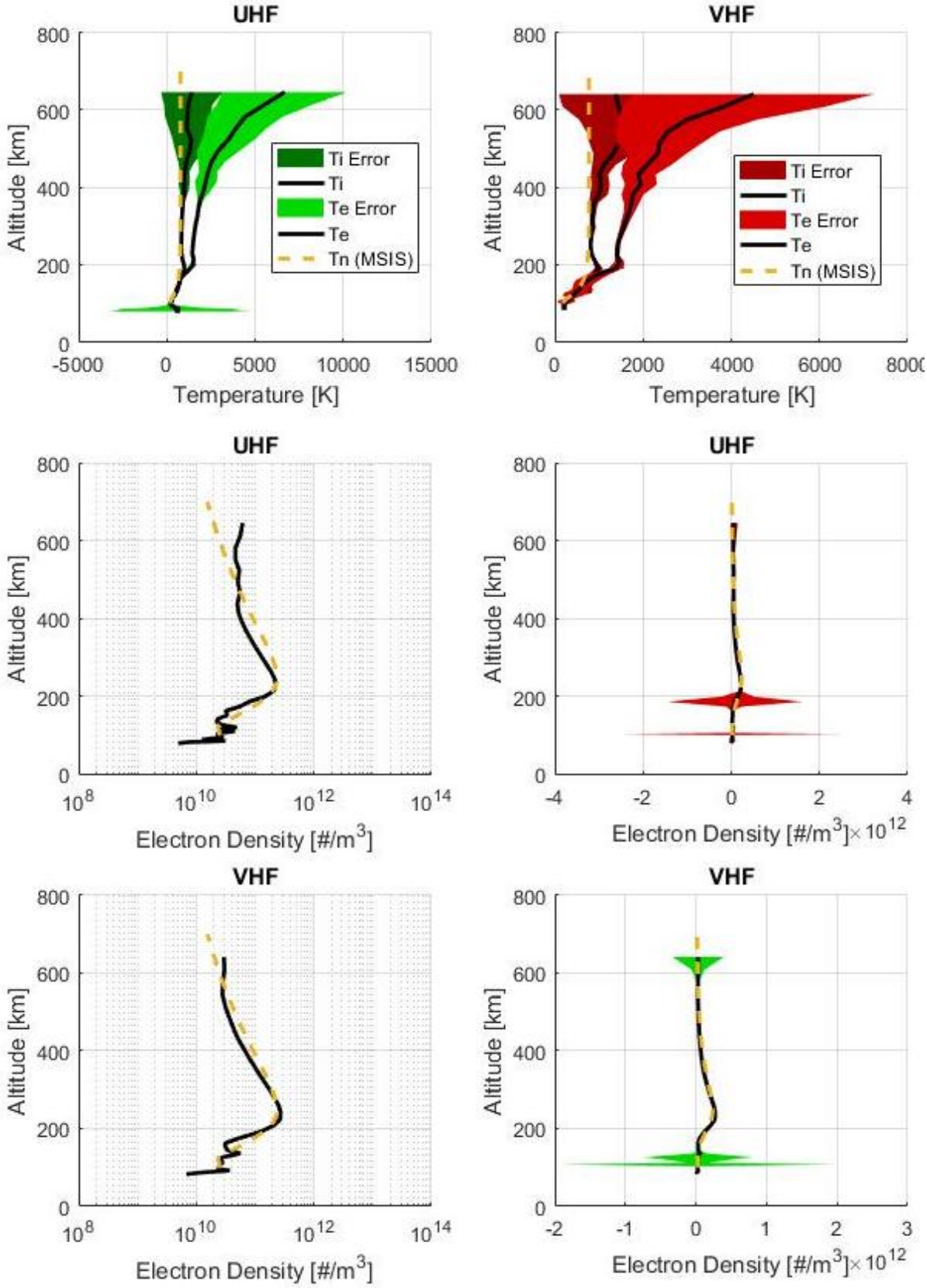


Figure 9 - Measured atmospheric values averaged over the entire scan time and with error bars showing standard deviations. For electron graphs, dashed lines represent the IRI model electron density while the black lines are the measured electron densities on both a log and linear scale (left and right respectively) to allow for clarity of observation and inclusion of the error.

The measured data from both radar show good results with the majority of the scan having negligible error. Unfortunately the main altitude range used in this experiment is 300-600km which has rather large standard deviations of temperature over 400km with the potential for the VHF to produce unphysical results such as electrons being cooler than ions. The electrons only contain errors outside of the analysis range, it is however troubling that the errors they do have are large enough to allow for impossible results such as a negative amount of electrons.

Overall, the electron graphs show similar values to the IRI model with only some misalignment within the analysis range however, at approximately 470km the UHF begins to show more electrons than the IRI model showing a diverging profile from this point on. The VHF also shows this at approximately 575km, but does not reach the same difference in results as the UHF. There is only a peak difference of about 250 $\#/m^3$ during the entire scan and that is while the results are on the order of 10^8 $\#/m^3$ so any differences in observations can be seen as negligible and the EISCAT radar can be assumed to be accurate.

6 RESULTS

Within this chapter will be the direct measurements from the scan (n_e, T_i, T_e and V_i) followed by a comparison of the results from both parameterisations, displayed side by side. The MATLAB parameterisation will be on the left while the Nicolls parameterisation will be on the right. It was originally planned to use an adapted version of Nicolls et al. (2006) that included the Joule heating; this unfortunately was removed after the scan from Sodankylä experienced some interference, ruining the results gained and therefore removing the ability for a tristatic scan. The analysed results are neutral temperatures, energy transfer rates, the neutral parameters as a whole as well as individual species comparing the seven data-sets against the MSIS model atmosphere, and the residuals.

6.1 DIRECT MEASUREMENTS

As show by Figure 10, Figure 11 and Figure 12**, t**he experiment started at 10:00 UST and ran until 14:00. By the time the experiment started the ionosphere would have been in sunlight for several hours and thus there measurements show no starting edge or rise in electrons at the start of the day. In contrast, the experiment ended late enough that there was significantly reduced electron density that shows some correlation with an increase of noise in the ion and electron temperature.

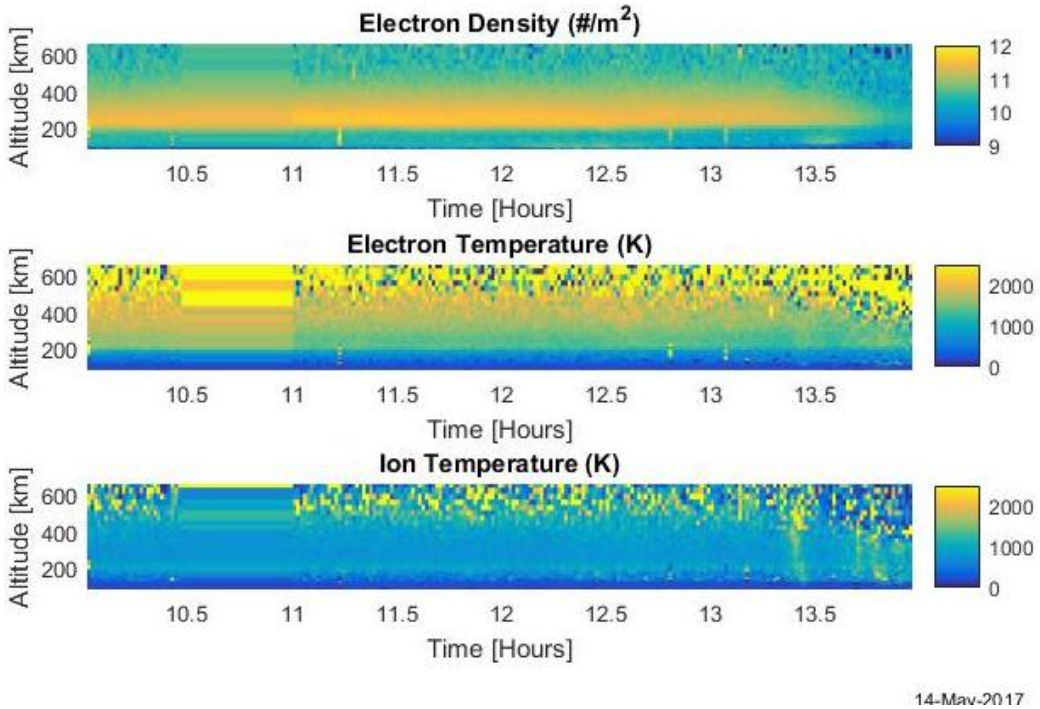


Figure 10 - Total measured data over the experiment for the VHF

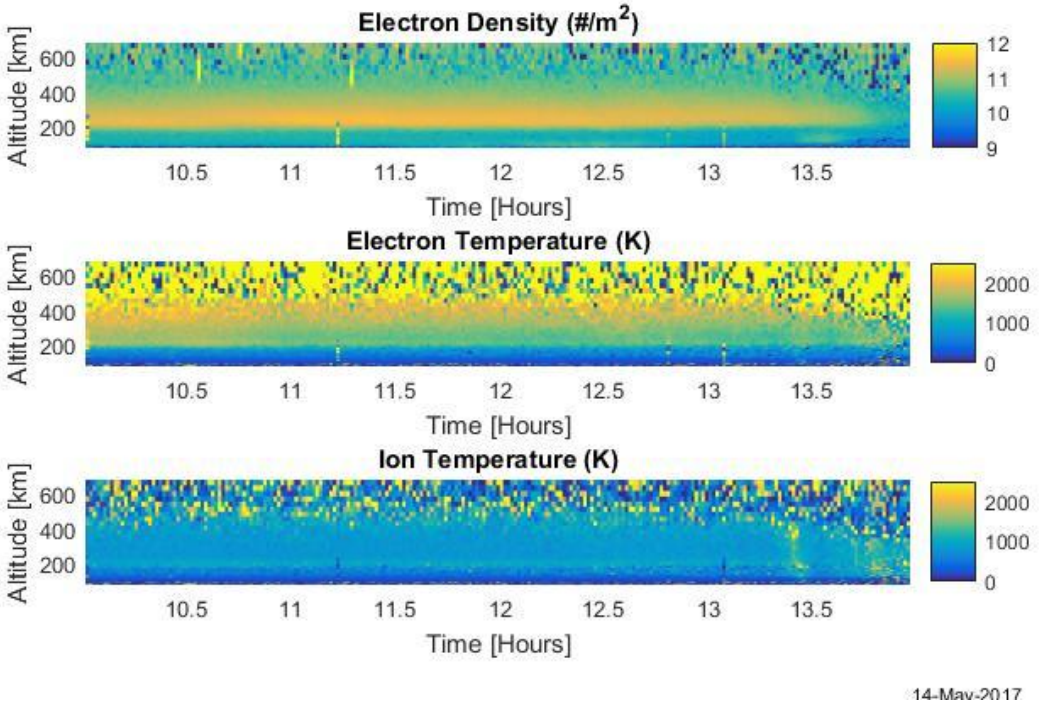


Figure 11 - Total measured data over the experiment for the UHF

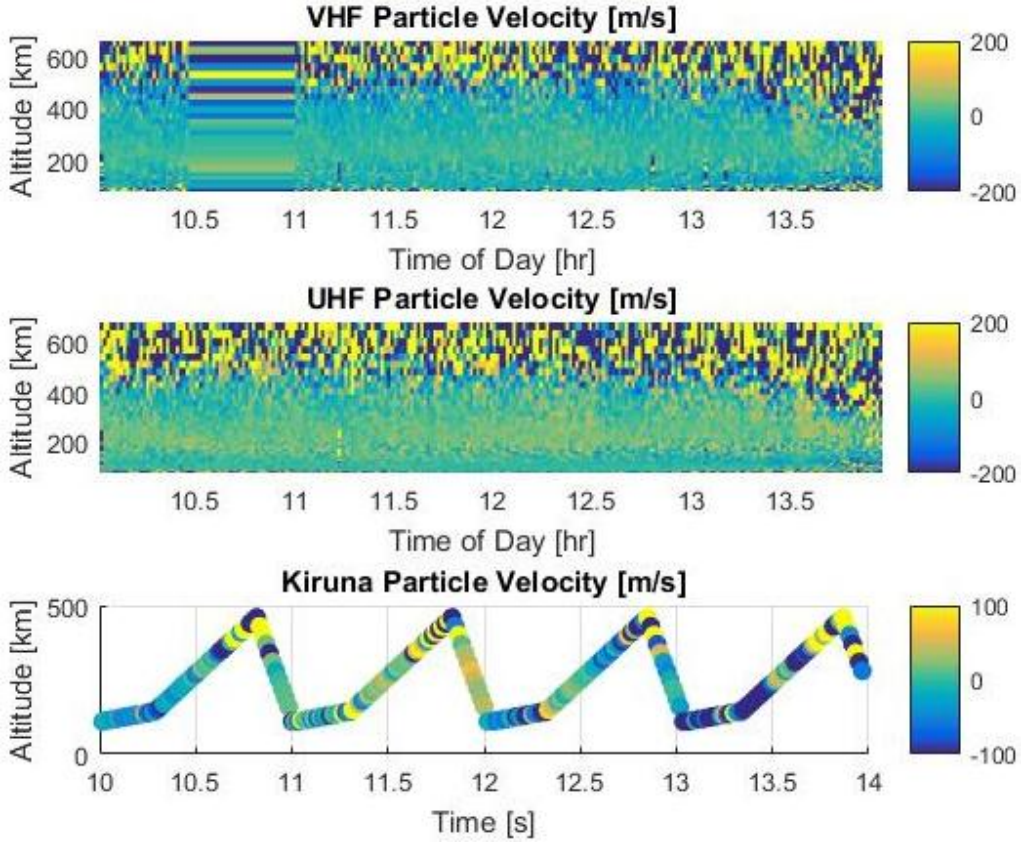


Figure 12 - Ion velocity from VHF, UHF and Kiruna radar

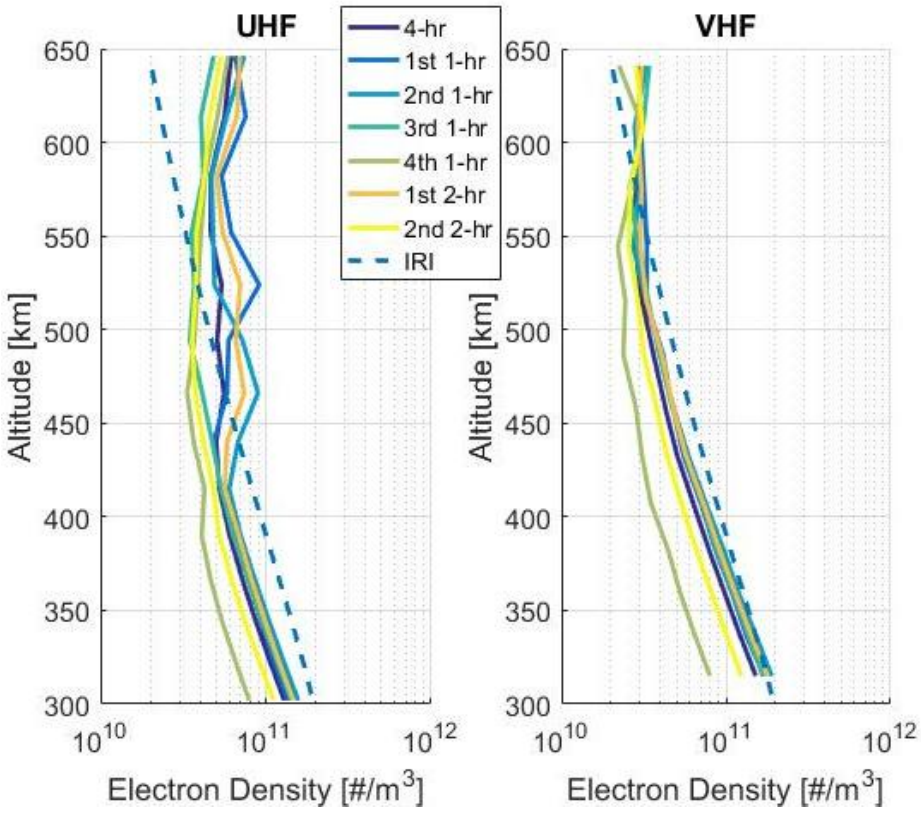


Figure 13 - Electron densities for the data-set averaged over different times

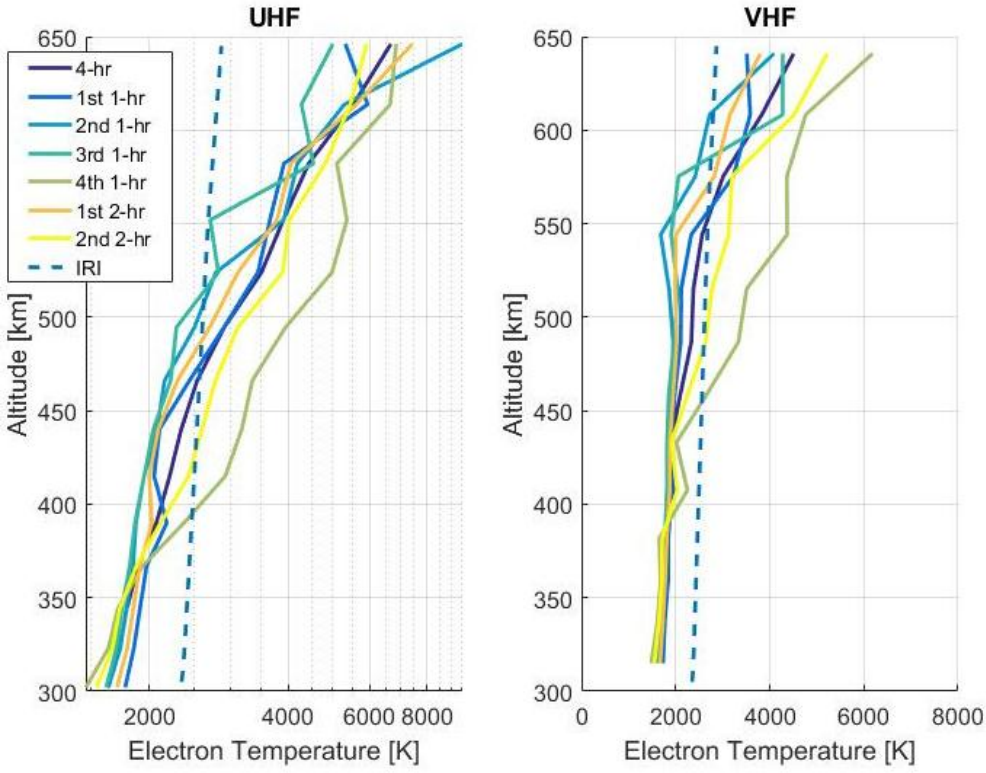


Figure 14 - Electron temperatures for the data-set averaged over different times

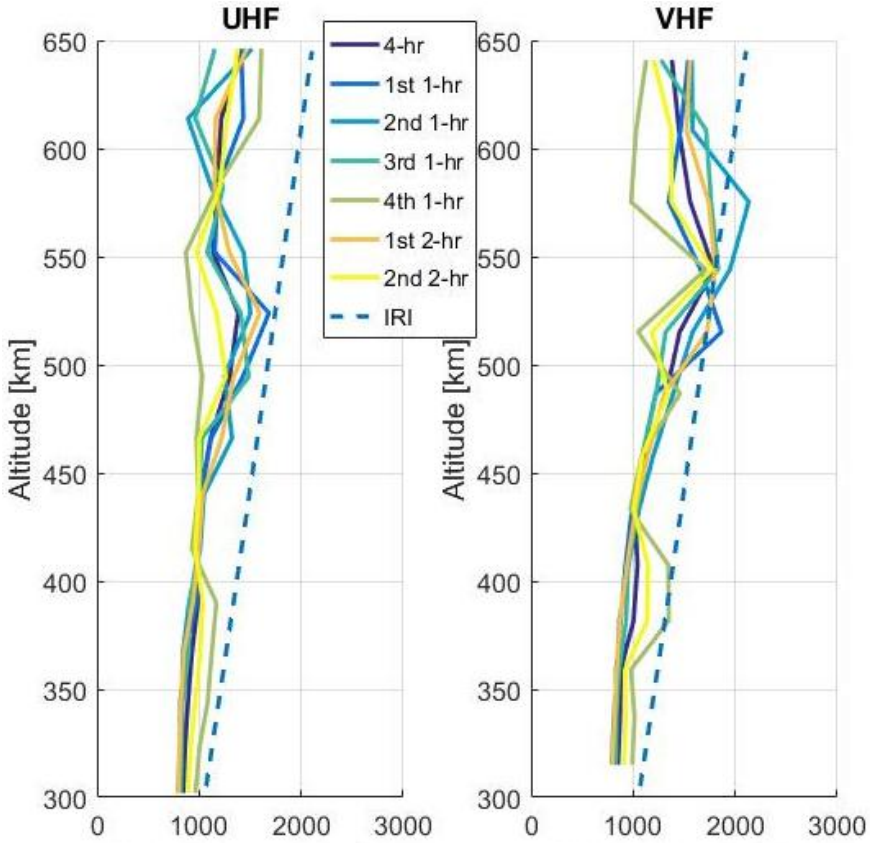


Figure 15 - Ion temperatures for the data-set averaged over different times

The investigation on time-frame has results as shown in Figure 13 showing electron densities. For both radar the full time-frame case is shown to be a good average value that smooths out any irregular curves that appear in other time-frames and this corresponds with Figure 14 and Figure 15 as well.

Electron density for the VHF acts much more expectedly with all curves having similar shape with only some difference in magnitude and a convergence about 600km. The shape of the UHF show some concern with slight bulges with 2nd and 3rd one-hour data-set as well as the 1st two-hour data-set that significantly affects the average.

The temperature graphs show quite mixed results with the ion temperatures in Figure 15 showing values less than the IRI model with the exception for a few cases. These differences are quite significant in some cases; at high altitudes this can be attributed to noise as shown in the previous figures but even at about 425km the IRI is showing temperatures roughly 40% higher. The opposite can be said for the electron temperatures that in UHF results shows 20% higher values than the IRI at 500km which is when the

electron density showed to be have similar values to IRI and no noticeable standard deviation.

6.2 RESULTS FROM PARAMETERISED FITTING

6.2.1 Data-sets averaged over four hours

6.2.1.1 Temperature

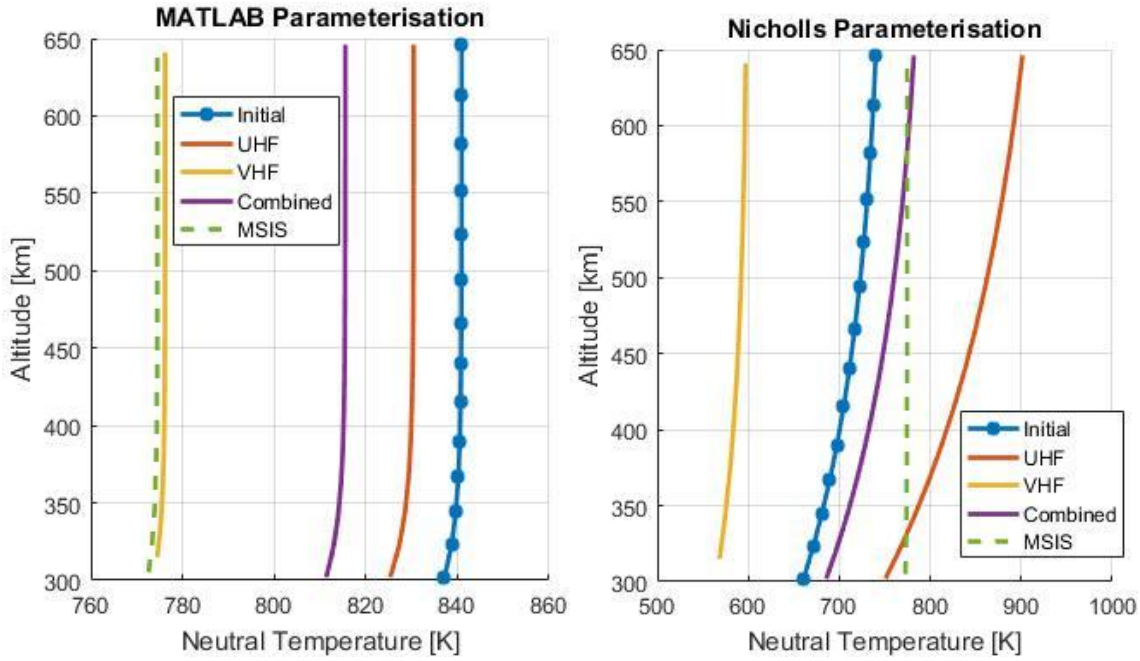


Figure 16 - Fitted neutral temperatures when averaged over the full time-frame.

As expected, all profiles for neutral temperature in Figure 16 to be an approximately near the exospheric temperature however, the MATLAB parameterisation moves its best estimates closer MSIS model's values where the Nicholls Parameterisation moves past the MSIS values for the UHF and in the opposite direction for the VHF.

6.2.1.2 Heat Transfer

The UHF and VHF with both methods show significantly different results, with the VHF showing a higher ion to neutral heat transfer rate than electron to ion however, although this net loss of heat is present in the MATLAB parameterisation, it is less than the Nicolls parameterisation.

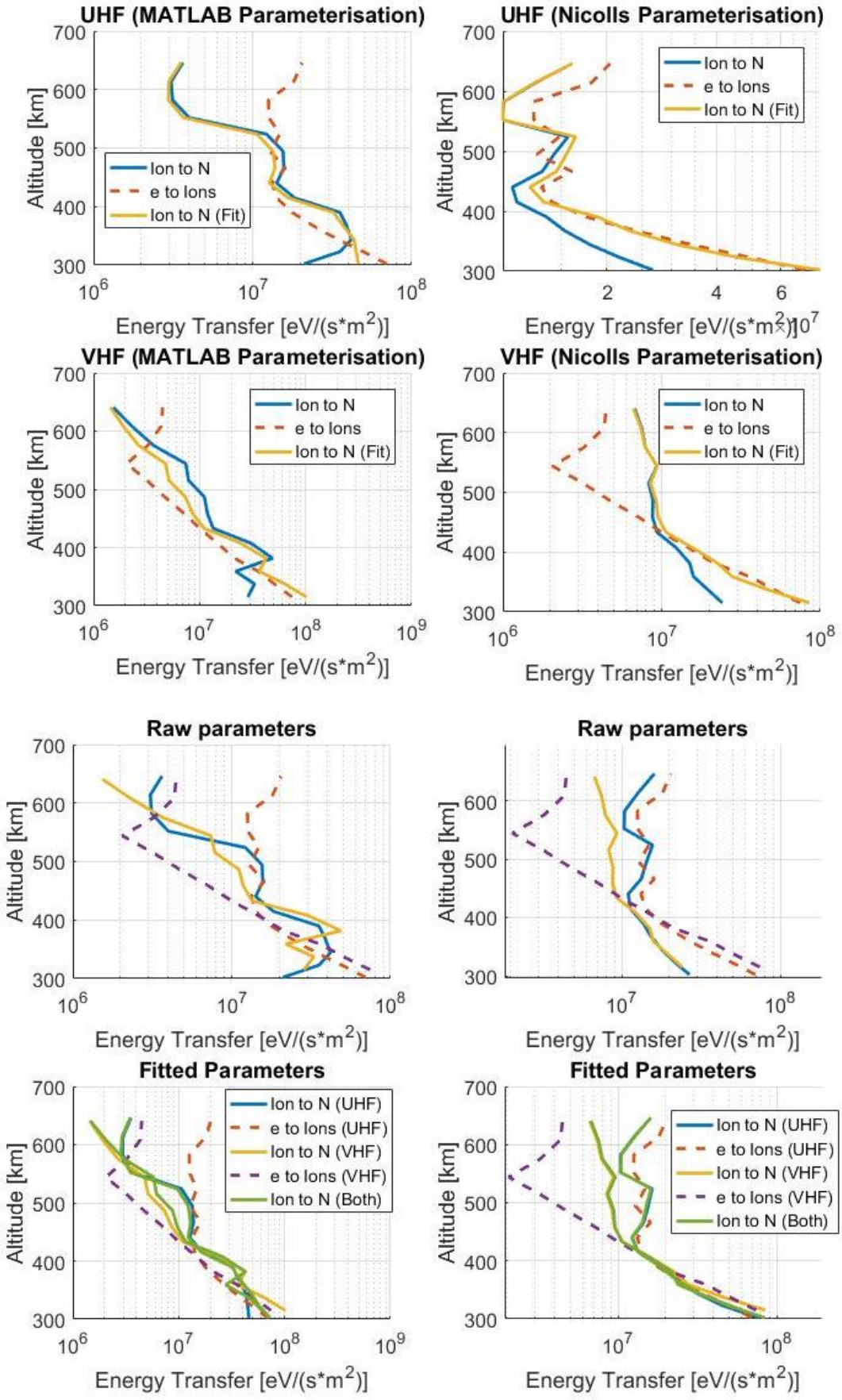


Figure 17 - Energy transfer rates. Top two rows are the fits for each radar. Bottom are all data-sets for the full time-frame split between initial guesses and best estimates for clarity

6.2.1.3 Neutral Densities

After fitting, all plots of neutral density put atomic oxygen as the dominating species, but for the Nicolls parameterisation, only until 400-450km where hydrogen populations become more prominent. The MATLAB parameterisation shows a higher transition point but as the MSIS Model Atmosphere has transition point above 600km altitude, both the Nicolls and MATLAB parameterisations seem unrealistic. In addition to this, the two parameterisations have very inconsistent profiles for minor species; the Nicolls parameterisation shows a lower population of helium than hydrogen with the MATLAB parameterisation. Over both radar, the MATLAB parameterisation has closer initial and fitted estimations, especially for atomic oxygen.

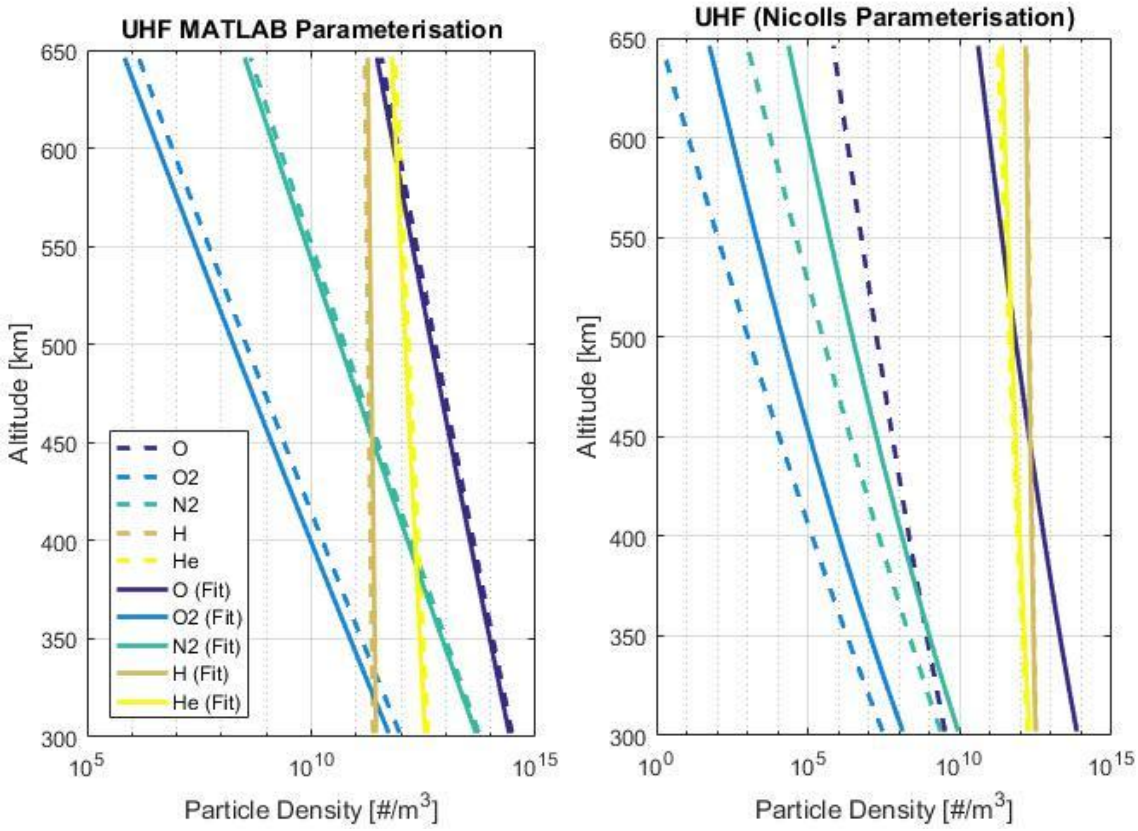


Figure 18 - Best estimate of neutral atmosphere from UHF data measurements

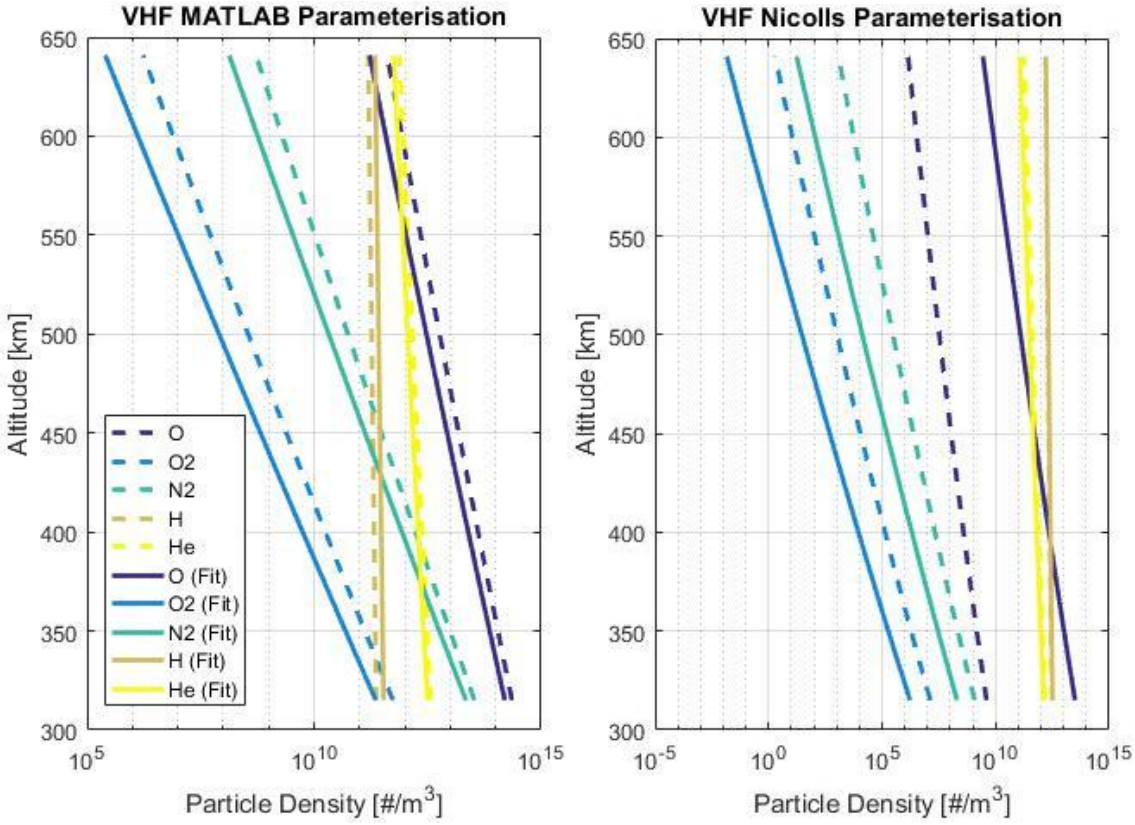


Figure 19 - Best estimate of neutral atmosphere from UHF data measurements

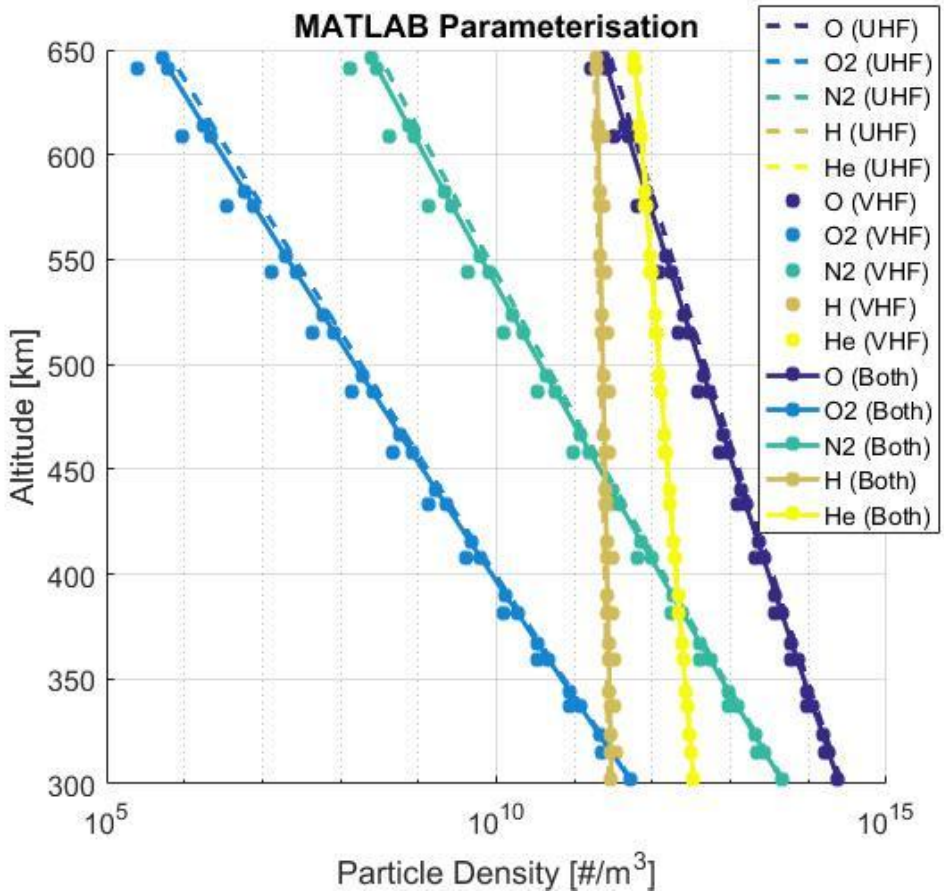


Figure 20 - All best estimates from four hour data-sets using MATLAB parameterisation

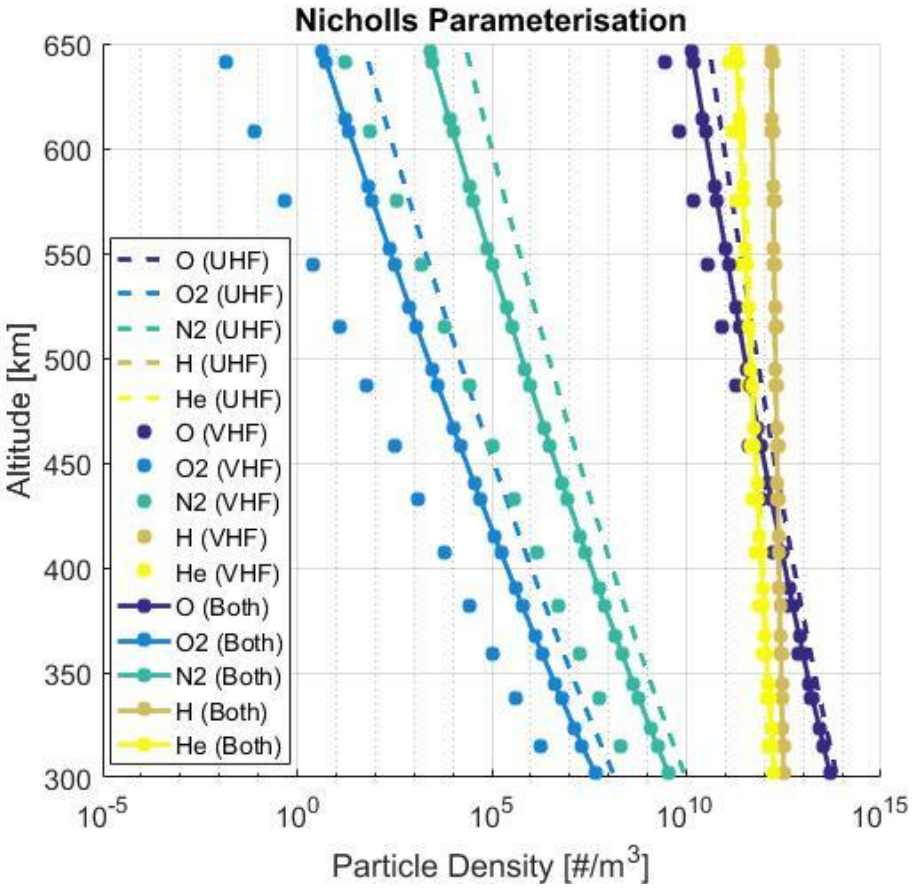


Figure 21 - All best estimates from four hour data-sets using Nicholls Parameterisation

6.2.2 Estimates from the partial data-sets

Comparisons between the two radar with each Parameterisation show similar temperatures except for the data-set averaged over the full four hours; that for the Nicolls parameterisation provided measurements that had a 200K difference between VHF and UHF. VHF measurement for this also shows a surprisingly similar measurement to the 2nd two-hour averaged data-set, to the extent these two profiles overlap. This difference is not as vast for the MATLAB parameterisation that shows a 50K difference. It is however, perhaps of more interest that the parameterisation shows a fitting of -35K and +15K for the VHF and UHF respectively. Both methods show the VHF with a much lower temperature for all profiles except the 2nd two-hour and 4th one-hour time-frames; the Nicolls parameterisation of these data-sets has both radar showing a temperature above 800K and increasing to ~1100K, this however seems unrealistic, especially as the 2nd two-hour data-set is the lowest temperature profile. The MATLAB parameterisation shows a clear increase over the day with each quarter case being greater than the previous, the

halves being averages between the quarters and the full case being a good average for the day.

The standard MSIS model estimates higher densities of nitrogen, atomic oxygen and molecular oxygen than both parameterisations with the Nicolls parameterisation showing about $10^5/m^3$ less for nitrogen. All data-sets also show MSIS with a much lower decrease in species' population over altitude. The individual data-sets show similar results in all species except atomic oxygen that has a large disparity between the 4th one-hour data-set and this occurs in both radar. Except for the Nicolls parameterisation of

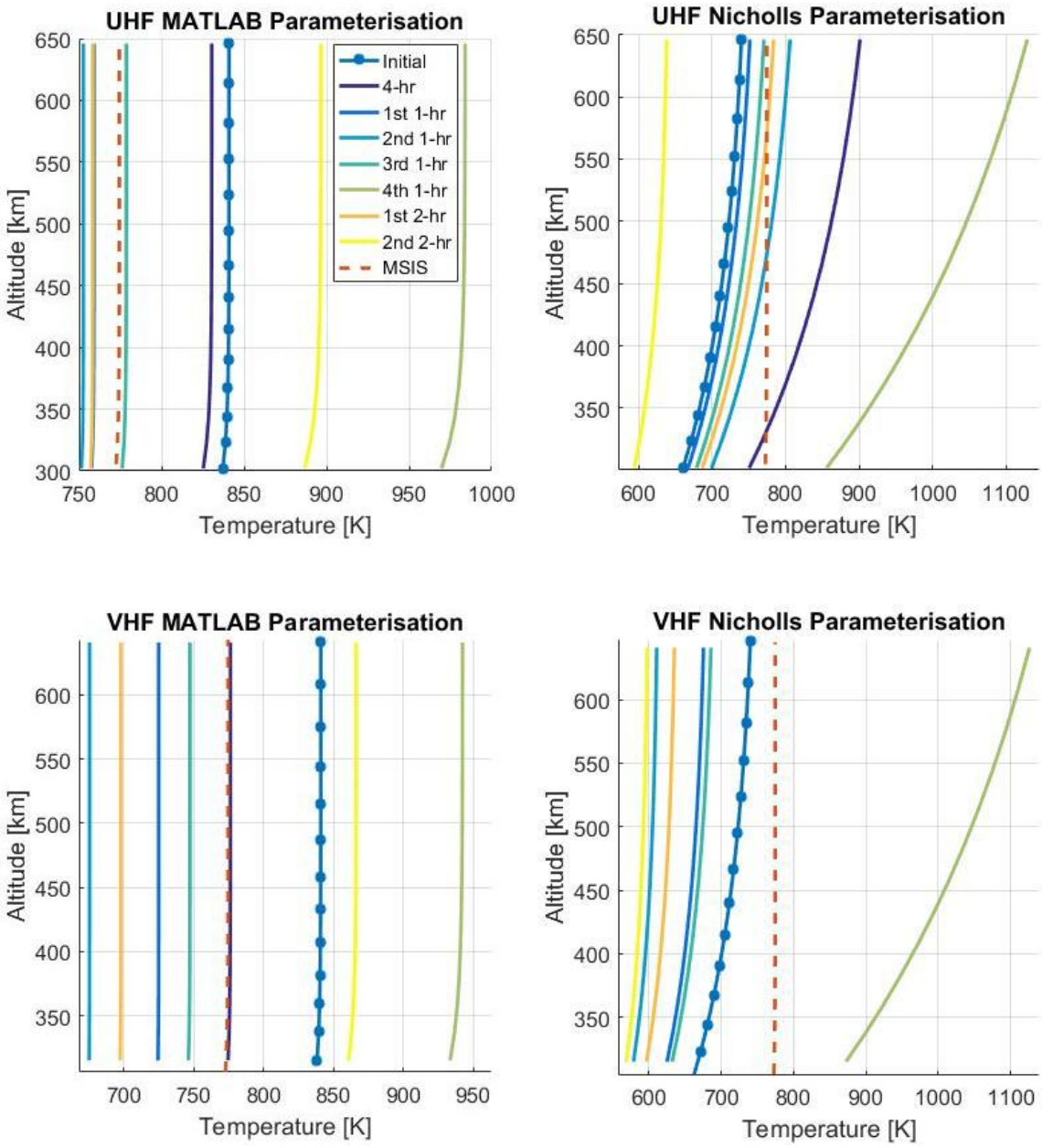


Figure 22 - Best estimates for neutral temperature for all single radar data-sets

atomic oxygen, all species' data-sets show very similar estimates with no sudden variations and no profiles from two different data-sets cross. The way each species varies with data-set is clearly different between parameterisation, for the MATLAB parameterisation, the later time-periods show higher levels of nitrogen, atomic oxygen and molecular oxygen.

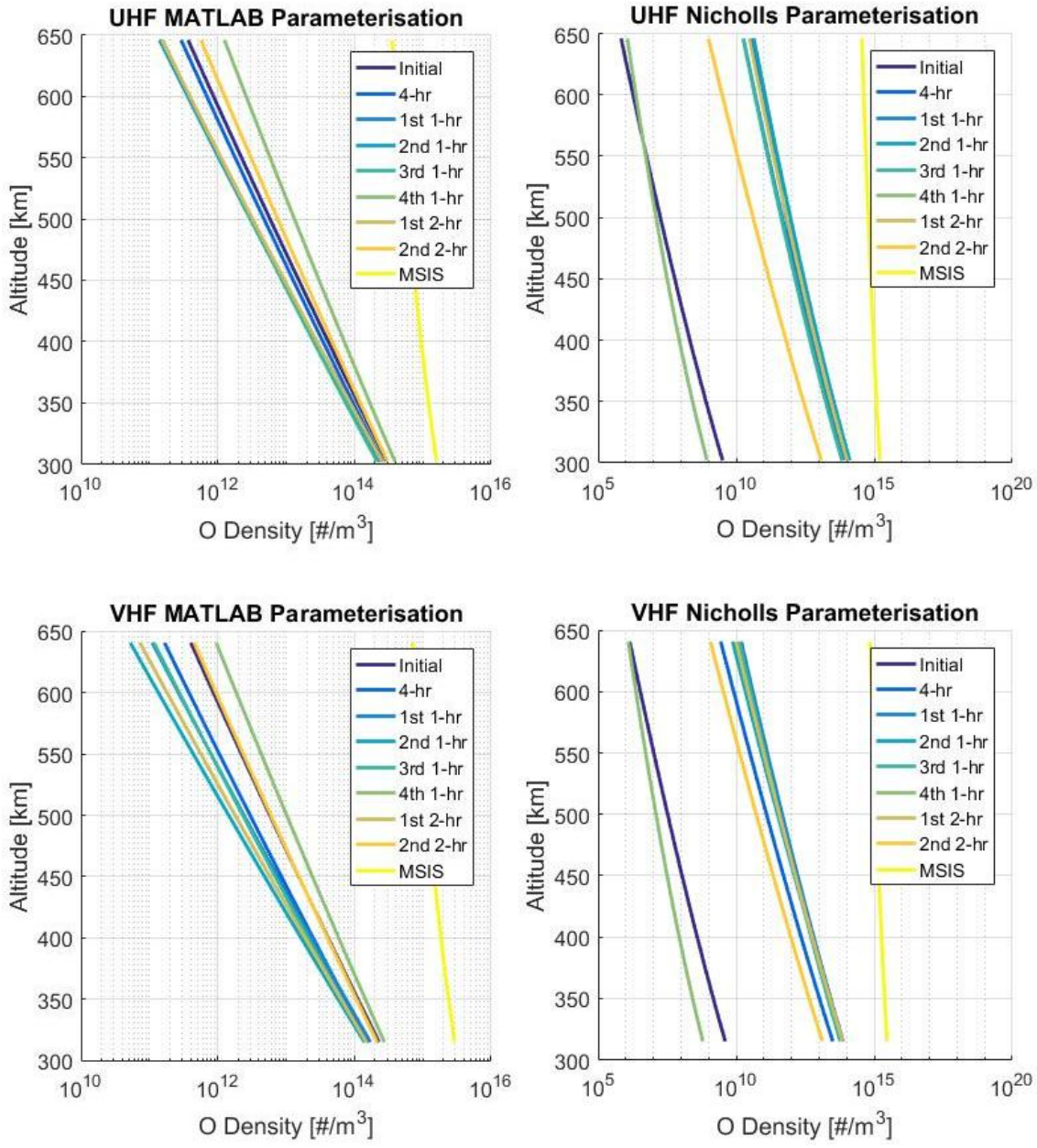


Figure 23 - Best estimates for atomic oxygen density for all single radar data-sets

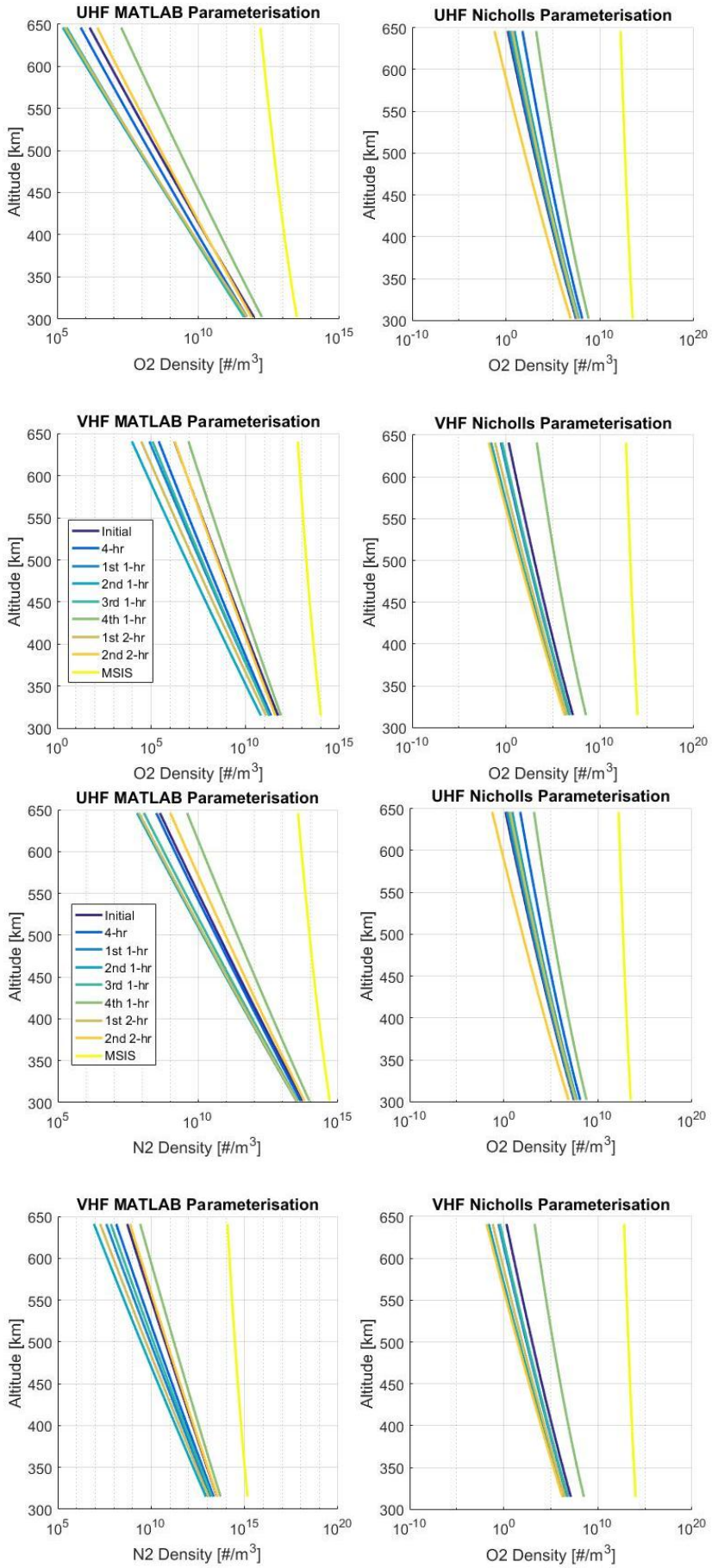


Figure 24 - Best estimates for particle density of other major species for all single radar data-sets

6.2.3 Comparison of residuals

As the electron to ion heat transfer is determined using only measured values, only the ion to neutral heat transfer is estimated and as both should be equal, a negative residual is an over estimation of the ion to neutral heat transfer and a positive value is an under estimation. The two parameterisations have very different residuals, for the Nicolls parameterisation, all data-sets other than the 4th one-hour and the 2nd two-hour on both radar show a large under estimation of the ion to neutral heat transfer in the range of about 350-400km. The largest under estimations are in the 2nd one-hour and 1st two-hour data-set; the 2nd two-hour data-set however, has a peak over estimation in both radar at this altitude. The Nicolls' VHF residuals typically have much larger peaks than the UHF that alternate between over and under estimation while its UHF seems to vary between under and over estimation more often but generally not reaching such high peaks. The MATLAB parameterisation shows much closer heat transfer estimates and no strange peaks until above 550km; these peaks are more common and have higher values in the UHF. It is expected for residuals to have several values over positive and negative one (~30% of measurements), as so many of these are within plus and minus one, there is a slight over estimation on the sigma values.

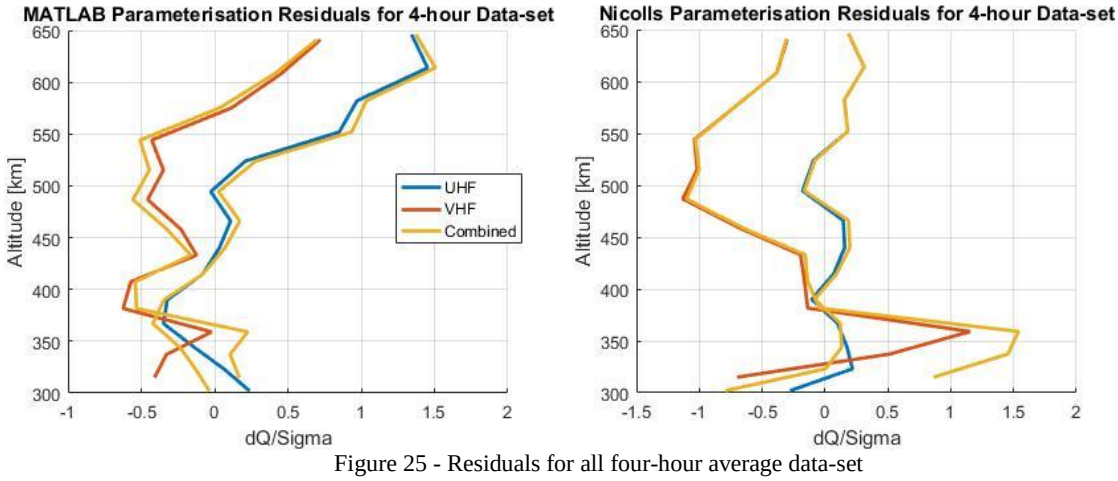


Figure 25 - Residuals for all four-hour average data-set

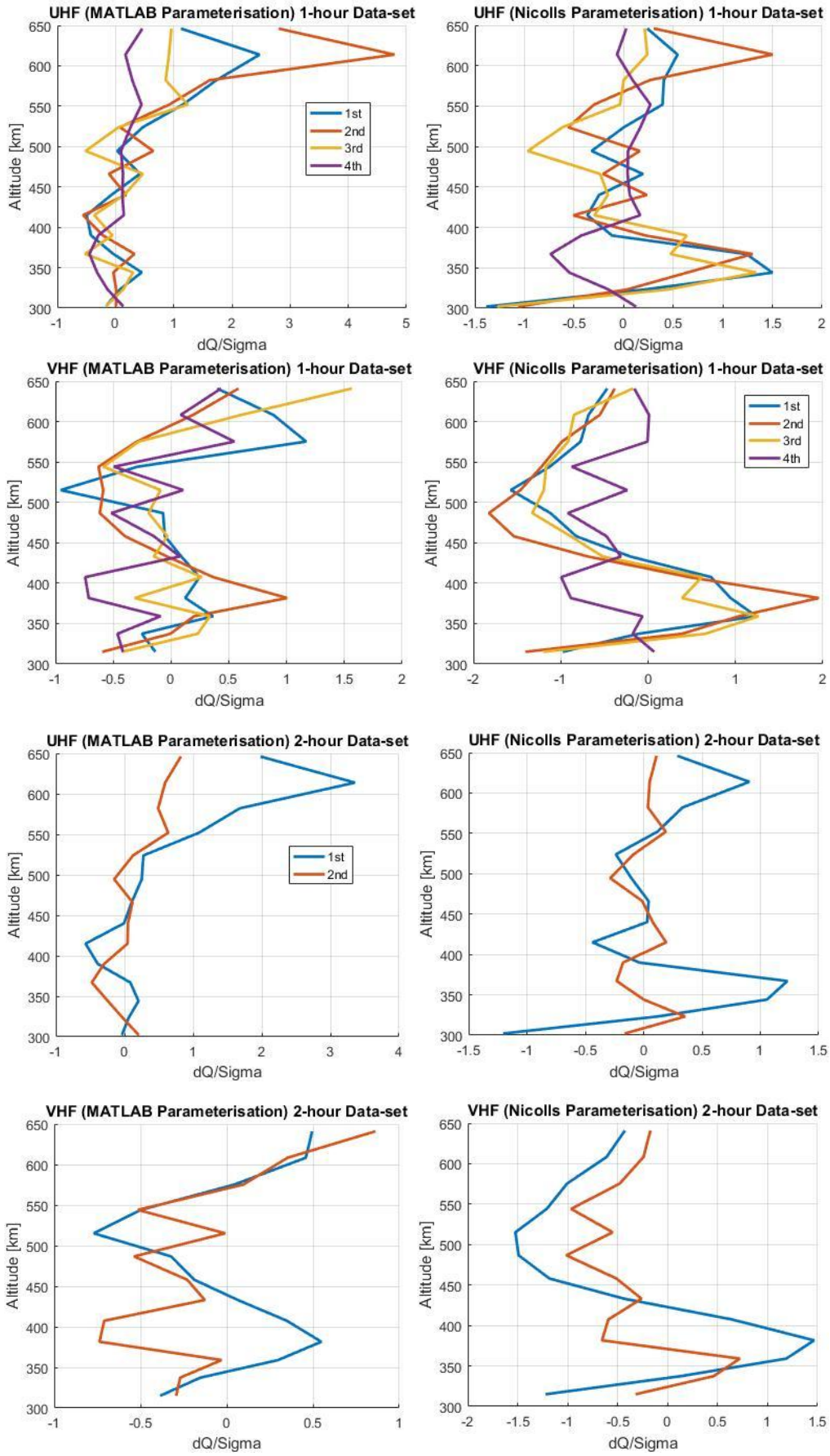


Figure 26 - Residuals for smaller time-frame data-sets on each radar

7 DISCUSSION

The MATLAB parameterisation shows on both radar that the temperature increases over the course of the day; although it is curious that the fit from the VHF lowers the full fit whereas the UHF's fit increases, it seems the UHF measures all temperatures as higher regardless of fitting method. At 300km, where the measured values have the lowest standard deviation, the VHF has a range of ~250K and 300K on MF and NM respectively while UHF only has around just over 200K with either method. These results would suggest an error with calibration between the two radar as both radar have very little standard deviation for measurements at this altitude and little noise is shown within Figure 10 and Figure 11.

The Nicolls parameterisation seems to have much more varied fits of neutral temperature than MF with the 2nd two-hour data-set being the lowest case on both radar, yet the 4th one-hour data-set being the highest. This wide scatter is most likely due to the Nicolls parameterisation not being robust enough to handle noise.

Both methods show the ions losing more energy to the neutrals than they are gaining from the electrons at altitudes above 425km, suggesting that there might be a significant Joule heating on the ions from oval currents or energy input from the magnetosphere as well. Unfortunately, due to the failure of the tristatic scan, the ionospheric ion drift velocities were not reliable enough to correct for this. It is also possible that the assumption of oxygen being the dominant component to neutral heating is not correct for these altitudes which would match some of the neutral density transition points, however these are very different from the MSIS model atmosphere densities so the corresponding ion to neutral heat transfer should also be called into question.

There is an irregularly large difference between the neutral densities in the Nicolls parameterisation and the MATLAB parameterisation with the best estimate for O_2 density at 300km on the order of $10^6/m^3$ and $10^{11}/m^3$ respectively; both estimates however, seem unrealistic and cause for concern given that the standard MSIS model gives an estimate with a density of $10^{13}/m^3$. The transition altitude between an atomic oxygen dominated and hydrogen dominated neutral atmosphere also greatly differs between

parameterisations with the MATLAB parameterisation estimating it is 200km higher than the Nicolls parameterisation. For both models, UHF increases the initial guess to get the best estimate while the VHF does the opposite; this is with all species except H and He, and these changes will be over an order of magnitude suggesting poor fitting. When comparing the different data-sets by species and with the standard MSIS model atmosphere, both methods show lower density of all species. The four-hour data-set for the MATLAB parameterisation seems to be a good approximation for conditions across the entire day as it is near the centre of the different data-sets in all species suggesting that this it is. The Nicolls parameterisation on the other hand has large contrasts between cases, with the 2nd two-hour data-set being the lowest case while the 4th one-hour data-set is the highest; This irregularity shows a lot of evidence for noise sensitivity as both these data-sets should be similar considering the 4th one-hour data-set measurements are included in the 2nd two-hour data-set.

It is very clear from the Nicolls parameterisation's graphs of the residuals and the several data-sets with systematic patterns in the residuals that it has not fitted well; the MATLAB parameterisation however, seems to have been fitted much better with only some large residuals at higher altitudes where noise is higher and measurements less reliable. The only Nicolls parameterisation data-sets that seem to have fitted is the 2nd two-hour data-set and the 4th one-hour data-set, potentially showing that the quieter conditions of the late day are preferable.

8 CONCLUSION

It is very clear from all fitted results and the graphs of the residuals that the Nicolls parameterisation did not successfully fit. As the 2nd two-hour data-set and the 4th one-hour data-set were the only cases that showed any signs of accurate fitting yet were the upper and lower extremes of the neutral temperature and densities, it does not seem that an accurate conclusion can be made from this method. The MATLAB parameterisation seemed to fit better but as all major species were below the standard MSIS atmosphere model by a factor of one hundred, this does not seem accurate either. Therefore, the experience from this experiment would suggest energy balance method is not an effective means for the determination of neutral parameters of the upper atmosphere at high latitudes. This result is however rather inconclusive due to the small number of points within the altitude range where the approximations are applicable. An increase of data points would hopefully allow the fitting to find a good average that can be applied across all data-sets. Even with this consideration made, the lack of a Joule heating estimation or the ability to account for ion heat-flux from the magnetosphere above, reduces the credibility of these results. The technical difficulties that caused the failure to accurately determine wind vectors need to be addressed and worked around before any solid conclusion can be made.

For future studies, a tristatic setup must be established. If there is similar outside RFI, the experiment should be carried out again, potentially using the advanced capabilities of KAIRA so that changes in velocity can be mapped more effectively. It would also be recommended to try and expand the altitude range or gather data from different days to allow more data and hopefully allow a more robust analysis.

9 APPENDIX A: MAJOR MATLAB CODES

9.1 ThesisRunNico

```
%% Will Stock Masters Thesis Script to Run Nicholls Parameterisation
%This script runs the least squares analysis contained within
%ThesisAnalysis and the functions
%
%Code Work flow
%1 Setup
%
%2 Run ThesisAnalysis for each radar and time frame (VHF, UHF, Both, and
%full, 4 quarters and 2 halves)
%
%3 Collect all output data from the ThesisAnalysis functions into groups
%e.g 'all VHF fitted neutral temperatures' etc
%
%4 Run ThesisAnalysis (Best outputs for both) for each radar and time frame
%(VHF, UHF, Both, and full, 4 quarters and 2 halves)
%
%5 Plot example graphs for "Analysis Methods" chapter
%
%6 Plot Heat Transfer Rates for all full time frame averaged data,
%  Plot dQ/Sigma, neutral atmosphere and temperatures for all data-sets


%% Setup
clc
clear
Root_Path = pwd;
Save_Path = strcat(Root_Path,'\Figures');

addpath(strcat(Root_Path,'\Meta'));
addpath(strcat(Root_Path,'\Matlab Functions'));
addpath(strcat(Root_Path,'\Matlab Functions\FMINSEARCHBND'));
addpath(strcat(Root_Path,'\Matlab Functions\T_e_ion'));
addpath(strcat(Root_Path,'\Matlab Functions\T_ion_neu'));
addpath(strcat(Root_Path,'\Matlab Functions\EARTH'));
addpath(strcat(Root_Path,'\Matlab Functions\cm_and_cb_utilities'));


Fnt_Size = 12;                              %Figure axis font size
Lat_Ram = 69.64;                            %Latitude of Ramsfjord Site [Deg]
Lon_Ram = 18.85;                            %Longitude of Ramsfjord Site [Deg]
Exp_yr = 2017;                              %Experiment year
Exp_d = 320;                                %Experiment Day of year
Exp_t = 36000;                              %Experiment Time [S]
h_lim = [300,650]*1e3;                      %Height restraints [km]
Sigma = 1;                                  %Placeholder for Sigma Value
MSISorNICO = 'NICO';

Exp_Data = {Lat_Ram,Lon_Ram,Exp_yr,...      %Experiment Data group to input
            Exp_d,Exp_t,h_lim};

Data_Path_V = strcat(Root_Path,'\Data\VHF\2016-11-15_beata_60@vhf');
Data_Path_U = strcat(Root_Path,'\Data\UHF\2016-11-15_beata_60@uhfa');
Data_Path_K = strcat(Root_Path,'\Data\KIR\2016-11-15_beata_60@kir');
Data_Path_S = strcat(Root_Path,'\Data\SOD\2016-11-15_beata_60@sod');
MSIS_Path = strcat(Root_Path,'\Meta\MSIS Neutral Profiles 15-11-16 full.txt');
IRI_Path = strcat(Root_Path,'\Meta\IRI Profile 15-11-16 with index.txt');
IRI = load(IRI_Path);
MSIS = load(MSIS_Path);

%Cutting reference data into same size matricies as radar data
MSIS_an = MSIS(h_lim(1)/1000<MSIS(:,1)&MSIS(:,1)<h_lim(2)/1000,:);
IRI_an = IRI(h_lim(1)/1000<IRI(:,1)&IRI(:,1)<h_lim(2)/1000,:);
PHp = sum([IRI(:,8:11),IRI(:,13)],2)/100;

%% VHF
%This section calls the Thesis Analysis function for all VHF data sets that
%outputs a large matrix of cells. This matrix is then unpacked by calling
%the AnalysisUnpacking function to give the final outputs that are plotted.

%Full time frame data set
```

```
[Atms_Fit_V,Atms_Raw_V,Best_Pars_V,dQSig_V,dTs_V,ne_Van,Qs_Fit_V,Ts_V,Z_Van,Sig_V,F10_7D,F10_7
,Ap,Anal_Setup] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
full',2);
%Unpacking variables
[Atm_Best_V,Atm_Start_V,Atm_Vi,Atm_VM,dTi_Van,dTe_Van,Q_Best_V,Q_Start_V,Ti_Van,Te_Van,Tn_fit_
V] = AnalysisUnpacking(Atms_Fit_V,Atms_Raw_V,dTs_V,Qs_Fit_V,Ts_V);
PHp_V = Atm_Vi(:,end);

% Getting VHF data for the time frame broken into 4 parts
% e.g: Each set of data is only averaged from 1/4 of the time period ending
% at "_[%]" of time.

[Atms_Fit_V25,Atms_Raw_V25,Best_Pars_V25,dQSig_V25,dTs_V25,ne_V25an,Qs_Fit_V25,Ts_V25,Z_V25an,
Sig_V25] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:52');
%Unpacking variables
[Atm_Best_V25,Atm_Start_V25,Atm_V25i,Atm_V25M,dTi_V25an,dTe_V25an,Q_Best_V25,Q_Start_V25,Ti_V2
5an,Te_V25an,Tn_fit_V25] =
AnalysisUnpacking(Atms_Fit_V25,Atms_Raw_V25,dTs_V25,Qs_Fit_V25,Ts_V25);

[Atms_Fit_V50,Atms_Raw_V50,Best_Pars_V50,dQSig_V50,dTs_V50,ne_V50an,Qs_Fit_V50,Ts_V50,Z_V50an,
Sig_V50] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 53:104');
%Unpacking variables
[Atm_Best_V50,Atm_Start_V50,Atm_V50i,Atm_V50M,dTi_V50an,dTe_V50an,Q_Best_V50,Q_Start_V50,Ti_V5
0an,Te_V50an,Tn_fit_V50] =
AnalysisUnpacking(Atms_Fit_V50,Atms_Raw_V50,dTs_V50,Qs_Fit_V50,Ts_V50);

[Atms_Fit_V75,Atms_Raw_V75,Best_Pars_V75,dQSig_V75,dTs_V75,ne_V75an,Qs_Fit_V75,Ts_V75,Z_V75an,
Sig_V75] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:156');
%Unpacking variables
[Atm_Best_V75,Atm_Start_V75,Atm_V75i,Atm_V75M,dTi_V75an,dTe_V75an,Q_Best_V75,Q_Start_V75,Ti_V7
5an,Te_V75an,Tn_fit_V75] =
AnalysisUnpacking(Atms_Fit_V75,Atms_Raw_V75,dTs_V75,Qs_Fit_V75,Ts_V75);

[Atms_Fit_V100,Atms_Raw_V100,Best_Pars_V100,dQSig_V100,dTs_V100,ne_V100an,Qs_Fit_V100,Ts_V100,
Z_V100an,Sig_V100] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
157:207');
%Unpacking variables
[Atm_Best_V100,Atm_Start_V100,Atm_V100i,Atm_V100M,dTi_V100an,dTe_V100an,Q_Best_V100,Q_Start_V1
00,Ti_V100an,Te_V100an,Tn_fit_V100] =
AnalysisUnpacking(Atms_Fit_V100,Atms_Raw_V100,dTs_V100,Qs_Fit_V100,Ts_V100);

% Getting VHF data for the time frame broken into 2 parts
% e.g: Each set of data is only averaged from 1/2 of the time period with
% markings of "_half1"/"_half2" for the lower/upper halves respectively.
[Atms_Fit_Vhalf1,Atms_Raw_Vhalf1,Best_Pars_Vhalf1,dQSig_Vhalf1,dTs_Vhalf1,ne_Vhalf1an,Qs_Fit_V
half1,Ts_Vhalf1,Z_Vhalf1an,Sig_Vhalf1] =
ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:104');
%Unpacking variables
[Atm_Best_Vhalf1,Atm_Start_Vhalf1,Atm_Vhalf1i,Atm_Vhalf1M,dTi_Vhalf1an,dTe_Vhalf1an,Q_Best_Vha
lf1,Q_Start_Vhalf1,Ti_Vhalf1an,Te_Vhalf1an,Tn_fit_Vhalf1] =
AnalysisUnpacking(Atms_Fit_Vhalf1,Atms_Raw_Vhalf1,dTs_Vhalf1,Qs_Fit_Vhalf1,Ts_Vhalf1);

[Atms_Fit_Vhalf2,Atms_Raw_Vhalf2,Best_Pars_Vhalf2,dQSig_Vhalf2,dTs_Vhalf2,ne_Vhalf2an,Qs_Fit_V
half2,Ts_Vhalf2,Z_Vhalf2an,Sig_Vhalf2] =
ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:207');
%Unpacking variables
[Atm_Best_Vhalf2,Atm_Start_Vhalf2,Atm_Vhalf2i,Atm_Vhalf2M,dTi_Vhalf2an,dTe_Vhalf2an,Q_Best_Vha
lf2,Q_Start_Vhalf2,Ti_Vhalf2an,Te_Vhalf2an,Tn_fit_Vhalf2] =
AnalysisUnpacking(Atms_Fit_Vhalf2,Atms_Raw_Vhalf2,dTs_Vhalf2,Qs_Fit_Vhalf2,Ts_Vhalf2);

%% UHF
%This section calls the Thesis Analysis function for all UHF data sets that
%outputs a large matrix of cells. This matrix is then unpacked by calling
%the AnalysisUnpacking function to give the final outputs that are plotted.

%Full time frame data set
[Atms_Fit_U,Atms_Raw_U,Best_Pars_U,dQSig_U,dTs_U,ne_Uan,Qs_Fit_U,Ts_U,Z_Uan,Sig_U,F10_7D,F10_7
,Ap,Anal_Setup] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
full',2);
%Unpacking variables
[Atm_Best_U,Atm_Start_U,Atm_Ui,Atm_UM,dTi_Uan,dTe_Uan,Q_Best_U,Q_Start_U,Ti_Uan,Te_Uan,Tn_fit_
U] = AnalysisUnpacking(Atms_Fit_U,Atms_Raw_U,dTs_U,Qs_Fit_U,Ts_U);
PHp_U = Atm_Ui(:,end);

% Getting VHF data for the time frame broken into 4 parts
% e.g: Each set of data is only averaged from 1/4 of the time period ending
% at "_[%]" of time.
```

```
[Atms_Fit_U25,Atms_Raw_U25,Best_Pars_U25,dQSig_U25,dTs_U25,ne_U25an,Qs_Fit_U25,Ts_U25,Z_U25an,
Sig_U25] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:52');
%Unpacking variables
[Atm_Best_U25,Atm_Start_U25,Atm_U25i,Atm_U25M,dTi_U25an,dTe_U25an,Q_Best_U25,Q_Start_U25,Ti_U2
5an,Te_U25an,Tn_fit_U25] =
AnalysisUnpacking(Atms_Fit_U25,Atms_Raw_U25,dTs_U25,Qs_Fit_U25,Ts_U25);

[Atms_Fit_U50,Atms_Raw_U50,Best_Pars_U50,dQSig_U50,dTs_U50,ne_U50an,Qs_Fit_U50,Ts_U50,Z_U50an,
Sig_U50] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 53:104');
%Unpacking variables
[Atm_Best_U50,Atm_Start_U50,Atm_U50i,Atm_U50M,dTi_U50an,dTe_U50an,Q_Best_U50,Q_Start_U50,Ti_U5
0an,Te_U50an,Tn_fit_U50] =
AnalysisUnpacking(Atms_Fit_U50,Atms_Raw_U50,dTs_U50,Qs_Fit_U50,Ts_U50);

[Atms_Fit_U75,Atms_Raw_U75,Best_Pars_U75,dQSig_U75,dTs_U75,ne_U75an,Qs_Fit_U75,Ts_U75,Z_U75an,
Sig_U75] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:156');
%Unpacking variables
[Atm_Best_U75,Atm_Start_U75,Atm_U75i,Atm_U75M,dTi_U75an,dTe_U75an,Q_Best_U75,Q_Start_U75,Ti_U7
5an,Te_U75an,Tn_fit_U75] =
AnalysisUnpacking(Atms_Fit_U75,Atms_Raw_U75,dTs_U75,Qs_Fit_U75,Ts_U75);

[Atms_Fit_U100,Atms_Raw_U100,Best_Pars_U100,dQSig_U100,dTs_U100,ne_U100an,Qs_Fit_U100,Ts_U100,
Z_U100an,Sig_U100] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
157:207');
%Unpacking variables
[Atm_Best_U100,Atm_Start_U100,Atm_U100i,Atm_U100M,dTi_U100an,dTe_U100an,Q_Best_U100,Q_Start_U1
00,Ti_U100an,Te_U100an,Tn_fit_U100] =
AnalysisUnpacking(Atms_Fit_U100,Atms_Raw_U100,dTs_U100,Qs_Fit_U100,Ts_U100);

% Getting UHF data for the time frame broken into 2 parts
% e.g: Each set of data is only averaged from 1/2 of the time period with
% markings of "_half1"/"_half2" for the lower/upper halves respectively.
[Atms_Fit_Uhalf1,Atms_Raw_Uhalf1,Best_Pars_Uhalf1,dQSig_Uhalf1,dTs_Uhalf1,ne_Uhalf1an,Qs_Fit_U
half1,Ts_Uhalf1,Z_Uhalf1an,Sig_Uhalf1] =
ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:104');
%Unpacking variables
[Atm_Best_Uhalf1,Atm_Start_Uhalf1,Atm_Uhalf1i,Atm_Uhalf1M,dTi_Uhalf1an,dTe_Uhalf1an,Q_Best_Uha
lf1,Q_Start_Uhalf1,Ti_Uhalf1an,Te_Uhalf1an,Tn_fit_Uhalf1] =
AnalysisUnpacking(Atms_Fit_Uhalf1,Atms_Raw_Uhalf1,dTs_Uhalf1,Qs_Fit_Uhalf1,Ts_Uhalf1);

[Atms_Fit_Uhalf2,Atms_Raw_Uhalf2,Best_Pars_Uhalf2,dQSig_Uhalf2,dTs_Uhalf2,ne_Uhalf2an,Qs_Fit_U
half2,Ts_Uhalf2,Z_Uhalf2an,Sig_Uhalf2] =
ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:207');
%Unpacking variables
[Atm_Best_Uhalf2,Atm_Start_Uhalf2,Atm_Uhalf2i,Atm_Uhalf2M,dTi_Uhalf2an,dTe_Uhalf2an,Q_Best_Uha
lf2,Q_Start_Uhalf2,Ti_Uhalf2an,Te_Uhalf2an,Tn_fit_Uhalf2] =
AnalysisUnpacking(Atms_Fit_Uhalf2,Atms_Raw_Uhalf2,dTs_Uhalf2,Qs_Fit_Uhalf2,Ts_Uhalf2);


%% Creating matricies for "Both"
%As the combined data-set has no data path, it does not need several steps
%from Thesis Analysis, therefore it needs concantination and then being
%input directly to the least square analysis function "BestOutputs"

Z_Ban = [Z_Van;Z_Uan];
Te_Ban = [Te_Van;Te_Uan];
Ti_Ban = [Ti_Van;Ti_Uan];
ne_Ban = [ne_Van;ne_Uan];
dTi_Ban = [dTi_Van;dTi_Uan];
dTe_Ban = [dTe_Van;dTe_Uan];
Atm_Bi = [Atm_Vi;Atm_Ui];
PHp_B = [PHp_V;PHp_U];


%% Least Squares Analysis for "Both"
%Inputs for analysis of error function
p_index = Anal_Setup{1};
fixed_pars = Anal_Setup{2};
start_guess = Anal_Setup{3};
LB = Anal_Setup{4};
UB = Anal_Setup{5};
options = Anal_Setup{6};
Z0 = 100;
Tinf = TExosphere(mean(F10_7D));
ORefDensity = Atm_VM(1:1);
```

```
err_ins_B =
{Z_Ban,Ti_Ban,Te_Ban,dTi_Ban,dTe_Ban,ne_Ban,PHp_B,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,MSISorNIC
O,Z0,Tinf,ORefDensity};

err_fcn_tag_B =
@(pars)err_fcn2(pars,p_index,fixed_pars,Z_Ban,Ti_Ban,Te_Ban,ne_Ban,PHp_B,Lat_Ram,Lon_Ram,Exp_y
r,Exp_d,Exp_t,Sigma);

[Best_Pars_B,Atm_Best_B,Q_Best_B,Atm_Start_B,Q_Start_B,dQSig_B] =
BestOutputs(err_fcn_tag_B,p_index,fixed_pars,start_guess,LB,UB,options,err_ins_B);

Tn_fit_B = Atm_Best_B(:,2);

%% Showing all best pars
disp('Lower Bounds=')
disp(LB(1,:))
Best_Pars_V
Best_Pars_V25
Best_Pars_V50
Best_Pars_V75
Best_Pars_V100
Best_Pars_Vhalf1
Best_Pars_Uhalf2
Best_Pars_U
Best_Pars_U25
Best_Pars_U50
Best_Pars_U75
Best_Pars_U100
Best_Pars_Uhalf1
Best_Pars_Uhalf2
Best_Pars_B
disp('Upper Bounds=')
disp(UB(1,:))

%% -------------------------Collecting Variables------------------------ %%
%% VHF
ne_All_VHF = [ne_Van,ne_V25an,ne_V50an,ne_V75an,ne_V100an...
              ,ne_Vhalf1an,ne_Vhalf2an];

ODen_All_VHF = [Atm_Start_V(:,3),Atm_Best_V(:,3)...
                ,Atm_Best_V25(:,3)...
                ,Atm_Best_V50(:,3)...
                ,Atm_Best_V75(:,3)...
                ,Atm_Best_V100(:,3)...
                ,Atm_Best_Vhalf1(:,3)...
                ,Atm_Best_Vhalf2(:,3)...
                ,Atm_VM(:,1)];

O2Den_All_VHF = [Atm_Start_V(:,4),Atm_Best_V(:,4)...
                 ,Atm_Best_V25(:,4)...
                 ,Atm_Best_V50(:,4)...
                 ,Atm_Best_V75(:,4)...
                 ,Atm_Best_V100(:,4)...
                 ,Atm_Best_Vhalf1(:,4)...
                 ,Atm_Best_Vhalf2(:,4)...
                 ,Atm_VM(:,2)];

N2Den_All_VHF = [Atm_Start_V(:,5),Atm_Best_V(:,5)...
                 ,Atm_Best_V25(:,5)...
                 ,Atm_Best_V50(:,5)...
                 ,Atm_Best_V75(:,5)...
                 ,Atm_Best_V100(:,5)...
                 ,Atm_Best_Vhalf1(:,5)...
                 ,Atm_Best_Vhalf2(:,5)...
                 ,Atm_VM(:,3)];

HDen_All_VHF = [Atm_Start_V(:,6),Atm_Best_V(:,6)...
                ,Atm_Best_V25(:,6)...
                ,Atm_Best_V50(:,6)...
                ,Atm_Best_V75(:,6)...
                ,Atm_Best_V100(:,6)...
                ,Atm_Best_Vhalf1(:,6)...
                ,Atm_Best_Vhalf2(:,6)...
                ,Atm_VM(:,4)];

HeDen_All_VHF = [Atm_Start_V(:,7),Atm_Best_V(:,7)...
```

```
                ,Atm_Best_V25(:,7)...
                ,Atm_Best_V50(:,7)...
                ,Atm_Best_V75(:,7)...
                ,Atm_Best_V100(:,7)...
                ,Atm_Best_Vhalf1(:,7)...
                ,Atm_Best_Vhalf2(:,7)...
                ,Atm_VM(:,5)];

Tn_All_VHF = [Atm_Start_V(:,2),Atm_Best_V(:,2)...
             ,Atm_Best_V25(:,2)...
             ,Atm_Best_V50(:,2)...
             ,Atm_Best_V75(:,2)...
             ,Atm_Best_V100(:,2)...
             ,Atm_Best_Vhalf1(:,2)...
             ,Atm_Best_Vhalf2(:,2)];

Q_OP2n_All_VHF = [Q_Start_V(:,1),Q_Best_V(:,1)...
                 ,Q_Start_V25(:,1),Q_Best_V25(:,1)...
                 ,Q_Start_V50(:,1),Q_Best_V50(:,1)...
                 ,Q_Start_V75(:,1),Q_Best_V75(:,1)...
                 ,Q_Start_V100(:,1),Q_Best_V100(:,1)...
                 ,Q_Start_Vhalf1(:,1),Q_Best_Vhalf1(:,1)...
                 ,Q_Start_Vhalf2(:,1),Q_Best_Vhalf2(:,1)];

Q_e2OP_All_VHF = [Q_Start_V(:,2),Q_Best_V(:,2)...
                 ,Q_Start_V25(:,2),Q_Best_V25(:,2)...
                 ,Q_Start_V50(:,2),Q_Best_V50(:,2)...
                 ,Q_Start_V75(:,2),Q_Best_V75(:,2)...
                 ,Q_Start_V100(:,2),Q_Best_V100(:,2)...
                 ,Q_Start_Vhalf1(:,2),Q_Best_Vhalf1(:,2)...
                 ,Q_Start_Vhalf2(:,2),Q_Best_Vhalf2(:,2)];


%% UHF
ne_All_UHF = [ne_Uan,ne_U25an,ne_U50an,ne_U75an,ne_U100an...
             ,ne_Uhalf1an,ne_Uhalf2an];

ODen_All_UHF = [Atm_Start_U(:,3),Atm_Best_U(:,3)...
               ,Atm_Best_U25(:,3)...
               ,Atm_Best_U50(:,3)...
               ,Atm_Best_U75(:,3)...
               ,Atm_Best_U100(:,3)...
               ,Atm_Best_Uhalf1(:,3)...
               ,Atm_Best_Uhalf2(:,3)...
               ,Atm_UM(:,1)];

O2Den_All_UHF = [Atm_Start_U(:,4),Atm_Best_U(:,4)...
                ,Atm_Best_U25(:,4)...
                ,Atm_Best_U50(:,4)...
                ,Atm_Best_U75(:,4)...
                ,Atm_Best_U100(:,4)...
                ,Atm_Best_Uhalf1(:,4)...
                ,Atm_Best_Uhalf2(:,4)...
                ,Atm_UM(:,2)];

N2Den_All_UHF = [Atm_Start_U(:,5),Atm_Best_U(:,5)...
                ,Atm_Best_U25(:,5)...
                ,Atm_Best_U50(:,5)...
                ,Atm_Best_U75(:,5)...
                ,Atm_Best_U100(:,5)...
                ,Atm_Best_Uhalf1(:,5)...
                ,Atm_Best_Uhalf2(:,5)...
                ,Atm_UM(:,3)];

HDen_All_UHF = [Atm_Start_U(:,6),Atm_Best_U(:,6)...
               ,Atm_Best_U25(:,6)...
               ,Atm_Best_U50(:,6)...
               ,Atm_Best_U75(:,6)...
               ,Atm_Best_U100(:,6)...
               ,Atm_Best_Uhalf1(:,6)...
               ,Atm_Best_Uhalf2(:,6)...
               ,Atm_UM(:,4)];

HeDen_All_UHF = [Atm_Start_U(:,7),Atm_Best_U(:,7)...
                ,Atm_Best_U25(:,7)...
                ,Atm_Best_U50(:,7)...
                ,Atm_Best_U75(:,7)...
```

```
                ,Atm_Best_U100(:,7)...
                ,Atm_Best_Uhalf1(:,7)...
                ,Atm_Best_Uhalf2(:,7)...
                ,Atm_UM(:,5)];

Tn_All_UHF = [Atm_Start_U(:,2),Atm_Best_U(:,2)...
             ,Atm_Best_U25(:,2)...
             ,Atm_Best_U50(:,2)...
             ,Atm_Best_U75(:,2)...
             ,Atm_Best_U100(:,2)...
             ,Atm_Best_Uhalf1(:,2)...
             ,Atm_Best_Uhalf2(:,2)];

Q_OP2n_All_UHF = [Q_Start_U(:,1),Q_Best_U(:,1)...
                 ,Q_Start_U25(:,1),Q_Best_U25(:,1)...
                 ,Q_Start_U50(:,1),Q_Best_U50(:,1)...
                 ,Q_Start_U75(:,1),Q_Best_U75(:,1)...
                 ,Q_Start_U100(:,1),Q_Best_U100(:,1)...
                 ,Q_Start_Uhalf1(:,1),Q_Best_Uhalf1(:,1)...
                 ,Q_Start_Uhalf2(:,1),Q_Best_Uhalf2(:,1)];

Q_e2OP_All_UHF = [Q_Start_U(:,2),Q_Best_U(:,2)...
                 ,Q_Start_U25(:,2),Q_Best_U25(:,2)...
                 ,Q_Start_U50(:,2),Q_Best_U50(:,2)...
                 ,Q_Start_U75(:,2),Q_Best_U75(:,2)...
                 ,Q_Start_U100(:,2),Q_Best_U100(:,2)...
                 ,Q_Start_Uhalf1(:,2),Q_Best_Uhalf1(:,2)...
                 ,Q_Start_Uhalf2(:,2),Q_Best_Uhalf2(:,2)];

%Final Parameters
Best_Pars =
[Best_Pars_V;Best_Pars_V25;Best_Pars_V50;Best_Pars_V75;Best_Pars_V100;Best_Pars_Vhalf1;Best_Pa
rs_Vhalf2;...

Best_Pars_U;Best_Pars_U25;Best_Pars_U50;Best_Pars_U75;Best_Pars_U100;Best_Pars_Uhalf1;Best_Par
s_Uhalf2;...
             Best_Pars_B];

Best_Atms_V =
[Atm_Best_V,Atm_Best_V25,Atm_Best_V50,Atm_Best_V75,Atm_Best_V100,Atm_Best_Vhalf1,Atm_Best_Vhal
f2];
Best_Atms_U =
[Atm_Best_U,Atm_Best_U25,Atm_Best_U50,Atm_Best_U75,Atm_Best_U100,Atm_Best_Uhalf1,Atm_Best_Uhal
f2];
Best_Atms_B = [Atm_Best_B];

%% -------------------------Analysis of Results------------------------- %%
%Determining average of the best parameters to see what are the best parameters for anytime of
day
Avg_Best_Pars = mean(Best_Pars,1);

%finding residual of Best Parameter investigation
%This is applicable to both radar
BPSig = Best_Pars./Avg_Best_Pars;

%Finding the standard deviations of the fitted neutrals, excluding initial
%and MSIS values from XDen_All_YHF matrix
OSig_V = std(ODen_All_VHF(:,2:2:end-1),0,2);
O2Sig_V = std(O2Den_All_VHF(:,2:2:end-1),0,2);
N2Sig_V = std(N2Den_All_VHF(:,2:2:end-1),0,2);
HSig_V = std(HDen_All_VHF(:,2:2:end-1),0,2);
HeSig_V = std(HeDen_All_VHF(:,2:2:end-1),0,2);

%Standard deviation for individual heat transfer rates of
%quartered data-sets
Sig_i2n_V_Quart = std(Q_OP2n_All_VHF(:,4:2:10),0,2);
Sig_e2i_V_Quart = std(Q_e2OP_All_VHF(:,4:2:10),0,2);

%Totalling the standard deviations of quartered data-sets
SigQ_V_Quart = sqrt(Sig_i2n_V_Quart.^2+Sig_e2i_V_Quart.^2);

%Determining residuals of quartered data-sets
dQSig_V_Quart = (mean(Q_e2OP_All_VHF(:,4:2:10),2)-
mean(Q_OP2n_All_VHF(:,4:2:10),2))./SigQ_V_Quart;

%Standard deviation for individual heat transfer rates of
%halved data-sets
```

```
Sig_i2n_V_Half = std(Q_OP2n_All_VHF(:,12:2:14),0,2);
Sig_e2i_V_Half = std(Q_e2OP_All_VHF(:,12:2:14),0,2);

%Totalling the standard deviations of halved data-sets
SigQ_V_Half = sqrt(Sig_i2n_V_Half.^2+Sig_e2i_V_Half.^2);

%Determining residuals of halved data-sets
dQSig_V_Half = (mean(Q_e2OP_All_VHF(:,12:2:14),2)-
mean(Q_OP2n_All_VHF(:,12:2:14),2))./SigQ_V_Half;

%Totalling all net heat transfers to find the best estimate for weighting
SigEst_V = SigQ_V_Quart/2+SigQ_V_Half/sqrt(2);

%Running the inverse problem again with the best estimated inputs and
%weighting
[Chi2_V,Atm_Final_V,Q_Final_V] =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z_Van,Ti_Van,Te_Van,ne_Van,PHp_V,SigEst_V,2);
dQSig_Final_V = (Q_Final_V(:,2)-Q_Final_V(:,1))./SigEst_V;

%Displaying all standard deviations and residuals
SigComparison_V = [SigEst_V,Sig_V,SigQ_V_Quart,SigQ_V_Half]
dQSigComparison_V = [dQSig_Final_V,dQSig_V,dQSig_V_Quart,dQSig_V_Half]

%% UHF
%Determining average of the best parameters to see what are the best parameters for anytime of
day
[SigO_U,SigO2_U,SigN2_U,SigH_U,SigHe_U] = Atm_Std(Best_Atms_U);

%Finding the standard deviations of the fitted neutrals, excluding initial
%and MSIS values from XDen_All_YHF matrix
OSig_U = std(ODen_All_UHF(:,2:2:end-1),0,2);
O2Sig_U = std(O2Den_All_UHF(:,2:2:end-1),0,2);
N2Sig_U = std(N2Den_All_UHF(:,2:2:end-1),0,2);
HSig_U = std(HDen_All_UHF(:,2:2:end-1),0,2);
HeSig_U = std(HeDen_All_UHF(:,2:2:end-1),0,2);

%Standard deviation for individual heat transfer rates of
%quartered data-sets
Sig_i2n_U_Quart = std(Q_OP2n_All_UHF(:,4:2:10),0,2);
Sig_e2i_U_Quart = std(Q_e2OP_All_UHF(:,4:2:10),0,2);

%Totalling the standard deviations of quartered data-sets
SigQ_U_Quart = sqrt(Sig_i2n_U_Quart.^2+Sig_e2i_U_Quart.^2);

%Determining residuals of quartered data-sets
dQSig_U_Quart = (mean(Q_e2OP_All_UHF(:,4:2:10),2)-
mean(Q_OP2n_All_UHF(:,4:2:10),2))./SigQ_U_Quart;


%Standard deviation for individual heat transfer rates of
%halved data-sets
Sig_i2n_U_Half = std(Q_OP2n_All_UHF(:,12:2:14),0,2);
Sig_e2i_U_Half = std(Q_e2OP_All_UHF(:,12:2:14),0,2);

%Totalling the standard deviations of halved data-sets
SigQ_U_Half = sqrt(Sig_i2n_U_Half.^2+Sig_e2i_U_Half.^2);

%Determining residuals of halved data-sets
dQSig_U_Half = (mean(Q_e2OP_All_UHF(:,12:2:14),2)-
mean(Q_OP2n_All_UHF(:,12:2:14),2))./SigQ_U_Half;

%Totalling all net heat transfers to find the best estimate for weighting
SigEst_U = SigQ_U_Quart/2+SigQ_U_Half/sqrt(2);

%Running the inverse problem again with the best estimated inputs and
%weighting
[Chi2_U,Atm_Final_U,Q_Final_U] =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z_Uan,Ti_Uan,Te_Uan,ne_Uan,PHp_U,SigEst_U,2);
dQSig_Final_U = (Q_Final_U(:,2)-Q_Final_U(:,1))./SigEst_U;

%Displaying all standard deviations and residuals
SigComparison_U = [SigEst_U,Sig_U,SigQ_U_Quart,SigQ_U_Half]
dQSigComparison_U = [dQSig_Final_U,dQSig_U,dQSig_U_Quart,dQSig_U_Half]

%% ------------------------------plotting------------------------------- %%
% "cutting" combined results so clear up graphs
% As combined matricies are vertically concantinated it creates a jarring
```

```
% line between data-sets. To stop this a [] value is put in at the
% "row2cut" using the cutrow function.
%
% cutrow simply extends the entered matrix by 1 row, moving all data at
% row2cut up, and then removing all values in row2cut.
row2cut = 14;
Atm_Start_B = cutrow(Atm_Start_B,row2cut);
Atm_Best_B = cutrow(Atm_Best_B,row2cut);
dQSig_B = cutrow(dQSig_B,row2cut);
Q_Start_B = cutrow(Q_Start_B,row2cut);
Q_Best_B = cutrow(Q_Best_B,row2cut);
Tn_fit_B = cutrow(Tn_fit_B,row2cut);
Te_Ban = cutrow(Te_Ban,row2cut);
Ti_Ban = cutrow(Ti_Ban,row2cut);
Z_Ban = cutrow(Z_Ban,row2cut);

%% Plotting Example Data for Chapter 5 (Analysis methods)

figure
set(gcf,'numbertitle','off','name','NM Fitted Heat data')
subplot(1,2,1)
hold on
plot(Tn_fit_U,Z_Uan/1000,'linewidth',2)
plot(Tn_fit_V,Z_Van/1000,'linewidth',2)
plot(Tn_fit_B,Z_Ban/1000,'linewidth',2)
ylabel('Altitude [km]')
xlabel('Neutral Temperature [K]')
legend('UHF','VHF','Combined')

subplot(1,2,2)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Best_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Best_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Best_V(:,1),Z_Van/1000,'linewidth',2)
semilogx(Q_Best_V(:,2),Z_Van/1000,'--','linewidth',2)
semilogx(Q_Best_B(:,1),Z_Ban/1000,'linewidth',2)
legend('Ion to N (UHF)','e to Ions (UHF)','Ion to N (VHF)','e to Ions (VHF)','Ion to N
(Both)')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')

figure
set(gcf,'numbertitle','off','name','NM Example Atm')
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
ph_AtmD6 = semilogx(Atm_Best_U(:,3:end),Z_Uan/1000,'--','linewidth',2);
ph_AtmD7 = semilogx(Atm_Best_V(:,3:end),Z_Van/1000,'*','linewidth',2);
ph_AtmD5 = semilogx(Atm_Best_B(:,3:end),Z_Ban/1000,'*-','linewidth',2);
cmlines(ph_AtmD5)
cmlines(ph_AtmD6)
cmlines(ph_AtmD7)
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O (UHF)','O2 (UHF)','N2 (UHF)','H (UHF)','He (UHF)',...
       'O (VHF)','O2 (VHF)','N2 (VHF)','H (VHF)','He (VHF)',...
       'O (Both)','O2 (Both)','N2 (Both)','H (Both)','He (Both)')

%% Plotting dQ/Sigmas
%Full
figure

set(gcf,'numbertitle','off','name','NM Error over altitudes Full')
hold on
plot(dQSig_U,Z_Uan/1000,'linewidth',2);
plot(dQSig_V,Z_Van/1000,'linewidth',2);
plot(dQSig_B,Z_Ban/1000,'linewidth',2);
title('Nicolls Parameterisation Residuals for 4-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('UHF','VHF','Combined')
grid on

%1/4s UHF
figure
```

```
set(gcf,'numbertitle','off','name','NM UHF Quart Errors')
hold on
plot(dQSig_U25,Z_U25an/1000,'linewidth',2);
plot(dQSig_U50,Z_U50an/1000,'linewidth',2);
plot(dQSig_U75,Z_U75an/1000,'linewidth',2);
plot(dQSig_U100,Z_U100an/1000,'linewidth',2);
title('UHF (Nicolls Parameterisation) 1-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd','3rd','4th')
grid on

%1/4s VHF
figure

set(gcf,'numbertitle','off','name','NM VHF Quart Errors')
hold on
plot(dQSig_V25,Z_V25an/1000,'linewidth',2);
plot(dQSig_V50,Z_V50an/1000,'linewidth',2);
plot(dQSig_V75,Z_V75an/1000,'linewidth',2);
plot(dQSig_V100,Z_V100an/1000,'linewidth',2);
title('VHF (Nicolls Parameterisation) 1-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd','3rd','4th')
grid on

%1/2s UHF
figure

set(gcf,'numbertitle','off','name','NM UHF Halves Errors')
hold on
plot(dQSig_Uhalf1,Z_Uhalf1an/1000,'linewidth',2);
plot(dQSig_Uhalf2,Z_Uhalf2an/1000,'linewidth',2);
title('UHF (Nicolls Parameterisation) 2-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd')
grid on

%1/2s VHF
figure

set(gcf,'numbertitle','off','name','NM VHF Halves Errors')
hold on
plot(dQSig_Vhalf1,Z_Vhalf1an/1000,'linewidth',2);
plot(dQSig_Vhalf2,Z_Vhalf2an/1000,'linewidth',2);
title('VHF (Nicolls Parameterisation) 2-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd')
grid on



%% Plotting Energy Transfers
figure
subplot(2,1,1)
set(gcf,'numbertitle','off','name','NM UHF Energy Transfer Rates')

set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_U(:,1),Z_Uan/1000,'linewidth',2)           %Q_OP2n
semilogx(Q_Start_U(:,2),Z_Uan/1000,'--','linewidth',2)      %Q_e2OP
semilogx(Q_Best_U(:,1),Z_Uan/1000,'linewidth',2)            %Fitted Q_OP2n
legend('Ion to N','e to Ions','Ion to N (Fit)')
title('UHF (Nicolls Parameterisation)')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

subplot(2,1,2)
set(gcf,'numbertitle','off','name','NM VHF Energy Transfer Rates')
```

```
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_V(:,1),Z_Van/1000,'linewidth',2)           %Q_OP2n
semilogx(Q_Start_V(:,2),Z_Van/1000,'--','linewidth',2)      %Q_e2OP
semilogx(Q_Best_V(:,1),Z_Van/1000,'linewidth',2)
legend('Ion to N','e to Ions','Ion to N (Fit)')
title('VHF (Nicolls Parameterisation)')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

% Plotting Neutral Atmosphere
figure

set(gcf,'numbertitle','off','name','NM UHF Neutral Densities')
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
ph_AtmD3 = semilogx(Atm_Start_U(:,3:end),Z_Uan/1000,'--','linewidth',2);
hold on
ph_AtmD4 = semilogx(Atm_Best_U(:,3:end),Z_Uan/1000,'linewidth',2);
cmlines(ph_AtmD3)
cmlines(ph_AtmD4)
title('UHF (Nicolls Parameterisation)')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O','O2','N2','H','He','O (Fit)','O2 (Fit)','N2 (Fit)','H (Fit)','He (Fit)')
grid on
figure

set(gcf,'numbertitle','off','name','NM VHF Neutral Densities')
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
ph_AtmD1 = semilogx(Atm_Start_V(:,3:end),Z_Van/1000,'--','linewidth',2);
hold on
ph_AtmD2 = semilogx(Atm_Best_V(:,3:end),Z_Van/1000,'linewidth',2);
cmlines(ph_AtmD1)
cmlines(ph_AtmD2)
title('VHF Nicolls Parameterisation')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O','O2','N2','H','He','O (Fit)','O2 (Fit)','N2 (Fit)','H (Fit)','He (Fit)')
grid on

%% Plotting Both Data
figure
set(gcf,'numbertitle','off','name','NM Tn Fitted')
hold on
plot(Tn_All_UHF(:,1),Z_Uan/1000,'-*','linewidth',2)
plot(Tn_fit_U,Z_Uan/1000,'linewidth',2)
plot(Tn_fit_V,Z_Van/1000,'linewidth',2)
plot(Tn_fit_B,Z_Ban/1000,'linewidth',2)
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
title('Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('Neutral Temperature [K]')
legend('Initial','UHF','VHF','Combined','MSIS')
grid on
hold off

figure
set(gcf,'numbertitle','off','name','NM Combined Energy Transfer Rates')
subplot(2,1,1)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Start_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Start_V(:,1),Z_Van/1000,'linewidth',2)
semilogx(Q_Start_V(:,2),Z_Van/1000,'--','linewidth',2)
title('Raw parameters')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off
```

```
subplot(2,1,2)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Best_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Best_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Best_V(:,1),Z_Van/1000,'linewidth',2)
semilogx(Q_Best_V(:,2),Z_Van/1000,'--','linewidth',2)
semilogx(Q_Best_B(:,1),Z_Ban/1000,'linewidth',2)
legend('Ion to N (UHF)','e to Ions (UHF)','Ion to N (VHF)','e to Ions (VHF)','Ion to N
(Both)')
title('Fitted Parameters')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

figure
set(gcf,'numbertitle','off','name','NM All Fitted Neutral Densities')

set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
ph_AtmD6 = semilogx(Atm_Best_U(:,3:end),Z_Uan/1000,'--','linewidth',2);
ph_AtmD7 = semilogx(Atm_Best_V(:,3:end),Z_Van/1000,'*','linewidth',2);
ph_AtmD5 = semilogx(Atm_Best_B(:,3:end),Z_Ban/1000,'*-','linewidth',2);
cmlines(ph_AtmD5)
cmlines(ph_AtmD6)
cmlines(ph_AtmD7)
title('Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O (UHF)','O2 (UHF)','N2 (UHF)','H (UHF)','He (UHF)',...
       'O (VHF)','O2 (VHF)','N2 (VHF)','H (VHF)','He (VHF)',...
       'O (Both)','O2 (Both)','N2 (Both)','H (Both)','He (Both)')
grid on

%% Total Collected (Fitted)
figure
set(gcf,'numbertitle','off','name','NM Fitted O Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO_U = semilogx(ODen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phO_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('O Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on
subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO_V = semilogx(ODen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phO_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('O Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','NM Fitted O2 Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO2_U = semilogx(O2Den_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phO2_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
```

```
xlabel('O2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO2_V = semilogx(O2Den_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phO2_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('O2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','NM Fitted N2 Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phN2_U = semilogx(N2Den_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phN2_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('N2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phN2_V = semilogx(N2Den_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phN2_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('N2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','NM Fitted H Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phH_U = semilogx(HDen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phH_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('H Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phH_V = semilogx(HDen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phH_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('H Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
```

```
set(gcf,'numbertitle','off','name','NM Fitted He Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phHe_U = semilogx(HeDen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phHe_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('He [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phHe_V = semilogx(HeDen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phHe_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('He Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','NM Fitted Tn')

subplot(2,1,1)
set(gca,'fontsize',Fnt_Size)
hold on
plot(Tn_All_UHF(:,1),Z_Uan/1000,'-*','linewidth',2);
phTn_U = plot(Tn_All_UHF(:,2:end),Z_Uan/1000,'linewidth',2);
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
cmlines(phTn_U)
title('UHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'fontsize',Fnt_Size)
hold on
plot(Tn_All_UHF(:,1),Z_Uan/1000,'-*','linewidth',2);
phTn_V = plot(Tn_All_VHF(:,2:end),Z_Van/1000,'linewidth',2);
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
cmlines(phTn_V)
title('VHF Nicholls Parameterisation')
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Intital','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on
```

9.2 ThesisRunMSIS

```
%% Will Stock Masters Thesis Script to Run MATLAB Parameterisation
%This script runs the least squares analysis contained within
%ThesisAnalysis and the functions
%
%Code Work flow
%1 Setup
%
%2 Analyse raw EISCAT radar data to get the full altitude measured results
%
%3 Use full altitude data to determine wind velocites and trajectories
%
%4 Run ThesisAnalysis (Best outputs for both) for each radar and time frame
%(VHF, UHF, Both, and full, 4 quarters and 2 halves)
%
%5 Collect all output data from the ThesisAnalysis functions into groups
%e.g 'all VHF fitted neutral temperatures' etc
%
%6 Run analysis on these collected figures, determining the stadnard
%deviations and residuals
%
%7 Plot diagram graphs for "Heat transfer at high altitudes" chapter
%
%8 Plot measured data for "Experiment observations" chapter
%
%9 Plot Heat Transfer Rates for all full time frame averaged data,
%  Plot dQ/Sigma, neutral atmosphere and temperatures for all data-sets

%% Setup
clc
clear
Root_Path = pwd;
Save_Path = strcat(Root_Path,'\Figures');

addpath(strcat(Root_Path,'\Matlab Functions'));
addpath(strcat(Root_Path,'\Matlab Functions\FMINSEARCHBND'));
addpath(strcat(Root_Path,'\Matlab Functions\T_e_ion'));
addpath(strcat(Root_Path,'\Matlab Functions\T_ion_neu'));
addpath(strcat(Root_Path,'\Matlab Functions\EARTH'));
addpath(strcat(Root_Path,'\Matlab Functions\cm_and_cb_utilities'));
addpath(strcat(Root_Path,'\Meta'));

Fnt_Size = 12;                              %Figure axis font size
Lat_Ram = 69.64;                            %Latitude of Ramsfjord Site [Deg]
Lon_Ram = 18.85;                            %Longitude of Ramsfjord Site [Deg]
Exp_yr = 2017;                              %Experiment year
Exp_d = 320;                                %Experiment Day of year
Exp_t = 36000;                              %Experiment Time [S]
h_lim = [300,650]*1e3;                      %Height restraints [km]
Sigma = 1;                                  %Placeholder for Sigma Value
MSISorNICO = 'MSIS';
Exp_Data = {Lat_Ram,Lon_Ram,Exp_yr,...      %Experiment Data group to input
            Exp_d,Exp_t,h_lim};

Data_Path_V = strcat(Root_Path,'\Data\VHF\2016-11-15_beata_60@vhf');
Data_Path_U = strcat(Root_Path,'\Data\UHF\2016-11-15_beata_60@uhfa');
Data_Path_K = strcat(Root_Path,'\Data\KIR\2016-11-15_beata_60@kir');
Data_Path_S = strcat(Root_Path,'\Data\SOD\2016-11-15_beata_60@sod');
MSIS_Path = strcat(Root_Path,'\Meta\MSIS Neutral Profiles 15-11-16 full.txt');
IRI_Path = strcat(Root_Path,'\Meta\IRI Profile 15-11-16 with index.txt');
IRI = load(IRI_Path);
MSIS = load(MSIS_Path);

%Cutting reference data into same size matricies as radar data
MSIS_an = MSIS(h_lim(1)/1000<MSIS(:,1)&MSIS(:,1)<h_lim(2)/1000,:);
IRI_an = IRI(h_lim(1)/1000<IRI(:,1)&IRI(:,1)<h_lim(2)/1000,:);
PHp = sum([IRI(:,8:11),IRI(:,13)],2)/100;

%% Remote sites
%As the wind data is not fitted at all, it does not need Thesis analysis
%and just needs converting from the Matlab files from the EISCAT system into readable data

%Kiruna data
[h_K,t_K,ne_K,Te_K,Ti_K,vi_K,dne_K,dTe_K,dTi_K,dvi_K,az_K,el_K,T_K,ranges_K] =
readGdata2(Data_Path_K);
```

```
%Sod data
[h_S,t_S,ne_S,Te_S,Ti_S,vi_S,dne_S,dTe_S,dTi_S,dvi_S,az_S,el_S,T_S,ranges_S] =
readGdata2(Data_Path_S);

%Outputs are in cell format, converting to matricies
h_K = cell2mat(h_K);
ne_K = cell2mat(ne_K);
Te_K = cell2mat(Te_K);
Ti_K = cell2mat(Ti_K);
vi_K = cell2mat(vi_K);
dvi_K = cell2mat(dvi_K);

h_S = cell2mat(h_S);
ne_S = cell2mat(ne_S);
Te_S = cell2mat(Te_S);
Ti_S = cell2mat(Ti_S);
vi_S = cell2mat(vi_S);
dvi_S = cell2mat(dvi_S);

%% ----------------------Determining wind velocity---------------------- %%
%Getting raw data for VHF and UHF so they have as much data as the remote
%sites
[h_V,t_V,ne_V,Te_V,Ti_V,vi_V,dne_V,dTe_V,dTi_V,dvi_V,az_V,el_V,T_V,ranges_V] =
readGdata(Data_Path_V); %Getting raw VHF data
[h_U,t_U,ne_U,Te_U,Ti_U,vi_U,dne_U,dTe_U,dTi_U,dvi_U,az_U,el_U,T_U,ranges_U] =
readGdata(Data_Path_U); %Getting raw VHF data

h_VW = h_V*1e3; %converting

%Calling the function wind_tr to get the ion velocities and trajectories
[v_ion_V,tr_wind_V] =
wind_tr(vi_V,vi_S,vi_S,mean(az_V),mean(el_V),mean(az_S),mean(el_S),mean(az_S),mean(el_S),dvi_V
,dvi_S,dvi_S);
[v_ion_U,tr_wind_U] =
wind_tr(vi_U,vi_S,vi_S,mean(az_U),mean(el_U),mean(az_S),mean(el_S),mean(az_S),mean(el_S),dvi_U
,dvi_S,dvi_S);

%% VHF
%This section calls the Thesis Analysis function for all VHF data sets that
%outputs a large matrix of cells. This matrix is then unpacked by calling
%the AnalysisUnpacking function to give the final outputs that are plotted.

%Full time frame data set
[Atms_Fit_V,Atms_Raw_V,Best_Pars_V,dQSig_V,dTs_V,ne_Van,Qs_Fit_V,Ts_V,Z_Van,Sig_V,F10_7D,F10_7
,Ap,Anal_Setup] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
full',2);
%Unpacking variables
[Atm_Best_V,Atm_Start_V,Atm_Vi,Atm_VM,dTi_Van,dTe_Van,Q_Best_V,Q_Start_V,Ti_Van,Te_Van,Tn_fit_
V] = AnalysisUnpacking(Atms_Fit_V,Atms_Raw_V,dTs_V,Qs_Fit_V,Ts_V);
PHp_V = Atm_Vi(:,end);

% Getting VHF data for the time frame broken into 4 parts
% e.g: Each set of data is only averaged from 1/4 of the time period ending
% at "_[%]" of time.

[Atms_Fit_V25,Atms_Raw_V25,Best_Pars_V25,dQSig_V25,dTs_V25,ne_V25an,Qs_Fit_V25,Ts_V25,Z_V25an,
Sig_V25] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:52');
%Unpacking variables
[Atm_Best_V25,Atm_Start_V25,Atm_V25i,Atm_V25M,dTi_V25an,dTe_V25an,Q_Best_V25,Q_Start_V25,Ti_V2
5an,Te_V25an,Tn_fit_V25] =
AnalysisUnpacking(Atms_Fit_V25,Atms_Raw_V25,dTs_V25,Qs_Fit_V25,Ts_V25);

[Atms_Fit_V50,Atms_Raw_V50,Best_Pars_V50,dQSig_V50,dTs_V50,ne_V50an,Qs_Fit_V50,Ts_V50,Z_V50an,
Sig_V50] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 53:104');
%Unpacking variables
[Atm_Best_V50,Atm_Start_V50,Atm_V50i,Atm_V50M,dTi_V50an,dTe_V50an,Q_Best_V50,Q_Start_V50,Ti_V5
0an,Te_V50an,Tn_fit_V50] =
AnalysisUnpacking(Atms_Fit_V50,Atms_Raw_V50,dTs_V50,Qs_Fit_V50,Ts_V50);

[Atms_Fit_V75,Atms_Raw_V75,Best_Pars_V75,dQSig_V75,dTs_V75,ne_V75an,Qs_Fit_V75,Ts_V75,Z_V75an,
Sig_V75] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:156');
%Unpacking variables
[Atm_Best_V75,Atm_Start_V75,Atm_V75i,Atm_V75M,dTi_V75an,dTe_V75an,Q_Best_V75,Q_Start_V75,Ti_V7
5an,Te_V75an,Tn_fit_V75] =
AnalysisUnpacking(Atms_Fit_V75,Atms_Raw_V75,dTs_V75,Qs_Fit_V75,Ts_V75);
```

```
[Atms_Fit_V100,Atms_Raw_V100,Best_Pars_V100,dQSig_V100,dTs_V100,ne_V100an,Qs_Fit_V100,Ts_V100,
Z_V100an,Sig_V100] = ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
157:207');
%Unpacking variables
[Atm_Best_V100,Atm_Start_V100,Atm_V100i,Atm_V100M,dTi_V100an,dTe_V100an,Q_Best_V100,Q_Start_V1
00,Ti_V100an,Te_V100an,Tn_fit_V100] =
AnalysisUnpacking(Atms_Fit_V100,Atms_Raw_V100,dTs_V100,Qs_Fit_V100,Ts_V100);

% Getting VHF data for the time frame broken into 2 parts
% e.g: Each set of data is only averaged from 1/2 of the time period with
% markings of "_half1"/"_half2" for the lower/upper halves respectively.
[Atms_Fit_Vhalf1,Atms_Raw_Vhalf1,Best_Pars_Vhalf1,dQSig_Vhalf1,dTs_Vhalf1,ne_Vhalf1an,Qs_Fit_V
half1,Ts_Vhalf1,Z_Vhalf1an,Sig_Vhalf1] =
ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:104');
%Unpacking variables
[Atm_Best_Vhalf1,Atm_Start_Vhalf1,Atm_Vhalf1i,Atm_Vhalf1M,dTi_Vhalf1an,dTe_Vhalf1an,Q_Best_Vha
lf1,Q_Start_Vhalf1,Ti_Vhalf1an,Te_Vhalf1an,Tn_fit_Vhalf1] =
AnalysisUnpacking(Atms_Fit_Vhalf1,Atms_Raw_Vhalf1,dTs_Vhalf1,Qs_Fit_Vhalf1,Ts_Vhalf1);

[Atms_Fit_Vhalf2,Atms_Raw_Vhalf2,Best_Pars_Vhalf2,dQSig_Vhalf2,dTs_Vhalf2,ne_Vhalf2an,Qs_Fit_V
half2,Ts_Vhalf2,Z_Vhalf2an,Sig_Vhalf2] =
ThesisAnalysis(Data_Path_V,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:207');
%Unpacking variables
[Atm_Best_Vhalf2,Atm_Start_Vhalf2,Atm_Vhalf2i,Atm_Vhalf2M,dTi_Vhalf2an,dTe_Vhalf2an,Q_Best_Vha
lf2,Q_Start_Vhalf2,Ti_Vhalf2an,Te_Vhalf2an,Tn_fit_Vhalf2] =
AnalysisUnpacking(Atms_Fit_Vhalf2,Atms_Raw_Vhalf2,dTs_Vhalf2,Qs_Fit_Vhalf2,Ts_Vhalf2);

%% UHF
%This section calls the Thesis Analysis function for all UHF data sets that
%outputs a large matrix of cells. This matrix is then unpacked by calling
%the AnalysisUnpacking function to give the final outputs that are plotted.

%Full time frame data set
[Atms_Fit_U,Atms_Raw_U,Best_Pars_U,dQSig_U,dTs_U,ne_Uan,Qs_Fit_U,Ts_U,Z_Uan,Sig_U,F10_7D,F10_7
,Ap,Anal_Setup] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
full',2);
%Unpacking variables
[Atm_Best_U,Atm_Start_U,Atm_Ui,Atm_UM,dTi_Uan,dTe_Uan,Q_Best_U,Q_Start_U,Ti_Uan,Te_Uan,Tn_fit_
U] = AnalysisUnpacking(Atms_Fit_U,Atms_Raw_U,dTs_U,Qs_Fit_U,Ts_U);
PHp_U = Atm_Ui(:,end);

% Getting VHF data for the time frame broken into 4 parts
% e.g: Each set of data is only averaged from 1/4 of the time period ending
% at "_[%]" of time.

[Atms_Fit_U25,Atms_Raw_U25,Best_Pars_U25,dQSig_U25,dTs_U25,ne_U25an,Qs_Fit_U25,Ts_U25,Z_U25an,
Sig_U25] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:52');
%Unpacking variables
[Atm_Best_U25,Atm_Start_U25,Atm_U25i,Atm_U25M,dTi_U25an,dTe_U25an,Q_Best_U25,Q_Start_U25,Ti_U2
5an,Te_U25an,Tn_fit_U25] =
AnalysisUnpacking(Atms_Fit_U25,Atms_Raw_U25,dTs_U25,Qs_Fit_U25,Ts_U25);

[Atms_Fit_U50,Atms_Raw_U50,Best_Pars_U50,dQSig_U50,dTs_U50,ne_U50an,Qs_Fit_U50,Ts_U50,Z_U50an,
Sig_U50] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 53:104');
%Unpacking variables
[Atm_Best_U50,Atm_Start_U50,Atm_U50i,Atm_U50M,dTi_U50an,dTe_U50an,Q_Best_U50,Q_Start_U50,Ti_U5
0an,Te_U50an,Tn_fit_U50] =
AnalysisUnpacking(Atms_Fit_U50,Atms_Raw_U50,dTs_U50,Qs_Fit_U50,Ts_U50);

[Atms_Fit_U75,Atms_Raw_U75,Best_Pars_U75,dQSig_U75,dTs_U75,ne_U75an,Qs_Fit_U75,Ts_U75,Z_U75an,
Sig_U75] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:156');
%Unpacking variables
[Atm_Best_U75,Atm_Start_U75,Atm_U75i,Atm_U75M,dTi_U75an,dTe_U75an,Q_Best_U75,Q_Start_U75,Ti_U7
5an,Te_U75an,Tn_fit_U75] =
AnalysisUnpacking(Atms_Fit_U75,Atms_Raw_U75,dTs_U75,Qs_Fit_U75,Ts_U75);

[Atms_Fit_U100,Atms_Raw_U100,Best_Pars_U100,dQSig_U100,dTs_U100,ne_U100an,Qs_Fit_U100,Ts_U100,
Z_U100an,Sig_U100] = ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,'
157:207');
%Unpacking variables
[Atm_Best_U100,Atm_Start_U100,Atm_U100i,Atm_U100M,dTi_U100an,dTe_U100an,Q_Best_U100,Q_Start_U1
00,Ti_U100an,Te_U100an,Tn_fit_U100] =
AnalysisUnpacking(Atms_Fit_U100,Atms_Raw_U100,dTs_U100,Qs_Fit_U100,Ts_U100);

% Getting UHF data for the time frame broken into 2 parts
% e.g: Each set of data is only averaged from 1/2 of the time period with
% markings of "_half1"/"_half2" for the lower/upper halves respectively.
```

```
[Atms_Fit_Uhalf1,Atms_Raw_Uhalf1,Best_Pars_Uhalf1,dQSig_Uhalf1,dTs_Uhalf1,ne_Uhalf1an,Qs_Fit_U
half1,Ts_Uhalf1,Z_Uhalf1an,Sig_Uhalf1] =
ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 1:104');
%Unpacking variables
[Atm_Best_Uhalf1,Atm_Start_Uhalf1,Atm_Uhalf1i,Atm_Uhalf1M,dTi_Uhalf1an,dTe_Uhalf1an,Q_Best_Uha
lf1,Q_Start_Uhalf1,Ti_Uhalf1an,Te_Uhalf1an,Tn_fit_Uhalf1] =
AnalysisUnpacking(Atms_Fit_Uhalf1,Atms_Raw_Uhalf1,dTs_Uhalf1,Qs_Fit_Uhalf1,Ts_Uhalf1);

[Atms_Fit_Uhalf2,Atms_Raw_Uhalf2,Best_Pars_Uhalf2,dQSig_Uhalf2,dTs_Uhalf2,ne_Uhalf2an,Qs_Fit_U
half2,Ts_Uhalf2,Z_Uhalf2an,Sig_Uhalf2] =
ThesisAnalysis(Data_Path_U,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,' 105:207');
%Unpacking variables
[Atm_Best_Uhalf2,Atm_Start_Uhalf2,Atm_Uhalf2i,Atm_Uhalf2M,dTi_Uhalf2an,dTe_Uhalf2an,Q_Best_Uha
lf2,Q_Start_Uhalf2,Ti_Uhalf2an,Te_Uhalf2an,Tn_fit_Uhalf2] =
AnalysisUnpacking(Atms_Fit_Uhalf2,Atms_Raw_Uhalf2,dTs_Uhalf2,Qs_Fit_Uhalf2,Ts_Uhalf2);


%% Creating matricies for "Both"
%As the combined data-set has no data path, it does not need several steps
%from Thesis Analysis, therefore it needs concantination and then being
%input directly to the least square analysis function "BestOutputs"

Z_Ban = [Z_Van;Z_Uan];
Te_Ban = [Te_Van;Te_Uan];
Ti_Ban = [Ti_Van;Ti_Uan];
ne_Ban = [ne_Van;ne_Uan];
dTi_Ban = [dTi_Van;dTi_Uan];
dTe_Ban = [dTe_Van;dTe_Uan];
Atm_Bi = [Atm_Vi;Atm_Ui];
PHp_B = [PHp_V;PHp_U];


%% Least Squares Analysis for "Both"
%Inputs for analysis of error function
p_index = Anal_Setup{1};
fixed_pars = Anal_Setup{2};
start_guess = Anal_Setup{3};
LB = Anal_Setup{4};
UB = Anal_Setup{5};
options = Anal_Setup{6};
Z0 = 100;
Tinf = TExosphere(mean(F10_7D));
ORefDensity = Atm_VM(1:1);


err_ins_B =
{Z_Ban,Ti_Ban,Te_Ban,dTi_Ban,dTe_Ban,ne_Ban,PHp_B,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,MSISorNIC
O,Z0,Tinf,ORefDensity};

err_fcn_tag_B =
@(pars)err_fcn2(pars,p_index,fixed_pars,Z_Ban,Ti_Ban,Te_Ban,ne_Ban,PHp_B,Lat_Ram,Lon_Ram,Exp_y
r,Exp_d,Exp_t,Sigma);

[Best_Pars_B,Atm_Best_B,Q_Best_B,Atm_Start_B,Q_Start_B,dQSig_B] =
BestOutputs(err_fcn_tag_B,p_index,fixed_pars,start_guess,LB,UB,options,err_ins_B);

Tn_fit_B = Atm_Best_B(:,2);

%% Showing all best pars
disp('Lower Bounds=')
disp(LB(1,:))
Best_Pars_V
Best_Pars_V25
Best_Pars_V50
Best_Pars_V75
Best_Pars_V100
Best_Pars_Vhalf1
Best_Pars_Uhalf2
Best_Pars_U
Best_Pars_U25
Best_Pars_U50
Best_Pars_U75
Best_Pars_U100
Best_Pars_Uhalf1
Best_Pars_Uhalf2
Best_Pars_B
disp('Upper Bounds=')
```

```
disp(UB(1,:))

%% -------------------------Collecting Variables------------------------ %%
%% VHF
ne_All_VHF = [ne_Van,ne_V25an,ne_V50an,ne_V75an,ne_V100an...
              ,ne_Vhalf1an,ne_Vhalf2an];

ODen_All_VHF = [Atm_Start_V(:,3),Atm_Best_V(:,3)...
                ,Atm_Best_V25(:,3)...
                ,Atm_Best_V50(:,3)...
                ,Atm_Best_V75(:,3)...
                ,Atm_Best_V100(:,3)...
                ,Atm_Best_Vhalf1(:,3)...
                ,Atm_Best_Vhalf2(:,3)...
                ,Atm_VM(:,1)];

O2Den_All_VHF = [Atm_Start_V(:,4),Atm_Best_V(:,4)...
                 ,Atm_Best_V25(:,4)...
                 ,Atm_Best_V50(:,4)...
                 ,Atm_Best_V75(:,4)...
                 ,Atm_Best_V100(:,4)...
                 ,Atm_Best_Vhalf1(:,4)...
                 ,Atm_Best_Vhalf2(:,4)...
                 ,Atm_VM(:,2)];

N2Den_All_VHF = [Atm_Start_V(:,5),Atm_Best_V(:,5)...
                 ,Atm_Best_V25(:,5)...
                 ,Atm_Best_V50(:,5)...
                 ,Atm_Best_V75(:,5)...
                 ,Atm_Best_V100(:,5)...
                 ,Atm_Best_Vhalf1(:,5)...
                 ,Atm_Best_Vhalf2(:,5)...
                 ,Atm_VM(:,3)];

HDen_All_VHF = [Atm_Start_V(:,6),Atm_Best_V(:,6)...
                ,Atm_Best_V25(:,6)...
                ,Atm_Best_V50(:,6)...
                ,Atm_Best_V75(:,6)...
                ,Atm_Best_V100(:,6)...
                ,Atm_Best_Vhalf1(:,6)...
                ,Atm_Best_Vhalf2(:,6)...
                ,Atm_VM(:,4)];

HeDen_All_VHF = [Atm_Start_V(:,7),Atm_Best_V(:,7)...
                 ,Atm_Best_V25(:,7)...
                 ,Atm_Best_V50(:,7)...
                 ,Atm_Best_V75(:,7)...
                 ,Atm_Best_V100(:,7)...
                 ,Atm_Best_Vhalf1(:,7)...
                 ,Atm_Best_Vhalf2(:,7)...
                 ,Atm_VM(:,5)];

Tn_All_VHF = [Atm_Start_V(:,2),Atm_Best_V(:,2)...
              ,Atm_Best_V25(:,2)...
              ,Atm_Best_V50(:,2)...
              ,Atm_Best_V75(:,2)...
              ,Atm_Best_V100(:,2)...
              ,Atm_Best_Vhalf1(:,2)...
              ,Atm_Best_Vhalf2(:,2)];

Q_OP2n_All_VHF = [Q_Start_V(:,1),Q_Best_V(:,1)...
                  ,Q_Start_V25(:,1),Q_Best_V25(:,1)...
                  ,Q_Start_V50(:,1),Q_Best_V50(:,1)...
                  ,Q_Start_V75(:,1),Q_Best_V75(:,1)...
                  ,Q_Start_V100(:,1),Q_Best_V100(:,1)...
                  ,Q_Start_Vhalf1(:,1),Q_Best_Vhalf1(:,1)...
                  ,Q_Start_Vhalf2(:,1),Q_Best_Vhalf2(:,1)];

Q_e2OP_All_VHF = [Q_Start_V(:,2),Q_Best_V(:,2)...
                  ,Q_Start_V25(:,2),Q_Best_V25(:,2)...
                  ,Q_Start_V50(:,2),Q_Best_V50(:,2)...
                  ,Q_Start_V75(:,2),Q_Best_V75(:,2)...
                  ,Q_Start_V100(:,2),Q_Best_V100(:,2)...
                  ,Q_Start_Vhalf1(:,2),Q_Best_Vhalf1(:,2)...
                  ,Q_Start_Vhalf2(:,2),Q_Best_Vhalf2(:,2)];
```

```
%% UHF
ne_All_UHF = [ne_Uan,ne_U25an,ne_U50an,ne_U75an,ne_U100an...
             ,ne_Uhalf1an,ne_Uhalf2an];

ODen_All_UHF = [Atm_Start_U(:,3),Atm_Best_U(:,3)...
               ,Atm_Best_U25(:,3)...
               ,Atm_Best_U50(:,3)...
               ,Atm_Best_U75(:,3)...
               ,Atm_Best_U100(:,3)...
               ,Atm_Best_Uhalf1(:,3)...
               ,Atm_Best_Uhalf2(:,3)...
               ,Atm_UM(:,1)];

O2Den_All_UHF = [Atm_Start_U(:,4),Atm_Best_U(:,4)...
                ,Atm_Best_U25(:,4)...
                ,Atm_Best_U50(:,4)...
                ,Atm_Best_U75(:,4)...
                ,Atm_Best_U100(:,4)...
                ,Atm_Best_Uhalf1(:,4)...
                ,Atm_Best_Uhalf2(:,4)...
                ,Atm_UM(:,2)];

N2Den_All_UHF = [Atm_Start_U(:,5),Atm_Best_U(:,5)...
                ,Atm_Best_U25(:,5)...
                ,Atm_Best_U50(:,5)...
                ,Atm_Best_U75(:,5)...
                ,Atm_Best_U100(:,5)...
                ,Atm_Best_Uhalf1(:,5)...
                ,Atm_Best_Uhalf2(:,5)...
                ,Atm_UM(:,3)];

HDen_All_UHF = [Atm_Start_U(:,6),Atm_Best_U(:,6)...
               ,Atm_Best_U25(:,6)...
               ,Atm_Best_U50(:,6)...
               ,Atm_Best_U75(:,6)...
               ,Atm_Best_U100(:,6)...
               ,Atm_Best_Uhalf1(:,6)...
               ,Atm_Best_Uhalf2(:,6)...
               ,Atm_UM(:,4)];

HeDen_All_UHF = [Atm_Start_U(:,7),Atm_Best_U(:,7)...
                ,Atm_Best_U25(:,7)...
                ,Atm_Best_U50(:,7)...
                ,Atm_Best_U75(:,7)...
                ,Atm_Best_U100(:,7)...
                ,Atm_Best_Uhalf1(:,7)...
                ,Atm_Best_Uhalf2(:,7)...
                ,Atm_UM(:,5)];

Tn_All_UHF = [Atm_Start_U(:,2),Atm_Best_U(:,2)...
             ,Atm_Best_U25(:,2)...
             ,Atm_Best_U50(:,2)...
             ,Atm_Best_U75(:,2)...
             ,Atm_Best_U100(:,2)...
             ,Atm_Best_Uhalf1(:,2)...
             ,Atm_Best_Uhalf2(:,2)];

Q_OP2n_All_UHF = [Q_Start_U(:,1),Q_Best_U(:,1)...
                 ,Q_Start_U25(:,1),Q_Best_U25(:,1)...
                 ,Q_Start_U50(:,1),Q_Best_U50(:,1)...
                 ,Q_Start_U75(:,1),Q_Best_U75(:,1)...
                 ,Q_Start_U100(:,1),Q_Best_U100(:,1)...
                 ,Q_Start_Uhalf1(:,1),Q_Best_Uhalf1(:,1)...
                 ,Q_Start_Uhalf2(:,1),Q_Best_Uhalf2(:,1)];

Q_e2OP_All_UHF = [Q_Start_U(:,2),Q_Best_U(:,2)...
                 ,Q_Start_U25(:,2),Q_Best_U25(:,2)...
                 ,Q_Start_U50(:,2),Q_Best_U50(:,2)...
                 ,Q_Start_U75(:,2),Q_Best_U75(:,2)...
                 ,Q_Start_U100(:,2),Q_Best_U100(:,2)...
                 ,Q_Start_Uhalf1(:,2),Q_Best_Uhalf1(:,2)...
                 ,Q_Start_Uhalf2(:,2),Q_Best_Uhalf2(:,2)];

%Final Parameters
Best_Pars =
[Best_Pars_V;Best_Pars_V25;Best_Pars_V50;Best_Pars_V75;Best_Pars_V100;Best_Pars_Vhalf1;Best_Pa
rs_Vhalf2;...
```

```
Best_Pars_U;Best_Pars_U25;Best_Pars_U50;Best_Pars_U75;Best_Pars_U100;Best_Pars_Uhalf1;Best_Par
s_Uhalf2;...
             Best_Pars_B];

Best_Atms_V =
[Atm_Best_V,Atm_Best_V25,Atm_Best_V50,Atm_Best_V75,Atm_Best_V100,Atm_Best_Vhalf1,Atm_Best_Vhal
f2];
Best_Atms_U =
[Atm_Best_U,Atm_Best_U25,Atm_Best_U50,Atm_Best_U75,Atm_Best_U100,Atm_Best_Uhalf1,Atm_Best_Uhal
f2];
Best_Atms_B = [Atm_Best_B];

%% -------------------------Analysis of Results------------------------- %%
%Determining average of the best parameters to see what are the best parameters for anytime of
day
Avg_Best_Pars = mean(Best_Pars,1);

%finding residual of Best Parameter investigation
%This is applicable to both radar
BPSig = Best_Pars./Avg_Best_Pars;

%Finding the standard deviations of the fitted neutrals, excluding initial
%and MSIS values from XDen_All_YHF matrix
OSig_V = std(ODen_All_VHF(:,2:2:end-1),0,2);
O2Sig_V = std(O2Den_All_VHF(:,2:2:end-1),0,2);
N2Sig_V = std(N2Den_All_VHF(:,2:2:end-1),0,2);
HSig_V = std(HDen_All_VHF(:,2:2:end-1),0,2);
HeSig_V = std(HeDen_All_VHF(:,2:2:end-1),0,2);

%Standard deviation for individual heat transfer rates of
%quartered data-sets
Sig_i2n_V_Quart = std(Q_OP2n_All_VHF(:,4:2:10),0,2);
Sig_e2i_V_Quart = std(Q_e2OP_All_VHF(:,4:2:10),0,2);

%Totalling the standard deviations of quartered data-sets
SigQ_V_Quart = sqrt(Sig_i2n_V_Quart.^2+Sig_e2i_V_Quart.^2);

%Determining residuals of quartered data-sets
dQSig_V_Quart = (mean(Q_e2OP_All_VHF(:,4:2:10),2)-
mean(Q_OP2n_All_VHF(:,4:2:10),2))./SigQ_V_Quart;

%Standard deviation for individual heat transfer rates of
%halved data-sets
Sig_i2n_V_Half = std(Q_OP2n_All_VHF(:,12:2:14),0,2);
Sig_e2i_V_Half = std(Q_e2OP_All_VHF(:,12:2:14),0,2);

%Totalling the standard deviations of halved data-sets
SigQ_V_Half = sqrt(Sig_i2n_V_Half.^2+Sig_e2i_V_Half.^2);

%Determining residuals of halved data-sets
dQSig_V_Half = (mean(Q_e2OP_All_VHF(:,12:2:14),2)-
mean(Q_OP2n_All_VHF(:,12:2:14),2))./SigQ_V_Half;

%Totalling all net heat transfers to find the best estimate for weighting
SigEst_V = SigQ_V_Quart/2+SigQ_V_Half/sqrt(2);

%Running the inverse problem again with the best estimated inputs and
%weighting
[Chi2_V,Atm_Final_V,Q_Final_V] =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z_Van,Ti_Van,Te_Van,ne_Van,PHp_V,SigEst_V,2);
dQSig_Final_V = (Q_Final_V(:,2)-Q_Final_V(:,1))./SigEst_V;

%Displaying all standard deviations and residuals
SigComparison_V = [SigEst_V,Sig_V,SigQ_V_Quart,SigQ_V_Half]
dQSigComparison_V = [dQSig_Final_V,dQSig_V,dQSig_V_Quart,dQSig_V_Half]

%% UHF
%Determining average of the best parameters to see what are the best parameters for anytime of
day
[SigO_U,SigO2_U,SigN2_U,SigH_U,SigHe_U] = Atm_Std(Best_Atms_U);

%Finding the standard deviations of the fitted neutrals, excluding initial
%and MSIS values from XDen_All_YHF matrix
OSig_U = std(ODen_All_UHF(:,2:2:end-1),0,2);
O2Sig_U = std(O2Den_All_UHF(:,2:2:end-1),0,2);
N2Sig_U = std(N2Den_All_UHF(:,2:2:end-1),0,2);
```

```
HSig_U = std(HDen_All_UHF(:,2:2:end-1),0,2);
HeSig_U = std(HeDen_All_UHF(:,2:2:end-1),0,2);

%Standard deviation for individual heat transfer rates of
%quartered data-sets
Sig_i2n_U_Quart = std(Q_OP2n_All_UHF(:,4:2:10),0,2);
Sig_e2i_U_Quart = std(Q_e2OP_All_UHF(:,4:2:10),0,2);

%Totalling the standard deviations of quartered data-sets
SigQ_U_Quart = sqrt(Sig_i2n_U_Quart.^2+Sig_e2i_U_Quart.^2);

%Determining residuals of quartered data-sets
dQSig_U_Quart = (mean(Q_e2OP_All_UHF(:,4:2:10),2)-
mean(Q_OP2n_All_UHF(:,4:2:10),2))./SigQ_U_Quart;


%Standard deviation for individual heat transfer rates of
%halved data-sets
Sig_i2n_U_Half = std(Q_OP2n_All_UHF(:,12:2:14),0,2);
Sig_e2i_U_Half = std(Q_e2OP_All_UHF(:,12:2:14),0,2);

%Totalling the standard deviations of halved data-sets
SigQ_U_Half = sqrt(Sig_i2n_U_Half.^2+Sig_e2i_U_Half.^2);

%Determining residuals of halved data-sets
dQSig_U_Half = (mean(Q_e2OP_All_UHF(:,12:2:14),2)-
mean(Q_OP2n_All_UHF(:,12:2:14),2))./SigQ_U_Half;

%Totalling all net heat transfers to find the best estimate for weighting
SigEst_U = SigQ_U_Quart/2+SigQ_U_Half/sqrt(2);

%Running the inverse problem again with the best estimated inputs and
%weighting
[Chi2_U,Atm_Final_U,Q_Final_U] =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z_Uan,Ti_Uan,Te_Uan,ne_Uan,PHp_U,SigEst_U,2);
dQSig_Final_U = (Q_Final_U(:,2)-Q_Final_U(:,1))./SigEst_U;

%Displaying all standard deviations and residuals
SigComparison_U = [SigEst_U,Sig_U,SigQ_U_Quart,SigQ_U_Half]
dQSigComparison_U = [dQSig_Final_U,dQSig_U,dQSig_U_Quart,dQSig_U_Half]

%% ------------------------------plotting------------------------------- %%
% "cutting" combined results so clear up graphs
% As combined matricies are vertically concantinated it creates a jarring
% line between data-sets. To stop this a [] value is put in at the
% "row2cut" using the cutrow function.
%
% cutrow simply extends the entered matrix by 1 row, moving all data at
% row2cut up, and then removing all values in row2cut.
row2cut = 14;
Atm_Start_B = cutrow(Atm_Start_B,row2cut);
Atm_Best_B = cutrow(Atm_Best_B,row2cut);
dQSig_B = cutrow(dQSig_B,row2cut);
Q_Start_B = cutrow(Q_Start_B,row2cut);
Q_Best_B = cutrow(Q_Best_B,row2cut);
Tn_fit_B = cutrow(Tn_fit_B,row2cut);
Te_Ban = cutrow(Te_Ban,row2cut);
Ti_Ban = cutrow(Ti_Ban,row2cut);
Z_Ban = cutrow(Z_Ban,row2cut);

%% Plotting Temperature Diagram
figure
set(gcf,'numbertitle','off','name','Temp Graph for annotation')
hold on
plot(mean(Te_U(2:end-1,:),2),h_U(2:end-1),'linewidth',2)
plot(mean(Ti_U(2:end-1,:),2),h_U(2:end-1),'linewidth',2)
plot(MSIS(MSIS(:,1)<max(h_U),6),MSIS(MSIS(:,1)<max(h_U),1),'linewidth',2)
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Te','Ti','Tn')
hold off
grid on

%% Measured Data
figure
set(gcf,'numbertitle','off','name','Full Range measurements')
subplot(3,2,1)
```

```
JackKnifePlotY(mean(Ti_U(2:end-1,:),2),h_U(2:end-1),mean(dTi_U(2:end-1,:),2),'k',[0,0.45,0]);
JackKnifePlotY(mean(Te_U(2:end-1,:),2),h_U(2:end-1),mean(dTe_U(2:end-1,:),2),'k',[0,0.85,0]);
plot(MSIS(:,6),MSIS(:,1),'--','linewidth',2);
title('UHF')
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Ti Error','Ti','Te Error','Te','Tn (MSIS)')
hold off
grid on

subplot(3,2,2)
hold on
JackKnifePlotY(mean(Ti_V(2:end-1,:),2),h_V(2:end-1),mean(dTi_V(2:end-1,:),2),'k',[0.65,0,0]);
JackKnifePlotY(mean(Te_V(2:end-1,:),2),h_V(2:end-1),mean(dTe_V(2:end-1,:),2),'k',[0.85,0,0]);
plot(MSIS(:,6),MSIS(:,1),'--','linewidth',2);
title('VHF')
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Ti Error','Ti','Te Error','Te','Tn (MSIS)')
hold off
grid on

subplot(3,2,3)
set(gca,'XScale','log')
hold on
JackKnifePlotY(mean(ne_U(1:end-1,:),2),h_U(1:end-1),mean(dne_U(1:end-1,:),2));
plot(IRI(:,2),IRI(:,1),'--','linewidth',2);
title('UHF')
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
hold off
grid on

subplot(3,2,4)
hold on
JackKnifePlotY(mean(ne_U(1:end-1,:),2),h_U(1:end-1),mean(dne_U(1:end-1,:),2));
plot(IRI(:,2),IRI(:,1),'--','linewidth',2);
title('UHF')
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
hold off
grid on

subplot(3,2,5)
set(gca,'XScale','log')
hold on
JackKnifePlotY(mean(ne_V(1:end-1,:),2),h_V(1:end-1),mean(dne_V(1:end-1,:),2),'k',[0,0.85,0]);
plot(IRI(:,2),IRI(:,1),'--','linewidth',2);
title('VHF')
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
hold off
grid on

subplot(3,2,6)
hold on
JackKnifePlotY(mean(ne_V(1:end-1,:),2),h_V(1:end-1),mean(dne_V(1:end-1,:),2),'k',[0,0.85,0]);
plot(IRI(:,2),IRI(:,1),'--','linewidth',2);
title('VHF')
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
hold off
grid on

%Plots showing all the different time averages of data
% Electrons
figure
set(gcf,'numbertitle','off','name','Electrons Densities')

subplot(1,2,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phe_U = semilogx(ne_All_UHF,Z_Uan/1000,'linewidth',2);
semilogx(IRI_an(:,2),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_U)
title('UHF')
```

```
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
grid on
subplot(1,2,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phe_V = semilogx(ne_All_VHF,Z_Van/1000,'linewidth',2);
semilogx(IRI_an(:,2),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_V)
title('VHF')
ylabel('Altitude [km]')
xlabel('Electron Density [#/m^3]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
grid on

%Electron Temp
figure
set(gcf,'numbertitle','off','name','Electrons Temps')

subplot(1,2,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phe_U = semilogx(Te_All_UHF,Z_Uan/1000,'linewidth',2);
semilogx(IRI_an(:,6),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_U)
title('UHF')
ylabel('Altitude [km]')
xlabel('Electron Temperature [K]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
grid on
subplot(1,2,2)

set(gca,'fontsize',Fnt_Size)
hold on
phe_V = semilogx(Te_All_VHF,Z_Van/1000,'linewidth',2);
semilogx(IRI_an(:,6),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_V)
title('VHF')
ylabel('Altitude [km]')
xlabel('Electron Temperature [K]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
grid on

%Ion Temp
figure
set(gcf,'numbertitle','off','name','Ion Temps')

subplot(1,2,1)
set(gca,'fontsize',Fnt_Size)
hold on
phe_U = plot(Ti_All_UHF,Z_Uan/1000,'linewidth',2);
plot(IRI_an(:,5),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_U)
title('UHF')
ylabel('Altitude [km]')
xlabel('Ion Temperature [K]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
grid on
subplot(1,2,2)
set(gca,'fontsize',Fnt_Size)
hold on
phe_V = plot(Ti_All_VHF,Z_Van/1000,'linewidth',2);
plot(IRI_an(:,5),IRI_an(:,1),'--','linewidth',2)
cmlines(phe_V)
title('VHF')
ylabel('Altitude [km]')
xlabel('Ion Temperature [K]')
legend('4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','IRI')
```

```
grid on


%% Plotting dQ/Sigmas
%Full
figure

set(gcf,'numbertitle','off','name','MF Error over altitudes Full')
hold on
plot(dQSig_U,Z_Uan/1000,'linewidth',2);
plot(dQSig_V,Z_Van/1000,'linewidth',2);
plot(dQSig_B,Z_Ban/1000,'linewidth',2);
title('MATLAB Parameterisation Residuals for 4-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('UHF','VHF','Combined')
grid on

%1/4s UHF
figure

set(gcf,'numbertitle','off','name','MF UHF Quart Errors')
hold on
plot(dQSig_U25,Z_U25an/1000,'linewidth',2);
plot(dQSig_U50,Z_U50an/1000,'linewidth',2);
plot(dQSig_U75,Z_U75an/1000,'linewidth',2);
plot(dQSig_U100,Z_U100an/1000,'linewidth',2);
title('UHF (MATLAB Parameterisation) 1-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd','3rd','4th')
grid on

%1/4s VHF
figure

set(gcf,'numbertitle','off','name','MF VHF Quart Errors')
hold on
plot(dQSig_V25,Z_V25an/1000,'linewidth',2);
plot(dQSig_V50,Z_V50an/1000,'linewidth',2);
plot(dQSig_V75,Z_V75an/1000,'linewidth',2);
plot(dQSig_V100,Z_V100an/1000,'linewidth',2);
title('VHF (MATLAB Parameterisation) 1-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd','3rd','4th')
grid on



%1/2s UHF
figure

set(gcf,'numbertitle','off','name','MF UHF Halves Errors')
hold on
plot(dQSig_Uhalf1,Z_Uhalf1an/1000,'linewidth',2);
plot(dQSig_Uhalf2,Z_Uhalf2an/1000,'linewidth',2);
title('UHF (MATLAB Parameterisation) 2-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd')
grid on

%1/2s VHF
figure

set(gcf,'numbertitle','off','name','MF VHF Halves Errors')
hold on
plot(dQSig_Vhalf1,Z_Vhalf1an/1000,'linewidth',2);
plot(dQSig_Vhalf2,Z_Vhalf2an/1000,'linewidth',2);
title('VHF (MATLAB Parameterisation) 2-hour Data-set')
ylabel('Altitude [km]')
xlabel('dQ/Sigma')
legend('1st','2nd')
grid on
```

```
%% Plotting Energy Transfers
figure
set(gcf,'numbertitle','off','name','MF Energy Transfer Rates')

subplot(2,1,1)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Start_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Best_U(:,1),Z_Uan/1000,'linewidth',2)
legend('Ion to N','e to Ions','Ion to N (Fit)')
title('UHF (MATLAB Parameterisation)')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

subplot(2,1,2)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_V(:,1),Z_Van/1000,'linewidth',2)           %Q_OP2n
semilogx(Q_Start_V(:,2),Z_Van/1000,'--','linewidth',2)      %Q_e2OP
semilogx(Q_Best_V(:,1),Z_Van/1000,'linewidth',2)
legend('Ion to N','e to Ions','Ion to N (Fit)')
title('VHF (MATLAB Parameterisation)')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

% Plotting Neutral Atmosphere
figure

set(gcf,'numbertitle','off','name','MF UHF Neutral Densities')
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
ph_AtmD3 = semilogx(Atm_Start_U(:,3:end),Z_Uan/1000,'--','linewidth',2);
hold on
ph_AtmD4 = semilogx(Atm_Best_U(:,3:end),Z_Uan/1000,'linewidth',2);
cmlines(ph_AtmD3)
cmlines(ph_AtmD4)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O','O2','N2','H','He','O (Fit)','O2 (Fit)','N2 (Fit)','H (Fit)','He (Fit)')
grid on
figure

set(gcf,'numbertitle','off','name','MF VHF Neutral Densities')
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
ph_AtmD1 = semilogx(Atm_Start_V(:,3:end),Z_Van/1000,'--','linewidth',2);
hold on
ph_AtmD2 = semilogx(Atm_Best_V(:,3:end),Z_Van/1000,'linewidth',2);
cmlines(ph_AtmD1)
cmlines(ph_AtmD2)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O','O2','N2','H','He','O (Fit)','O2 (Fit)','N2 (Fit)','H (Fit)','He (Fit)')
grid on



%% Plotting Both Data
figure
set(gcf,'numbertitle','off','name','MF Tn Fitted')
hold on
plot(Tn_All_UHF(:,1),Z_Uan/1000,'-*','linewidth',2)
plot(Tn_fit_U,Z_Uan/1000,'linewidth',2)
plot(Tn_fit_V,Z_Van/1000,'linewidth',2)
plot(Tn_fit_B,Z_Ban/1000,'linewidth',2)
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
title('MATLAB Parameterisation')
```

```
ylabel('Altitude [km]')
xlabel('Neutral Temperature [K]')
legend('Initial','UHF','VHF','Combined','MSIS')
grid on
hold off

figure
set(gcf,'numbertitle','off','name','MF Combined Energy Transfer Rates')
subplot(2,1,1)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Start_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Start_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Start_V(:,1),Z_Van/1000,'linewidth',2)
semilogx(Q_Start_V(:,2),Z_Van/1000,'--','linewidth',2)
title('Raw parameters')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

subplot(2,1,2)
set(gca,'fontsize',Fnt_Size)
set(gca,'XScale','log')
hold on
semilogx(Q_Best_U(:,1),Z_Uan/1000,'linewidth',2)
semilogx(Q_Best_U(:,2),Z_Uan/1000,'--','linewidth',2)
semilogx(Q_Best_V(:,1),Z_Van/1000,'linewidth',2)
semilogx(Q_Best_V(:,2),Z_Van/1000,'--','linewidth',2)
semilogx(Q_Best_B(:,1),Z_Ban/1000,'linewidth',2)
legend('Ion to N (UHF)','e to Ions (UHF)','Ion to N (VHF)','e to Ions (VHF)','Ion to N
(Both)')
title('Fitted Parameters')
ylabel('Altitude [km]')
xlabel('Energy Transfer [eV/(s*m^2)]')
grid on
hold off

figure
set(gcf,'numbertitle','off','name','MF All Fitted Neutral Densities')

set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
ph_AtmD6 = semilogx(Atm_Best_U(:,3:end),Z_Uan/1000,'--','linewidth',2);
ph_AtmD7 = semilogx(Atm_Best_V(:,3:end),Z_Van/1000,'*','linewidth',2);
ph_AtmD5 = semilogx(Atm_Best_B(:,3:end),Z_Ban/1000,'*-','linewidth',2);
cmlines(ph_AtmD5)
cmlines(ph_AtmD6)
cmlines(ph_AtmD7)
title('MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('Particle Density [#/m^3]')
legend('O (UHF)','O2 (UHF)','N2 (UHF)','H (UHF)','He (UHF)',...
       'O (VHF)','O2 (VHF)','N2 (VHF)','H (VHF)','He (VHF)',...
       'O (Both)','O2 (Both)','N2 (Both)','H (Both)','He (Both)')
grid on

%% Total Collected (Fitted)
figure
set(gcf,'numbertitle','off','name','MF Fitted O Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO_U = semilogx(ODen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phO_U)
title('UHF MATLAB Parameterisation') %ADD WHAT PARAM IT IS TO COMBINED TITLES
ylabel('Altitude [km]')
xlabel('O Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on
subplot(2,1,2)
set(gca,'XScale','log')
```

```
set(gca,'fontsize',Fnt_Size)
hold on
phO_V = semilogx(ODen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phO_V)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('O Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')

grid on

figure
set(gcf,'numbertitle','off','name','MF Fitted O2 Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO2_U = semilogx(O2Den_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phO2_U)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('O2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phO2_V = semilogx(O2Den_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phO2_V)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('O2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')

grid on

figure
set(gcf,'numbertitle','off','name','MF Fitted N2 Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phN2_U = semilogx(N2Den_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phN2_U)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('N2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phN2_V = semilogx(N2Den_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phN2_V)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('N2 Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','MF Fitted H Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
```

```
hold on
phH_U = semilogx(HDen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phH_U)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('H Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phH_V = semilogx(HDen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phH_V)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('H Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','MF Fitted He Density')

subplot(2,1,1)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phHe_U = semilogx(HeDen_All_UHF,Z_Uan/1000,'linewidth',2);
cmlines(phHe_U)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('He [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'XScale','log')
set(gca,'fontsize',Fnt_Size)
hold on
phHe_V = semilogx(HeDen_All_VHF,Z_Van/1000,'linewidth',2);
cmlines(phHe_V)
title('VHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('He Density [#/m^3]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

figure
set(gcf,'numbertitle','off','name','MF Fitted Tn')

subplot(2,1,1)
set(gca,'fontsize',Fnt_Size)
hold on
plot(Tn_All_UHF(:,1),Z_Uan/1000,'-*','linewidth',2);
phTn_U = plot(Tn_All_UHF(:,2:end),Z_Uan/1000,'linewidth',2);
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
cmlines(phTn_U)
title('UHF MATLAB Parameterisation')
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on

subplot(2,1,2)
set(gca,'fontsize',Fnt_Size)
hold on
plot(Tn_All_VHF(:,1),Z_Van/1000,'-*','linewidth',2);
phTn_V = plot(Tn_All_VHF(:,2:end),Z_Van/1000,'linewidth',2);
plot(MSIS_an(:,6),MSIS_an(:,1),'--','linewidth',2)
cmlines(phTn_V)
title('VHF MATLAB Parameterisation')
```

```
ylabel('Altitude [km]')
xlabel('Temperature [K]')
legend('Initial','4-hr','1st 1-hr','2nd 1-hr','3rd 1-hr','4th 1-hr',...
       '1st 2-hr','2nd 2-hr','MSIS')
grid on


%% Wind
figure
set(gcf,'numbertitle','off','name','Ion Velocities')
set(gca,'fontsize',Fnt_Size)

subplot(3,1,1)
pcolor(rem(t_V/3600,24),h_V,vi_V); shading flat
caxis([-200 200])
colorbar
title('VHF Particle Velocity [m/s]')
ylabel('Altitude [km]')
xlabel('Time of Day [hr]')

subplot(3,1,2)
pcolor(rem(t_U/3600,24),h_U,vi_U); shading flat
caxis([-200 200])
colorbar
title('UHF Particle Velocity [m/s]')
ylabel('Altitude [km]')
xlabel('Time of Day [hr]')

subplot(3,1,3)
scatter(rem((t_K)/3600,24),h_K,55,vi_K,'filled')
caxis([-100 100])
colorbar
title('Kiruna Particle Velocity [m/s]')
ylabel('Altitude [km]')
xlabel('Time [s]')
grid on

figure
scatter(rem((t_S(1:end-5))/3600,24),h_S(1:end-5),55,vi_S(1:end-5),'filled')
title('Particle Velocity')
ylabel('Altitude [km]')
xlabel('Time [s]')
caxis([100000 9999999])
grid on
```

9.3 ThesisAnalysis

```
function [Atms_Fit,Atms_Raw,Best_Pars,dQSig,dTs,ne_an,Qs_Fit,Ts,Z_an,Sig,varargout] =
ThesisAnalysis(Data_Path,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,TimeFrame,IndexOptions)
%calling [Atms_Fit,Atms_Raw,Best_Pars,dQSig,dTs,ne_an,Qs_Fit,Ts,Z_an,Sig,varargout] =
ThesisAnalysis(Data_Path,MSIS_Path,IRI_Path,Exp_Data,MSISorNICO,TimeFrame,IndexOptions)
%
%Inputs
%Data_Path      full data path of radar matlab matrix to be read. Will read
%               all matricies in a given folder
%MSIS_Path      full file data for MSIS data
%IRI_Path       full file data for IRI data
%Exp_Data       Carry-all cell array for experiment data,
%               see below (line 30-35)
%MSISorNICO     Option for MSIS or Nicolls experiement
%TimeFrame      Time frame for dataset to be averaged over
%               Time frame options, full or matrix index
%IndexOptions   Do you want the analysis setup and geomagnetic indecies for
%               later functions?
%
%Outputs
%Atms_Fit       =Cell array of Fittend and guessed values for
[Z,Tn,ODen,O2Den,N2Den,HDen,HeDen]
%Atms_Raw       =Cell array of MSIS and IRI data
%Best_Pars      =Fitted output parameters [1,X]
%dQSig          =Residual for this run
%dTs            =Standard deviations for temperatures
%ne_an          = Measured electron density within H range and averaged over time trame
%Qs_Fit         =Best estimates of heat transfers [Q_OP2n,Q_e2OP]
%Ts             =Cell array of temperatures {Te, Ti, Tn_Fitted}
%Z_an           =altitude within height limits
%Sig            =Standard deviation of heat transfers
%varargout



%Unpacking Inputs
Lat_Ram = Exp_Data{1};                  %Latitude of Ramsfjord Site [Deg]
Lon_Ram = Exp_Data{2};                  %Longitude of Ramsfjord Site [Deg]
Exp_yr = Exp_Data{3};                   %Experiment year
Exp_d = Exp_Data{4};                    %Experiment Day of year
Exp_t = Exp_Data{5};                    %Experiment Time [S]
h_lim = Exp_Data{6};                    %Height restraints [km]
Sigma = 1;                              %Placeholder for Sigma Value


if nargin < 7 || isempty(IndexOptions)
    IndexOptions = 1;
end



%% Reading and plotting Tromso Guisdap data
%Running ThesisProcess function to convert EISCAT matricies and put data
%within height limits
[Z_an,Ti_an,Te_an,ne_an,Atm_i,Atm_M,dTi_an,dTe_an,vi_an,dvi_an,az_an,el_an,F10_7D,F10_7,Ap] =
ThesisProcess(Data_Path,MSIS_Path,IRI_Path,h_lim,TimeFrame);
PHp = Atm_i(:,end);                 %% of ions that are not O+, as determined by IRI
Z0 = 100;                           %Reference height
Tinf = TExosphere(mean(F10_7D));    %Exospheric temperature determined by the average value of
F10.7 from MSIS data
ORefDensity = Atm_M(1:1);           %denisty of oxygen at reference height (100km)

%Breakdown of outputs
% Atm_i = [Op,Hp,Hep,O2p,NOp,Np,PHp]
% Atm_M = [ODen_M,O2Den_M,N2Den_M,HDen_M,HeDen_M,ArDen_M,NDen_M]
% Atm_N = [ODen_N,O2Den_N,N2Den_N,HDen_N,HeDen_N]
% Best_Atm = [Z,Tn,ODen,O2Den,N2Den,HDen,HeDen]
% Q_Best = [Q_OP2n,Q_e2OP]

%% --------------------Determining Heat Transfer Rates----------------- %%
options = optimset('fminsearch');
options = optimset(options,'Display','iter');
options = optimset(options,'MaxFunEvals',10000);
options = optimset(options,'MaxIter',10000);

if MSISorNICO == 'MSIS'
```

```
%% Least Squares Steup (MSIS)
%pars ['F10_7D','F10_7','Ap']
p_index = [1:3];
fixed_pars = [F10_7,F10_7D,Ap];
start_guess = [F10_7,F10_7D,Ap];
LB = [50,50,0];                              %Lower Bounds for search function
UB = [150,150,15];                           %Upper Bounds for seach function
err_fcn_tag =
@(pars)err_fcn2(pars,p_index,fixed_pars,Z_an,Ti_an,Te_an,ne_an,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_
d,Exp_t,Sigma);

else
%% Least Squares analysis (M. Nicholls)

%pars ['Tinf','ORefDensity']
p_index = [1:2];
fixed_pars = [Tinf,ORefDensity];
start_guess = [Tinf,ORefDensity];
LB = [600,1e14];                 %Lower Bounds for search function
UB = [1300,1e21];                %Upper Bounds for search function
err_fcn_tag = @(pars)err_fcn(pars,p_index,fixed_pars,Z0,Z_an,Ti_an,Te_an,ne_an,PHp,Sigma);

end
%Creating carry all cell array for analysis constants
Anal_Setup = {p_index,fixed_pars,start_guess,LB,UB,options};

%% Least Squares Analysis Run
%Creating carry all cell array for error function inputs
err_ins =
{Z_an,Ti_an,Te_an,dTi_an,dTe_an,ne_an,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,MSISorNICO,Z0,Tin
f,ORefDensity};

%Running BestOutputs function for fitting and refinement of weighting
[Best_Pars1,Atm_Best1,Q_Best1,Atm_Start,Q_Start,dQSig1,Sig1] =
BestOutputs(err_fcn_tag,p_index,fixed_pars,start_guess,LB,UB,options,err_ins);

%remaking the error tag to include refined weighting
if MSISorNICO == 'MSIS'
err_fcn_tag2 =
@(pars)err_fcn2(pars,p_index,fixed_pars,Z_an,Ti_an,Te_an,ne_an,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_
d,Exp_t,Sig1);
else
err_fcn_tag2 = @(pars)err_fcn(pars,p_index,fixed_pars,Z0,Z_an,Ti_an,Te_an,ne_an,PHp,Sig1);
end

%Running fitting and refinement with the refined weighting to get refined
%parameters
[Best_Pars,Atm_Best,Q_Best,Atm_Start2,Q_Start2,dQSig,Sig] =
BestOutputs(err_fcn_tag2,p_index,fixed_pars,start_guess,LB,UB,options,err_ins);

Tn_fit = Atm_Best(:,2); %fitted neutral temperature

%Creating carry-all cell arrays for output
Atms_Fit = {Atm_Best,Atm_Start};
Atms_Raw = {Atm_i,Atm_M};
dTs = {dTi_an,dTe_an};
Qs_Fit = {Q_Best,Q_Start};
Ts = {Ti_an,Te_an,Tn_fit};

if IndexOptions == 2
  varargout{1} = F10_7D;         %Single value for daily F10.7 index from MSIS
  varargout{2} = F10_7;          %Single value for 90 day F10.7 index from MSIS
  varargout{3} = Ap;             %Single value Ap index from MSIS
  varargout{4} = Anal_Setup;     %carry all cell array for analysis constants
else

end
```

9.4 ThesisProcess

```
function [Z_an,Ti_an,Te_an,ne_an,Atm_i,Atm_M,dTi_an,dTe_an,vi_an,dvi_an,az,el,F10_7D,F10_7,Ap]
= ThesisProcess(Data_Path,MSIS_Path,IRI_Path,h_lim,TimeFrame)
%Calling ThesisProcess
%Inputs
%Data_Path      full data path of radar matlab matrix to be read. Will read
%               all matricies in a given folder
%
%MSIS_Path      full file data for MSIS data
%
%IRI_Path       full file data for IRI data
%
%h_lim          [2 x 1] matrix of lower and upper height
%TimeFrame
%
%Outputs
%Z_an           = Altitude range within lower and upper height
%Ti_an          = Measured ion temp within H range averaged over time trame
%Te_an          = Measured electron temp within H range
%ne_an          = Measured electron density within H range and averaged over time trame
%Atm_i          = IRI data for H range [X,7]
%Atm_M          = IRI data for H range [Y,6]
%dTi_an         = Measured ion temp standard deviation within H range and averaged over time
trame
%dTe_an         = Measured electron temp standard deviation within H range and averaged over
time trame
%vi_an          = Measured Ion velocity component within H range and averaged over time trame
%dvi_an         = Measured Ion velocity component standard deviation within H range and
averaged over time trame
%az             = Radar Azimuth over time
%el             = Radar elevation over time
%F10_7D         = Daily F10.7 number from MSIS
%F10_7          = 90 day F10.7 number from MSIS
%Ap =           = Ap index from MSIS

%Function to read all matlab matricies within folder
try %Will catch if data is being output as cells and put into different function that converts
cells
[h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = readGdata(Data_Path);
catch
[h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = readGdata2(Data_Path);
end

%Analysis Constants
Z0 = 100e3;                                              %Chosen reference height [m]

%Are the outputs cells?
if iscell(h)
    h = cell2mat(h);
    ne = cell2mat(ne);
    Te = cell2mat(Te);
    Ti = cell2mat(Ti);
    vi = cell2mat(vi);
    dvi = cell2mat(dvi);

end

%Seeing what the time stamp is and putting it in a way that can be enetered
%into plots
if strcmp(' full',TimeFrame) || strcmp(' Full',TimeFrame)

    TF = ':';

else
    TF = str2num(TimeFrame);
end

%Input values taken from Scan
Z = h*1e3;                                               %Renaming altitude data from scan to
suit my fascist ideals [km]
Z_an = Z(h_lim(1)<Z&Z<h_lim(2));                         %Analysis heights [km]
Ti_an = mean(Ti((h_lim(1)<Z&Z<h_lim(2)),TF),2);          %Average Ion temperature fitted to
measurable height profile [K]
Te_an = mean(Te((h_lim(1)<Z&Z<h_lim(2)),TF),2);          %Average Electron temperature fitted to
measurable height profile [K]
```

```
ne_an = mean(ne((h_lim(1)<Z&Z<h_lim(2)),TF),2);        %Average Electron Density fitted to
measurable height profile [#/m^3]
dTe_an = mean(dTe((h_lim(1)<Z&Z<h_lim(2)),TF),2);      %Average standard deviation for
electron temp fitted to measurable height profile [#/m^3]
dTi_an = mean(dTi((h_lim(1)<Z&Z<h_lim(2)),TF),2);      %Average standard deviation for ion
temp fitted to measurable height profile [#/m^3]
vi_an = mean(vi((h_lim(1)<Z&Z<h_lim(2)),TF),2);        %Average ion velocity across measurable
height profile [m/s]
dvi_an = mean(dvi((h_lim(1)<Z&Z<h_lim(2)),TF),2);      %Average standard deviation for ion
velocity fitted to measurable height profile [m/s]
t_an = t(TF);                                          %Time frame of scan

%MSIS setup values
MSIS = load(MSIS_Path);
F10_7D = MSIS(1,12);                                   %F10.7 Daily value
F10_7 = MSIS(1,13);                                    %F10.7 tri-Monthly value
Ap = mean(MSIS(1,14:20));                              %Ap index Daily Value
ODen_M = MSIS(h_lim(1)<Z&Z<h_lim(2),2)*1e6;            %MSIS O Density Profile [#/m^3]
N2Den_M = MSIS(h_lim(1)<Z&Z<h_lim(2),3)*1e6;           %MSIS N2 Density Profile [#/m^3]
O2Den_M = MSIS(h_lim(1)<Z&Z<h_lim(2),4)*1e6;           %MSIS O2 Density Profile [#/m^3]
HeDen_M = MSIS(h_lim(1)<Z&Z<h_lim(2),8)*1e6;           %MSIS He Density Profile [#/m^3]
ArDen_M = MSIS(h_lim(1)<Z&Z<h_lim(2),9)*1e6;           %MSIS Ar Density Profile [#/m^3]
HDen_M = MSIS(h_lim(1)<Z&Z<h_lim(2),10)*1e6;           %MSIS H Density Profile [#/m^3]
NDen_M = MSIS(h_lim(1)<Z&Z<h_lim(2),11)*1e6;           %MSIS N Density Profile [#/m^3]

Atm_M = [ODen_M,O2Den_M,N2Den_M,HDen_M,HeDen_M,ArDen_M,NDen_M];

%IRI loading and values
IRI = load(IRI_Path);
Op = IRI(h_lim(1)<Z&Z<h_lim(2),7).*ne_an/100;          %IRI O+ Density Profile [#/m^3]
Hp = IRI(h_lim(1)<Z&Z<h_lim(2),8).*ne_an/100;          %IRI H+ Density Profile [#/m^3]
Hep = IRI(h_lim(1)<Z&Z<h_lim(2),9).*ne_an/100;         %IRI He+ Density Profile [#/m^3]
O2p = IRI(h_lim(1)<Z&Z<h_lim(2),10).*ne_an/100;        %IRI O2+ Density Profile [#/m^3]
NOp = IRI(h_lim(1)<Z&Z<h_lim(2),11).*ne_an/100;        %IRI NO+ Density Profile [#/m^3]
Np = IRI(h_lim(1)<Z&Z<h_lim(2),13).*ne_an/100;         %IRI N+ Density Profile [#/m^3]
PHp = (Hp+Hep+O2p+NOp+Np)./ne_an;                      %Percentage of other ions

Atm_i = [Op,Hp,Hep,O2p,NOp,Np,PHp];



%% Plotting Raw Data

 figure
 set(gcf,'numbertitle','off','name',strcat(Data_Path(58:60),TimeFrame,' Measured Data'))
 subplot(3,1,1)
 pcolor(rem(t_an/3600,24),h,log10(max(0,ne(:,TF)))); shading flat;
 title('Electron Density (#/m^2)')
 caxis([9 12])
 ylabel('Altitude [km]')
 xlabel('Time [Hours]')
 colorbar

 subplot(3,1,2)
 pcolor(rem(t_an/3600,24),h,max(0,Te(:,TF))); shading flat;
 title('Electron Temperature (K)')
 caxis([0 2500])
 ylabel('Altitude [km]')
 xlabel('Time [Hours]')
 colorbar

 subplot(3,1,3)
 pcolor(rem(t_an/3600,24),h,max(0,Ti(:,TF))); shading flat;
 title('Ion Temperature (K)')
 caxis([0 2500])
 ylabel('Altitude [km]')
 xlabel('Time [Hours]')
 colorbar

if strcmp(' full',TimeFrame) || strcmp(' Full',TimeFrame)
 figure
 set(gcf,'numbertitle','off','name',strcat(Data_Path(58:60),TimeFrame,' Reference Data'))
 subplot(1,2,1)
 set(gca,'XScale','log')
 hold on
 ph_Atm1 = semilogx(Atm_M,Z_an/1000,'.-','linewidth',2);
```

```
 cmlines(ph_Atm1)
 title('MSIS')
 ylabel('Altitude [km]')
 xlabel('Particle Density [#/m^3]')
 legend('O (MSIS)','O2 (MSIS)','N2 (MSIS)','H (MSIS)','He (MSIS)','Ar (MSIS)','N (MSIS)')
 grid on
 hold off

 subplot(1,2,2)
 set(gca,'XScale','log')
 hold on
 semilogx(Op,Z_an/1000,'.-','linewidth',2)
 semilogx(Hp,Z_an/1000,'*','linewidth',2)
 semilogx(Hep,Z_an/1000,'--','linewidth',2)
 semilogx(O2p,Z_an/1000,':','linewidth',2)
 semilogx(NOp,Z_an/1000,'x','linewidth',2)
 semilogx(Np,Z_an/1000,'-','linewidth',2)
 title('IRI')
 ylabel('Altitude [km]')
 xlabel('Particle Density [#/m^3]')
 legend('Op','Hp','Hep','O2p','NOp','Np')
 grid on
 hold off

end
```

9.5 BestOutputs

```
function [Best_Pars,Atm_Best,Q_Best,Atm_Start,Q_Start,dQSig,Sig] =
BestOutputs(err_fcn_tag,p_index,fixed_pars,start_guess,LB,UB,options,err_ins)
%Calling: [Best_Pars,Atm_Best,Q_Best,Atm_Start,Q_Start,dQSig,Sig] =
BestOutputs(err_fcn_tag,p_index,fixed_pars,start_guess,LB,UB,options,err_ins)
%
%Inputs
%err_fcn_tag        =Error tag function handle to @pars
%p_index            =Which parameters are fixed and variable? Those within
%                    index are variable
%fixed_pars         =The fixed parameters for this run
%start_guess        =The start guess for the run
%LB                 =Lower bounds for fminsearchbnd
%UB                 =Upper bounds for fminsearchbnd
%options            =Normal fminsearch option structure
%err_ins            =Carry-all cell array containin info bellow
%                   (Line 26-41)
%
%Outputs
%Best_Pars          =Fitted output parameters [1,X]
%Atm_Best           =Fittend values for [Z,Tn,ODen,O2Den,N2Den,HDen,HeDen]
%Q_Best             =Fitted values for [Q_OP2n,Q_e2OP]
%Atm_Start          =Initial guess values for [Z,Tn,ODen,O2Den,N2Den,HDen,HeDen]
%Q_Start            =Initial guess values for [Q_OP2n,Q_e2OP]
%dQSig              =Residual for this run
%Sig                =Standard deviation for this run

%Unpacking inputted "carry-all" cell array
Z = err_ins{1};
Ti = err_ins{2};
Te = err_ins{3};
dTi = err_ins{4};
dTe = err_ins{5};
ne = err_ins{6};
PHp = err_ins{7};
Lat_Ram = err_ins{8};
Lon_Ram = err_ins{9};
Exp_yr = err_ins{10};
Exp_d = err_ins{11};
Exp_t = err_ins{12};
MSISorNICO = err_ins{13};
Z0 = err_ins{14};
Tinf = err_ins{15};
ORefDensity = err_ins{16};
Sigma = 1;



%Checking to see what type of err_fcn we are using
```

```
if MSISorNICO == 'MSIS'
Best_Pars(1,1:3) = fminsearchbnd(err_fcn_tag,start_guess,LB,UB,options);

[err(1,:),Q(:,1:2)]   = err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti-
1*dTi,Te+1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma);
[err(2,:),Q(:,3:4)]   =
err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+0*dTi,Te+1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,
Exp_t,Sigma);
[err(3,:),Q(:,5:6)]   =
err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+1*dTi,Te+1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,
Exp_t,Sigma);
[err(4,:),Q(:,7:8)]   = err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti-
1*dTi,Te+0*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma);
[err(5,:),Q(:,9:10)]  =
err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+0*dTi,Te+0*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,
Exp_t,Sigma);
[err(6,:),Q(:,11:12)] =
err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+1*dTi,Te+0*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,
Exp_t,Sigma);
[err(7,:),Q(:,13:14)] = err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti-1*dTi,Te-
1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma);
[err(8,:),Q(:,15:16)] = err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+0*dTi,Te-
1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma);
[err(9,:),Q(:,17:18)] = err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti+1*dTi,Te-
1*dTe,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma);

else
Best_Pars(1,1:2) = fminsearchbnd(err_fcn_tag,start_guess,LB,UB,options);

[err(1,:),Q(:,1:2)]   = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti-
1*dTi,Te+1*dTe,ne,PHp,Sigma);
[err(2,:),Q(:,3:4)]   =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+0*dTi,Te+1*dTe,ne,PHp,Sigma);
[err(3,:),Q(:,5:6)]   =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+1*dTi,Te+1*dTe,ne,PHp,Sigma);
[err(4,:),Q(:,7:8)]   = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti-
1*dTi,Te+0*dTe,ne,PHp,Sigma);
[err(5,:),Q(:,9:10)]  =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+0*dTi,Te+0*dTe,ne,PHp,Sigma);
[err(6,:),Q(:,11:12)] =
err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+1*dTi,Te+0*dTe,ne,PHp,Sigma);
[err(7,:),Q(:,13:14)] = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti-1*dTi,Te-
1*dTe,ne,PHp,Sigma);
[err(8,:),Q(:,15:16)] = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+0*dTi,Te-
1*dTe,ne,PHp,Sigma);
[err(9,:),Q(:,17:18)] = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti+1*dTi,Te-
1*dTe,ne,PHp,Sigma);

end

%Getting standard deviations from running 9 cases, for each +/- comination
%of dTi and dTe applied to their respective temperature
Sig_i2n = std(Q(:,1:2:size(Q,2)),0,2);
Sig_e2i = std(Q(:,2:2:size(Q,2)),0,2);

%Totalling the two standard deviations
Sig = sqrt(Sig_i2n.^2+Sig_e2i.^2);


%Running error function again using new standard deviation as weighting to
%find best parameters
if MSISorNICO == 'MSIS'
[Error1,Atm_Start,Q_Start] =
err_fcn2(start_guess,p_index,fixed_pars,Z,Ti,Te,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sig,
2);
[Error2,Atm_Best,Q_Best] =
err_fcn2(Best_Pars,p_index,fixed_pars,Z,Ti,Te,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sig,2)
;
else
[Error1,Atm_Start,Q_Start] = err_fcn(start_guess,p_index,fixed_pars,Z0,Z,Ti,Te,ne,PHp,Sig,2);
[Error2,Atm_Best,Q_Best] = err_fcn(Best_Pars,p_index,fixed_pars,Z0,Z,Ti,Te,ne,PHp,Sig,2);
end

%Calculation of residuals
dQSig = (Q_Best(:,2)-Q_Best(:,1))./Sig;
```

10 APPENDIX B: MINOR MATLAB CODES (ALPHABETISED)

```
function M_X=AMU2kg(amu)

%Function for converting atomic mass to kg

M_X=(amu/6.022e23)/1e3;

function [Atm_Best,Atm_Start,Atm_i,Atm_M,dTi_an,dTe_an,Q_Best,Q_Start,Ti_an,Te_an,Tn_fit] =
AnalysisUnpacking(Atms_Fit,Atms_Raw,dTs,Qs_Fit,Ts)

%Little wrapper function to save space on main code converts the incoming
%arrays into seperate matricies

Atm_Best = Atms_Fit{1};
Atm_Start = Atms_Fit{2};

Atm_i = Atms_Raw{1};
Atm_M = Atms_Raw{2};

dTi_an = dTs{1};
dTe_an = dTs{2};

Q_Best = Qs_Fit{1};
Q_Start = Qs_Fit{2};


Ti_an = Ts{1};
Te_an = Ts{2};
Tn_fit = Ts{3};

function [HL,CLIN] = cmlines(varargin)
% CMLINES   Change the color of plotted lines using the colormap.
%
%   SYNTAX:
%                 cmlines
%                 cmlines(CMAP)
%                 cmlines(H,...)
%     [HL,CLIN] = cmlines(...);
%
%   INPUT:
%     CMAP - Color map name or handle to be used, or a Nx3 matrix of colors
%            to be used for each of the N lines or color char specifiers.
%            DEFAULT: jet.
%     H    - Handles of lines or from a axes to search for lines or from
%            figures to search for exes. If used, must be the first input.
%            DEFAULT: gca (sets colors for lines in current axes)
%
%   OUTPUT (all optional):
%     HL   - Returns the handles of lines. Is a cell array if several axes
%            handle were used as input.
%     CLIN - Returns the RGB colors of the lines. Is a cell array if
%            several axes handle were used as input.
%
%   DESCRIPTION:
%     Ths function colored the specified lines with the spectrum of the
%     given colormap. Ideal for lines on the same axes which means increase
%     (or decrease) monotonically.
%
%   EXAMPLE:
%     plot(reshape((1:10).^2,2,5))
%     cmlines
%
%   NOTE:
%     * Optional inputs use its DEFAULT value when not given or [].
%     * Optional outputs may or not be called.
%
%   SEE ALSO:
%     PLOT and COLORMAP.
%     and
%     CMAPPING
```

```
%     at http://www.mathworks.com/matlabcentral/fileexchange
%
%
%   ---
%   MFILE:   cmlines.m
%   VERSION: 1.0 (Jun 08, 2009) (<a
href="matlab:web(['www.mathworks.com/matlabcentral/fileexchange/loadAuthor.do',char(63),'objec
tType',char(61),'author',char(38),'objectId=1093874'])">download</a>)
%   MATLAB:  7.7.0.471 (R2008b)
%   AUTHOR:  Carlos Adrian Vargas Aguilera (MEXICO)
%   CONTACT: nubeobscura@hotmail.com

%   REVISIONS:
%   1.0      Released. (Jun 08, 2009)

%   DISCLAIMER:
%   cmlines.m is provided "as is" without warranty of any kind, under the
%   revised BSD license.

%   Copyright (c) 2009 Carlos Adrian Vargas Aguilera

% INPUTS CHECK-IN
% --------------------------------------------------------------------

% Set defaults:
HL   = {};
Ha   = gca;
CMAP = colormap;

% Checks number of inputs:
if nargin>2
 error('CVARGAS:cmlines:tooManyInputs', ...
  'At most 2 inputs are allowed.')
end
if nargout>2
 error('CVARGAS:cmlines:tooManyOutputs', ...
  'At most 2 outputs are allowed.')
end

% Checks handles of lines, axes or figure inputs:
Hl = [];
if (nargin~=0) && ~isempty(varargin{1}) && all(ishandle(varargin{1}(:))) ...
 && ((length(varargin{1})>1) || ~isa(varargin{1},'function_handle'))
 Ha = [];
 for k = 1:length(varargin{1})
  switch get(varargin{1}(k),'Type')
   case 'line'
    Hl = [Hl varargin{1}(k)];
   case 'axes'
    Ha = [Ha varargin{1}(k)];
   case {'figure','uipanel'}
    Ha = [Ha findobj(varargin{1}(k),'-depth',1,'Type','axes',...
                     '-not',{'Tag','Colorbar','-or','Tag','legend'})];
   otherwise
     warning('CVARGAS:cmlines:unrecognizedHandleInput',...
      'Ignored handle input.')
  end
 end
 varargin(1) = [];
end

% Looks for CMAP input:
if nargin && ~isempty(varargin) && ~isempty(varargin{1})
 CMAP = varargin{1};
end

% Gets line handles:
if ~isempty(Hl)
 HL{1} = Hl;
end
if ~isempty(Ha)
 for k = 1:length(Ha)
  Hl = findobj(Ha(k),'Type','line');
  if ~isempty(Hl)
   HL{end+1} = Hl;
  end
 end
```

```
end
if isempty(HL)
 if ~nargout
  clear HL
 end
 return
end

% ---------------------------------------------------------------------------
% MAIN
% ---------------------------------------------------------------------------

% Sets color lines for each set of lines:
Nlines = length(HL);
CLIN   = cell(1,Nlines);
for k  = 1:length(HL)

 % Interpolates the color map:
 CLIN{k} = cmapping(length(HL{k}),CMAP);

 % Changes lines colors:
 set(HL{k},{'Color'},mat2cell(CLIN{k},ones(1,size(CLIN{k},1)),3))

end

% OUTPUTS CHECK-OUT
% ---------------------------------------------------------------------------

if ~nargout
 clear HL
elseif Nlines==1
 HL   = HL{1};
 CLIN = CLIN{1};
end


% [EOF]   cmlines.m

function X = cutrow(X,row2cut)
%Calling: X = cutline(X,row2cut)
%
%A function to row a matrix X at the row2cut so that concantinated elements
%will make two distinct lines.

X(row2cut+1:end+1,:)=X(row2cut:end,:);
X(row2cut) = nan;

function varargout = err_fcn(VarPars,p_index,fixed_pars,Z0,Z,Ti,Te,ne,PHp,Sigma,outargtype)

%function err_sq = err_fcn(pars,Z0,Z,Ti,Te,ne,PHp)

if nargin < 11 || isempty(outargtype)
    outargtype = 1;
end
%% Least Squares
pars = fixed_pars;
pars(p_index)=VarPars(p_index);
Tinf = pars(1);
ORefDensity = pars(2);

%% Free Param
Tn0 = 185.3;
dTndZ = (203.1-185.3)/5000;                              %Temperature Gradient at reference
height [K/m]
O2RefDensity = (2.135e12)*1e6;                           %O2 density at refrencce height [#/m^3]
N2RefDensity = (8.835e12)*1e6;                           %N2 density at refrencce height [#/m^3]
HRefDensity =  (2.185e7)*1e6;                            %H density at refrencce height  [#/m^3]
HeRefDensity = (1.098e8)*1e6;                            %He density at refrencce height [#/m^3]


%% Paramatarised Variables
Zeta = GeoPotH(Z0,Z);
S = InvScaleH(Tinf,Tn0,dTndZ);
Tn = TofZeta(Tn0,Tinf,S,Zeta);

ODen = OofZeta(Tn0,S,Zeta,Tinf,ORefDensity,Z0);
O2Den = O2ofZeta(Tn0,S,Zeta,Tinf,O2RefDensity,Z0);
```

```
N2Den = N2ofZeta(Tn0,S,Zeta,Tinf,N2RefDensity,Z0);
HDen = HofZeta(Tn0,S,Zeta,Tinf,HRefDensity,Z0);
HeDen = HeofZeta(Tn0,S,Zeta,Tinf,HeRefDensity,Z0);

%% Energy Transfer Rates
Q_Inel = Q_inel(ne,HDen,Ti,Tn,ODen,PHp);          %Heat tranfer from Inelastic Conditions
[eV/m^3*s]

Q_Op2O = Q_Oion2O(ODen,ne,PHp,Ti,Tn);             %Heat Transfer between O+ and O atoms
[eV/m^3*s]
Q_Op2O2 = Q_Oion2O2(O2Den,ne,PHp,Ti,Tn);          %Heat Transfer between O+ and O2 atoms
[eV/m^3*s]
Q_Op2N2 = Q_Oion2N2(N2Den,ne,PHp,Ti,Tn);          %Heat Transfer between O+ and N2 atoms
[eV/m^3*s]
Q_Op2H = Q_Oion2H(HDen,ne,PHp,Ti,Tn);             %Heat Transfer between O+ and H atoms
[eV/m^3*s]
Q_Op2He = Q_Oion2He(HeDen,ne,PHp,Ti,Tn);          %Heat Transfer between O+ and He atoms
[eV/m^3*s]

Q_OP2n = (Q_Op2O+Q_Op2O2+Q_Op2N2+Q_Op2H+Q_Op2He+Q_Inel);

Q_e2OP = (Q_e2Oion(ne,Te,Ti,PHp));                %Heat Transfer between e and O+ atoms
[eV/m^3*s]



err_sq = sum((abs(Q_e2OP)-Q_OP2n).^2./Sigma.^2);

if outargtype == 1
  varargout{1} = err_sq;
  varargout{2} = [Q_OP2n,Q_e2OP];
elseif outargtype == 2
  varargout{1} = err_sq;
  varargout{2} = [Z,Tn,ODen,O2Den,N2Den,HDen,HeDen];
  varargout{3} = [Q_OP2n,Q_e2OP];

end

function varargout =
err_fcn2(VarPars,p_index,fixed_pars,Z,Ti,Te,ne,PHp,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,Sigma,ou
targtype)
%function err_sq = err_fcn(pars,Z0,Z,Ti,Te,ne,PHp)

if nargin < 15 || isempty(outargtype)
    outargtype = 1;
end

[Exp_yr,Exp_d,Exp_t] = ExpDateTime(Z,Exp_yr,Exp_d,Exp_t);
pars = fixed_pars;
pars(p_index)=VarPars(p_index);

%% Least Squares
F10_7 = pars(1);
F10_7D = pars(2);
Ap = pars(3)*ones(1,7); %,fixed_pars(4:end)];%pars(3:9);
flags = ones(1,23);
flags(2:9) = -1;
flags(12:13) = -1;
flags(15) = -1;


[T,Rho] = atmosnrlmsise00(Z,Lat_Ram,Lon_Ram,Exp_yr,Exp_d,Exp_t,F10_7,F10_7D,Ap,flags);
Tn = T(:,2);
HeDen = Rho(:,1);
ODen = Rho(:,2);
N2Den = Rho(:,3);
O2Den = Rho(:,4);
ArDen = Rho(:,5);
HDen = Rho(:,7);

Atm = [Tn,ODen,O2Den,N2Den,HDen,HeDen];

%% Energy Transfer from Ions to N
Q_Inel = Q_inel(ne,HDen,Ti,Tn,ODen,PHp);          %Heat tranfer from Inelastic Conditions
[eV/m^3*s]
```

```
Q_Op2O = Q_Oion2O(ODen,ne,PHp,Ti,Tn);              %Heat Transfer between O+ and O atoms
[eV/m^3*s]
Q_Op2O2 = Q_Oion2O2(O2Den,ne,PHp,Ti,Tn);           %Heat Transfer between O+ and O2 atoms
[eV/m^3*s]
Q_Op2N2 = Q_Oion2N2(N2Den,ne,PHp,Ti,Tn);           %Heat Transfer between O+ and N2 atoms
[eV/m^3*s]
Q_Op2H = Q_Oion2H(HDen,ne,PHp,Ti,Tn);              %Heat Transfer between O+ and H atoms
[eV/m^3*s]
Q_Op2He = Q_Oion2He(HeDen,ne,PHp,Ti,Tn);           %Heat Transfer between O+ and He atoms
[eV/m^3*s]

Q_OP2n = (Q_Op2O+Q_Op2O2+Q_Op2N2+Q_Op2H+Q_Op2He+Q_Inel);

%% Energy Transfer from e to Ion
Q_e2OP = (Q_e2Oion(ne,Te,Ti,PHp));             %Heat Transfer between e and O+ atoms
[eV/m^3*s]


err_sq = sum((abs(Q_e2OP)-Q_OP2n).^2./Sigma.^2);

%% Outputs

if outargtype == 1
  varargout{1} = [err_sq];
  varargout{2} = [Q_OP2n,Q_e2OP];
elseif outargtype == 2
  varargout{1} = [err_sq];
  varargout{2} = [Z,Atm];
  varargout{3} = [Q_OP2n,Q_e2OP];
end

function [x,fval,exitflag,output] = fminsearchbnd(fun,x0,LB,UB,options,varargin)
% FMINSEARCHBND: FMINSEARCH, but with bound constraints by transformation
% usage: x=FMINSEARCHBND(fun,x0)
% usage: x=FMINSEARCHBND(fun,x0,LB)
% usage: x=FMINSEARCHBND(fun,x0,LB,UB)
% usage: x=FMINSEARCHBND(fun,x0,LB,UB,options)
% usage: x=FMINSEARCHBND(fun,x0,LB,UB,options,p1,p2,...)
% usage: [x,fval,exitflag,output]=FMINSEARCHBND(fun,x0,...)
%
% arguments:
%  fun, x0, options - see the help for FMINSEARCH
%
%  LB - lower bound vector or array, must be the same size as x0
%
%       If no lower bounds exist for one of the variables, then
%       supply -inf for that variable.
%
%       If no lower bounds at all, then LB may be left empty.
%
%       Variables may be fixed in value by setting the corresponding
%       lower and upper bounds to exactly the same value.
%
%  UB - upper bound vector or array, must be the same size as x0
%
%       If no upper bounds exist for one of the variables, then
%       supply +inf for that variable.
%
%       If no upper bounds at all, then UB may be left empty.
%
%       Variables may be fixed in value by setting the corresponding
%       lower and upper bounds to exactly the same value.
%
% Notes:
%
%  If options is supplied, then TolX will apply to the transformed
%  variables. All other FMINSEARCH parameters should be unaffected.
%
%  Variables which are constrained by both a lower and an upper
%  bound will use a sin transformation. Those constrained by
%  only a lower or an upper bound will use a quadratic
%  transformation, and unconstrained variables will be left alone.
%
%  Variables may be fixed by setting their respective bounds equal.
%  In this case, the problem will be reduced in size for FMINSEARCH.
%
%  The bounds are inclusive inequalities, which admit the
```

```
%  boundary values themselves, but will not permit ANY function
%  evaluations outside the bounds. These constraints are strictly
%  followed.
%
%  If your problem has an EXCLUSIVE (strict) constraint which will
%  not admit evaluation at the bound itself, then you must provide
%  a slightly offset bound. An example of this is a function which
%  contains the log of one of its parameters. If you constrain the
%  variable to have a lower bound of zero, then FMINSEARCHBND may
%  try to evaluate the function exactly at zero.
%
%
% Example usage:
% rosen = @(x) (1-x(1)).^2 + 105*(x(2)-x(1).^2).^2;
%
% fminsearch(rosen,[3 3])     % unconstrained
% ans =
%    1.0000    1.0000
%
% fminsearchbnd(rosen,[3 3],[2 2],[])     % constrained
% ans =
%    2.0000    4.0000
%
% See test_main.m for other examples of use.
%
%
% See also: fminsearch, fminspleas
%
%
% Author: John D'Errico
% E-mail: woodchips@rochester.rr.com
% Release: 4
% Release date: 7/23/06

% size checks
xsize = size(x0);
x0 = x0(:);
n=length(x0);

if (nargin<3) || isempty(LB)
  LB = repmat(-inf,n,1);
else
  LB = LB(:);
end
if (nargin<4) || isempty(UB)
  UB = repmat(inf,n,1);
else
  UB = UB(:);
end

if (n~=length(LB)) || (n~=length(UB))
  error 'x0 is incompatible in size with either LB or UB.'
end

% set default options if necessary
if (nargin<5) || isempty(options)
  options = optimset('fminsearch');
end

% stuff into a struct to pass around
params.args = varargin;
params.LB = LB;
params.UB = UB;
params.fun = fun;
params.n = n;
% note that the number of parameters may actually vary if
% a user has chosen to fix one or more parameters
params.xsize = xsize;
params.OutputFcn = [];

% 0 --> unconstrained variable
% 1 --> lower bound only
% 2 --> upper bound only
% 3 --> dual finite bounds
% 4 --> fixed variable
params.BoundClass = zeros(n,1);
for i=1:n
```

```
  k = isfinite(LB(i)) + 2*isfinite(UB(i));
  params.BoundClass(i) = k;
  if (k==3) && (LB(i)==UB(i))
    params.BoundClass(i) = 4;
  end
end

% transform starting values into their unconstrained
% surrogates. Check for infeasible starting guesses.
x0u = x0;
k=1;
for i = 1:n
  switch params.BoundClass(i)
    case 1
      % lower bound only
      if x0(i)<=LB(i)
        % infeasible starting value. Use bound.
        x0u(k) = 0;
      else
        x0u(k) = sqrt(x0(i) - LB(i));
      end

      % increment k
      k=k+1;
    case 2
      % upper bound only
      if x0(i)>=UB(i)
        % infeasible starting value. use bound.
        x0u(k) = 0;
      else
        x0u(k) = sqrt(UB(i) - x0(i));
      end

      % increment k
      k=k+1;
    case 3
      % lower and upper bounds
      if x0(i)<=LB(i)
        % infeasible starting value
        x0u(k) = -pi/2;
      elseif x0(i)>=UB(i)
        % infeasible starting value
        x0u(k) = pi/2;
      else
        x0u(k) = 2*(x0(i) - LB(i))/(UB(i)-LB(i)) - 1;
        % shift by 2*pi to avoid problems at zero in fminsearch
        % otherwise, the initial simplex is vanishingly small
        x0u(k) = 2*pi+asin(max(-1,min(1,x0u(k))));
      end

      % increment k
      k=k+1;
    case 0
      % unconstrained variable. x0u(i) is set.
      x0u(k) = x0(i);

      % increment k
      k=k+1;
    case 4
      % fixed variable. drop it before fminsearch sees it.
      % k is not incremented for this variable.
  end

end
% if any of the unknowns were fixed, then we need to shorten
% x0u now.
if k<=n
  x0u(k:n) = [];
end

% were all the variables fixed?
if isempty(x0u)
  % All variables were fixed. quit immediately, setting the
  % appropriate parameters, then return.

  % undo the variable transformations into the original space
  x = xtransform(x0u,params);
```

```
  % final reshape
  x = reshape(x,xsize);

  % stuff fval with the final value
  fval = feval(params.fun,x,params.args{:});

  % fminsearchbnd was not called
  exitflag = 0;

  output.iterations = 0;
  output.funcCount = 1;
  output.algorithm = 'fminsearch';
  output.message = 'All variables were held fixed by the applied bounds';

  % return with no call at all to fminsearch
  return
end

% Check for an outputfcn. If there is any, then substitute my
% own wrapper function.
if ~isempty(options.OutputFcn)
  params.OutputFcn = options.OutputFcn;
  options.OutputFcn = @outfun_wrapper;
end

% now we can call fminsearch, but with our own
% intra-objective function.
[xu,fval,exitflag,output] = fminsearch(@intrafun,x0u,options,params);

% undo the variable transformations into the original space
x = xtransform(xu,params);

% final reshape to make sure the result has the proper shape
x = reshape(x,xsize);

% Use a nested function as the OutputFcn wrapper
  function stop = outfun_wrapper(x,varargin);
    % we need to transform x first
    xtrans = xtransform(x,params);

    % then call the user supplied OutputFcn
    stop = params.OutputFcn(xtrans,varargin{1:(end-1)});

  end

end % mainline end

% ======================================
% ========= begin subfunctions =========
% ======================================
function fval = intrafun(x,params)
% transform variables, then call original function

% transform
xtrans = xtransform(x,params);

% and call fun
fval = feval(params.fun,reshape(xtrans,params.xsize),params.args{:});

end % sub function intrafun end

% ======================================
function xtrans = xtransform(x,params)
% converts unconstrained variables into their original domains

xtrans = zeros(params.xsize);
% k allows some variables to be fixed, thus dropped from the
% optimization.
k=1;
for i = 1:params.n
  switch params.BoundClass(i)
    case 1
      % lower bound only
      xtrans(i) = params.LB(i) + x(k).^2;

      k=k+1;
```

```
    case 2
      % upper bound only
      xtrans(i) = params.UB(i) - x(k).^2;

      k=k+1;
    case 3
      % lower and upper bounds
      xtrans(i) = (sin(x(k))+1)/2;
      xtrans(i) = xtrans(i)*(params.UB(i) - params.LB(i)) + params.LB(i);
      % just in case of any floating point problems
      xtrans(i) = max(params.LB(i),min(params.UB(i),xtrans(i)));

      k=k+1;
    case 4
      % fixed variable, bounds are equal, set it at either bound
      xtrans(i) = params.LB(i);
    case 0
      % unconstrained variable.
      xtrans(i) = x(k);

      k=k+1;
  end
end

end % sub function xtransform end

function Zeta=GeoPotH(Z0,Z)

%Function for Geoptoential Height (Zeta) [m]
%
% Calling:
%  Zeta=GeoPotH(Z0,Z)
%Inputs:
% Reference Height (Z0) [m]
% Height (Z) [m] 1 element matrix
Re=6371e3; %Radius of the earth [m]

Zeta=(Z-Z0)./(((Z-Z0)/Re)+1);

function [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] =
guisdap_param2cell(mat_files,time_limits)
% guisdap_param2cell reading of GUISDAP mat-files,
% extracting time, altitude, electron concentration, electron and
% ion temperatures and ion velocities and their errors.
%
% Calling:
%  [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] =
guisdap_param2cell(mat_files[,time_limits])
% Input:
%  mat_files - list of mat-files as returned from DIR - that is a
%              struct array with fields 'name', 'date', 'bytes',
%              'isdir' and 'datenum', here only 'name' is used, so
%              any struct array with a field 'name' will work.
%  time_limits - [2*n x [yyyy mm dd hh mm ss]] array with start and
%                stop-times of periods of interest - only data from
%                files with names between the start and stop-times
%                will be read.
%
% Output:
%  h   - Altitudes, array with altitudes (km)
%  t   - Time (unix time?), array with observation times [1 x n]
%  ne  - Electron density, array with electron density
%        profiles for each time-step (m^-3)
%  Te  - Electron temperature (K), array with Te altitude profiles
%  Ti  - Ion temperature (K), array with Ti altitude profiles
%  vi  - Ion velocities (m/s) along the beam
%  dne - standard deviation of electron densities
%  dTe - standard deviation of electron temperatures
%  dTi - standard deviation of ion temperatures
%  dvi - standard deviation of ion velocities
%  az  - azimuth angle of radar (degrees)
%  el  - elevation angle of radar (degrees)
%  T   - date and time array [YYYY,MM,DD,hh,mm,ss] [n_time x 6] (UT)
%  ranges - array with ranges (km)
%
% Example,
%   q = dir('/dir/Dir/*.mat');
```

```
%   q = dir('../dir/05*.mat');
%   q = dir('./*.mat');
%   [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T] = guisdap_param2cell(q,...
%                                         [2015 02 16 16 0 0;2015 02 16 17 0 0]);
%
%   subplot(3,1,1)
%   pcolor(rem(t/3600,24),h,log10(max(1e8,ne))),shading flat
%   caxis([9 12])
%   timetick
%   colorbar_labeled('m^{-3}','log','fontsize',12)
%   ylabel('alt (km)')
%   ylabel('')
%   subplot(3,1,2)
%   pcolor(rem(t/3600,24),h,Te),shading flat
%   caxis([500 4500])
%   timetick
%   ylabel('altitude (km)')
%   colorbar_labeled('K','linear','fontsize',12)
%   subplot(3,1,3)
%   pcolor(rem(t/3600,24),h,Ti),shading flat
%   caxis([250 1500])
%   timetick
%   xlabel('Time (UT)')
%   colorbar_labeled('K','linear','fontsize',12)


% Copyright B. Gustavsson 20100527

if nargout
  OK = 0;
end


if nargin > 1 && ~isempty(time_limits)

  StrMat_w_t = char({mat_files.name});
  StrMat_w_t = StrMat_w_t(:,1:end-4);
  data_times = str2num(StrMat_w_t);
  [t_start_stop] = tosecs(time_limits);
  idxInRange = [];
  for iR = 1:2:length(t_start_stop),
    idxInRange = [idxInRange(:);find(t_start_stop(iR)<=data_times & data_times <=
t_start_stop(iR+1))];
  end
  mat_files = mat_files(idxInRange);
end

for i1 = length(mat_files):-1:1,

  load(mat_files(i1).name)
  try
    ne{i1} = r_param(:,1);
    Ti{i1} = r_param(:,2);
    Te{i1} = r_param(:,2).*r_param(:,3);
    vi{i1} = r_param(:,5);
    dne{i1} = r_error(:,1);
    dTi{i1} = r_error(:,2);
    dTe{i1} = r_error(:,3);
    dvi{i1} = r_error(:,5);
    az(i1) = r_az;
    el(i1) = r_el;
  catch
    error(['All files need to have the same altitude resolution of ne'])
  end
  t(i1) = date2unix(r_time(1,:));
  T(i1,:) = r_time(1,:);
  h{i1} = r_h;
  ranges{i1} = r_range;
end
% keyboard


function [unix_time]=date2unix(utc_date)

year = utc_date(1);
month = utc_date(2);
day = utc_date(3);
```

```
hour = utc_date(4);
minute = utc_date(5);
second = utc_date(6);
%Specify day 0
number_of_day_before_day_one=datenum(1970,01,00); %Start time of
                                                  %unix time
absolute_number_of_day=datenum(year,month,day); %Default day one for datenum is january 1st of
year 0
julian_day=absolute_number_of_day - number_of_day_before_day_one; %Number of day since day one

seconds_of_day = 3600*hour+60*minute+second;
unix_time = seconds_of_day + 24*3600*julian_day;

function [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] =
guisdap_param2cell(mat_files,time_limits)
% GUISDAP_PARAM2CELL2REGULAR reading of GUISDAP mat-files,
% extracting time, altitude, electron concentration, electron and
% ion temperatures and ion velocities and their errors.
%
% Calling:
%  [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] =
guisdap_param2cell2regular(mat_files[,time_limits])
% Input:
%  mat_files - list of mat-files as returned from DIR - that is a
%              struct array with fields 'name', 'date', 'bytes',
%              'isdir' and 'datenum', here only 'name' is used, so
%              any struct array with a field 'name' will work.
%  time_limits - [2*n x [yyyy mm dd hh mm ss]] array with start and
%                stop-times of periods of interest - only data from
%                files with names between the start and stop-times
%                will be read.
%
% Output:
%  h   - Altitudes, array with altitudes (km)
%  t   - Time (unix time?), array with observation times [1 x n]
%  ne  - Electron density, array with electron density
%        profiles for each time-step (m^-3)
%  Te  - Electron temperature (K), array with Te altitude profiles
%  Ti  - Ion temperature (K), array with Ti altitude profiles
%  vi  - Ion velocities (m/s) along the beam
%  dne - standard deviation of electron densities
%  dTe - standard deviation of electron temperatures
%  dTi - standard deviation of ion temperatures
%  dvi - standard deviation of ion velocities
%  az  - azimuth angle of radar (degrees)
%  el  - elevation angle of radar (degrees)
%  T   - date and time array [YYYY,MM,DD,hh,mm,ss] [n_time x 6] (UT)
%  ranges - array with ranges (km)
%
% Example,
%   q = dir('/dir/Dir/*.mat');
%   q = dir('../dir/05*.mat');
%   q = dir('./*.mat');
%   [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T] = guisdap_param2cell2regular(q,...
%                                         [2015 02 16 16 0 0;2015 02 16 17 0 0]);
%
%   subplot(3,1,1)
%   pcolor(rem(t/3600,24),h,log10(max(1e8,ne))),shading flat
%   caxis([9 12])
%   timetick
%   colorbar_labeled('m^{-3}','log','fontsize',12)
%   ylabel('alt (km)')
%   ylabel('')
%   subplot(3,1,2)
%   pcolor(rem(t/3600,24),h,Te),shading flat
%   caxis([500 4500])
%   timetick
%   ylabel('altitude (km)')
%   colorbar_labeled('K','linear','fontsize',12)
%   subplot(3,1,3)
%   pcolor(rem(t/3600,24),h,Ti),shading flat
%   caxis([250 1500])
%   timetick
%   xlabel('Time (UT)')
%   colorbar_labeled('K','linear','fontsize',12)
```

```
% Copyright B. Gustavsson 20100527

if nargout
  OK = 0;
end


if nargin > 1 && ~isempty(time_limits)

  StrMat_w_t = char({mat_files.name});
  StrMat_w_t = StrMat_w_t(:,1:end-4);
  data_times = str2num(StrMat_w_t);
  [t_start_stop] = tosecs(time_limits);
  idxInRange = [];
  for iR = 1:2:length(t_start_stop),
    idxInRange = [idxInRange(:);find(t_start_stop(iR)<=data_times & data_times <=
t_start_stop(iR+1))];
  end
  mat_files = mat_files(idxInRange);
end

for i1 = length(mat_files):-1:1,

  load(mat_files(i1).name)
  try
    n_e{i1} = r_param(:,1);
    T_i{i1} = r_param(:,2);
    T_e{i1} = r_param(:,2).*r_param(:,3);
    v_i{i1} = r_param(:,5);
    dn_e{i1} = r_error(:,1);
    dT_i{i1} = r_error(:,2);
    dT_e{i1} = r_error(:,3);
    dv_i{i1} = r_error(:,5);
    az(i1) = r_az;
    el(i1) = r_el;
  catch
    error(['All files need to have the same altitude resolution of ne'])
  end
  t(i1) = date2unix(r_time(1,:));
  T(i1,:) = r_time(1,:);
  h{i1} = r_h;
  range_s{i1} = r_range;
end
% keyboard
for i1 = 1:length(t),
  ne(:,i1)  = interp1(h{i1},n_e{i1},h{1});
  Te(:,i1)  = interp1(h{i1},T_e{i1},h{1});
  Ti(:,i1)  = interp1(h{i1},T_i{i1},h{1});
  vi(:,i1)  = interp1(h{i1},v_i{i1},h{1});
  dne(:,i1) = interp1(h{i1},dn_e{i1},h{1});
  dTe(:,i1) = interp1(h{i1},dT_e{i1},h{1});
  dTi(:,i1) = interp1(h{i1},dT_i{i1},h{1});
  dvi(:,i1) = interp1(h{i1},dv_i{i1},h{1});
  ranges(:,i1) = interp1(h{i1},range_s{i1},h{1});
end

h = h{1};
dTe = dTi.*(Te./Ti);


function [unix_time]=date2unix(utc_date)

year = utc_date(1);
month = utc_date(2);
day = utc_date(3);
hour = utc_date(4);
minute = utc_date(5);
second = utc_date(6);
%Specify day 0
number_of_day_before_day_one=datenum(1970,01,00); %Start time of
                                                  %unix time
absolute_number_of_day=datenum(year,month,day); %Default day one for datenum is january 1st of
year 0
julian_day=absolute_number_of_day - number_of_day_before_day_one; %Number of day since day one

seconds_of_day = 3600*hour+60*minute+second;
unix_time = seconds_of_day + 24*3600*julian_day;
```

```
function He_Z = HeofZeta(Tn0,S,Zeta,Tinf,He_Density,Z0)

%Function for molecular oxygen density as an expression of geopotential HeeigHet
%
%Inputs:
%ExospHeeric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from MSIS Model
%Inverse Scale HeeigHet (S) [1/km]
%Oxygen Density at reference HeeigHet (O2_Density) [#/m^3]

mHe = AMU2kg(4);            %Mass of atomic oxygen [kg]
Re = 6371e3;                 %Radius of tHee eartHe [m]
g0 = 3.99e14/((Z0+Re)^2);    %Gravity at reference HeeigHet [m/s^2]
Kb = 1.38064852e-23;         %Boltzmann Constant [m^2 kg /(s^2 K)]


gamma_p1 = 1+(mHe*g0)/(Kb*Tinf*S);        %1+Gamma Factor
B = (exp(Zeta.*S)-1).*Tinf-Tn0;
C = (Tn0./B).^gamma_p1;

He_Z = He_Density.*C.*exp(Zeta.*S); %Oxygen density vs Geopotential HeeigHet

function H_Z = HofZeta(Tn0,S,Zeta,Tinf,H_Density,Z0)

%Function for molecular oxygen density as an expression of geopotential HeigHt
%
%Inputs:
%Exospheric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from MSIS Model
%Inverse Scale Height (S) [1/km]
%Oxygen Density at reference Height (O2_Density) [#/m^3]

mH = AMU2kg(1);            %Mass of atomic oxygen [kg]
Re = 6371e3;                 %Radius of the earth [m]
g0 = 3.99e14/((Z0+Re)^2);    %Gravity at reference Height [m/s^2]
Kb = 1.38064852e-23;         %Boltzmann Constant [m^2 kg /(s^2 K)]


gamma_p1 = 1+(mH*g0)/(Kb*Tinf*S);        %1+Gamma Factor
B = (exp(Zeta.*S)-1).*Tinf-Tn0;
C = (Tn0./B).^gamma_p1;

H_Z = H_Density.*C.*exp(Zeta.*S); %Oxygen density vs Geopotential Height

function S=InvScaleH(Tinf,Tn0,dTndZ)

%Function for Inverse Scale Height [1/km]

%Inputs
%Exospheric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from ISA table
%Gradient in neutral temp (KS) [K/km] evaluated from reference height

S=dTndZ*(1/(Tinf-Tn0));

function [Pa,Li,t] = JackKnifePlotY(varargin)
% JackKnifePlotY plots jackknife errorbars around a given curve
%
%     [Pa,Li,t] = JackKnifePlotY(x,y,L,U,'r','g')
%     Plots a gray Jackknife around the line displayed in black very useful for
%     funky nature style error bars which are shaded.
%
%         Pa is a patch object for more help on patch objects see below
%         Li is a line object, more help on line object is available in MATLAB
%
%     USAGE :
%               1)    [Pa,Li] = JackKnifePlotY(x,y,E)
%                      Calculates the Lower and upper errorbars as
%                      L = Y-E and U = Y+E. It then takes a default gray color
%                      as patch color, and the line color as black and plots
%                      it around the line using a patch object.
%
%               2)    [Pa,Li] = JackKnifePlotY(x,y,E,LineColor,PatchColor)
%                      Calculates the Lower and upper errorbars as
%                      L = Y-E and U = Y+E. It then takes PatchColor
%                      as patch color, and the Line Color from the LineColor
```

```
%                     variable. It then plots it around the line
%                     using a patch object.
%
%               3)    [Pa,Li] = JackKnifePlotY(x,y,L,U)
%                     User Supplied bounds are taken as L and U, It then takes
%                     a default gray color as patch color, and the line color
%                     as black and plots it around the line using a patch object.
%
%               4)    [Pa,Li] = JackKnifePlotY(x,y,L,U,LineColor,PatchColor)
%                     User Supplied bounds are taken as L and U, It then takes
%                     PatchColor as patch color, and the Line Color from the LineColor
%                     variable. It then plots it around the line using a
%                     patch object.
%       CAVEATS
%                 1) Can be Slow sometimes for length(Array) > 10000,
%                 2) Needs better vectorization
%       EXAMPLE
%                           t = [-5:0.05:5];
%                           Y = sin(t);
%                           E = 0.4*rand(1,length(t));
%                           [Pa,Li] = JackKnifePlotY(t,Y,E);
%                           xlabel('time');
%                           ylabel('Amplitude');
%                           title('Using Errors alone');
%
%                           figure;
%                           L = Y - E;
%                           U = Y + E;
%                           [Pa,Li] = JackKnifePlotY(t,Y,L,U);
%                           xlabel('time');
%                           ylabel('Amplitude');
%                           title('Using Lower and Upper Confidence Intervals');
%                           hold on;
%
%                           Y1 = 2*Y;
%                           L = Y1 - 0.2;
%                           U = Y1 + 0.2;
%                           [Pa,Li] = JackKnifePlotY(t,Y1,L,U,[255 51 51]./255,[255 153
102]./255);
%                           hold on;
%                           [Pa,Li] = JackKnifePlotY(t,Y1*2,E,[51 51 153]./255,[102 153
204]./255);
%                           [Pa,Li] = JackKnifePlotY(t,Y1*2,E,'r','g');
%
% See also ERRORBAR, PATCH, LINE
%
%
% Version 0.001 Chandramouli Chandrasekaran (Chandt) - 13 April 2006.


switch(nargin)
 case 3,
  % If there are 3 inputs it means its just the errors
  x = varargin{1};
  y = varargin{2};
  E = varargin{3};
  L = x - E;
  U = x + E;
  LineColor = 'k';
  PatchColor = [0.85 0 0];
 case 4,
  % If there are 4 inputs it means they entered the Lower and upper
  % bounds
  x = varargin{1};
  y = varargin{2};
  L = varargin{3};
  U = varargin{4};
  LineColor = 'k';
  PatchColor = [0 0 0.85];
 case 5,
  x = varargin{1};
  y = varargin{2};
  E = varargin{3};
  L = x - E;
  U = x + E;
  LineColor =  varargin{4};
  PatchColor = varargin{5};
```

```
 case 6,
  % If there are 6 inputs then
  x = varargin{1};
  y = varargin{2};
  L = varargin{3};
  U = varargin{4};
  LineColor =  varargin{5};
  PatchColor = varargin{6};
end
tic;


  iL = find(~isfinite(L));
  iU = find(~isfinite(U));
  if ~isempty(iL) | ~isempty(iL)
    iNan = unique([iL,iU]);
    i2plot = [[1,min(iNan+1,length(y))];[max(iNan-1,1),length(y)]];

    for i1 = 1:length(i2plot)

      Ycoords = [y(i2plot(1,i1):i2plot(2,i1)); y(i2plot(2,i1):-1:i2plot(1,i1))];
      Xcoords = [U(i2plot(1,i1):i2plot(2,i1)); L(i2plot(2,i1):-1:i2plot(1,i1))];

      Pa = patch(Xcoords,Ycoords,PatchColor);
      set(Pa,'linestyle','none','linewidth',2);
      hold on;
      Li =
plot(x(i2plot(1,i1):i2plot(2,i1)),y(i2plot(1,i1):i2plot(2,i1)),'color',LineColor,'linewidth',2
);
      hold on;

    end
  else

    Ycoords = [y; y(end:-1:1)];
    Xcoords = [U; L(end:-1:1)];
    Pa = patch(Xcoords,Ycoords,PatchColor);
    set(Pa,'linestyle','none','linewidth',2);
    hold on;
    Li = plot(x,y,'color',LineColor,'linewidth',2);
    hold on;

  end
  t = toc;

function [x,y,z] = llh_to_local_coord(lat0,long0,alt0,lat,long,alt)
% LLH_TO_LOCAL_COORD transforms the positions (LAT, LONG, ALT) to (X,Y,Z)
% in a Cartesian coordinate system x || east , y || north, z ||
% zenit centered in (lat0,long0) at altitude ALT0 above sea level.
%
% CALLING:
% [x,y,z] = llh_to_local_coord(lat0,long0,alt0,lat,long,alt)
%
% INPUT:
%    LAT0 latitude of reference point (origin of coordinates), in degrees
%    LONG0 longiture,of reference point (origin of coordinates) in degrees
%    ALT0 altitude,  of reference point (origin of coordinates) in km
%    LAT latitude, in degrees
%    LONG longiture, in degrees
%    ALT altitude, in km
% OUTPUT:
%  X - Horisontal east distance from (lat0,long0,alt0) (km)
%  Y - Horisontal north distance from (lat0,long0,alt0) (km)
%  Z - Horisontal altitude above (lat0,long0,alt0) (km)


%   Copyright ï¿½ 19970907 Bjorn Gustavsson, <bjorn.gustavsson@irf.se>
%   This is free software, licensed under GNU GPL version 2 or later
% f = 1/298.257;
% e = (2*f-f*f).^.5;
% a = 6378.137;


phi0 = pi/180*lat0*ones(size(lat));
Lambda0 = pi/180*long0*ones(size(long));
phi = pi/180*lat;
Lambda = pi/180*long;
```

```
[e1,e2,e3] = e_local(lat0,long0,0);

ro(1,:) = XX(phi0,Lambda0,alt0);
ro(2,:) = YY(phi0,Lambda0,alt0);
ro(3,:) = ZZ(phi0,Lambda0,alt0);

r(1,:) = XX(phi,Lambda,alt);
r(2,:) = YY(phi,Lambda,alt);
r(3,:) = ZZ(phi,Lambda,alt);

r_ro = r - ro;

x = dot(r_ro,e1);
y = dot(r_ro,e2);
z = dot(r_ro,e3);

function [trmtr] = maketransfmtr(lat0,long0,lat,long,already_degrees)
% MAKETRANSFMTR - the transformation rotation matrixes
% from the local coordinate system in LAT, LONG to the local
% coordinate system in LAT0, LONG0,
% OBS! default Input in radians!!!
%
% CALLING:
% [trmtr] = maketransfmtr(lat0,long0,lat,long)
% [trmtr] = maketransfmtr(lat0,long0,lat,long,already_degrees)
%
% INPUT:
%    LAT0 latitude of reference point (origin of coordinates)
%    LONG0 longiture,of reference point (origin of coordinates)
%    LAT latitude,
%    LONG longiture,
%    DEGS_2_RADIANS - flag to switch to conversion from input in
%                     degrees
% ANGLES in _RADIANS BY DEFAULT!!!

%   Copyright ï¿½ 20000323 Bjorn Gustavsson, <bjorn.gustavsson@irf.se>
%   This is free software, licensed under GNU GPL version 2 or later

if nargin>4 && already_degrees
  % Do no conversion from radians to degrees!
  %phi0 = lat0;
  %Lambda0 = long0;
  %phi = lat;
  %Lambda = long;
else
  lat0 = lat0*180/pi;
  long0 = long0*180/pi;
  lat = lat*180/pi;
  long = long*180/pi;

  %phi0 = lat0;
  %Lambda0 = long0;
  %phi = lat;
  %Lambda = long;
end

[e1,e2,e3]    = e_local(lat0,long0,0);
[e1p,e2p,e3p] = e_local(lat,long,0);

trmtr(1,1) = dot(e1p,e1);
trmtr(1,2) = dot(e2p,e1);
trmtr(1,3) = dot(e3p,e1);
trmtr(2,1) = dot(e1p,e2);
trmtr(2,2) = dot(e2p,e2);
trmtr(2,3) = dot(e3p,e2);
trmtr(3,1) = dot(e1p,e3);
trmtr(3,2) = dot(e2p,e3);
trmtr(3,3) = dot(e3p,e3);

function [X,STDX,MSE,S] = myLSCOV(A,B,V)
%MYLSCOV - lscov Least squares with known covariance.
%  X = lscov(A,B,V) returns the ordinary least squares solution to the
%  linear system of equations A*X = B, i.e., X is the N-by-1 vector that
%  minimizes the sum of squared errors (B - A*X)'*inv(V)*(B - A*X),
%  where A is M-by-N, and B is M-by-1.
```

```
%
%  X = lscov(A,B,V), where V is an M-by-M symmetric (or hermitian) positive
%  definite matrix, returns the generalized least squares solution to the
%  linear system A*X = B with covariance matrix proportional to V, i.e., X
%  minimizes (B - A*X)'*inv(V)*(B - A*X).
%
%  X = lscov(A,B,V), where V is a vector length M of real positive weights,
%  returns the weighted least squares solution to the linear system A*X =
%  B, i.e., X minimizes (B - A*X)'*inv(diag(V))*(B - A*X).
%
%  [X,STDX] = lscov(...) returns the estimated standard errors of X.  When
%  A is rank deficient, STDX contains zeros in the elements corresponding
%  to the necessarily zero elements of X. (not tested here! TODO: check)
%
%  [X,STDX,MSE] = lscov(...) returns the mean squared error.  If B is assumed
%  to have covariance matrix SIGMA^2 * V (or SIGMA^2 * DIAG(1./W)), then MSE
%  is an estimate of SIGMA^2.
%
%  [X,STDX,MSE,S] = lscov(...) returns the estimated covariance matrix
%  of X.  When A is rank deficient, S contains zeros in the rows and
%  columns corresponding to the necessarily zero elements of X.  lscov
%  cannot return S if it is called with multiple right-hand sides (i.e.,
%  size(B,2) > 1).
%
%  The standard formulas for these quantities, when A and V are full rank,
%  are:
%
%     X = inv(A'*inv(V)*A)*A'*inv(V)*B
%     MSE = B'*(inv(V) - inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))*B./(M-N)
%     S = inv(A'*inv(V)*A)*MSE
%     STDX = sqrt(diag(S))
%
%  lscov assumes that the covariance matrix of B is known only up to a
%  scale factor.  MSE is an estimate of that unknown scale factor, and
%  lscov scales the outputs S and STDX appropriately.  However, if V is
%  known to be exactly the covariance matrix of B, then that scaling is
%  unnecessary.  To get the appropriate estimates in this case, you should
%  rescale S and STDX by 1/MSE and sqrt(1/MSE), respectively.
%
%  Class support for inputs A,B,V,W:
%     float: double, singleSee also

if min(size(V)) == 1
  V = diag(V);
end
X = inv(A'*inv(V)*A)*A'*inv(V)*B;
MSE = B'*(inv(V) - inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))*B;
S = inv(A'*inv(V)*A);
STDX = sqrt(diag(S));

function N2_Z = N2ofZeta(Tn0,S,Zeta,Tinf,N2_Density,Z0)

%Function for molecular oxygen density as an expression of geopotential height
%
%Inputs:
%Exospheric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from MSIS Model
%Inverse Scale Height (S) [1/km]
%Oxygen Density at reference height (O2_Density) [#/m^3]

mN2 = AMU2kg(28);            %Mass of atomic oxygen [kg]
Re = 6371e3;                 %Radius of the earth [m]
g0 = 3.99e14/((Z0+Re)^2);    %Gravity at reference height [m/s^2]
Kb = 1.38064852e-23;         %Boltzmann Constant [m^2 kg /(s^2 K)]


gamma_p1 = 1+(mN2*g0)/(Kb*Tinf*S);        %1+Gamma Factor
B = (exp(Zeta.*S)-1).*Tinf-Tn0;
C = (Tn0./B).^gamma_p1;

N2_Z = N2_Density.*C.*exp(Zeta.*S); %Oxygen density vs Geopotential Height

function O2_Z = O2ofZeta(Tn0,S,Zeta,Tinf,O2_Density,Z0)

%Function for molecular oxygen density as an expression of geopotential height
%
%Inputs:
```

```
%Exospheric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from MSIS Model
%Inverse Scale Height (S) [1/km]
%Oxygen Density at reference height (O2_Density) [#/m^3]

mO2 = AMU2kg(32);                  %Mass of atomic oxygen [kg]
Re = 6371e3;                       %Radius of the earth [m]
g0 = 3.99e14/((Z0+Re)^2);    %Gravity at reference height [m/s^2]
Kb = 1.38064852e-23;               %Boltzmann Constant [m^2 kg /(s^2 K)]


gamma_p1 = 1+(mO2*g0)/(Kb*Tinf*S);        %1+Gamma Factor
B = (exp(Zeta.*S)-1).*Tinf-Tn0;
C = (Tn0./B).^gamma_p1;

O2_Z = O2_Density.*C.*exp(Zeta.*S); %Oxygen density vs Geopotential Height

function O_Z=OofZeta(Tn0,S,Zeta,Tinf,O_Density,Z0)

%Function for atomic oxygen density as an expression of geopotential height

%Inputs:
%Exospheric temperature (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from MSIS Model
%Inverse Scale Height (S) [1/km]
%Oxygen Density at reference height (O_Density) [#/m^3]

mO=AMU2kg(16);                  %Mass of atomic oxygen [kg]
Re=6371e3;                      %Radius of the earth [m]
g0= 3.99e14/((Z0+Re)^2);        %Gravity at reference height [m/s^2]
Kb=1.38064852e-23;              %Boltzmann Constant [m^2 kg /(s^2 K)]


gamma_p1 = 1+(mO*g0)/(Kb*Tinf*S);        %1+Gamma Factor
B=(exp(Zeta.*S)-1).*Tinf-Tn0;
C=(Tn0./B).^gamma_p1;

O_Z=O_Density.*C.*exp(Zeta.*S); %Oxygen density vs Geopotential Height

function Q_e2OP=Q_e2Oion(ne,Te,Ti,PHp)

%Function for heat transfer between electrons and O+
%


Kb = 1.38064852e-23; %Boltzman's Constant [m^2 kg/s^2*K]
e0 = 8.85418782e-12; %Permittivity of free space [s^4 A^2/m^3 kg]
q = 1.60217662e-19;  %Charge of electron [C]
y = 0.57722;         %Eular's Constant


L_Debye = sqrt((e0*Kb*Te)./(ne*q^2));               %DeBye Length [m]
ln_LAMBDA = log(16*pi()*ne.*L_Debye.^3/y^2)-0.5;    %The Coulomb Logarithm

A = ln_LAMBDA*3.197e-8/1e6;                         %Constant [eVK^0.5 m/s]

Q_e2OP = (1-PHp).*A.*ne.^2.*(Te-Ti).*(Te.^(-3/2));   %Energy Transfer Rate [eV/cm^3*s]

function QO_H = Q_inel(ne,HDen,Ti,Tn,ODen,PHp)
%Function for determining inelastic heat exchange between oxygen ions and
%Atomic Hydrogen
%From Bailey, 1983; Raitt et al., 1975

C = 3.8e-15/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/K*S [eV m^3/K*S]

QO_H = C*ne.*(HDen.*(1-PHp).*Ti.*sqrt(Tn)-8/9*ODen.*PHp);


function Q_ion2N = Q_ion2N(Trein,profile)

Q_Hp2H = Hp2H(Trein,profile);
Q_Hp2N2 = Hp2N2(Trein,profile);
Q_Hp2O = Hp2O(Trein,profile);
Q_Hp2O2 = Hp2O2(Trein,profile);
Q_NOp2N2 = NOp2N2(Trein,profile);
Q_NOp2O = NOp2O(Trein,profile);
Q_NOp2O2 = NOp2O2(Trein,profile);
```

```
Q_O2p2N2 = O2p2N2(Trein,profile);
Q_O2p2O = O2p2O(Trein,profile);
Q_O2p2O2 = O2p2O2(Trein,profile);
Q_Op2H = Op2H(Trein,profile);
Q_Op2He = Op2He(Trein,profile);
Q_Op2N2 = Op2N2(Trein,profile);
Q_Op2O = Op2O(Trein,profile);
Q_Op2O2 = Op2O2(Trein,profile);

Q_ion2N = Q_Hp2H + Q_Hp2N2 + Q_Hp2O + Q_Hp2O2+ Q_NOp2N2 + Q_NOp2O...
+ Q_NOp2O2 + Q_O2p2N2 + Q_O2p2O + Q_O2p2O2 + Q_Op2H + Q_Op2He...
+ Q_Op2N2 + Q_Op2O + Q_Op2O2;
% Looking at units of constant within functions, these provide an output in
% [eV/cm^3*s]

function QOP_H = Q_Oion2H(HDen,ne,PHp,Ti,Tn)
%Function for determining inelastic heat exchange between oxygen ions and
%Molecular Nitrogen
%From Banks, 1966; Banks and Kockarts, 1973; Bailey and Selleck, 1990;
%Schunk and Nagy, 2000


C = 3.3e-14/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/K*S [eV m^3/K*S]


QOP_H = C*HDen.*ne.*(1-PHp).*(Ti-Tn);

function QOP_He = Q_Oion2He(HeDen,ne,PHp,Ti,Tn)
%Function for determining inelastic heat exchange between oxygen ions and
%Molecular Nitrogen
%From Banks, 1966; Banks and Kockarts, 1973; Bailey and Selleck, 1990;
%Schunk and Nagy, 2000


C = 2.8e-14/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/K*S [eV m^3/K*S]


QOP_He = C*HeDen.*ne.*(1-PHp).*(Ti-Tn);

function QOP_N2 = Q_Oion2N2(N2Den,ne,PHp,Ti,Tn)
%Function for determining inelastic heat exchange between oxygen ions and
%Molecular Nitrogen
%From Banks, 1966; Banks and Kockarts, 1973; Bailey and Selleck, 1990;
%Schunk and Nagy, 2000


C = 6.6e-14/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/K*S [eV m^3/K*S]


QOP_N2 = C*N2Den.*ne.*(1-PHp).*(Ti-Tn);

function QOP_O = Q_Oion2O(ODen,ne,PHp,Ti,Tn)
%Function for determining inelastic heat exchange between oxygen ions and
%Atomic oxygen
%From Banks, 1966; Banks and Kockarts, 1973; Bailey and Selleck, 1990;
%Schunk and Nagy, 2000


C = 2.1e-15/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/(S*K^(3/2)) [eV
m^3/(S*K^(3/2))]
F = 1;

QOP_O = C*F*ODen.*ne.*(1-PHp).*sqrt(Ti+Tn).*(Ti-Tn);

function QOP_O2 = Q_Oion2O2(O2Den,ne,PHp,Ti,Tn)
%Function for determining inelastic heat exchange between oxygen ions and
%Molecular Nitrogen
%From Banks, 1966; Banks and Kockarts, 1973; Bailey and Selleck, 1990;
%Schunk and Nagy, 2000


C = 2.1e-15/1e6; %Constant given in Nicholls (2006) originally in eV cm^3/K*S [eV m^3/K*S]


QOP_O2 = C*O2Den.*ne.*(1-PHp).*(Ti-Tn);
```

```
function [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = readGdata(data_path)

%Script to convert Guisdap files to readable data

%Adding file paths to matlab directory
addpath(data_path);

%Getting Stuc format of file names
wpwd = pwd;
cd(data_path)
Files=dir('*.mat');

%Using Bjorn G. function to convert files to readable data
[h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = guisdap_param2cell2regular(Files);
cd(wpwd)


function [h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = readGdata2(data_path)

%Script to convert Guisdap files to readable data

%Adding file paths to matlab directory
addpath(data_path);

%Getting Stuc format of file names
wpwd = pwd;
cd(data_path)
Files=dir('*.mat');

%Using Bjorn G. function to convert files to readable data
[h,t,ne,Te,Ti,vi,dne,dTe,dTi,dvi,az,el,T,ranges] = guisdap_param2cell(Files);
cd(wpwd)

function Tinf=TExosphere(F10_7)

%Function to find exospheric temperature
%F10.7 value for 15/11/17 from http://ccmc.gsfc.nasa.gov/modelweb/models/msis_vitmo.php

Tinf=500+(3.4*F10_7); %[K]

function timetick(ax,precision)
% TIMETICK    Time formatted tick labels
%
% TIMETICK sets current axes to time formatted tick labels.
% X Axis must be time, in datenum format
%
% TIMETICK(AX) sets axes with handle AX to time formatted tick labels
%
% TIMETICK(AX,precision) sets axes with handle AX to time formatted tick
% labels with precision values to the right of the decimal for seconds
% specifier

% Michelle Hirsch
% The MathWorks
% Copyright 2003-2014 The MathWorks, Inc

% Default - use GCA
if nargin==0 || isempty(ax) || ax ==0
    ax = gca;
end;

if nargin<2
    precision = 4;
end;

% Get tick values
xt = get(ax,'XTick');

% Convert to time string
xtl = timestr(xt,precision);

% Set ticks to the time string
set(ax,'XTickLabel',xtl);

function S = timestr(D,precision)
%timestr          String representation of time.   HH:MM:SS.SSSS
%
```

```
% TIMESTR(D) converts D, a serial date number (as returned by DATENUM)
% into a time string with the format HH:MM:SS.SSSS
%
% TIMESTR(D,precision) uses precision values to the right of the decimal

if nargin==1
    precision=4;
end;

if precision > 0
    totalwidth = precision + 3;   %2 to left of decimal, plus decimal
else
    totalwidth = 2;               %2 to left of decimal (decimal won't display)

end
precision = num2str(precision);
totalwidth = num2str(totalwidth);

D = D(:);

% Obtain components of date number
[y,mo,d,h,min,s] = datevecmx(D,1.1);  mo(mo==0) = 1;

% Generate formatted string
% sw = floor(s);      %Whole
% sf = floor((s-sw)*1000);          %Fraction
% M = [h';min';sw';sf'];       %sprintf works columnwise
% fmt = '%02d:%02d:%02d.%04d';
M = [h';min';s'];       %sprintf works columnwise

% Figure out how long to make seconds format.
% Since we are building a string array, every element must be the same
% length
sw = floor(s);      %Whole
sf = floor((s-sw)*1000);          %Fraction

fmt = ['%02d:%02d:%0' totalwidth '.' precision 'f'];
S = sprintf(fmt,M(:,1));
for ii=2:length(D)
    t= sprintf(fmt,M(:,ii));
    S = [S;t];
end;

% My formatting is a bit messed up.  I can't figure out how to add
% zeros when necessary for seconds - I end up with blanks.
% Replace blanks with 0
% blnks = find(double(S)==32);        % 32 - ASCII for 0
% S(blnks) = '0';

function Tn=TofZeta(Tn0,Tinf,S,Zeta)


%Function for neutral temp profile for geopotential height (Zeta)

%Inputs:
%Scale Height (S) 1 Element matrix,
%Geopotential height (Zeta)1 Element Matrix
%Exospheric Temp (Tinf) [K]
%Neutral Temp at reference altitude (Tn0) [K] from ISA table

Tn=Tinf-(Tinf-Tn0)*exp(-S.*Zeta); %[K]

function [v_ion,tr_wind,varargout] = wind_tr(vT,vS,vK,azT,elT,azS,elS,azK,elK,dvT,dvS,dvK)
% WIND_TR - transform tristatic EISCAT drifts to cartesian winds
% WIND_TR takes EISCAT tristatic bisector velocities and transforms
% them to ion drift in cartesian coordinates.
%
% Calling:
%  [V_ion,tr_wind,stdV_ion,mseV_ion] = wind_tr(vT,vS,vK,azT,elT,azS,elS,azK,elK,dvT,dvS,dvK)
% Input:
% vT  - Ion velocities at the tristatic intersection altitude
%       [1 x n] (m/s) (positive up) from EISCAT Tromsø, or a cell
%       array {v_i,alt} where v_i [n_alts x n_times] is the
%       altitude-time variation of the ion velocity, and alt is the
%       corresponding altitudes [n_alts x 1] (km), in the latter
%       case an automated triangulation of the best altitude is
%       done, and the corresponding ion velocities is
```

```
%       extra/inter-polated.
% vS  - Ion velocities from EISCAT Sodankylæ [1 x n_times] (m/s)
%       (positive up)
% vK  - Ion velocities from EISCAT Kiruna [1 x n_times] (m/s)
%       (positive up)
% azT - Tromsø azimuth [1 x 1] (degrees), optional defaults to -174.89
% elT - Tromsø elevation [1 x 1] (degrees), optional defaults to 77.5
% azS - Sodankulæ azimuth [1 x 1] (degrees), optional defaults to 1
% elS - Sodankylæ elevation [1 x 1] (degrees), optional defaults to 1
% azK - Kiruna azimuth [1 x 1] (degrees), optional defaults to 1
% elK - Kiruna elevation [1 x 1] (degrees), optional defaults to 1
% dvT - Wind errors from Tromsø, should be standard deviations, should be
%       same size as the wind array part of vT
% dvS - Wind errors (standard deviations) from Sodankyla, should be sized as vS
% dvK - Wind errors (standard deviations) from Sodankyla, should be sized as vT
%
% Output:
% v_ion - [3 x n_times] array of wind vector, in East, North and zenith
%           direction.
% tr_wind - [3 x 3] array that can be used to get v_ion, by just
%           multiplying tr_wind with [vT, vS, vK].
% stdV_ion - [3 x n_times] array of wind vector standard deviation,
%            in East, North and zenith direction.
% Adapted after original from T Sergienko.

% Position of EISCAT ISR
Lat_tr =  69.583;
Long_tr = 19.21;
Alt_tr = 0.03;

% Position of Sodanylæ recieving station
Lat_so =  67.367;
Long_so = 26.65;
Alt_so = 0.18;

% Position of Sodanylæ recieving station
Lat_ki =  67.863;
Long_ki = 20.44;
Alt_ki = 0.412;

% Get the antenna pointing positions for Tromsø
if nargin < 4 | isempty(azT)
  Az_tr = -174.89/180*pi; % Default, should be a warning!
else
  Az_tr = azT*pi/180;
end
if nargin < 5 | isempty(elT)
  El_tr =  77.50/180*pi; % Default, should be a warning!
else
  El_tr = elT*pi/180; % TODO: check that this gets handled right
                      % even with the wrong name!
end

% Get the antenna pointing positions for Sodankylæ
if nargin < 6 | isempty(azT)
  Az_so = -53.62/180*pi; % Default, should be a warning!
else
  Az_so = azS*pi/180;
end
if nargin < 7 | isempty(elT)
  El_so =  28.66/180*pi; % Default, should be a warning!
else
  El_so = elS*pi/180; % TODO: check that this gets handled right
                      % even with the wrong name!
end

% Get the antenna pointing positions for Kiruna
if nargin < 6 | isempty(azT)
  Az_ki = -19.64/180*pi; % Default, should be a warning!
else
  Az_ki = azK*pi/180;
end
if nargin < 7 | isempty(elT)
  El_ki =  53.56/180*pi; % Default, should be a warning!
else
  El_ki = elK*pi/180; % TODO: check that this gets handled right
                      % even with the wrong name!
```

```
end


% Tromso - coordinate center [0,0,0]
% This gives the position of the Sod and Kir stations in a
% horizontal coordinate system centred in Ramfjord with x to the
% local geographic east.
[x_so,y_so,z_so] = llh_to_local_coord(Lat_tr,Long_tr,Alt_tr,Lat_so,Long_so,Alt_so);
[x_ki,y_ki,z_ki] = llh_to_local_coord(Lat_tr,Long_tr,Alt_tr,Lat_ki,Long_ki,Alt_ki);
% And the rotation matrices from local horisontal systems in SOD
% and KIR.
[trmtr_so_tr]  =  maketransfmtr(Lat_tr/180*pi,Long_tr/180*pi,Lat_so/180*pi,Long_so/180*pi);
[trmtr_ki_tr]  =  maketransfmtr(Lat_tr/180*pi,Long_tr/180*pi,Lat_ki/180*pi,Long_ki/180*pi);

% TRO line-of-sight vector
ex_tr = sin(Az_tr)*cos(El_tr);
ey_tr = cos(Az_tr)*cos(El_tr);
ez_tr = sin(El_tr);

eT = [ex_tr,ey_tr,ez_tr];
% SOD line-of-sight vector, in SOD horisontal coordinates
ex_so = sin(Az_so)*cos(El_so);
ey_so = cos(Az_so)*cos(El_so);
ez_so = sin(El_so);
% Rotate it to Tromsø horisontal coordinates
y1 = trmtr_so_tr*[ex_so ey_so ez_so]';
eS = y1';

% Tristatic ISR measures the bisector ion velocities, so here we
% calulate the unit vectors of the TRO-SOD bisector
% 1 add e_Tro to e_Sod
ex_so = y1(1)+ex_tr;
ey_so = y1(2)+ey_tr;
ez_so = y1(3)+ez_tr;
% 2 normalize
mod = sqrt(ex_so^2+ey_so^2+ez_so^2);
ex_so = ex_so/mod;
ey_so = ey_so/mod;
ez_so = ez_so/mod;

% KIR line-of-sight vector - same as for SOD
ex_ki = sin(Az_ki)*cos(El_ki);
ey_ki = cos(Az_ki)*cos(El_ki);
ez_ki = sin(El_ki);
x1 = trmtr_ki_tr*[ex_ki ey_ki ez_ki]';
eK = x1';

ex_ki = x1(1)+ex_tr;
ey_ki = x1(2)+ey_tr;
ez_ki = x1(3)+ez_tr;
mod = sqrt(ex_ki^2+ey_ki^2+ez_ki^2);
ex_ki = ex_ki/mod;
ey_ki = ey_ki/mod;
ez_ki = ez_ki/mod;

e_Tro  = [ex_tr, ey_tr, ez_tr];
e_bSod = [ex_so, ey_so, ez_so];
e_bKir = [ex_ki, ey_ki, ez_ki];

% The radar ion-velocity observations are just the scalar product
% between the 3-D v_ion and e_bisector, that is
% v_Tro  = e_Tro  * v_ion,
% v_bSod = e_bSod * v_ion,
% v_bKir = e_bKir * v_ion,
%
% This 3-by-3 system of equations should be straightforward to
% solve with in matrix notation:
%
% V_obs = M * V_ion -> V_ion = M\V_obs
M = [e_Tro;
     e_bSod;
     e_bKir];


tr_wind = M^-1;

% If we haven't calculated the intersection altitude and extracted
```

```
% the corresponding ion velocities we can allways do it
% automatically here:
if iscell(vT)
  [r1,l,mindiff] = stereoscopic([0,0,0],eT,[x_so,y_so,z_so],eS);
  [r2,l,mindiff] = stereoscopic([0,0,0],eT,[x_ki,y_ki,z_ki],eK);
  best_alt = r1(3)/2+r2(3)/2;
  alts = vT{2};
  v_i = inpaint_nans(vT{1});
  v_T = interp1(alts,v_i,best_alt);
end

if nargin < 10
  % Without the ion drift errors just use direct linear least
  % squares to determine the ion drift.
  v_ion = M\[v_T',vS',vK']';
  if nargout > 2
    varargout{1} = [];
  end
  if nargout > 3
    varargout{2} = [];
  end
else
  % Matlab has a function lscov that can take the variance of the
  % observations and make a proper solution taking that into account

  % First determine where we have nans in the v, the corresponding
  % estimates we will then give infinite variance, so their impact
  % on the solution will be removed.
  %##  TODO: those times should also be given for output so we can
  %##        keep track of where the solution is crap!
  % inanS = find(isnan(vS));
  % inanK = find(isnan(vK));
  if ~iscell(vT)
    % Then it should be an array so we can just search
    % for the nans directly
    inanT = find(isnan(vT));
  else
    % We have to interpolate just as before...
    inanT = find(isnan(interp1(alts,vT{1},best_alt)));
    % ...and dvT should also be an [n_alts x n_times] array
    % so we have to interpolate for the dvT at the tristatic
    % altitude:
    dv_T = interp1(alts,dvT,best_alt);
    dv_T = interp1(alts,dvT,best_alt).*(~isnan(interp1(alts,dvT,best_alt)));
    dv_T(isnan(dv_T)) = interp1(alts,dvT,best_alt(isnan(dv_T)),'nearest');
  end
  % Set the variances(stanard deviation???) to infinity for when
  % there is nans in the wind measurements!
  dv_T(inanT) = inf;
  dv_S(isnan(vS)) = inf;
  dv_K(isnan(vK)) = inf;
  % ...and the errors.
  dv_T(isnan(dv_T)) = inf;
  dv_S(isnan(dv_S)) = inf;
  dv_K(isnan(dv_K)) = inf;

  % Should be possible to vectorize this, but loops for now.
  for i1 = 1:length(vS)
    % lscov wants the
    w = [1./dvT(i1)^2,1./dvS(i1)^2,1./dvK(i1)^2];
    %[v_ion(i1,:),stdV_ion(i1,:),mseV_ion(i1,:)] = lscov(M,[v_T(i1),vS(i1),vK(i1)]',w);
    V = diag(1./w);
    [v_ion(i1,:),stdV_ion(i1,:),mseV_ion(i1,:)] = myLSCOV(M,[vT(i1),vS(i1),vK(i1)]',V);
    %keyboard
  end
  if nargout > 2
    varargout{1} = stdV_ion;
  end
  if nargout > 3
    varargout{2} = mseV_ion;
  end
end
```

11 REFERENCES

Aikio, A., Cai, L. & Nygrén, T., 2012. Statistical Distribution of Height-integrated Energy Exchange Rates. *J. Geophys. Res.,* Volume 117, p. A10325.

Akasofu, N. & Champan, S., 1972. *Solar-Terrestrial Physics.* Oxford: Clarendon Press.

Baily, G., 1983. The Effect of a Meridial E x B Drift on the Thermal Plasma at L = 1.4. *Planet. Space. Sci.,* 31(4), pp. 389-409.

Banks, P., 1966a. Collisional Frequecies and Energy Transfer: Electrons. *Planet. Space Sci.,* 14(11), pp. 1085-1103.

Banks, P., 1966b. Collisional Frequecies and Energy Transfer: Ions. *Planet. Space Sci.,* 14(11), pp. 1105-1122.

Bates, D., 1959. Some Problems Concerning the Terrestrial Atmosphere Above About the 100km level. *Proc. R. Soc. London,* 253(1275), pp. 451-462.

Beynon, W. & Williams, E., 1970. Electron density in the lower ionosphere and the winter anomaly in HF absorption. *J. Atmos. Terr. Phys.,* 32(7), pp. 1325-1329.

Brekke, A., 2013. *Physics of the Upper Polar Atmosphere.* Berlin Heidelberg: SPRINGER.

Brekke, A. & Rino, C., 1978. High-resolution altitude profiles of the. *J. Geophys.,* Volume 83, pp. 2517-2524.

Burnside, R., Teply, C. & Wickwar, V., 1987. The O(+)-O collision cross-section: Can it be inferred from aeronomical measurements?. *Ann. Geophys.,* 5(6), pp. 343-350.

Evans, J., 1969. Theory and practice of ionosphere study by Thompson Scatter radar. *Proceedings of the IEEE,* 57(4), pp. 496-530.

Fuji, R., Nozawa, S., Matuura, N. & Brekke, A., 1999. Statistical Characteristics of Electromagnetic Energy Transfer between the magnetosphere, the Ionosphere and the Thermosphere. *J. Geophys. Res.,* Volume 104, pp. 2357-2365.

Gordon, W., 1958. Incoherent Scattering of Radio Waves by Free Electrons with Applications to Space Exploration by Radar. *Proceedings of the IRE,* 46(11), pp. 1824-1829.

Hedin, A., 1991. Extension of the MSIS Thermosphere Model into the middle and lower atmosphere. *J. geophs. Res.,* 96(A2), pp. 1159-1172.

Itikawa, Y., 1975. Electron-ion Energy Transfer Rate. *J. Atmos. Terr. Phys.,* 37(12), pp. 1601-1602.

Kazil, J., Kopp, E., Chabrillat, S. & Bishop, J., 2003. The University of Bern Atmospheric Ion Model: Time-dependent modeling of the ions in the mesosphere and lower thermosphere. *J. Geophys. Res.,* 108(D14), p. 4432.

King, J. & Kohl, H., 1965. Upper Atmospheric Winds and Ionospheric Drifts Caused by Neutral Air Pressure Gradients. *Nature,* 206(4985), pp. 699-701.

Nicolls, M. et al., 2006. Daytime F region ion energy balance at Arecibo for moderate to high solar flux conditions. *J. Geophys. Res.,* 111(A10), p. A10307.

Oliver, W. & Glotfelty, K., 1996. O(+)-O collision cross section and long-term F-region O density variations deduced from the ionosphere energy budget. *J. Geophys. Res.,* 101(A10), pp. 21,769-21,784.

Ridley, A., 2007. Effects of seasonal changes in the ionosphericconductances on magnetospheric field-aligned currents. *Geophys. Res. Lett.,* Volume 34, p. L05101.

Roble, R., 1975. The calculated and observed diurnal variation of the ionosphere over Millstone Hill on 23–24 March 1970. *Planet. Space Sci.,* 23(7), pp. 1017-1033.

Salah, J., 1993. Interim standard for the ion-neutral atomic oxygen collision frequency. *Geophys. Res. Lett.,* 20(15), pp. 1543-1546.

Schunk, R., 1975. Transport equations for aeronomy. *Planet. Space Sci.,* 23(3), pp. 437-485.

Schunk, R. & Nagy, A., 2009. *Ionospheres: Physics, Plasma Physics, and Chemistry.* New York: Cambridge Univ..

Schunk, R. & Walker, J., 1973. Theoretical ion densities in the lower ionosphere. *Planet. Space Sci.,* 21(11), pp. 1875-1896.

Strømme, A., 2005. *EISCAT incoherent scatter radar school: Ionospheric and magnetospheric studies with incoherent scatter radars.* Sodankylä, Sodankylä Geophysical Observatory.

Sulzer, M. & Gonzalez, S., 1996. Simultaneous measurements of 0 + and H+temperatures. *Geophys. Res. Lett.,* 23(22), pp. 3235-3238.

Thayer, J., 1998. Height-resolved Joule Heating Rates in the High-latitude E Region and the Influence of Neutral Winds. *J. Geophys. Res.,* Volume 103, pp. 471-487.

Thayer, J. & Semeter, J., 2004. The convergence of magnetospheric energy flux in the polar atmosphere. *J. Atmos. Sol. Terr. Phys.,* Volume 66, pp. 807-824.

Tjulin, A., 2016. *EISCAT Experiments.* Kiruna: EISCAT Scientific Association.

Vallinkoski, M., 1988. Statistics of incoherent scatter multiparameter fits. *J. Atmos. Terr. Phys,* Volume 50, pp. 839-851.

Vasyliunas, V. & Song, P., 2005. Meaning of Ionospheric Joule Heating. *J. Geophys. Res.,* Volume 110, p. A020301.

Vickers, H. et al., 2013. Therospheric atomic oxygen density estimates using the EISCAT Svalbard Radar. *J. Geophys. Res. Space Physics,* Volume 118, pp. 1319-1330.

Walker, J., 1965. Analytical Representation of Upper Atmosphere Densities based on Jacchia's Static Diffusion Models. *J. Geophs. Res.,* 22(4), pp. 3396-3400.