9-25-2014 12:00 AM

Gravity wave spectra morphology in the Arctic and non-Arctic lower atmosphere

Melanie C. Wright, *The University of Western Ontario*

Supervisor: Wayne K. Hocking, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Science degree in Physics

Gravity wave spectra morphology in the Arctic and non-Arctic lower atmosphere

(Monograph)

by

Melanie Wright

Faculty of Science
Department of Physics and Astronomy

Submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Physics

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada
August, 2014

Abstract

The spectral analysis of data from three VHF radars (one high-Arctic and two mid-latitude) show general support for the universal spectrum theory for gravity waves in the lower atmosphere (altitudes of 2.0-11.0 km), provided that the impact of the off-vertical beam and noise are taken into consideration. This analysis also reveals that local gravity wave generation is of secondary, but still significant, importance for determining the spectra.

A total of eight spectral methods were considered and scrutinized for the purposes of determining gravity wave spectra from VHF radar data. A definition for the "best" method was given and examined. The method selected as the "best" for the analysis presented was a date-compensated discrete Fourier transform with a Hamming window.

Keywords

gravity waves, universal spectrum, analysis methods, VHF, radar

Acknowledgments

I would like to thank the family and friends who have been most supportive of me completing my masters and during my time at Western, specifically (in no specific order): Gram Char and Grandpa Gary, Neil Bhatt, Jason Kirkness, John and Betty Kirkness, Emma Cookson, Amanda DiCarlo, Mat Abado, Rob Weryk, and the lovely people at Dance Steps (specifically Donna Bayley and Krista Conti).

I want to thank Emily McCullough for the use of her Matlab plotting function (imagescnanEmily), which has been included in Appendix B. I would also like to thank CANDAC for partial funding, including: Bob Sica, Pierre Fogal, Jim Drummond, and Ashley Kilgour. I would also like to thank the ladies in the Physics and Astronomy Department office (Clara, Jodi, Jackie, Lisa, Nelia, Loveleen, and Donna) who have been extremely helpful, as well as my advisory committee: Bob Sica and John de Bruyn.

Above all else, I would like to extend my deepest gratitude to my supervisor, Wayne Hocking, who has supported and pushed me the whole way through. I couldn't have asked for a better supervisor. And to think, it all started from a broken little finger.

Table of Contents

List of Figures

Chapter 1

1 Introduction

In 1960, Hines suggested internal gravity waves (or buoyancy waves) as a major contributor to upper atmosphere motions, carrying energy and momentum large distances through the atmosphere. The literature fixated on singular waves until, in 1982, VanZandt proposed a "universal spectrum"—a shape to the intensity of gravity waves versus their wavenumber that is independent of geographic location, meteorological conditions, altitude, and time—based on the oceanographic work of Garrett and Munk (1972, 1975). The literature absorbed the concept of universality: many (e.g. Medvedev and Klassen 2000) produced power spectral density forms based on theories such as Weinstock's nonlinear wave diffusion, Hines' Doppler shifted theory, and the inconsistent linear instability theory (Hines 1991). Focus then shifted towards deviations from the spectrum (e.g. Eckermann 1995), which leaves us thinking, "How *'universal'* is the universal spectrum anyway?"

I investigate the form of the gravity wave spectra at different geographic locations (Negrocreek, ON and Eureka, NU) and altitudes (selected altitudes between 1-14 km) in different seasons. My analysis using Ferraz-Mello's (1981) data-compensated discrete Fourier transform with a Hamming window reveals that "universality" is not a far-fetched idea; however, local gravity wave generation also affects the gravity wave spectra.

This thesis is comprised of seven chapters. Chapter 2 reviews the gravity wave literature and provides motivation for this work. The necessary prerequisite background on radars is given in Chapter 3. Chapter 4 outlines the mathematical background to the analysis methods including reasoning for applying windows and selecting various analysis techniques. Chapter 5 applies the methods described in Chapter 4 to determine the "best" method of computing atmospheric gravity wave spectra. The results are presented and interpreted in Chapter 6. Chapter 7 concludes this thesis and outlines potential future work.

Chapter 2

2 Gravity waves

Gravity waves (also known as buoyancy waves), named for the restoring force responsible for the wave motion, are perturbations in the atmosphere (density, pressure, velocity, and temperature) with typical horizontal wavelengths of a few to hundreds of kilometres, vertical wavelengths of approximately $1-30$ km, and periods of five minutes to many hours. In general, these waves can propagate along the interface of two mediums (such as on the surface of a lake or ocean, where they are called surface gravity waves) or through a medium (such as through the ocean or atmosphere, where they are called internal gravity waves), provided that the density decreases with increasing height. This thesis specifically examines the geographical and temporal variation of the spectrum of atmospheric internal gravity waves, which will be referred to as "gravity waves".

2.1 Why gravity waves

Atmospheric gravity waves have a large impact on aircraft and weather. Early interest in mountain lee waves (a particular type of gravity wave, generated by flow over mountains) originated from sailplane pilots, who used the waves to soar to record altitudes (Gossard and Hooke 1975). Waves can transport energy faster than mean flow transport. In particular, gravity waves carry energy away from the source (a mountain range, thunderstorm, etc.) and distribute it throughout the atmosphere, produce phenomena such as turbulence in the night time atmospheric boundary layer or clear air turbulence (Nappo 2002), which can be hazardous to aircraft (Gossard and Hooke 1975).

In addition to playing a major role in upper atmosphere dynamics (Hines 1960), gravity waves can slow—and even reverse—mean wind speeds throughout the atmosphere, including in the troposphere (e.g., Lindzen 1981, Holton 1983). A better understanding of gravity waves could lead to better weather forecasting models.

2.2 Gravity wave overview

While both surface and internal gravity waves propagate in a similar manner, it may be easier for the reader to conceptualize gravity waves using surface gravity waves, for example, on the surface of a lake. Consider a small packet of water displaced vertically above the lake's surface. The mass of water above the mean level causes the surface of the water to fall. When the water surface reaches equilibrium, it still has a downwards momentum and continues to fall. The surrounding water applies a restoring force to the water surface, causing it to slow the downwards motion, eventually stop, and then return towards the equilibrium position. The surface overshoots equilibrium again, causing it to be displaced above the water surface again, hence establishing an oscillation. This motion propagates along the surface.

Internal gravity waves propagate in the same manner, however, the atmosphere must satisfy a second condition (in addition to decreasing density with height); having a specific lapse rate. A lapse rate is the rate at which air temperature decreases with increasing altitude. For internal gravity waves to propagate through the atmosphere, the lapse rate of the air packet as it moves must be steeper than the lapse rate of the background atmosphere. As air rises, it expands and becomes less dense. After rising, the air packet needs to be denser than the surrounding air to oscillate. If it isn't, the air packet will continue rising, instead of oscillating, and internal gravity waves cannot occur. Where the lapse rates are such that internal gravity waves can propagate, the atmosphere is called "stable". Otherwise, the atmosphere is called "unstable".

2.3 Causes of gravity waves

One well accepted source of gravity waves is airflow over terrain, such as mountains and hills (Gossard and Hooke 1975). Airflow over mountains causes the initial perturbation necessary to create waves, as demonstrated in Figure 2.1. These waves are also known as "mountain lee waves". In a steady state, these waves are stationary with respect to the terrain, but propagate with respect to the mean airflow. In a time-dependent state, airflow over terrain should cause propagating waves with respect to the ground. On various

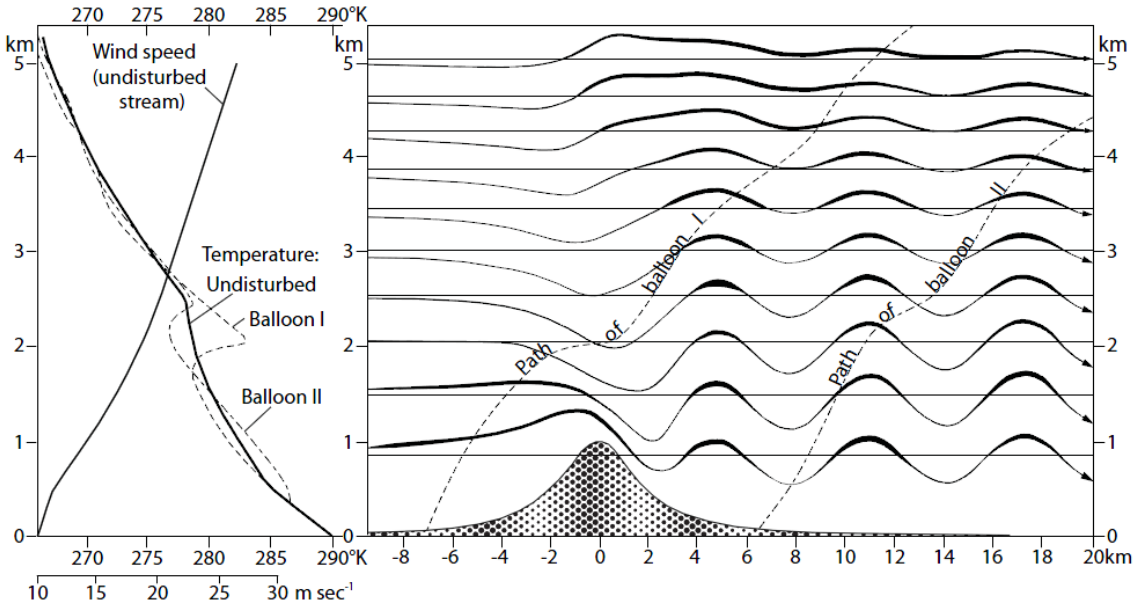


Figure 2.1: An example of mean wind flow over a mountain causing lee-waves and the accompanying mean wind and temperature structure. From Hocking (in prep.), adapted from Röttger (2000) (who adapted it from Scorer (1997)).

occasions Hines suggested (e.g. Hines 1991) that this may cause waves in the upper atmosphere.

Other potential sources of gravity waves are: squall lines and frontal systems, velocity jets, large explosions (Nappo 2002), turbulence, penetrative convection, wave-wave interactions, geostrophic adjustment, shear instability, Ekman-layer instability, and wave generation by boundary-layer turbulence (Gossard and Hooke 1975).

2.4 A brief history of the understanding of gravity waves

In 1960, Hines explored the theory of gravity waves as a major contributor to upper atmosphere motions. He used linear perturbation theory, which assumes that the wave amplitudes (the perturbations in pressure, temperature, etc.) are sufficiently small, such that all variables dependent on nonlinear combinations of wave amplitudes are negligible. In 1976, Weinstock advanced the theory of gravity waves by using a nonlinear theory,

though he still treated the nonlinear pressure terms as negligible, since the pressure fluctuations are relatively small.

The literature focused on singular gravity wave sightings and measurements until 1982, when Van Zandt introduced the idea of a "universal spectrum" of gravity waves, based on the oceanic work of Garrett and Munk (1972, 1975). The universal spectrum quantifies the spectral "tail" (lower wavelengths) as a power law and claims the shape of the tail is roughly invariant with meteorological conditions, latitude and longitude, time, and, to some extent, altitude.

In 1991, Hines developed a Doppler spread theory for gravity waves. In general, literature focus shifted to this concept of universality, with many research groups developing similar models to describe the universality. Focus then shifted to deviations from universality.

2.5 Mathematical Background

The mathematics of gravity waves starts with the standard fluid dynamics equations (a version of Newton's second law, the first law of thermodynamics combined with the speed of sound in air, conservation of mass, and heat diffusion, respectively), viz.:

$$\frac{D\vec{u}}{Dt} + 2\vec{\Omega} \times \vec{u} - \vec{g} + \frac{1}{\rho}\vec{\nabla}p - \nu\nabla^2\vec{u} = 0 \tag{2.1}$$

$$\frac{D\rho}{Dt} = \frac{1}{c_s^2}\frac{Dp}{Dt} \tag{2.2}$$

$$\frac{D\rho}{Dt} + \rho\vec{\nabla} \cdot \vec{u} = 0 \tag{2.3}$$

$$\frac{D\Theta}{Dt} = \frac{\kappa}{\rho}\nabla^2\Theta \tag{2.4}$$

where D/Dt represents differentiation following the motion, $\vec{u}$ is the velocity, ρ is the density, $\vec{\Omega}$ is the Earth's angular rotation rate, $\vec{g}$ is the acceleration due to gravity (and is given by $(0, 0, -g)$), p is the pressure, c_s is the speed of sound, $\vec{\nabla}$ is the gradient

differential operator, Θ is the potential temperature, κ is the heat diffusion coefficient, and ν is the kinematic viscosity coefficient.

For gravity waves, solutions are assumed to be of the form:

$$\Psi = \Psi_0 \, e^{i(\vec{k}\cdot\vec{x} - \omega t + \phi)} \tag{2.5}$$

where Ψ can be any one of the velocity components, the pressure, the density, or temperature, and may be complex. $\vec{k}$ is the wavenumber vector (k, l, m) and ω refers to the ground-based angular frequency of the wave. These forms are substituted into the standard fluid dynamics equations above, keeping only first-order perturbation terms in an attempt to linearize the equations for gravity wave analysis. If the effects of viscosity are ignored, a mean wind of zero is assumed (for simplicity), and the wave propagates in the x—z plane (with z being vertical), then the standard fluid dynamics equations (i.e. Equations 2.1 through 2.4) simplify to five equations; three momentum equations:

$$-i\omega\hat{u} - f\hat{v} = -ik\hat{\psi} \tag{2.6}$$

$$-i\omega\hat{v} + f\hat{u} = 0 \tag{2.7}$$

$$-i\omega\hat{w} + \hat{r}g = -\left(im\hat{\psi} - \frac{\hat{\psi}}{\bar{H}}\right) \tag{2.8}$$

a form of the first law of thermodynamics:

$$-i\omega\hat{r} - \frac{\hat{w}\omega_B^2}{g} = -i\frac{\omega\hat{\psi}}{\overline{c_s^2}} \tag{2.9}$$

and a continuity equation:

$$-i\omega\hat{r} + ik\hat{u} - \frac{\hat{w}}{H} + im\hat{w} = 0 \tag{2.10}$$

where a hat (e.g., $\hat{u}$) denotes a perturbation value, $i = \sqrt{-1}$, f is the Coriolis parameter $2\Omega \sin\theta$ (where θ is the latitude), $\hat{\psi} = \hat{p}/\bar{p}$, $\hat{r} = \hat{\rho}/\bar{p}$, and $\overline{c_s^2}$ is the mean squared speed

of sound at the height of the wave. The complex velocity perturbation is $\vec{u} = (\hat{u}, \hat{v}, \hat{w})$. The $\vec{k}$-vector is $(k, 0, m)$, where it is assumed that the wave propagates in the x—z plane for simplicity. The complex vertical wavenumber, m, is $m_R + i\, m_I$ (complex to indicate that the wave is oscillating and the amplitude increases exponentially with increasing height), where $m_I = -1/(2\bar{H})$. $\bar{H}$ is the scale height given by:

$$\bar{H} = \frac{R\bar{T}}{g} \tag{2.11}$$

The Brunt-Vaisala frequency, ω_B, is the natural oscillation frequency of a displaced air parcel in the atmosphere (below 100 km altitude, values are typically 5-10 minutes) and satisfies:

$$\omega_B^2 = \frac{g}{\bar{T}}\frac{\partial \bar{T}}{\partial z} + \epsilon\frac{g}{\bar{H}} = \frac{g}{\bar{\Theta}}\frac{d\bar{\Theta}}{dz} \tag{2.12}$$

where $\epsilon = R/c_p$, R is the gas constant for air, c_p is the specific heat of air at constant pressure, and Θ is potential temperature.

Solutions to these equations are various waves that satisfy two particular relations: the dispersion relation and polarization relations. The dispersion relation relates the wave frequencies and wavenumbers. It also restricts wave frequencies to the range between the Brunt-Vaisala frequency and the "inertial frequency", the lower frequency limit set by the Coriolis parameter. The polarization relations relate wave-velocity amplitudes to the temperature, density, and pressure amplitudes. Both the dispersion and polarization relations can differ slightly, depending on the terms ignored/retained in the linearized approximations to the equations of motion.

For the approximated equations of motion, the dispersion relation is:

$$m_R^2 = \frac{\omega_B^2 - \omega^2}{\omega^2 - f^2}k^2 - \frac{1}{4\,H^2} + \frac{\omega^2}{c_s^2} \tag{2.13}$$

A simpler form of the approximation, called the Boussinesq approximation, is:

$$m_R^2 = \frac{\omega_B^2 - \omega^2}{\omega^2 - f^2} k^2 \tag{2.14}$$

for a wide range of frequencies larger than f and smaller than ω_B, this dispersion relations can be approximated by:

$$\frac{m_R}{k} = \frac{\omega_B}{\omega} \tag{2.15}$$

The polarization relation is approximated by the following equations:

$$\hat{w} = -\frac{k}{m_R}\hat{u} = \frac{\omega}{(\bar{u}-c)m_R}\hat{u} \tag{2.16}$$

$$\hat{v} = -i\frac{f}{\omega}\hat{u} \tag{2.17}$$

$$\hat{\Theta} = \frac{d\bar{\Theta}}{dz}\hat{\xi} = -\frac{i}{\omega}\frac{d\bar{\Theta}}{dz}\hat{w} = -\frac{i}{\omega}\frac{\omega_B^2}{g}\bar{\Theta}\hat{w} = \frac{-i}{m_R(\bar{u}-c)} \cdot \frac{d\bar{\Theta}}{dz}\hat{u} \tag{2.18}$$

$$\frac{\hat{\Theta}}{\bar{\Theta}} = -\frac{\hat{\rho}}{\bar{\rho}} \tag{2.19}$$

$$\hat{p} = \hat{u}\bar{\rho}(c - \bar{u}) \tag{2.20}$$

where $\hat{\xi}$ is the vertical displacement, and $c - \bar{u} = \omega/k$ is the "intrinsic phase speed" of the wave, or the wave phase speed with respect to the mean wind at the height of the wave. Equations 2.16 and 2.17 are exact for the Boussinesq approximation and Equations 2.18, 2.19, and 2.20 are only valid for $f \ll \omega \ll \omega_B$.

More precise relations can be developed; however, these approximations are useful since they are simple, yet are still fairly accurate over a wide range of frequencies. Unlike the derivations above, the mean wind is allowed to be non-zero, equal to $(\bar{u}, 0, 0)$.

2.6 The spectral tail

While the theory of gravity wave spectra in the atmosphere and oceans is very similar, VanZandt (1982) summarized three key differences: (1) Doppler shifting by mean flows is more important in the atmosphere, (2) nonlinear wave interactions occur over shorter timescales, and (3) the atmospheric buoyancy frequency is roughly independent of height, whereas it decreases exponentially with depth in the ocean.

In the atmosphere, the slope of the tail is roughly invariant with altitude and flattens off at lower frequencies (see the sample tail in Figure 2.2). At higher altitudes, the tail is longer (i.e. flattens off at a lower frequency). Van Zandt (1982) claimed this

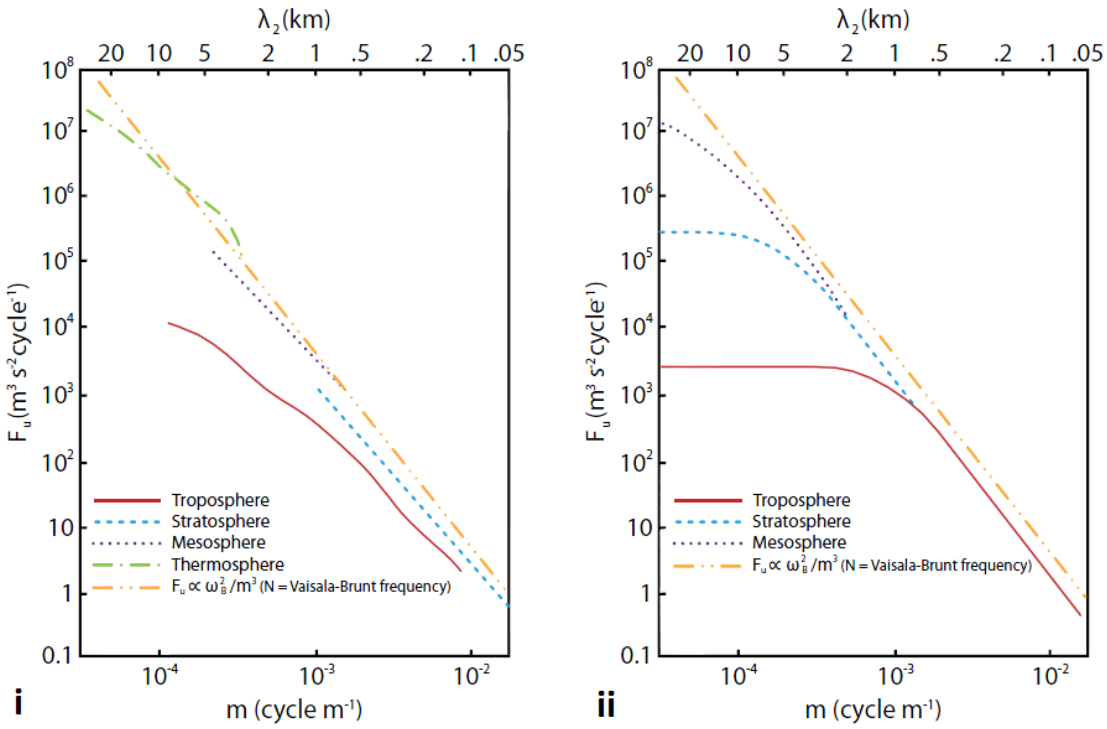


Figure 2.2: (i) Measured atmospheric spectra (F_u) of horizontal gravity wave fluctuations as a function of vertical wavenumber (m) for various heights. Note that the "roll off" point, which marks the transition between the m^{-3} part of the spectrum and the flatter part changes with altitude. (ii) Model spectra proposed by Smith et al. (1987). These graphs were taken from Hocking (in prep.), who adapted them from Smith and Van Zandt (1987).

was the result of saturation of particular frequencies and left the physical cause for separate identification.

Hodges (1967, 1969) introduced linear perturbation theory to model gravity waves, which assumes that the wave amplitudes are small enough such that the nonlinear terms are negligible. When gravity waves propagate upwards in the atmosphere, the wave amplitudes grow exponentially (due to exponentially decreasing density). Some authors (e.g. Lindzen 1981, 1984) used linear instability theory to account for the form and intensity of the tail; at a critical height, the linear approximation breaks down and above this height, nonlinearities cause saturation. Hines (1991) reworked the linear instability formulation and found inconsistencies. Hines (1991) created a Doppler shifted theory which accounts for the intensity and the form of the spectral tail, claiming that linear instability theory may account for the length of the tail.

Another potential cause of gravity wave saturation is shedding. As waves increase in altitude, their amplitude increases as well. At a critical height (dependent on the frequency of the wave), the wave grows too big and any energy that would normally cause an increase in amplitude dissipates as turbulence instead.

Extending Weinstock's (1976) nonlinear wave diffusion theory and Hines' (1991) Doppler spreading theory, Medvedev and Klaassen (2000) explained saturation by

turbulence induced in waves that exceeded the convective instability thresholds. They parameterized the power spectral density, S, as:

$$S = \frac{A\,\omega_B^2}{m^3} \tag{2.21}$$

where A is roughly a constant (which slowly varies with m and the mean wind), ω_B is the Brunt-Vaisala frequency, and m is vertical wave number. Throughout the literature, values of A range from $\frac{1}{9}$ to $\frac{1}{2}$ (see Medvedev & Klassen, 2000), with values typically being $\frac{1}{6}$ (e.g. Smith et al. 1987). Other literature even has A dependent on seasons. This model gives reasonable magnitudes for acceleration of mean flow and some observations of saturated, near-monochromatic waves support this model. This model is also attractive

for use in parameterization schemes because it has a clear physical interpretation. However, it has some disadvantages too; the theory only works for near monochromatic waves, it requires the gravity wave amplitudes to be nearly constant at and above the breaking level, and, when using this model, the vertical wave drag profiles and enhanced diffusion become step functions, which cause continuity problems in large scale models.

Chapter 3

3 Radars

3.1 The antennas

The radars used to gather data for this thesis do not use a large dish for transmitting and receiving the signal, but rather use a large array of smaller antennas that work together as one coherent unit. This is called a "phased array" and its beam can be steered without actually tilting the ground upon which the antennas rest. The specific radars for this study are comprised of 128 antennas (called Yagi antennas). In the arrays used, each antenna acts as both a transmitter and receiver. The antennas have three horizontal bars; from the top down, they are: the director, the driven element, and the reflector.

Figure 3.1: Yagi antennas. Part of the antenna array located at Eureka, NU.

3.2 How they work

The transmitter, under the control of the computer, sends an electronic pulse to the antenna array and each antenna transmits the pulse as an electromagnetic wave. Since there are many antennas transmitting the same pulse simultaneously (an exception, i.e. beam steering, is discussed below), the transmitted pulse appears as a plane wave propagating vertically upward from the array, as demonstrated in Figure 3.2. Each of the planar waves is an interference pattern caused by the individual antenna wavelets. The waves reflect off targets (or more specifically, gradients in refractive index) in the atmosphere. Some of the reflected wave propagates back towards the antenna array.

The reflected wave induces a current in the driven element (the middle horizontal bar). The director and reflector play the role of narrowing the radar beam, and so in some senses "concentrating" the signal. After going through hardware processing, the voltage is digitized at a rate corresponding to (in our case) 500 m intervals.

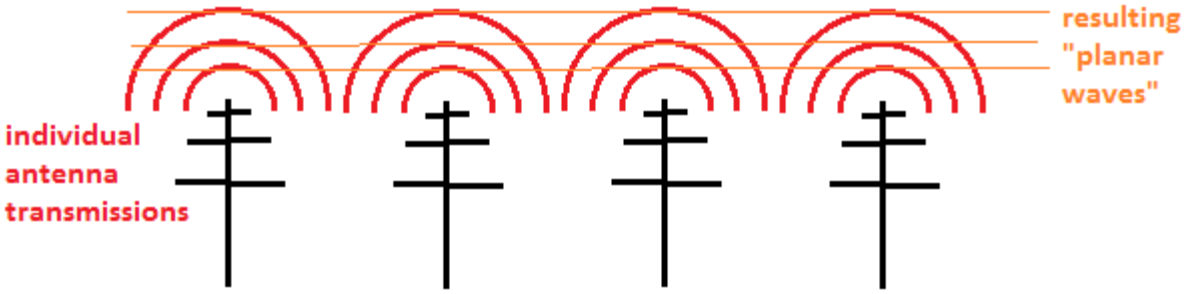


Figure 3.2: A schematic describing how the transmitted pulses from the antenna array appear to form a planar wave.

3.3 Beam steering

The 3D interference pattern created by the antennas is called the beam pattern. The beam pattern represents the "pattern of sensitivity" of the radar, with targets in the regions of highest sensitivity being most strongly detected; i.e. the direction of the beam pattern defines where the radar is “looking”. The beam pattern is the diffraction pattern of the antennas and, therefore, depends on the layout of the antennas and their orientation with respect to each other. Beam patterns often have one large central beam and several

smaller beams (called sidelobes), which point in different directions than the main beam. Since the sidelobes are smaller (i.e. lower power), most of the atmospheric detections occur through the main lobe. The effect of sidelobes is discussed below.

In the case described above, the main beam points perpendicular to the plane of the radar (normally vertical). To steer the beam (to "look" in other directions), a time delay is applied between nearby antennas transmitting the same pulse, as seen in Figure 3.3.

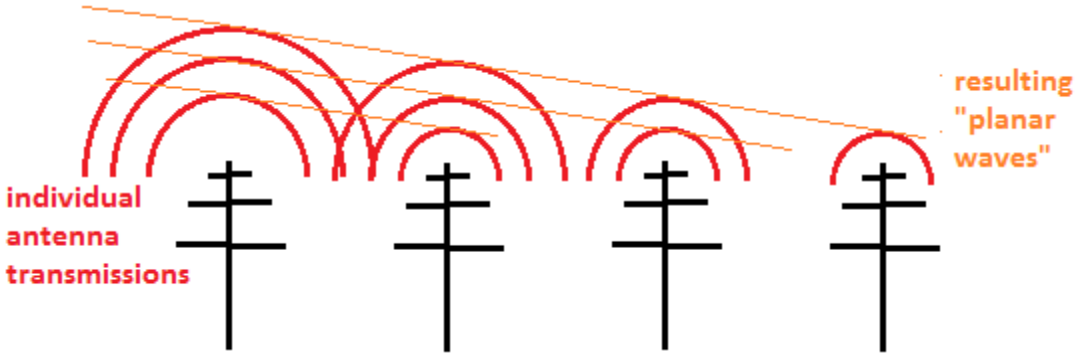


Figure 3.3: A schematic describing how the radar beam is steered through the sky. Rather than transmitting the pulse simultaneously, the left-most antenna started transmitting, followed by the second from the left, the third from the left, and then the rightmost antenna. Typically delays are a small fraction of one wave-period; in our case typically 4 ns or so.

3.4 Detection difficulties

3.4.1 Detection through sidelobes

Since the main beam is large in strength (so it has more power) in comparison to the sidelobes, it is assumed that targets detected are located within the main beam. However, a strong reflector (e.g. a plane, a building, etc.) in a side lobe will have the same effect as a weaker reflector in the main beam.

Once the radar is built (i.e. the position of the antennas is set), there is not much that can be done about detection through sidelobes. In some cases, a special procedure

called "interferometry" can be used to obtain better directional information, but this requires extra antennas and is not needed for our particular studies.

3.4.2 Non-atmospheric objects

It's a bird! It's a plane! It's not an atmospheric target. The targets in the atmosphere are weak reflectors (small density changes) and typically move relatively slowly (most velocities are less than 4 m/s). When something with strong reflectance is in the radar volume or an object moves unusually fast through the radar volume, it has a stronger influence on the data than the atmospheric targets. However, their distinct nature often allows them to be recognized and ignored.

The power returned to the antenna array depends on the square of the potential refractive index gradient. The refractive index in the lower atmosphere is:

$$n = 1 + 77.6 \times 10^{-6} \frac{p}{T} + 3.73 \times 10^{-1} \frac{e}{T^2} \tag{3.1}$$

where p is the pressure in millibars, T is the temperature in Kelvin, and e is the partial water vapour pressure in millibars. Sometimes this equation also contains a plasma term, which is unnecessary in the region of interest for this study. The gradient is given by (Tatarski 1961, with updated constants from Larsen and Röttger 1982):

$$M = \frac{-77.6\times10^{-6}}{T^2} \cdot p \cdot \left[1 + \frac{15500}{T} q\right] \left[\frac{dT}{dz} + \Gamma_{\mathrm{a}} - \frac{7800}{1+\frac{15500}{T}q} \cdot \frac{dq}{dz}\right] \tag{3.2}$$

where T is the temperaure in degrees celsius, Γ_a is the adiabatic lapse rate, and $q = e/(1.62p)$ is the specific humidity. The power received by the radar, P_R, is proportional to the square of the refractive index gradient, viz.:

$$P_R \propto M^2 \tag{3.3}$$

Therefore, larger refractive index gradients, such as those between the atmosphere and a bird or plane, return more power to the radar than smaller refractive index gradients, such as the atmospheric targets we are interested in.

Often, highly reflective, non-atmospheric objects (e.g. birds, airplanes) do not stay in the radar volume for very long, perhaps a few tens of seconds. Such targets can be removed in two ways: (1) recognizing the characteristic signature of the entity and removing it from the data-stream or (2) simply ignoring the entire data set. Typically calculation of a wind measurement requires $30 - 40$ s of data, so if such a group is too severely contaminated, the entire $30 - 40$ s data set is ignored.

Another type of non-atmospheric target that gets detected is ground clutter (e.g. mountains and buildings). Reflections from ground clutter enter the radar beam through the sidelobes and are seen as having zero velocity. Various tricks can be used to remove these contaminants, but they are only partly successful. In this thesis, we primarily use data collected using an off-vertical beam, since the horizontal movement of the air (generally) produces non-zero radial velocities and ground clutter (which occurs at 0 Hz) can be notched out. Vertical beams are harder to use because the vertical winds are often close to zero and merge in with the ground-clutter, making separation of the different spectral sources more difficult, as shown in Figure 3.4.

3.4.3 Poor low altitude range data: impedance mismatch

If the impedance of the antenna and the cables are not exactly the same, then when the transmitted pulse passes from the coaxial cable to the antenna, some of the signal is reflected. The reflected signal propagates back along the coaxial cable and is reflected at the transmitter end since it is not expecting to receive a signal of that strength (the receiver is disconnected when the voltage is over a particular threshold). The reflected signal propagates back to the antenna where some is reflected again and some is transmitted. This oscillation leads to a ringing after the pulse has been transmitted. Sometimes this ringing interferes with receiving reflections from the pulse scattered from atmospheric targets. This often results in unreliable data for the lower end of the radar's range ($1 - 3$ km for the radars used in this thesis), so conclusions here-in are primarily based on higher-altitude data.

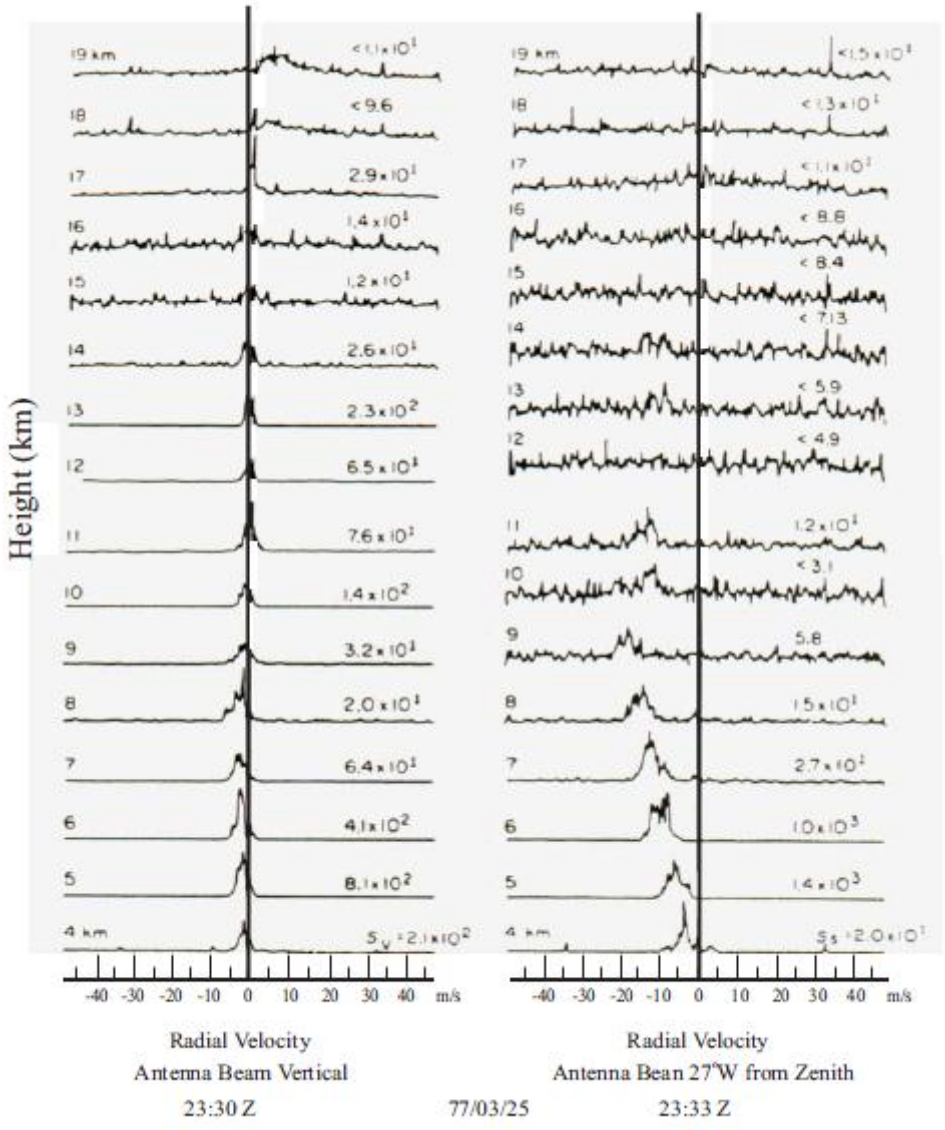


Figure 3.4: An example of radial velocities when the radar beam is vertical (left hand side) and off-vertical (right hand side) with ground clutter removed. Each successive spectrum corresponds to a different height. The numbers to the left of each spectrum indicate the recorded power. At the upper heights, the signal appears noisier due to a loss of signal (for reasons discussed in Section 3.4.4). The off-vertical velocities appear to become noisier than the vertical velocities at lower altitudes because of the anisotropic scattering nature of the atmospheric targets. From Hocking (in prep.), originally from Gage and Green (1978).

3.4.4 Poor high altitude range data: low power

If the scatterers are particularly weak, the reflected wave may go undetected by the radar. This is particularly true for the higher altitudes of the radar's range ($12 - 14$ km for the radars used in this thesis) for a variety of reasons.

The power transmitted from the radar diminishes in strength proportionally to the range squared. Upon reflection, the power of the reflected wave also diminishes proportionally to the range squared, so the power received by the radar diminishes as the range to the power -4. This is partially compensated for by the fact that the radar volume (where the atmospheric targets are detected) increases proportionally to the range squared. This still results in a net loss of returned power. Specifically, the received power is inversely proportional to the square of the range—scatterers at 10 km altitude return 100 times less signal than scatterers at 1 km altitude.

The properties of the atmosphere also come into play. Air density decreases exponentially with altitude—air density at 10 km is 4 times less than at the ground—resulting in a loss of scattering capability. Furthermore, the scattering cross-section also depends on water vapour content to some extent. Water vapour densities are considerably less at 10 km altitude. All these factors combine to mean that above (typically) $10 - 12$ km altitude, the signal is too weak for our radars to detect.

3.4.5 Range determination and range-aliasing

The radar sends out a pulse, then "listens" for a response. The distance from the radar to a target (the range of the target) is determined from the expression:

$$r = \frac{ct}{2} \tag{3.4}$$

where c is the speed of light in air (the speed at which the pulse propagates) and t is the time between the radar pulse-transmission and pulse reception. In practice, there will be multiple targets and so multiple received pulses at different lags. The factor of 2 comes from the fact that the observed time is twice the time the wave took to get from the radar to the target (as the wave has to travel there and back again). In this equation, r is used to

denote range as opposed to an h (implying height) since if the beam is pointing in a direction other than vertical, range and height are different.

The detection process assumes that everything detected is a reflection from a target due to the last pulse that was transmitted. This is not always true; sometimes a pulse from a very distant target may return to the radar *after* a subsequent pulse was transmitted, resulting in range aliasing (incorrectly calculating the range) for detections farther away than:

$$r^* = \frac{c\Delta t}{2} \tag{3.5}$$

where Δt is the time between successive radar pulses. Since the radar is not designed to detect reflections above r^*, the returned pulses from beyond r^* are generally weak; however, a strong reflection (e.g. meteor) at a higher altitude may still reflect the pulse such that it is detected by the radar.

For an example, consider a radar that transmits pulses at an interval allowing it to see up to 20 km away. If a meteor at 90 km range reflected the pulse, the radar would interpret the meteor reflection as happening at 10 km range (a reflection from the most recently transmitted pulse) as opposed to 90 km (a reflection from the fourth last pulse, which is the true pulse of origin for the scattered pulse).

3.5 Why radar? (As opposed to other methods)

Consistent, short time interval data are needed for this study. Radar is the only method that gathers atmospheric measurements of this nature.

In-situ measurements, such as rockets and weather balloons, can yield very detailed data. With in-situ measurements, the experimenter has no control over what section of atmosphere it samples and the time between successive launches ($6 - 12$ hours) is too long for this study.

Other ground-based measurements (mainly lidar) can provide successive, highly detailed wind measurements, but until recently they were only useful on clear, dark

nights. For mid-latitude sites, the lidar would be unoperational every day during sunlit hours, which would not be ideal. For Arctic sites, the lidar would be unoperational for the entire summer (as the sun does not set). These gaps in data collection are not well suited for this study. Recent developments have allowed newer lidars to work in the daytime as well, but they are still restricted by fog and clouds and—for the present—are primarily useful below 1 km or so in height (Stiller et al. 2012).

3.6 Our sites

This thesis uses three Canadian radar sites; Eureka, Negrocreek, and Markstay (as shown in Figure 3.5). All three sites consist of 128 Yagi antennas and operate somewhere in the band 46-51 MHz, with specific frequencies being different at each radar.

Figure 3.5: A map (courtesy of Google) indicating the locations of the three radar sites used.

3.6.1 Eureka, Nunavut

Eureka, Nunavut (pictured below) is a high-Arctic site at GPS coordinates 80.00 N, 85.80 W. The radar at Eureka is surrounded by Arctic tundra. The radar operates at 51.0 MHz and has been operational since 2007.

Figure 3.6: A panoramic view of the radar station at Eureka, NU with Blacktop mountain in the distance. The computer and transmitter are located inside the blue-green building to the right.

3.6.2 Negrocreek, Ontario

Negrocreek (pictured in Figure 3.7) is located in southern Ontario at GPS coordinates 44.36 N, 80.86 W near the town of Owen Sound, near the Great Lakes. The land at Negrocreek is swampy and surrounded by trees. The radar operates at 48.92 MHz and has been operational since 2008.

3.6.3 Markstay, Ontario

Markstay (picture in Figure 3.8) is also located in Ontario, north of Negrocreek at GPS coordinates 46.54N, 80.54 W, a short distance from Sudbury. The area surrounding Markstay is mostly forest and Canadian shield. The radar operates at 45.47 MHz and has been operational since 2010.

Figure 3.7: Part of the radar station at Negrocreek, ON.

Figure 3.8: Part of the radar station at Markstay, ON.

Chapter 4

4 Spectral analysis methods

A repetitive function or signal can be represented as a superposition of simple sine and cosine waves, sometimes referred to as harmonics. However, in a more developed theory, even non-periodic functions can be represented in this way, provided that an infinite number of frequencies are available. The Fourier transform of a function determines which frequencies, and the amplitude of each frequency that the harmonics require, in order to sum to the original function. For a continuous function in time, $g(t)$, the Fourier transform has many equivalent definitions; however, this thesis will use the definition:

$$G(f) \equiv \int_{-\infty}^{\infty} g(t) \mathrm{e}^{-2\pi i f t} dt \tag{4.1}$$

for all real t and f, where t represents time (in seconds) and f represents frequency (in hertz).

The power spectrum (relative power of each harmonic vs. frequency) is often used to analyse the harmonics of a signal. The power spectrum intensity is given by:

$$I(f) = \|G(f)\|^2 \tag{4.2}$$

The remainder of this chapter discusses issues arising from using the Fourier transform with our specific data (finite, discrete, and non-uniformly spaced data), and our solutions to those problems (windowing functions and the date-compensated discrete Fourier transform).

4.1 Finite Data

In practice, the measured signal is not infinite in extent, and the Fourier transform is usefully redefined as:

$$\hat{G}(f) = \int_0^T \hat{g}(t) \mathrm{e}^{-2\pi i f t} dt \tag{4.3}$$

where T represents the signal length and the hat on the g signifies the finite data signal (which, for this section, is continuous; discretizing the signal causes other effects, which are discussed later). For the purposes of this thesis, $\hat{g}(t)$ is one component of the radar-derived wind velocity. In terms of the ideal, infinite signal, the finite signal is:

$$\hat{g}(t) = g(t)h(t) \tag{4.4}$$

where $h(t)$ is a boxcar window function:

$$h(t) = \begin{cases} 1 & \text{for } 0 \leq t \leq T \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

so that:

$$\hat{G}(f) = \int_{-\infty}^{\infty} g(t)h(t)\mathrm{e}^{-2\pi i f t}dt \tag{4.6}$$

From this perspective, it is evident that the finite nature of the data intrinsically applies a window function. This has deleterious effects in the frequency domain, causing a broadening of spectral frequencies and "frequency leakage".

Multiplying the signal by the window function applies the window to the signal. In the frequency domain, this is equivalent to convolving the signal's Fourier transform and the window function's Fourier transform. To see the full effects of a window, consider a pure sine wave. The Fourier transform of a pure sine wave is a delta function at the wave's frequency, as shown in Figure 4.1 (ii). The Fourier transform of a boxcar window function is a sinc function, as shown in Figure 4.1 (iv). The Fourier transform of a pure sine wave, after applying a boxcar window to it, is a sinc function (the boxcar's Fourier transform) centred at the sine wave's frequency, as shown in Figure 4.1 (vi). Note that the boxcar Fourier transform has a main lobe centred at zero with smaller "sidelobes" on either side. While a wider main lobe broadens spectral peaks, the sidelobes spread spectral frequencies to higher and lower frequencies.

However, a boxcar is only one type of window. Sometimes, spectral clarity results from using a more restrictive window, with tapering at the edges. It may seem

that such a process would further deteriorate the spectrum, but advantages ensue due to reduction of the sidelobes. Since using a windowing function is unavoidable, a variety of window functions were examined.

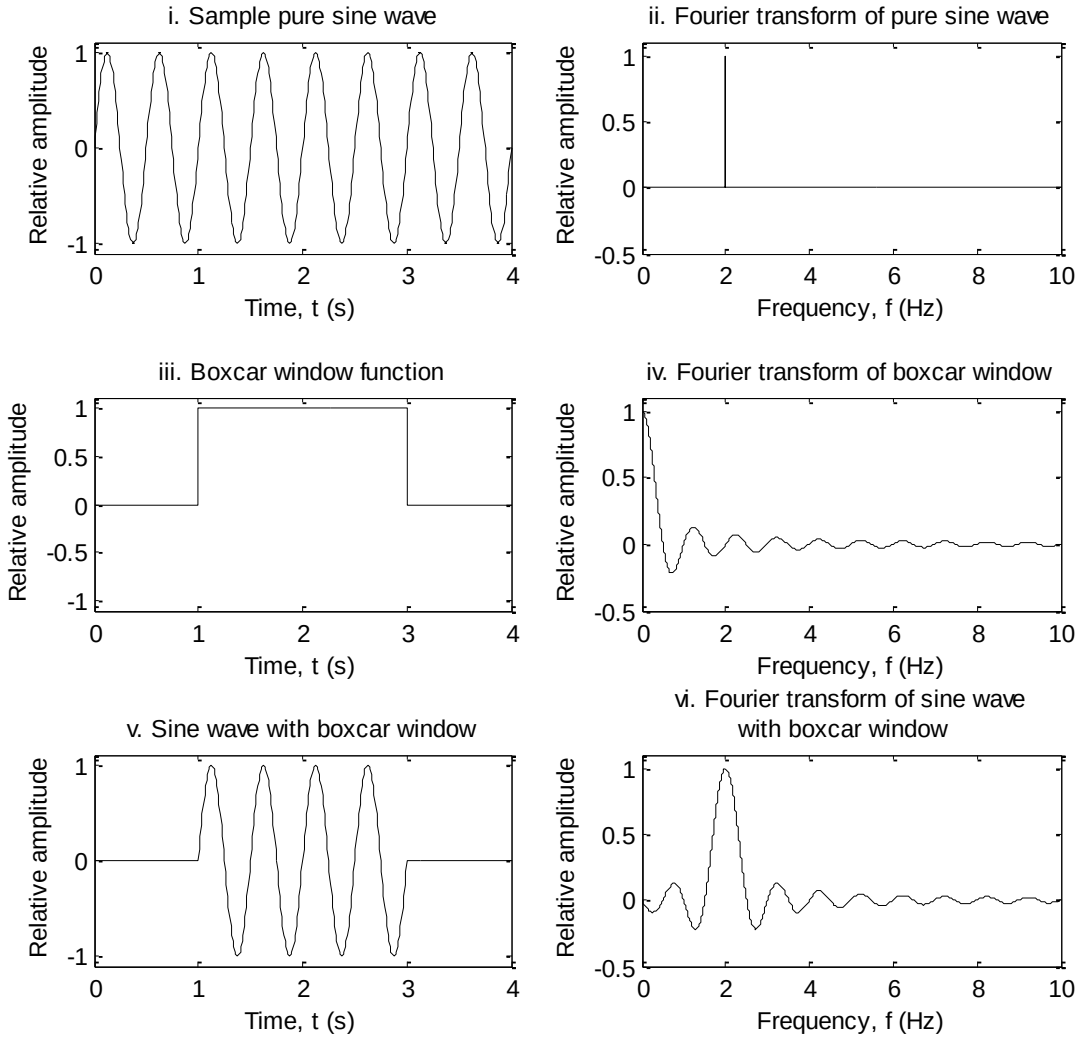


Figure 4.1: A demonstration of how a multiplication in the time domain is equivalent to a convolution in the frequency domain. (i) A pure sine wave, (iii) a boxcar window function, and (v) the product of (i) and (iii). The plots in the right column are the corresponding Fourier transforms of the plots shown to their immediate left.

4.2 Windows

Creating the "best" window is a result of optimizing the main lobe width and relative amplitude of the sidelobes. Since the data set is finite, there will always be sidelobes; however, they can be suppressed at the expense of widening the main lobe (broadening the main lobe is equivalent to smearing frequencies into neighbouring frequencies in the frequency domain). Similarly, the main lobe can be narrowed at the expense of increasing the sidelobes (potentially leading to frequency aliasing). Both large sidelobes and a wide main lobe are detrimental to the spectra. Four windows are examined in this thesis to determine the optimal window for the gravity wave spectra. The windows examined here are: boxcar, boxcar with 10% cosine tapering, Hann (sometimes called Hanning), and Hamming. The windows and their Fourier transforms are displayed in Figure 4.2. The normalization and Fourier transform of each window are displayed step-by-step in the Appendix.

4.2.1 Boxcar window

The boxcar window (Figure 4.2(i)) for a signal length of T is:

$$h(t) = \begin{cases} 1/T & \text{if } |t| \leq T/2 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

The boxcar window may appear to be ideal because it gives each data point equal weighting. However, the sudden start and stop of the data causes detrimental ringing in the frequency domain.

4.2.2 Boxcar with 10% cosine taper window

The boxcar with 10% cosine taper (Figure 4.2 (iii)) for a signal length of T is:

$$h(t) = \begin{cases} \frac{10}{9T}\cos\left(\frac{10\pi}{T}t\right) & \text{if } -\frac{T}{2} \leq t < -\frac{2T}{5} \\ \frac{10}{9T} & \text{if } |t| \leq \frac{2T}{5} \\ \frac{10}{9T}\cos\left(\frac{10\pi}{T}t\right) & \text{if } \frac{2T}{5} < t \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4.8}$$

This window gives 80% of the data equal weighting and uses the outside 20% (10% at the beginning and 10% at the end) to smooth the ringing effect of the sudden start and stop.

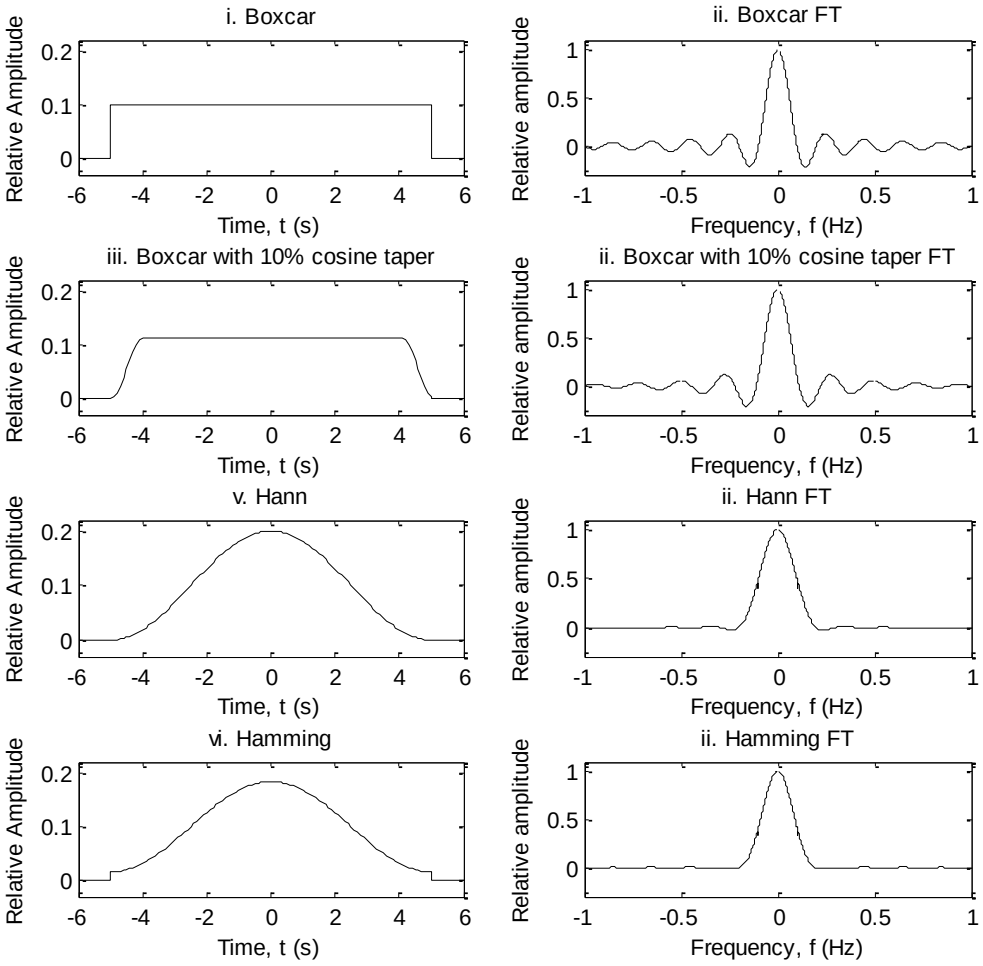


Figure 4.2: The left column contains representations of all the windows applied to data sets in this thesis. While the windows in plots (i), (iii), (v), and (vii) span 10 seconds in this sample, when applied to the data in this thesis, they span the same length of time as the data (typically a month). The plots in the right column are the Fourier transform of the plots shown to their immediate left.

4.2.3 Hann window

The Hann window, named for Julius von Hann, (Figure 4.2(v)) is the discrete window function:

$$w(n) = \sin^2\left(\frac{\pi n}{N-1}\right) \tag{4.9}$$

for $\{n \in Z \,|0 \leq n \leq N\}$, where N is the sample size. For the purposes of this section (to demonstrate the effects of a windowing function rather than the effects of discretization of data), the Hann window is treated as the continuous function:

$$h(t) = \begin{cases} \frac{2}{T}\cos^2\left(\frac{\pi t}{T}\right) & \text{for } |t| \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4.10}$$

where T is the signal length and the window has been shifted so that its centre is at $t = 0$. The Hann window widens the main lobe to minimize the amplitude of all side lobes.

4.2.4 Hamming window

The Hamming window, named for Richard W. Hamming, (Figure 4.2(vii)) is the discrete window function:

$$w(n) = \frac{25}{46} - \frac{21}{46}\cos\left(\frac{2\pi n}{N-1}\right) \tag{4.11}$$

for $\{n \in Z \,|0 \leq n \leq N\}$, where N is the sample size. For the purposes of this section, the Hamming window is treated as the continuous function:

$$h(t) = \begin{cases} A\left(\frac{25}{46} + \frac{21}{46}\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |t| \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{4.12}$$

where T is the signal length, A is the normalization constant $\frac{92}{25T}$, and the window has been shifted so that its centre is at $t = 0$. Unlike the other windows in this thesis, the Hamming window tapers off at the beginning and end of the data set, but it is non-zero at

$t = \pm\frac{T}{2}$. While the Hann window aims to reduce the effect of all sidelobes, the Hamming window aims to reduce the first side lobe as much as possible.

4.3 Discrete Data

A second issue arises when trying to use the Fourier transform; the Fourier transform is defined for a continuous signal and the radar collects discrete data. Discretizing the data is equivalent to multiplying the continuous signal (i.e. the north component of the wind velocity in our case) by a sampling function; specifically a series of Dirac delta functions evenly spaced at an interval of Δt. The sampling function is:

$$w(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta t) \tag{4.13}$$

The Fourier transform of the sampling function is another sampling function:

$$W(f) = \sum_{k=-\infty}^{\infty} \delta(f - k\Delta f) \tag{4.14}$$

where the spacing in the temporal and frequency domains are related by:

$$\Delta f = \frac{1}{\Delta t} \tag{4.15}$$

Recalling that a multiplication in the temporal domain is a convolution in the frequency domain, multiplying the signal by a sampling window leads to repetition of the signal's Fourier transform in the frequency domain, as Figure 4.3 demonstrates.

In order to determine a frequency, it must be sampled at least twice per cycle. If a frequency is sampled less frequently, it appears as a wave of lower frequency. For a visual example, see Figure 4.4. Any wave of frequency f_{actual}, which is above half the sampling frequency (termed the Nyquist frequency), appears as the lower frequency:

$$f_{apparent} = f_{actual} \bmod f_N \tag{4.16}$$

where f_N is the Nyquist frequency ($f_{sampling}/2$). This implies that all frequencies larger than the Nyquist frequency are mapped onto the range $[0, f_N]$, as demonstrated in Figure

4.5 to Figure 4.7. Figure 4.5 is a sample spectrum. If the data were discretized at a rate of 20 Hz, the resulting calculated spectrum would be given by Figure 4.6. Figure 4.7 shows how the sample spectrum and resulting spectrum are related. Non-equally sampled data may produce harmonics beyond the Nyguist frequency. This would add to the noise level at all frequencies and the harmonics would be aliased to the wrong frequency.

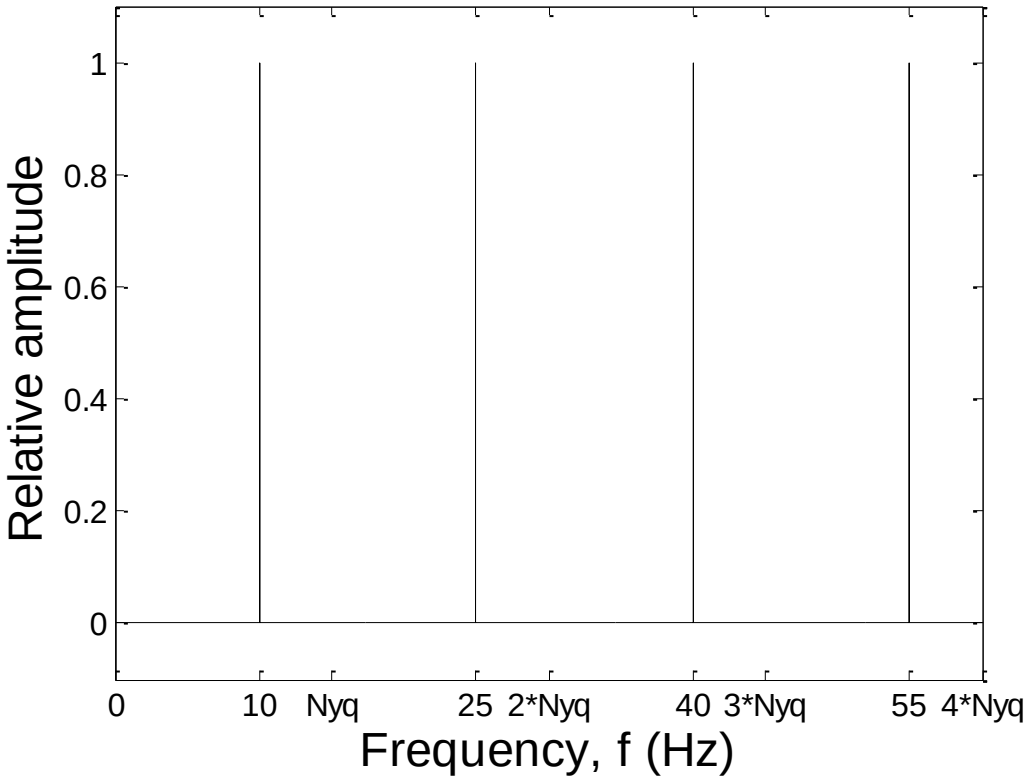


Figure 4.3: The Fourier transform of a pure sine wave with a frequency of 10 Hz, sampled at a frequency of 30 Hz. The Nyquist frequency (denoted as Nyq in the figure) is half the sampling frequency (15 Hz). Beyond the Nyquist frequency, the spectrum repeats itself as described in the text.

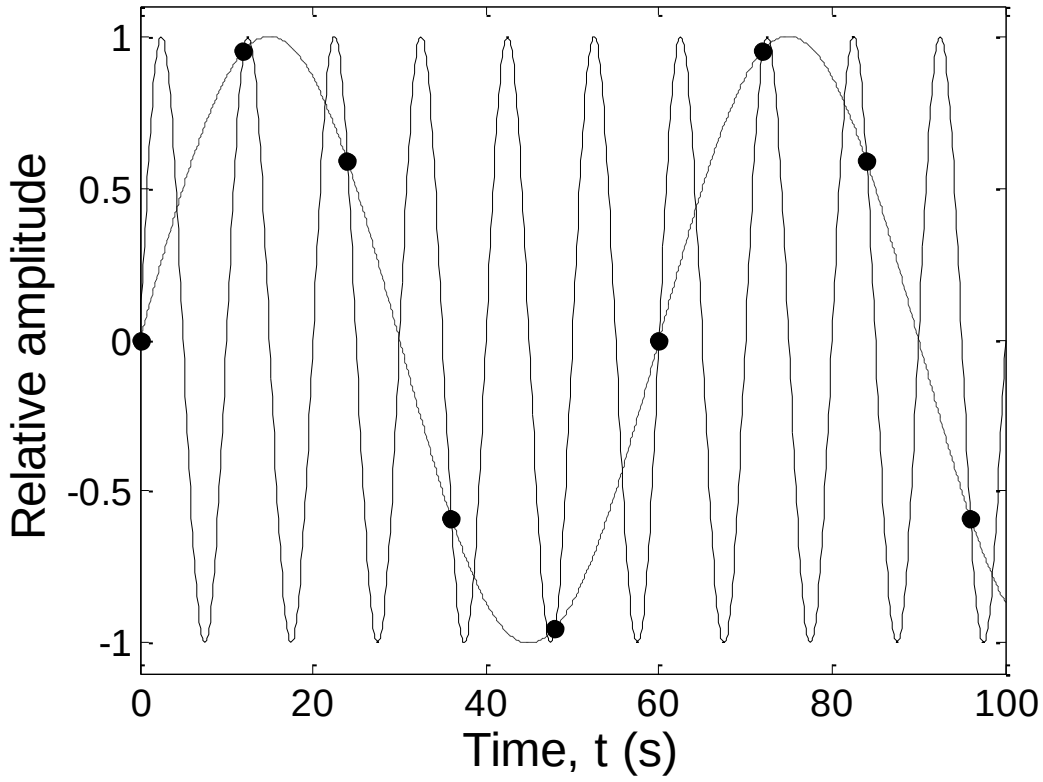


Figure 4.4: The solid line shows a pure sine wave with frequency 1/10 Hz. The black circles represent the sine wave if it were sampled at a rate of 1/12 Hz. However, the black dots also correspond to a sine wave of frequency 1/60 Hz $\left(\frac{1}{10}-\frac{1}{12}\right)$, shown by the dashed line. If a 1/10 Hz sine wave was sampled at 1/12 Hz, the Fourier transform would mistake the original sine wave as a 1/60 Hz sine wave. This leads to frequency aliasing when data are digitized too slowly.

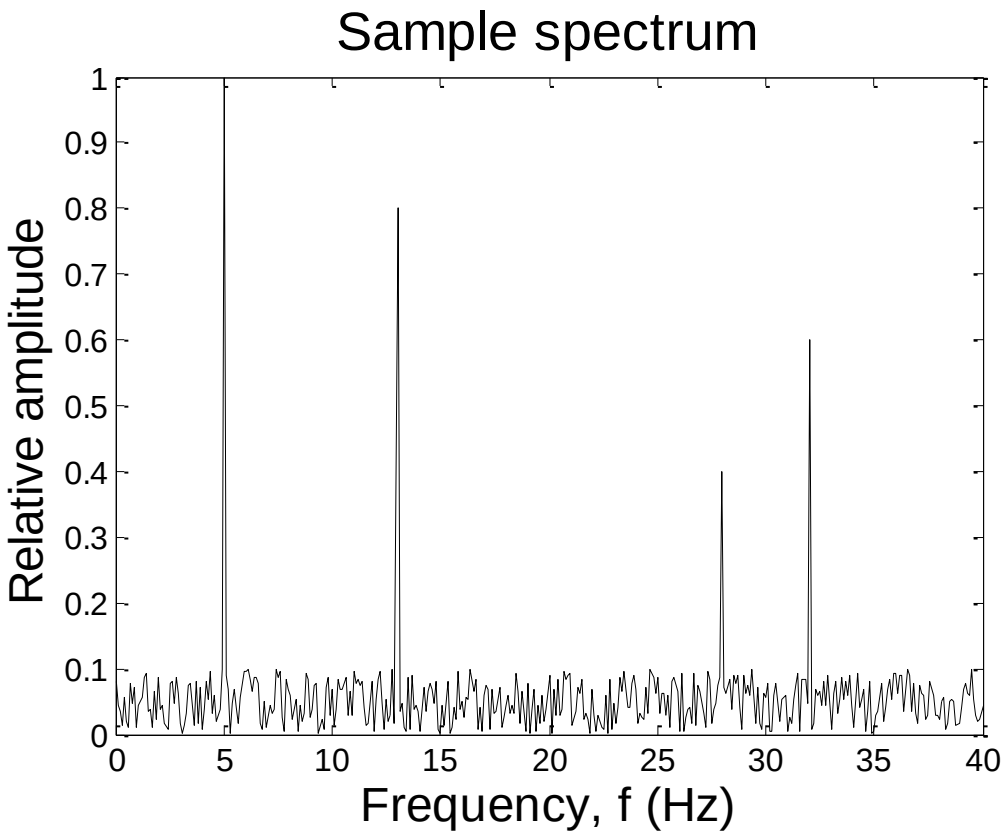


Figure 4.5: A sample spectrum created to demonstrate the effects of sampling at an insufficiently high frequency. The spectral peaks occur at frequencies: 5 Hz, 13 Hz, 28 Hz, and 32 Hz. See Figure 4.6 for the final result of sampling at a frequency of 10 Hz. See Figure 4.7 to see why that is the case.

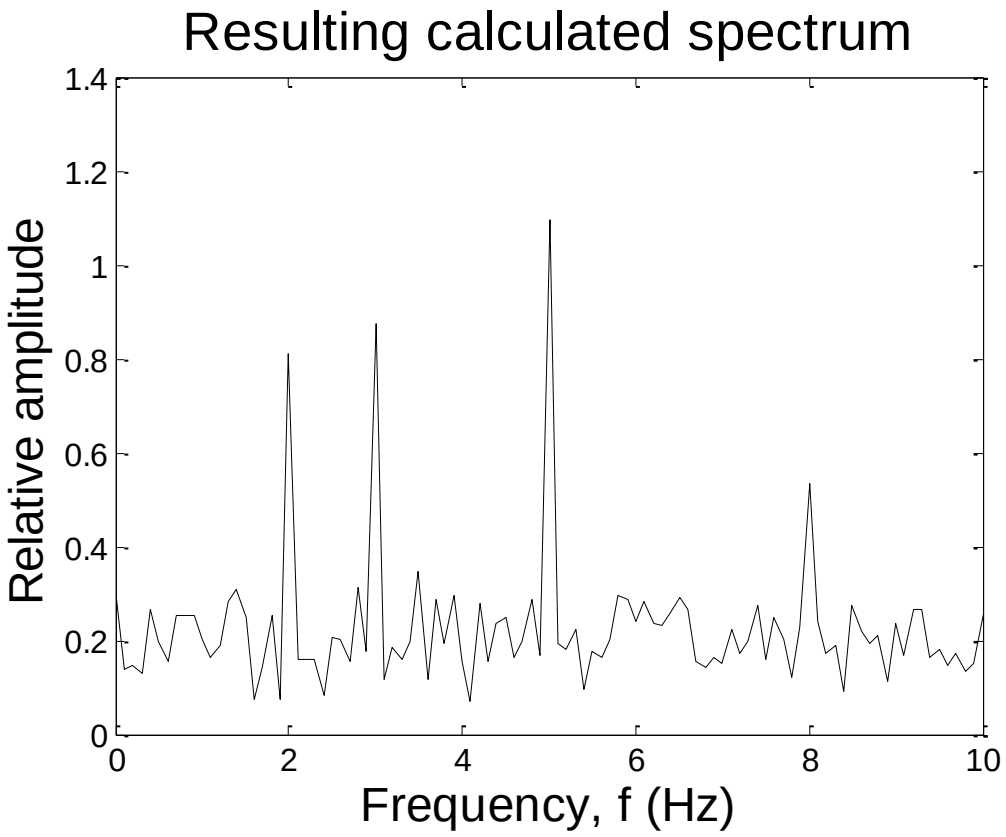


Figure 4.6: The resulting calculated spectrum when the data set that was created in Figure 4.5 is sampled at a frequency of 20 Hz. Notice that the four peaks present in the sample spectrum (5 Hz, 13 Hz, 28 Hz, and 32 Hz in Figure 4.5) appear at 5 Hz, 3 Hz, 8 Hz, and 2 Hz, respectively. Figure 4.7 shows the relation between the sample spectra and this calculated spectra.

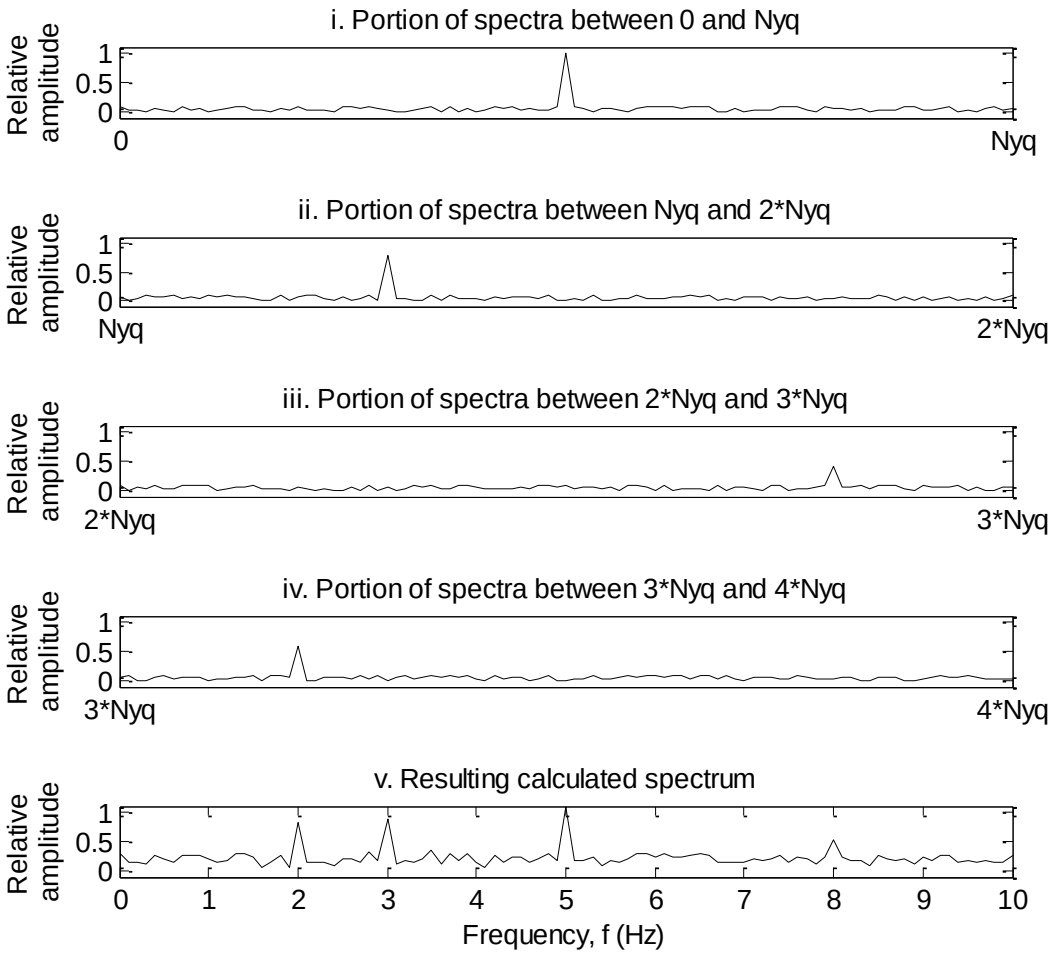


Figure 4.7: (i) The portion of the original spectrum in Figure 4.5 between 0 Hz and the Nyquist frequency (10 Hz). (ii) The portion of the original spectrum in Figure 4.5 between the Nyquist frequency and twice the Nyquist frequency. (iii) The portion of the original spectrum in Figure 4.5 between twice the Nyquist frequency and thrice the Nyquist frequency. (iv) The portion of the original spectrum in Figure 4.5 between thrice the Nyquist frequency and four times the Nyquist frequency. (v) The spectrum resulting from sampling the data for Figure 4.5 at 20 Hz and (as a result) the sum of plots (i) through (iv) as they are lined up. Since the data in this example are sampled at 20 Hz, frequencies in plots ii, iii, and iv are mapped onto the range $[0, 10\ \text{Hz}]$. This can be seen in more detail in Figure 4.6.

4.4 Fast Fourier transform

To summarize thus far, the Fourier transform of a set of data from the radar is:

$$\tilde{G}(f) = \int_{-\infty}^{\infty} \tilde{g}(t) \mathrm{e}^{-2\pi i f t} dt \tag{4.17}$$

where $\tilde{g}(t)$ is the radar data given by:

$$\tilde{g}(t) = g(t)h(t)w(t) \tag{4.18}$$

where $g(t)$ is the continuous atmospheric phenomenon, $h(t)$ is the window function, and $w(t)$ is the sampling function. A simpler, more computer friendly method of computing Equation 4.17 is to implement the definition of the discrete Fourier transform (DFT) instead of the continuous version above. The DFT is defined as:

$$\hat{G}_k \equiv \sum_{n=0}^{N-1} \hat{g}_n e^{-\frac{2\pi i k n}{N}} \tag{4.19}$$

for N discrete points, where $k \in \mathrm{Z}$, and $\hat{g}_n$ is the discrete windowed radar data (i.e. $g(t)h(t)$).

4.5 Non-Uniformly Spaced Data

The radar data are non-uniformly spaced in time for a variety of reasons. On long time scales, interruptions to the radar's operation (due to, for example, power interruption or equipment failure) require on-site technical intervention from London, ON. This is not always feasible on a short timescale, leading to missing periods of data. While Negrocreek and Markstay are approximately a 2-3 and 6-8 hour drive one way respectively, Eureka is an expensive, fly-in only service run. Figure 4.8 depicts when the radar sites were operational.

Even when the radars are operational, the data sampling rate is inconsistent, as shown in Figure 4.9**Error! Reference source not found.**. In some cases, the data were gathered for previous studies, which may have been collected at different sampling rates. Some days the radar was shut down for a few hours to allow for repairs and maintenance.

If a wind velocity differs from the nearest 10 neighbouring measurements by more than 10 m/s (user-definable), it is discarded as it is unlikely to be atmospheric data in the

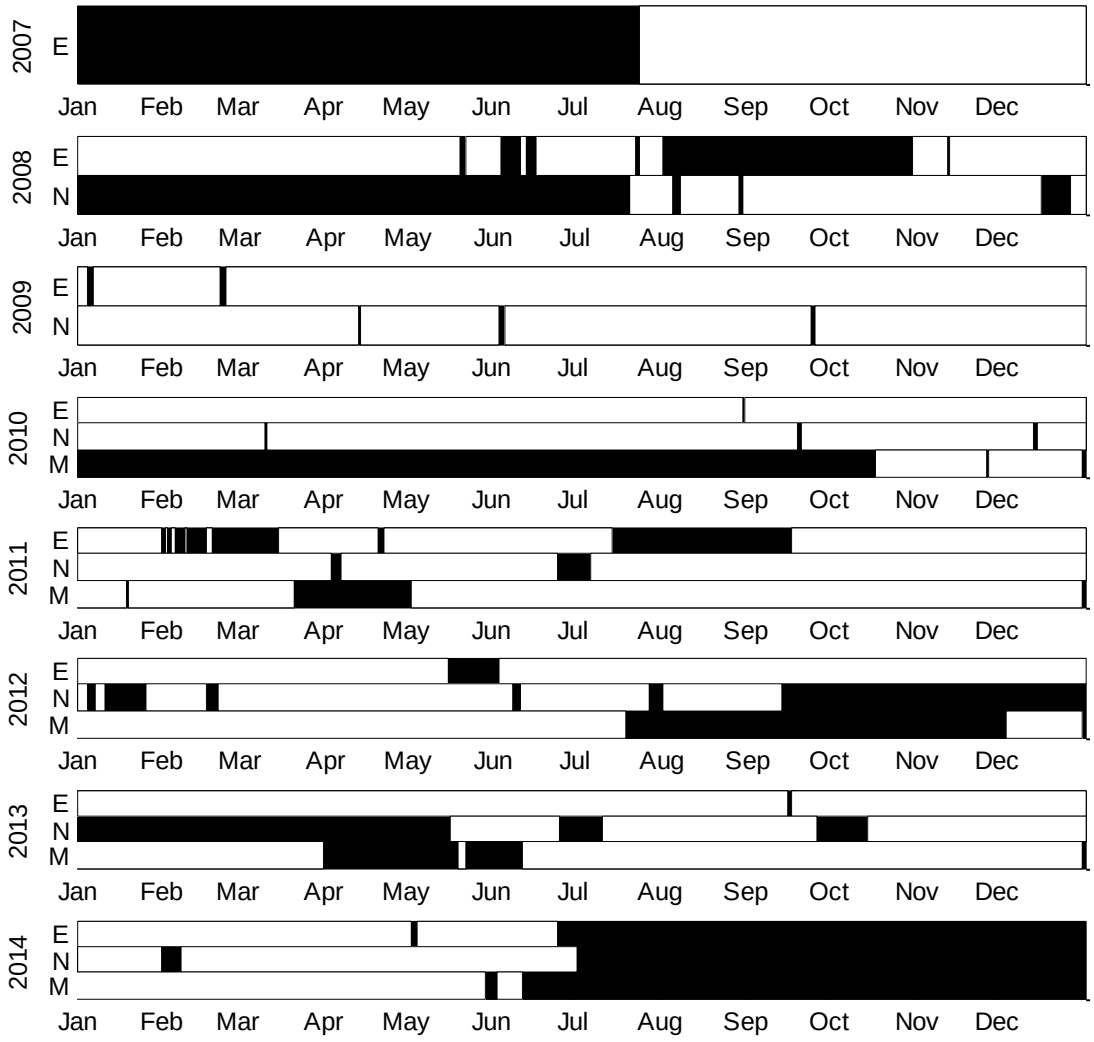


Figure 4.8: Black represents the radar site being non-operational. Where the radar sites are operational (white), uniformly spaced data may still be unavailable. The number on the ordinate is the year. For ease of reading, the radar site names (Eureka, Negrocreek, and Markstay) are shortened to just the first initial (E, N, and M respectively).

expected radar volume (recall range aliasing, non-atmospheric scatterers, and detection through sidelobes). For some atmospheric conditions, the radar is unable to measure within the extremes of its range (e.g. not enough power, pulse saturation), resulting in missing data for low (roughly 1.0-3.0 km altitude) and high (roughly 12.0-14.0 km altitude) ends of the radar's range.

While there are numerous methods of coping with non-uniformly spaced data (e.g., Lomb-Scargle (Lomb 1976, Scargle 1982), MUSIC (Marple 1987,etc.), this thesis examines two solutions: (1) using averaging of bins and interpolating when necessary to use the fast Fourier transform and (2) using the date-compensated discrete Fourier transform.

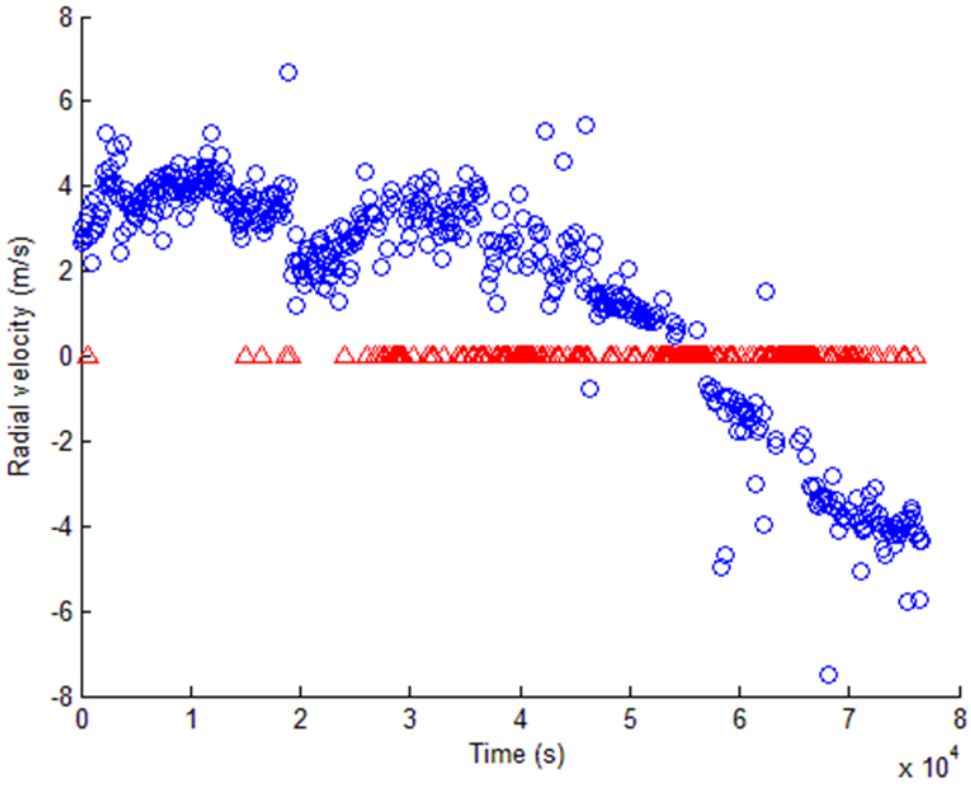


Figure 4.9: A typical data set over the span of a day. Each of the points is a minute apart. The blue circles represent minutes where there is an accepted radial velocity. The red triangles represent minutes for which data are missing (for a variety of reasons—see text).

4.6 Bin averaging and interpolation

In an attempt to reduce the amount of interpolating necessary due to short-term (on the order of minutes to a few hours) non-uniform data, the radar data are averaged in bins of 15 minutes. When there are empty bins, the gaps are interpolated. The interpolation is linear between the data points immediately before and after the empty bins, with random noise added. The same standard deviation is used for the interpolated data as exists in the bins near the missing one(s) (5 points before and 5 points after). Missing gaps of data are not interpolated if the gap spans six hours or longer. As such, one month may have multiple "strands" of data to compute the spectra and spectral slope. In this case, the longest data strand is selected as representative for the month. The Fourier transform is then calculated using Matlab's built in FFT function.

Matlab's built-in FFT uses a combination of algorithms, including: the Cooley-Tukey algorithm (Cooley & Tukey 1965), a variation of Cooley-Tukey (Oppenheim 1989), a prime factor algorithm (Oppenheim 1989), a split-radix algorithm (Duhamel & Vetterli 1990), and Rader's algorithm (Rader 1968). For more information on the details of solving the DFT, see Matlab's help page on FFT.

4.7 Date-compensated discrete Fourier transform

The second method this thesis uses to cope with non-uniformly spaced data is the date-compensated discrete Fourier transform (DCDFT), as proposed by Ferraz-Mello (1981), which computes a power spectral density estimate for discrete, unequally time-spaced data. For a particular frequency, ω, this method involves fitting the following three curves to the data (after they are orthonormalized in a Gram-Schmidt fashion and the mean of the data is removed), as presented in Ferraz-Mello (1981):

$$H_0(t) = 1 \tag{4.20}$$

$$H_1(t) = \cos(2\pi\omega t) \tag{4.21}$$

$$H_2(t) = \sin(2\pi\omega t) \tag{4.22}$$

This fit provides a power spectral density estimate to the data for frequency ω. To obtain a power spectrum for this thesis, this process is repeated for frequencies evenly spaced between $\pm\frac{N-1}{T}$ where N is number of data points in the signal. In the special case of equally time-spaced data, the DCDFT simplifies to the discrete Fourier transform.

4.8 Spectral slope

Once the Fourier transform of the radial velocities is determined and converted into a power spectrum, the slope is determined. All of the slopes are determined using a least squares fit to the frequencies between 6 hours and 2 days (approximately 4.6×10^{-6} Hz $- 5.8 \times 10^{-6}$ Hz). The upper bound of the period range (i.e. 2 days) is the upper boundary to gravity wave periods. The lower bound (i.e. 6 hours) was chosen such that it does not include the shallowing of the noise floor. A sample spectrum is shown in Figure 4.10 with the green line showing the spectral slope for the section included in the calculation.

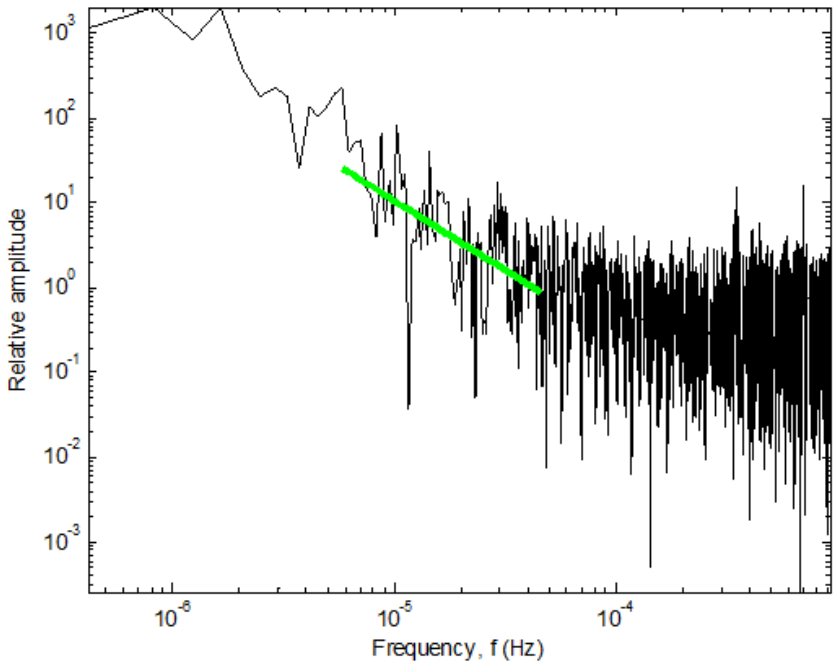


Figure 4.10: A sample spectrum (Eureka, February 2010, 3.0 km range) with the spectral slope shown in green for the period range 6 hours to 2 days.

Chapter 5

5 Spectral method selection

This thesis examines two different ways of determining the monthly spectra with four different windows at three different radar sites. To save on computation, the different methods are illustrated in detail for only one of the sites (Negrocreek, a mid-latitude site) for one year, with some comparisons to a second site (Eureka, a high-latitude site). All sites, and a wider coverage of years, are presented once the methodologies are optimized.

The year 2009 was selected as the comparison year since both Eureka and Negrocreek were operational for the majority of that year. The "best" method of determining spectra is chosen from this comparison and then used to compare all three radar sites (Eureka, Negrocreek, and Markstay) over their operational periods (7, 6, and 4 years respectively).

5.1 Definition of "best"

Each of the monthly spectra for Negrocreek for each altitude and each window type was examined visually and the "breakpoint" frequency was selected manually. The breakpoint frequency separates the portion of the spectrum with an identifiable non-zero slope—sometimes referred to as the "signal" portion—and from the noise floor. Figure 5.1 shows a sample spectra with the breakpoint indicated. The breakpoint value is used to determine which method is "best"; the higher frequency the breakpoint, the more of the spectra can be seen (i.e. is not hidden in noise). Figure 5.2 to Figure 5.5 compare the breakpoint frequency between different windows and methods.

The plots depicting the breakpoint frequency as a function of altitude and month, Figure 5.2 to Figure 5.5, show the difference in the breakpoint frequency between two methods (i.e. FFT or DCDFT) with the same window. The methods being compared are noted on the colour bar axis. The site and window that is being held constant is noted at the top of the figures. Similarly, Figure 5.10 to Figure 5.16 show the difference in the breakpoint frequency between two windows with the same method. The windows being

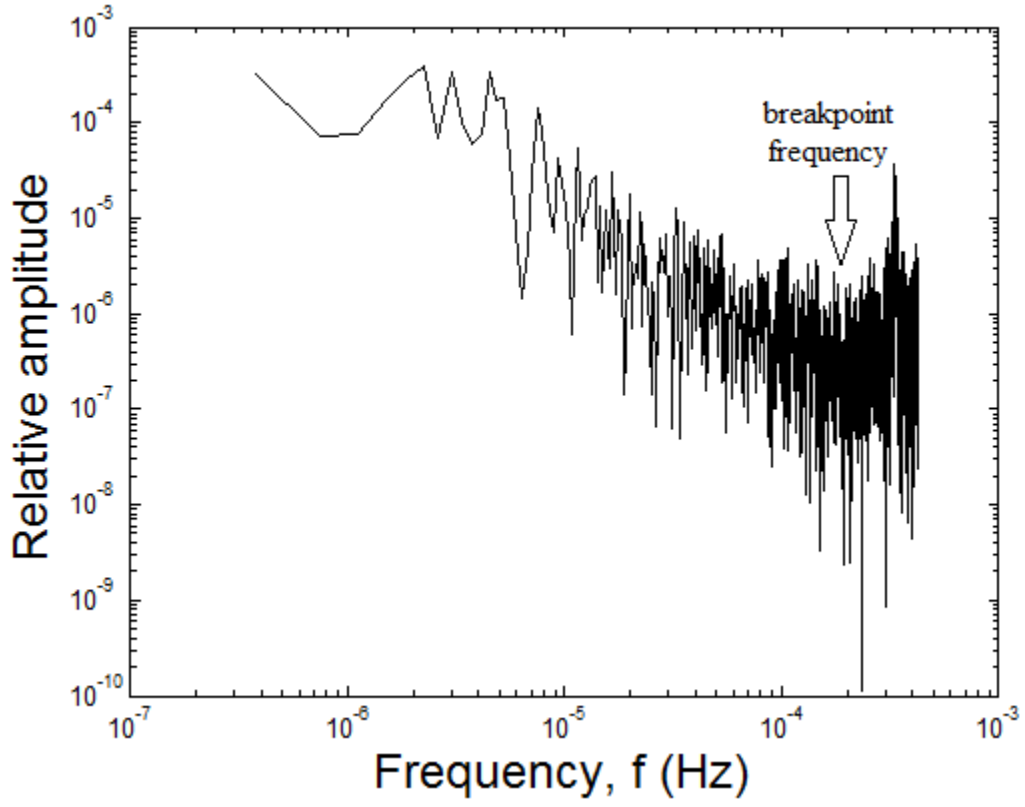


Figure 5.1: A typical monthly spectrum, indicating the breakpoint frequency. This spectrum is for Negrocreek, January 2009 using the DCDFT method with the Hamming window.

compared are noted on the colour bar axis (the boxcar with 10% cosine taper window is shortened to "10% cos"). The site and method that is being held constant is noted at the top of the figures. The colour indicates which of the method and window combinations has a higher break point (i.e. lower noise floor and are preferred). A more intense colour implies that one method/window combination has a higher breakpoint frequency than the other. However, the most intense colours do not necessarily imply a large difference in breakpoint frequency. For spectra that did not have a noise floor (i.e. the majority of the spectra is noise), the breakpoint frequency was set to zero. Where this is the case for one window (at a particular range gate in a specific month) and not the second, that colour-coding at the appropriate altitude, month coordinate will appear as an intense colour favouring the second window. Yellow values imply that the two windows have the same

breakpoint frequency. White values imply that the spectra for both windows did not have a noise floor. These plots demonstrate this definition of best.

There is a second criterion to the "best" as well. Atmospheric phenomena, such as gravity waves and their spectra, are expected to be fairly continuous as a function of altitude and time. Since the spectra used in this thesis are monthly averages, the spectra are not necessarily expected to be continuous from month to month; however, they should still be roughly continuous as a function of altitude. Spectral methods that yield erratic spectral slopes as a function of altitude are probably not an accurate description of the physical scenario. Therefore, the second criteria to being the "best" is that the slopes are smooth functions with altitude.

5.2 The "best" method

Under the breakpoint definition of "best", the fast Fourier transform (FFT) method appears better than the date-compensated discrete Fourier transform (DCDFT) for all windows.

While the FFT appears to be the better choice, the spectral slopes computed with the FFT are more erratic than those computed with the DCDFT. Figure 5.6 to Figure 5.9 depict the spectral slopes using the FFT and DCDFT methods for a representative window for both Negrocreek and Eureka in 2009. These plots depict the erratic behaviour of the spectral slopes when analyzed with the FFT method. The apparent better choice of breakpoint frequency for the FFT is possibly an artifact, as is described in Chapter 6.

It is also possible that the FFT frequencies have been aliased, due to the time-interval binning of data. More analysis on the spectral methods is needed to reveal if this is the case. For these reasons, the DCDFT has been chosen as the "best" method.

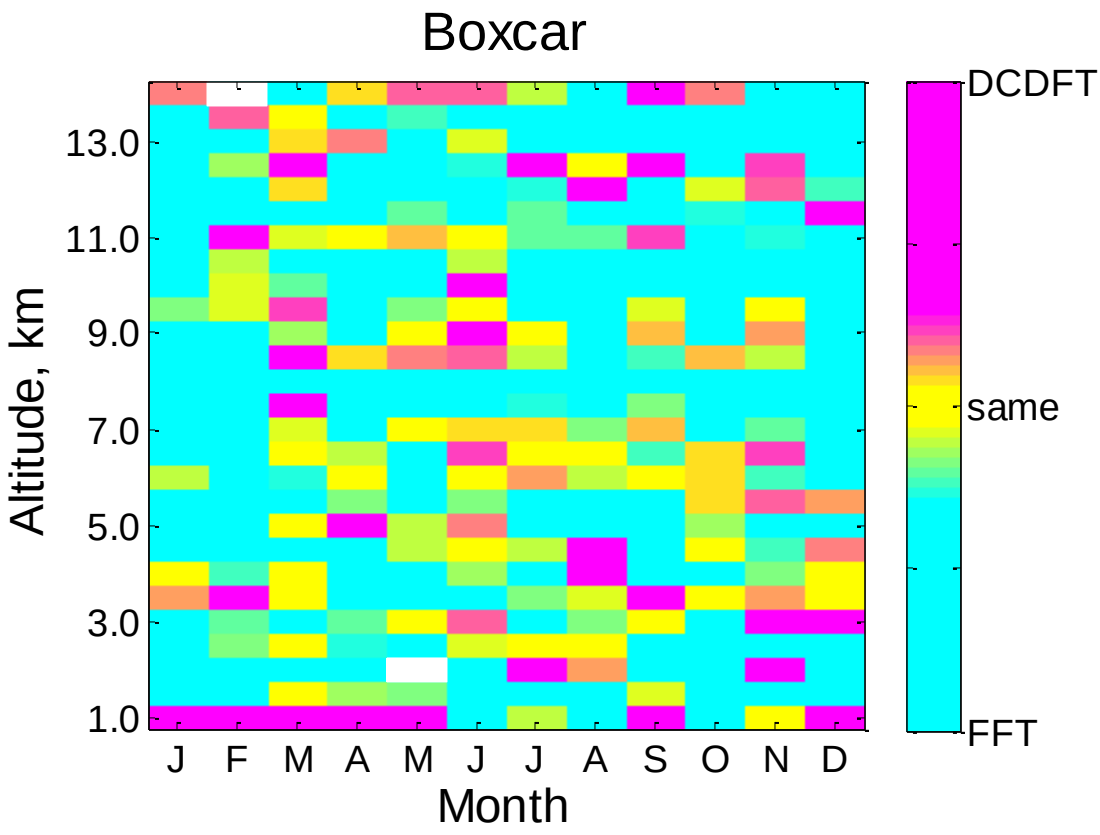


Figure 5.2: Comparisons between frequency break points at the junction between the signal and the noise floor for the DCDFT compared to the FFT for the boxcar window. In this case, 80 comparisons show a preference (i.e. a larger break point frequency) for the DCDFT while 241 show a preference for the FFT. Since this plot is overwhelmingly cyan, the FFT yields a lower noise barrier than the DCDFT.

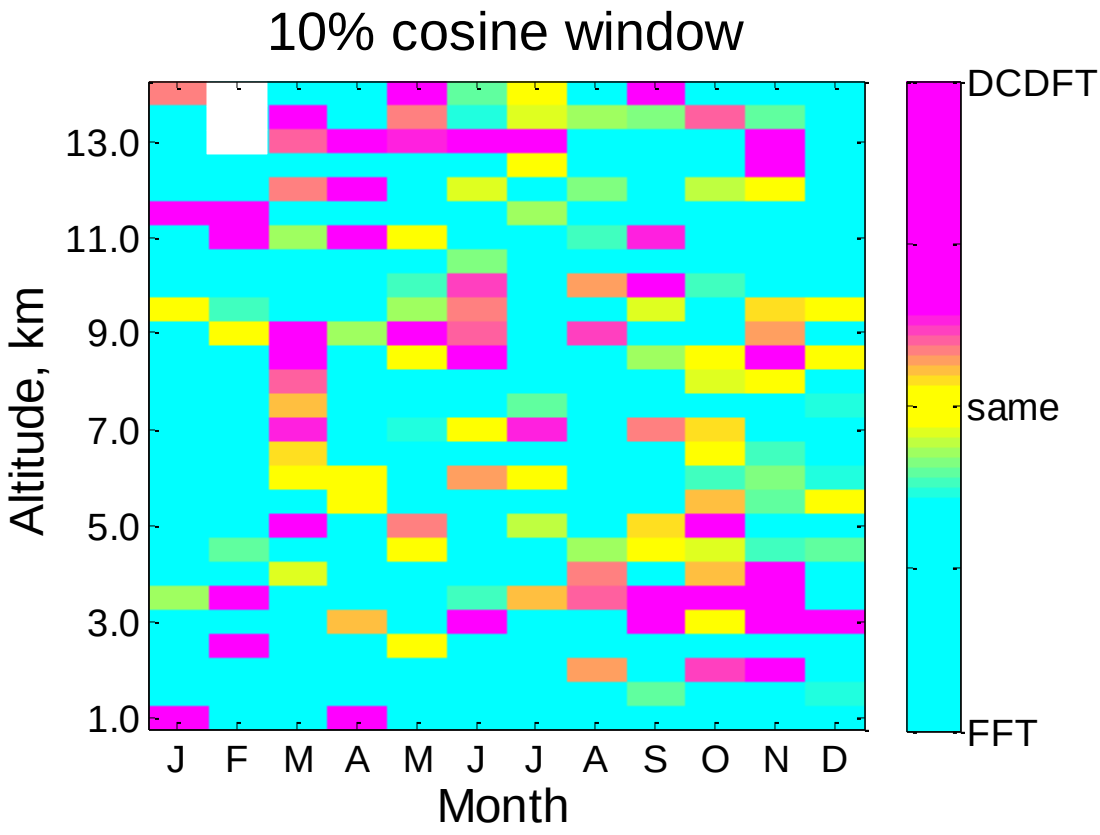


Figure 5.3: Comparisons between frequency break points at the junction between the signal and the noise floor for the DCDFT compared to the FFT for the boxcar with a 10% cosine taper window. In this case, 75 comparisons show a preference (i.e. a larger break point frequency) for the DCDFT while 243 show a preference for the FFT. Since this plot is overwhelmingly cyan, the FFT yields a lower noise barrier than the DCDFT.

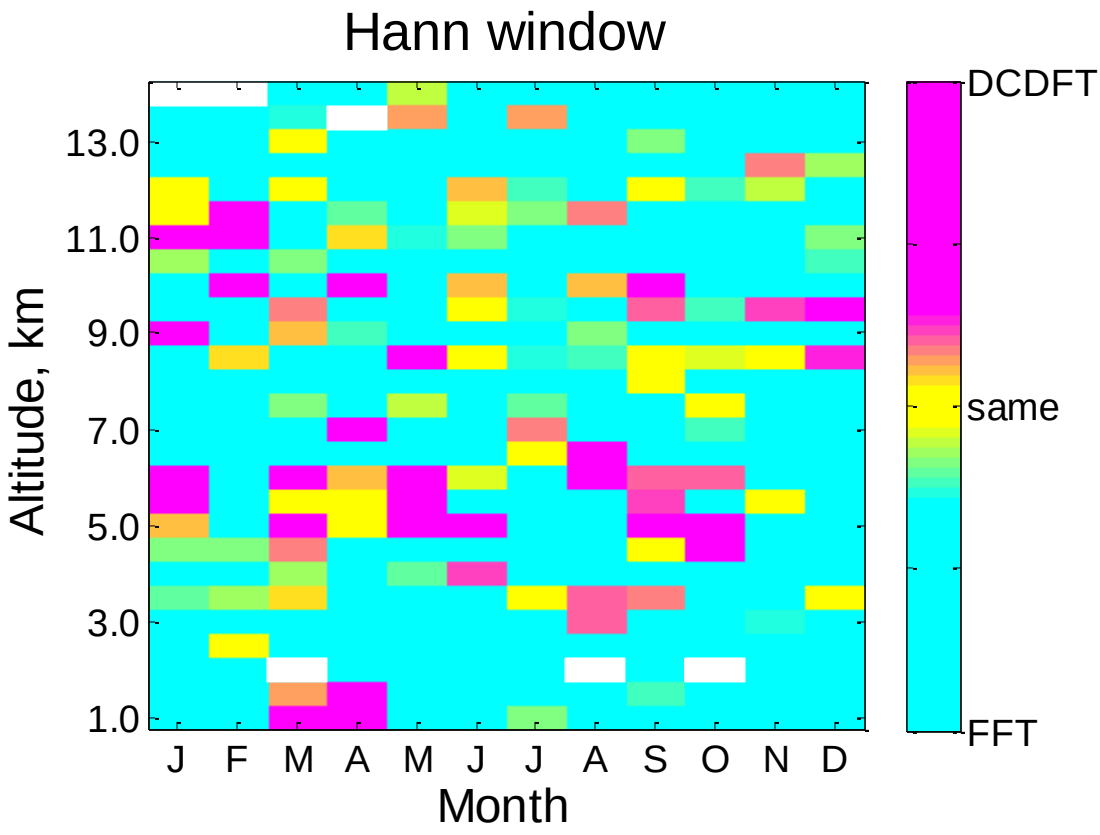


Figure 5.4: Comparisons between frequency break points at the junction between the signal and the noise floor for the DCDFT compared to the FFT for the Hann window. In this case, 60 comparisons show a preference (i.e. a larger break point frequency) for the DCDFT while 255 show a preference for the FFT. Since this plot is overwhelmingly cyan, the FFT yields a lower noise barrier than the DCDFT.

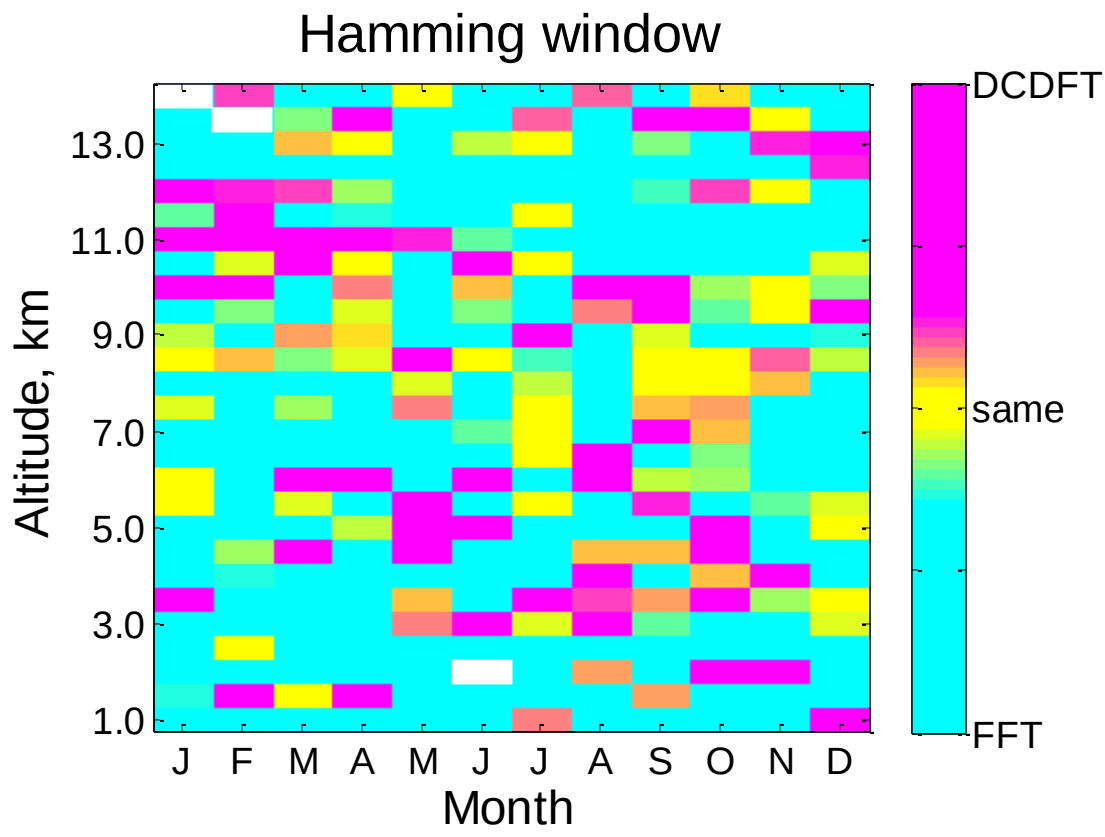


Figure 5.5: Comparisons between frequency break points at the junction between the signal and the noise floor for the DCDFT compared to the FFT for the Hamming window. In this case, 92 comparisons show a preference (i.e. a larger break point frequency) for the DCDFT while 255 show a preference for the FFT. Since this plot is overwhelmingly cyan, the FFT yields a lower noise barrier than the DCDFT.

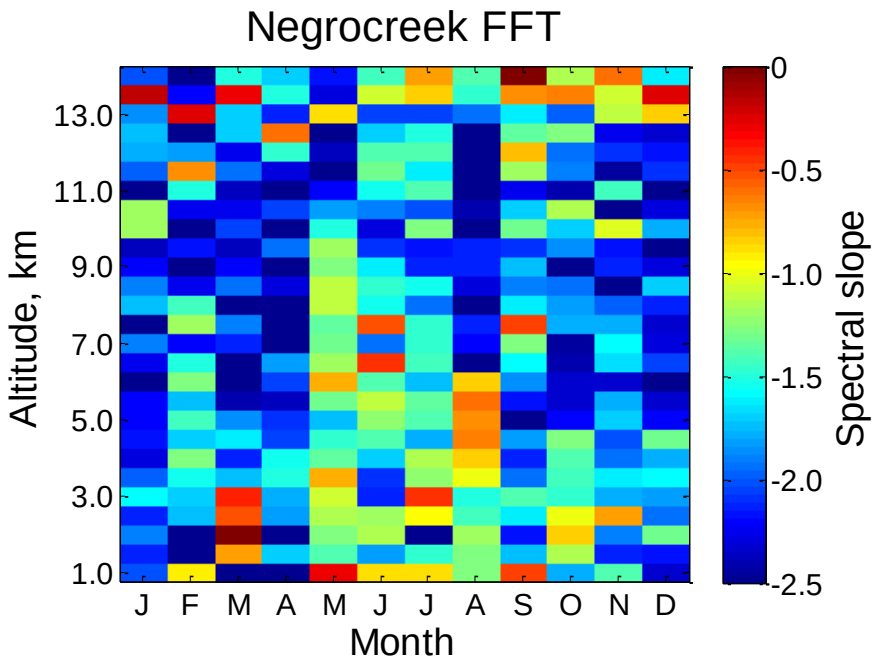


Figure 5.6: Spectral slope values as a function of altitude and month for Negrocreek 2009, using the FFT and a representative window (the Hann window).

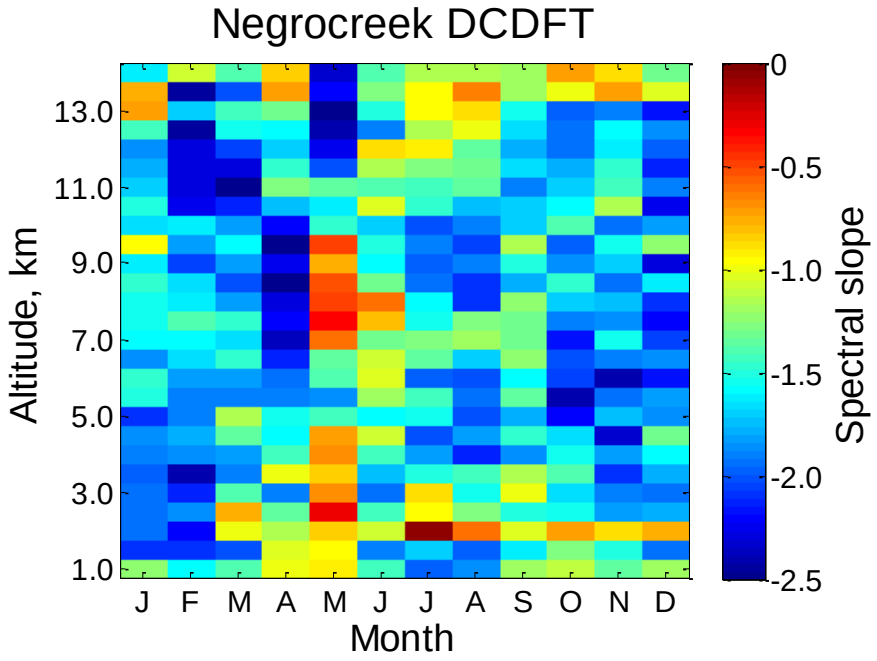


Figure 5.7: Spectral slope values as a function of altitude and month for Negrocreek 2009, using the DCDFT and a representative window (the Hann window).

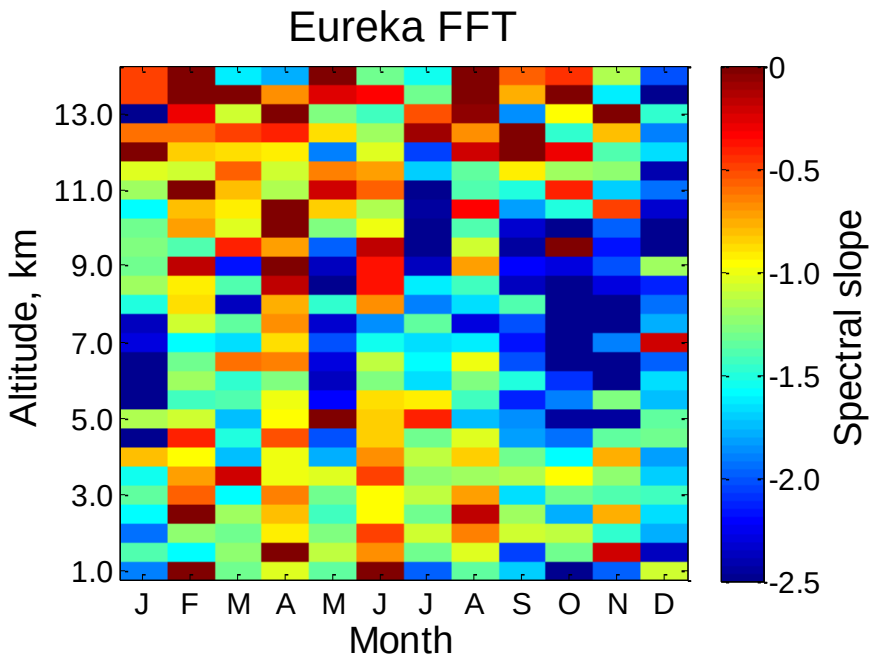


Figure 5.8: Spectral slope values as a function of altitude and month for Eureka 2009, using the FFT and a representative window (the Hann window).

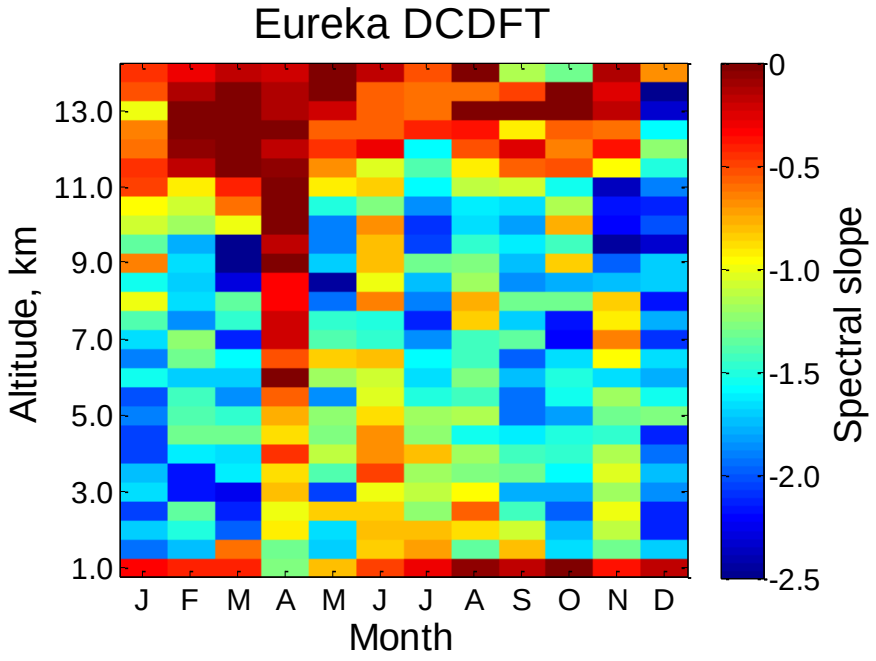


Figure 5.9: Spectral slope values as a function of altitude and month for Eureka 2009, using the DCDFT and a representative window (the Hann window).

5.3 The “best” window

The breakpoints for each method (FFT and DCDFT) were compared for different windows to determine which window is the best for each method. Under the previously given definition of best, the Hann window (which aims to suppress the effect of all sidelobes) is the best window to use with the FFT method and the Hamming window (which aims to suppress the first sidelobe) is the best window to use with the DCDFT method.

For both the FFT and DCDFT, the boxcar window was worse than the boxcar with 10% cosine taper. It was therefore discarded after “round one”. For the FFT, since the Hann window and Hamming window comparison lacked such a clear divide, both windows were compared to the boxcar with 10% cosine taper. For the DCDFT, the Hann window was worse than the Hamming window. Therefore, the Hann window was discarded and the boxcar with 10% cosine taper window was compared with the Hamming window. Figure 5.10 to Figure 5.16 show each of these comparisons.

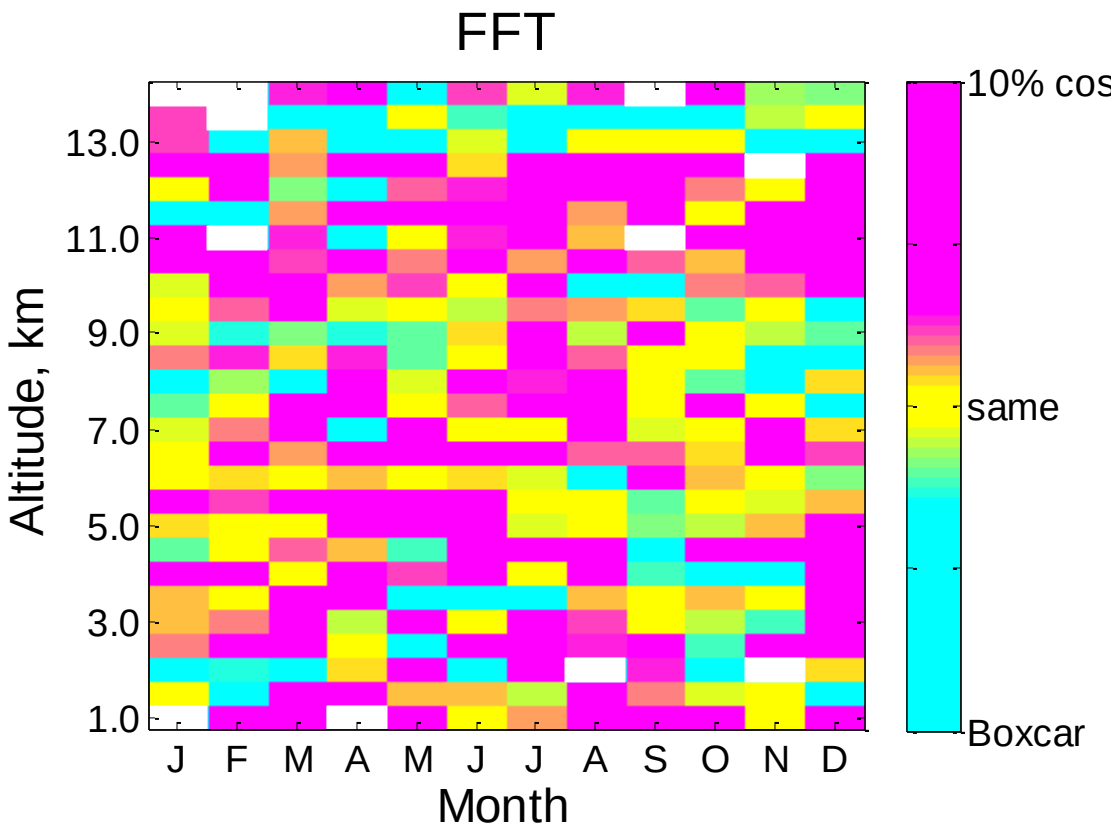


Figure 5.10: Comparisons between breakpoint frequency for a boxcar with 10% cosine tapering window and a boxcar window for the FFT method, as described in the text. In this case, 202 comparisons show a preference (i.e. a larger breakpoint frequency) for the boxcar with 10% cosine tapering window while 104 show a preference for the regular boxcar window. Since this plot is overwhelmingly magenta, the boxcar with 10% cosine tapering yields a lower noise barrier than a regular boxcar window, therefore, it is disregard for further analysis with the FFT.

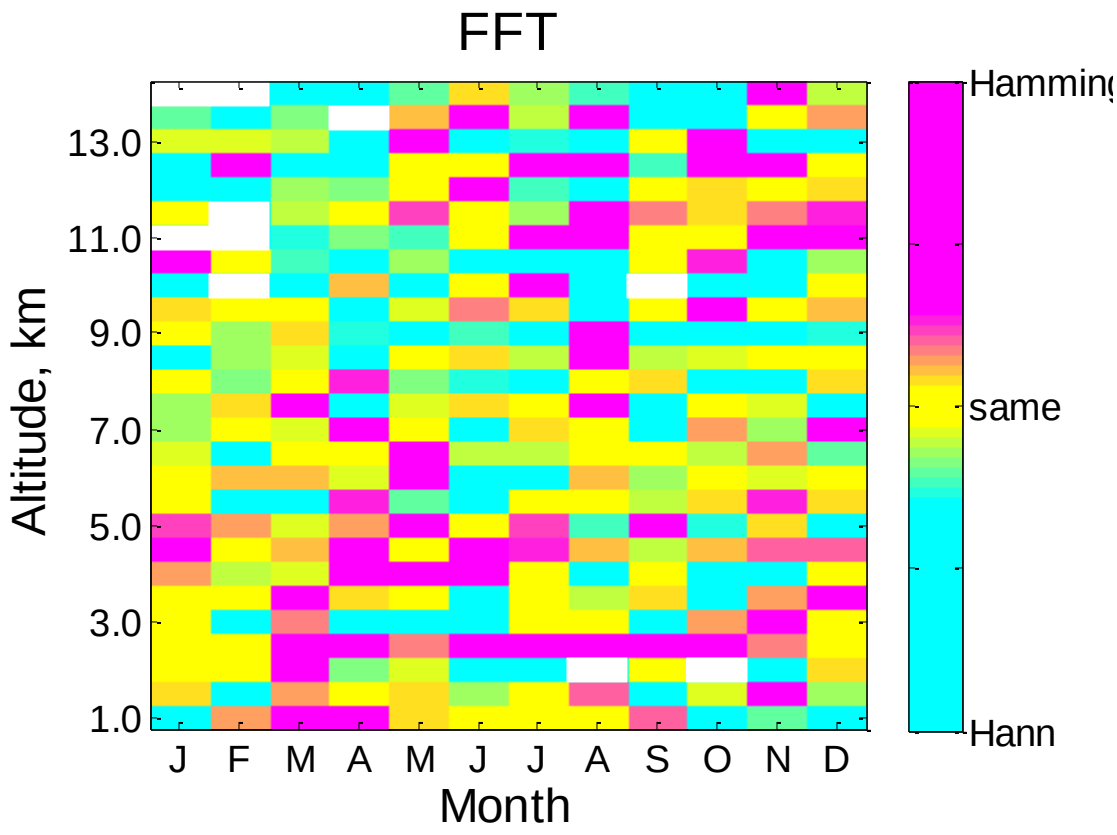


Figure 5.11: Comparisons between breakpoint frequency for a Hamming window and a Hann window for the FFT method, as described in the text. In this case, 133 comparisons show a preference (i.e. a larger break point frequency) for the Hamming window while 168 show a preference for the Hann window. The Hann window yields a lower noise barrier than the Hamming window. Both of these windows are compared to the boxcar with 10% cosine tapering window.

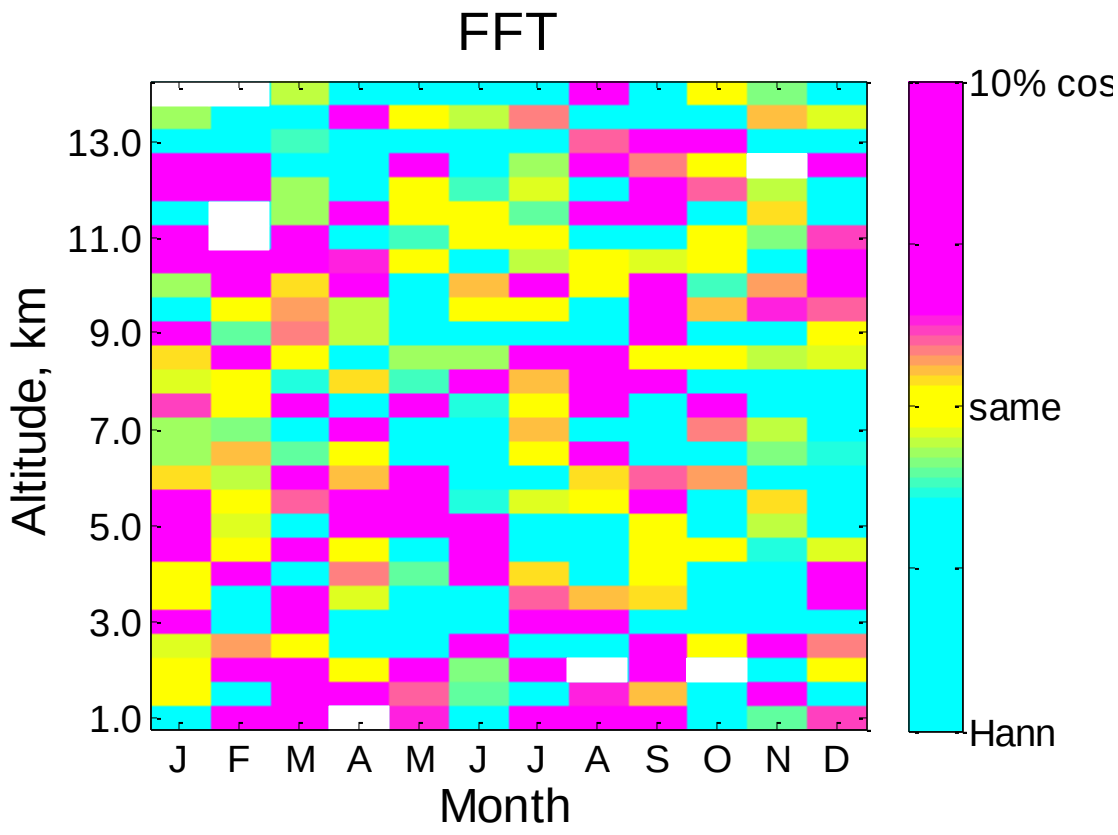


Figure 5.12: Comparisons between breakpoint frequency for a boxcar with 10% cosine tapering window and a Hann window for the FFT method, as described in the text. In this case, 137 comparisons show a preference (i.e. a larger break point frequency) for the boxcar with 10% cosine tapering window while 176 show a preference for the Hann window. The Hann window yields a lower noise barrier than the Hamming window.

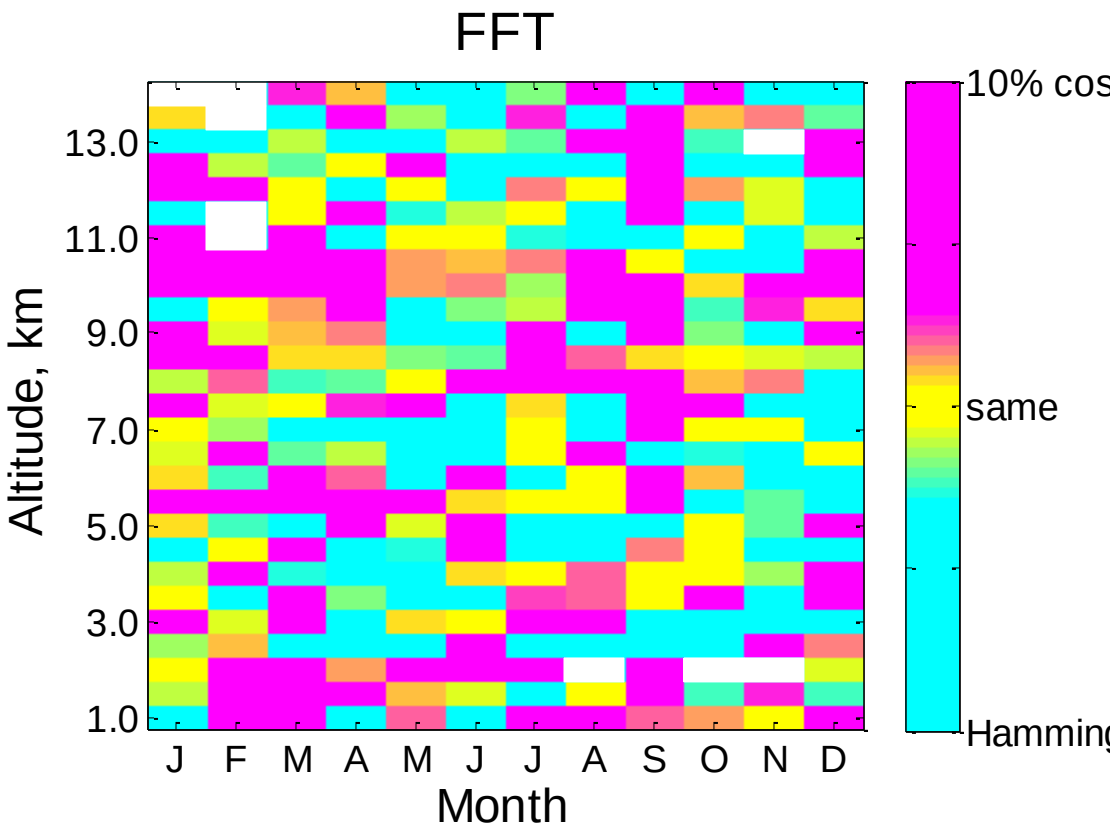


Figure 5.13: Comparisons between breakpoint frequency for a boxcar with 10% cosine tapering window and a Hamming window for the FFT method, as described in the text. In this case, 153 comparisons show a preference (i.e. a larger break point frequency) for the boxcar with 10% cosine tapering window while 158 show a preference for the Hamming window. The Hamming window yields a lower noise barrier than the boxcar with 10% cosine taper window; however, the Hann window yields lower break point values than both of these windows, as shown in Figure 5.11 and Figure 5.12.

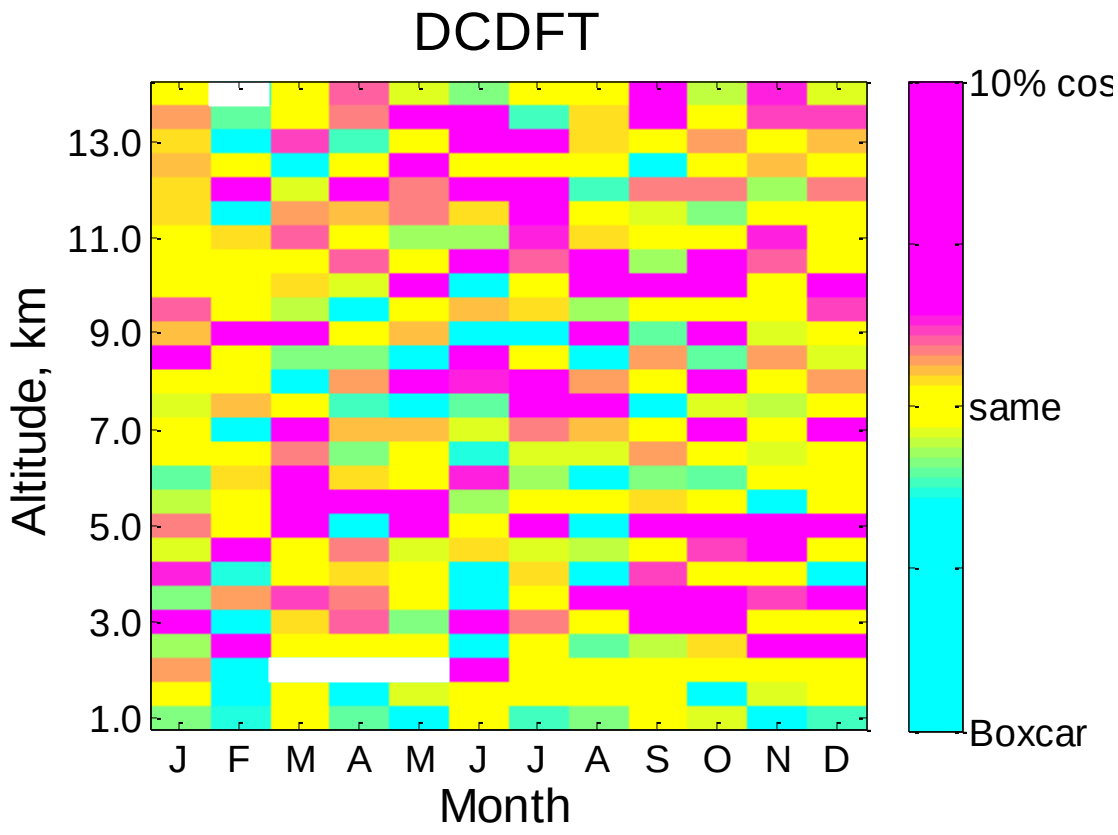


Figure 5.14: Comparisons between breakpoint frequency for a boxcar with 10% cosine tapering window and a boxcar window for the DCDFT method, as described in the text. In this case, 184 comparisons show a preference (i.e. a larger break point frequency) for the boxcar with 10% cosine tapering window while 125 show a preference for the regular boxcar window. The boxcar with 10% cosine tapering yields a lower noise barrier than a regular boxcar window, therefore, it is disregarded the boxcar window for further analysis using the DCDFT.

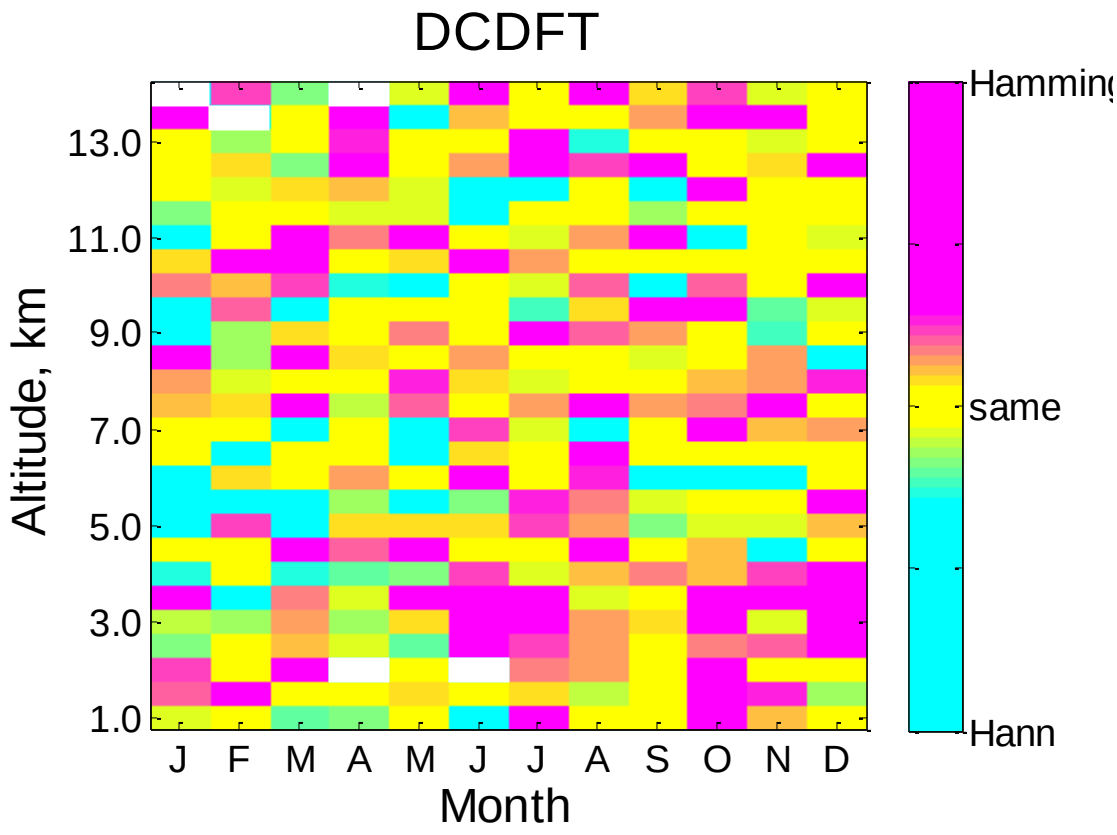


Figure 5.15: Comparisons between breakpoint frequency for a Hamming window and a Hann window for the DCDFT method, as described in the text. In this case, 194 comparisons show a preference (i.e. a larger break point frequency) for the Hamming window while 120 show a preference for the Hann window. The Hamming window yields a lower noise barrier than a Hann window.

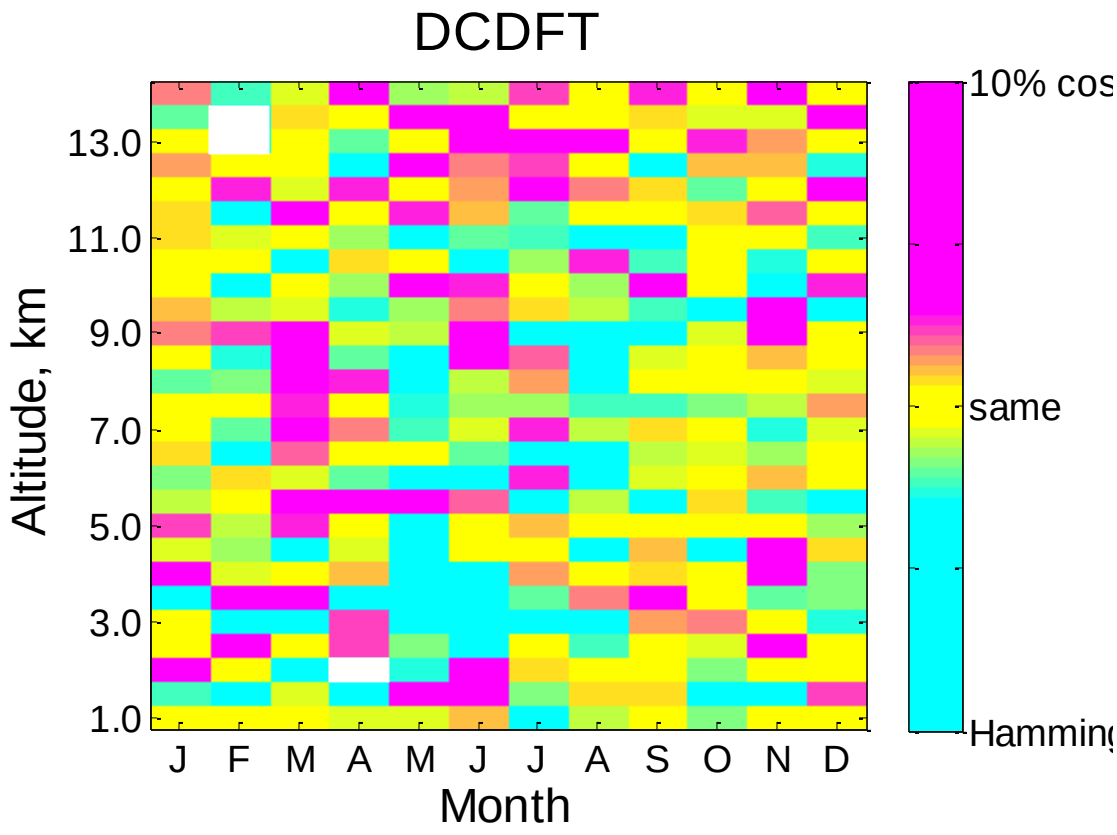


Figure 5.16: Comparisons between breakpoint frequency for a boxcar with 10% cosine tapering window and a Hamming window for the DCDFT method, as described in the text. In this case, 140 comparisons show a preference (i.e. a larger break point frequency) for the boxcar with 10% cosine tapering window while 170 show a preference for the regular Hamming window. The Hamming window yields a lower noise barrier than a boxcar with 10% cosine tapering window. The Hamming window also yields a lower noise barrier than the other windows, therefore, it is chosen as the "best" window to use with the DCDFT.

Chapter 6

6 Results and Interpretation

6.1 Breakpoint frequency

Breakpoint frequencies were hand-selected for monthly spectra recorded at Negrocreek during 2009 for each window type and spectral method. The purpose of this determination was initially to evaluate the best spectral method to use, since optimal fitting depends very much on a suitable determination of the breakpoint. The breakpoint frequency defines where the spectra becomes visible (i.e. where the noise floor starts), and optimal fitting should include only the portion of the spectrum where the signal dominates over the noise. Knowledge about the location of the breakpoint not only allows better determination of the spectral slopes, but also has other information of value. For example, superior windows should have break-points at higher frequencies. The behaviour of the location of the break-point might also give geophysical information, particularly if it shows systematic variations in height and/or time.

The breakpoint frequency does not appear to be a good indicator of what method (i.e. fast Fourier transform (FFT) or date-compensated discrete Fourier transform (DCDFT)) is better; the FFT produces breakpoints at higher frequencies, suggesting that the FFT with interpolated data is superior to the DCDFT. However, there may be a reasonable explanation. For the FFT, we binned the data into 15 minute intervals and time-averaged the data points. This results in points being relocated in time. For example, if a point occurs at 6:02 and we use 15 minute bins starting on the hour, quarter past, half-hour, and quarter to, then this point is relocated to 6:07 (i.e. the midpoint of the 6:00-6:15 bin). The problem is compounded by the fact that different bins have a different number of points, since the data are not equally spaced in time; one bin might have 3 points while another may have 7. The net result of this is that points are allocated to the wrong time-marker, which must introduce additional (incorrect) Fourier components to the spectrum. This adds spectral power to the spectrum in a non-uniform manner, so the noise is not white noise (i.e. frequency dependent). Indeed the noise is

frequency dependent and the problem is exacerbated by the fact that, due to limited funding, we had to reduce the rate of data collection in the later years of the study. Hence the fact that the breakpoint for the standard FFT is further into the higher frequencies is not an indicator of better data, but rather an indicator of added non-real spectral content due to incorrect sampling. In order to confirm this, we looked at the variability of the spectral slopes. As discussed in the previous chapter, it is clear that there is greater variability in the spectral slopes determined by the FFT method than from the DCDFT method. For these reasons, we adopted the DCDFT as our standard for future analysis—partly because the data points are correctly allocated in time and partly because of the evidence provided by the variability in slopes.

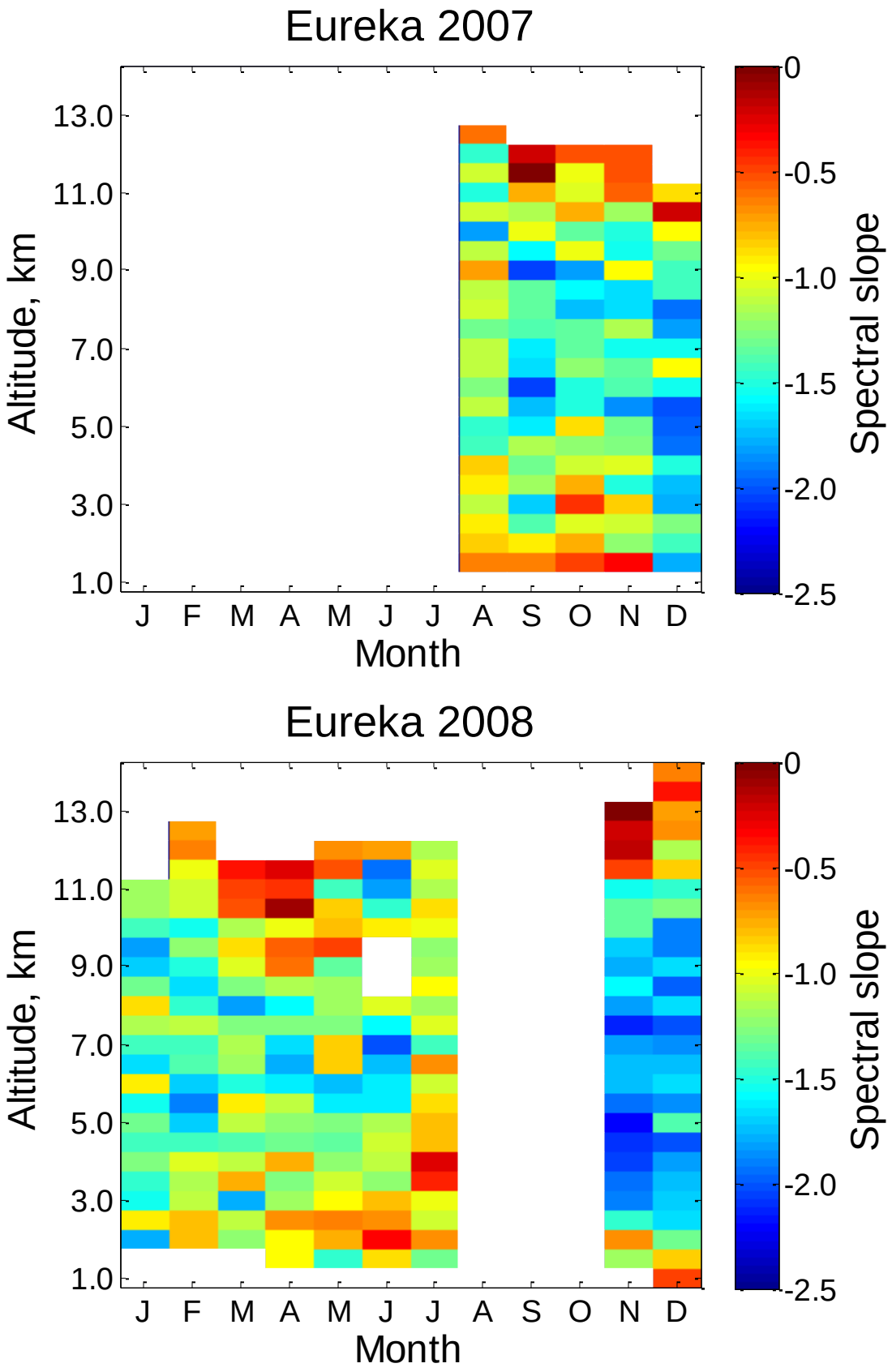
6.2 Spectral slopes

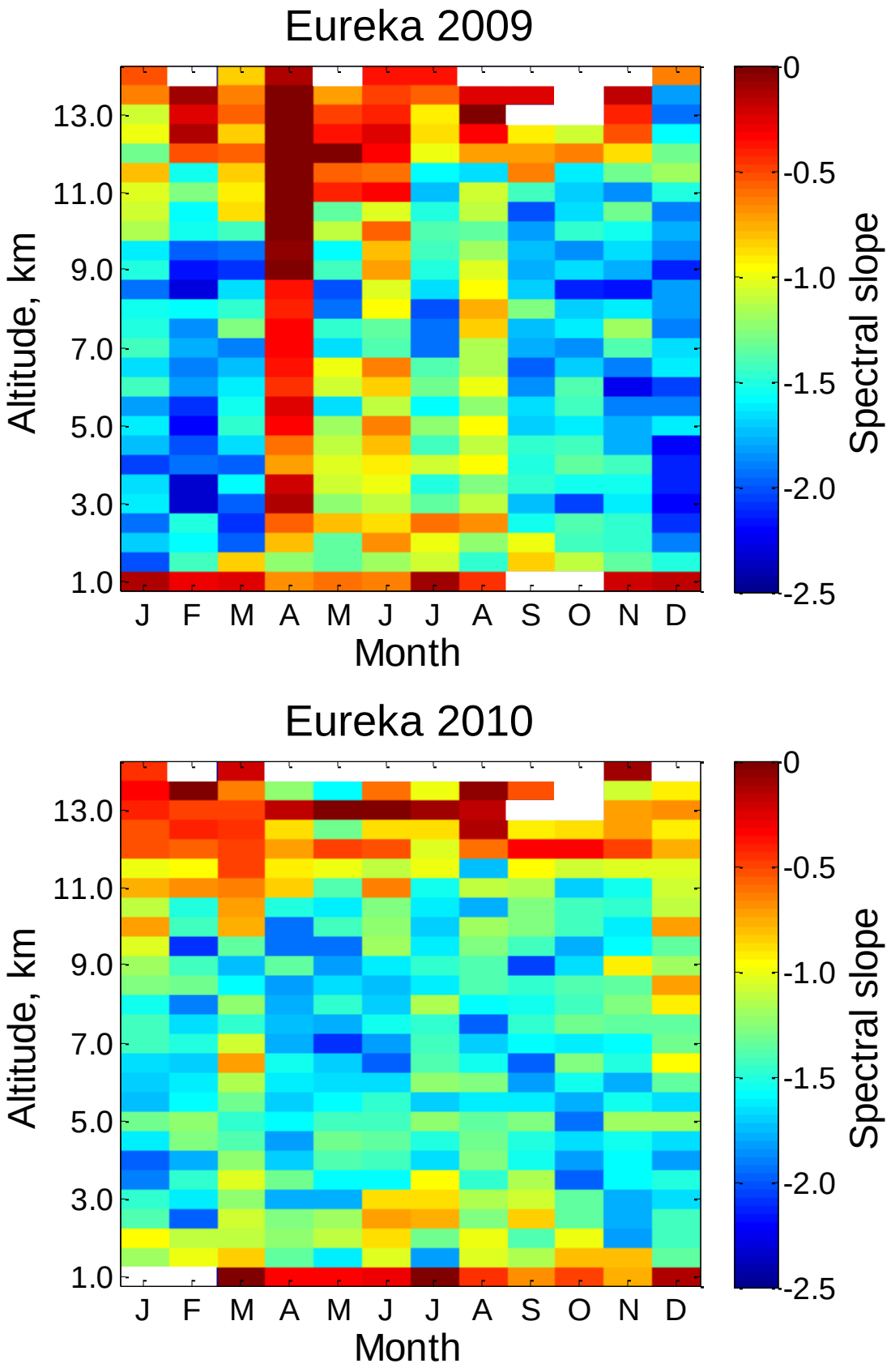
6.2.1 Spectral slope values

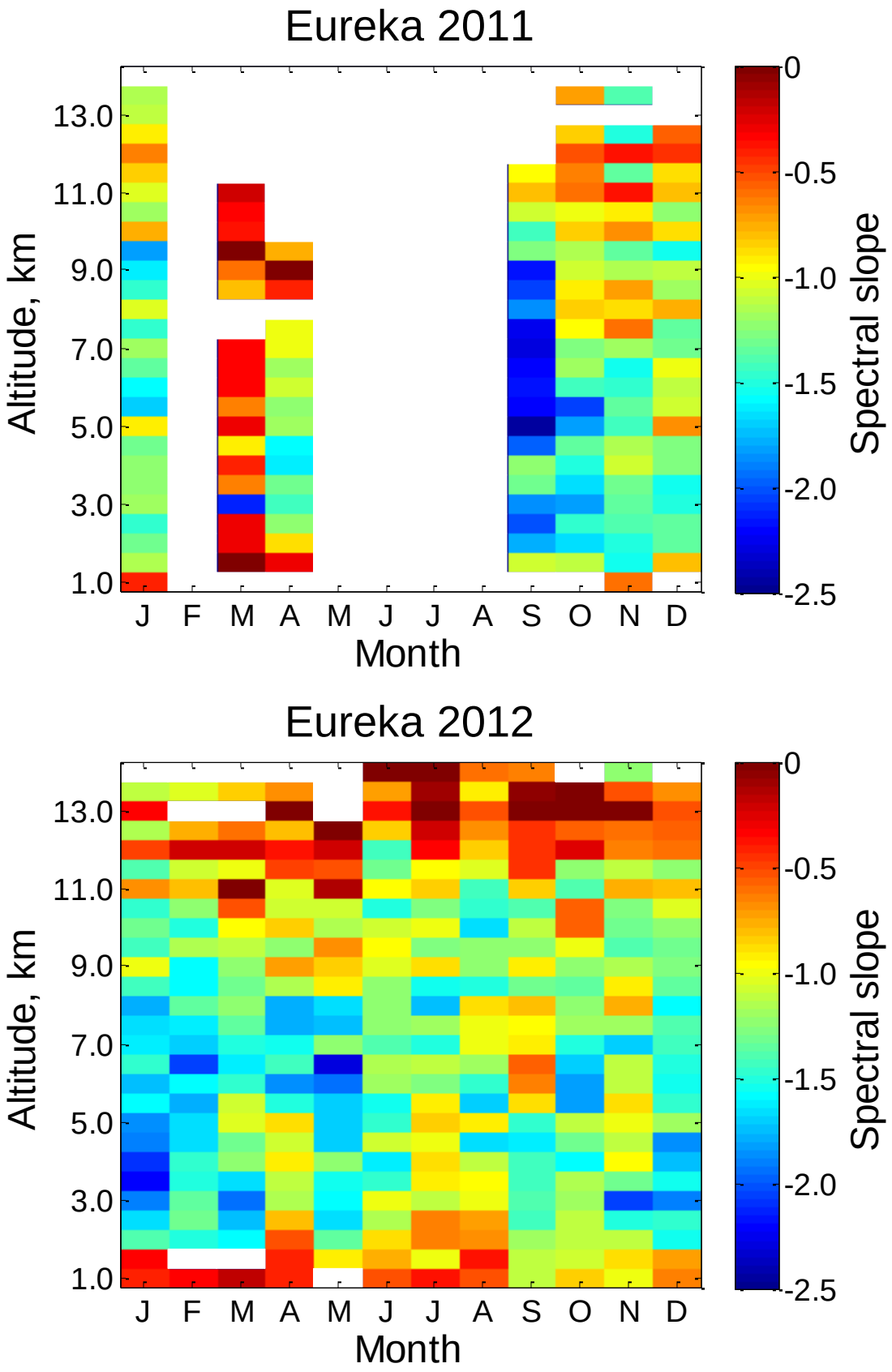
The following 20 figures show monthly spectral slope values for altitude versus month using the date-compensated discrete Fourier transform with the Hamming window. The site and year depicted for each plot are noted above the graph.

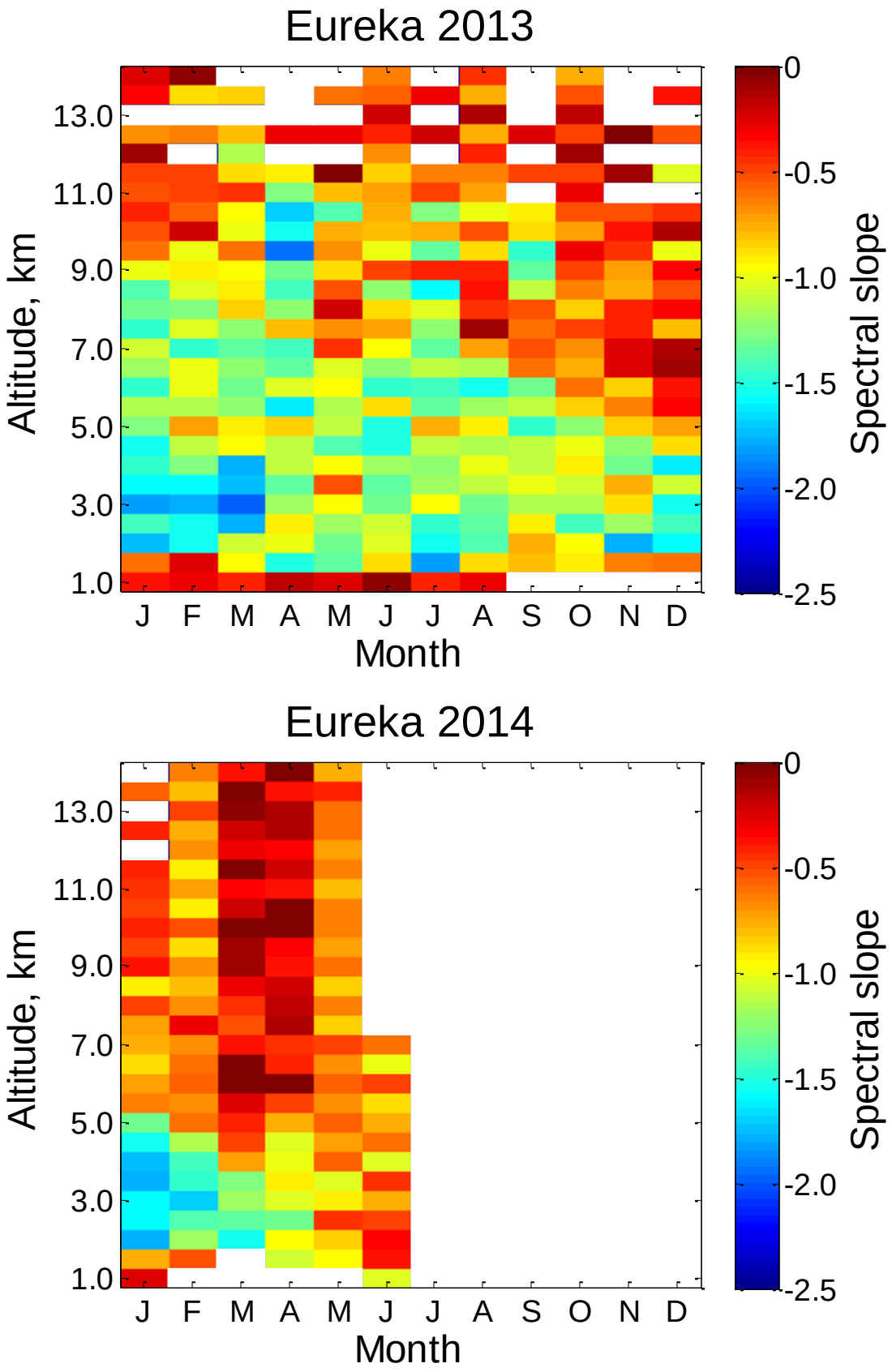
Altitude and month pairs that did not have data or had fewer than 240 data points (equivalent to one data point every three hours for a month with 30 days, if they were evenly spaced) are plotted as white. The theoretical slope value (for the middle and upper atmosphere) is $-5/3$ (i.e. -1.67, see Figure 2.2), corresponding to a dark cyan.

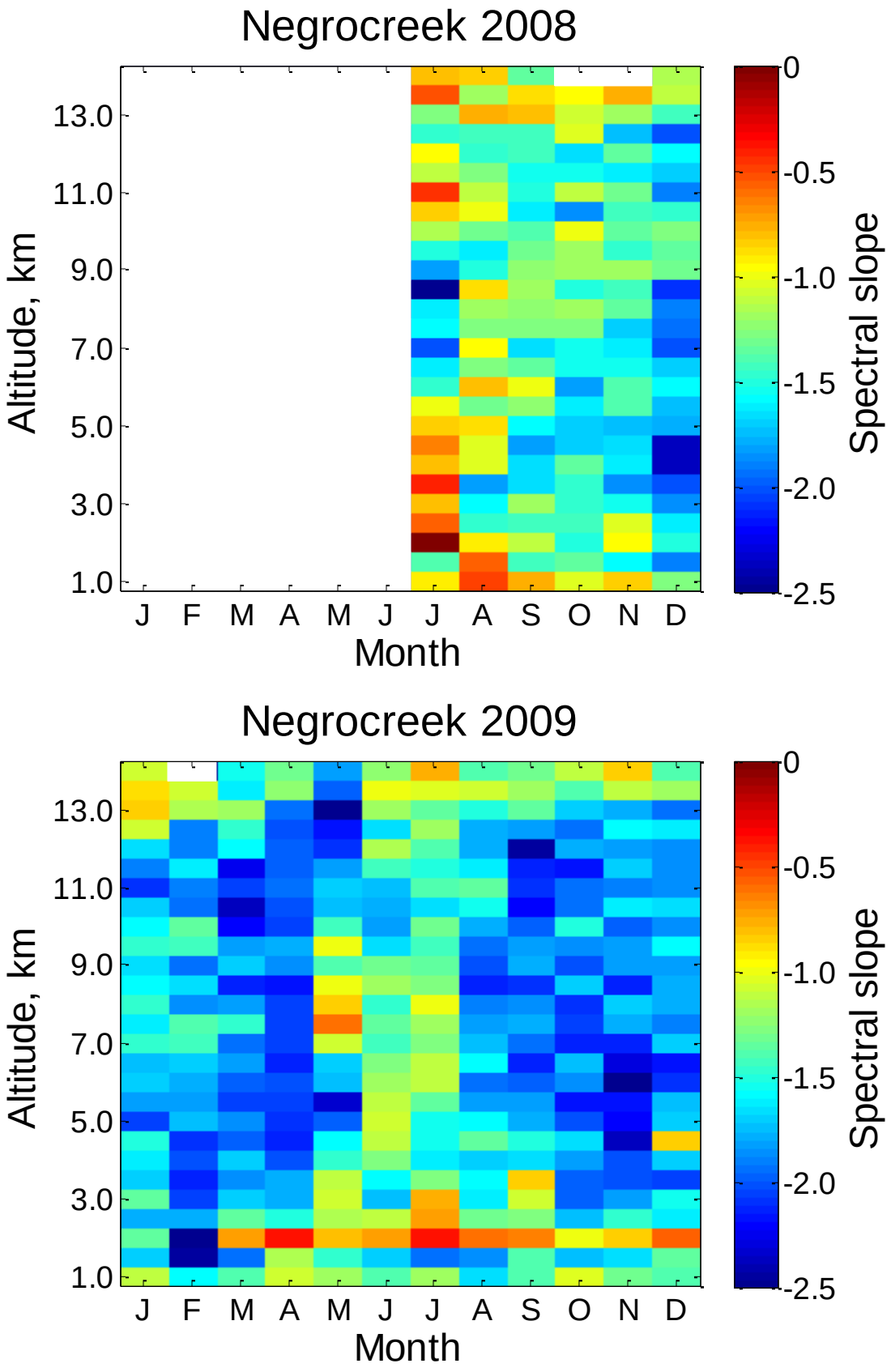
These slopes were analysed with respect to altitude, season, and geographic location in subsequent sections.

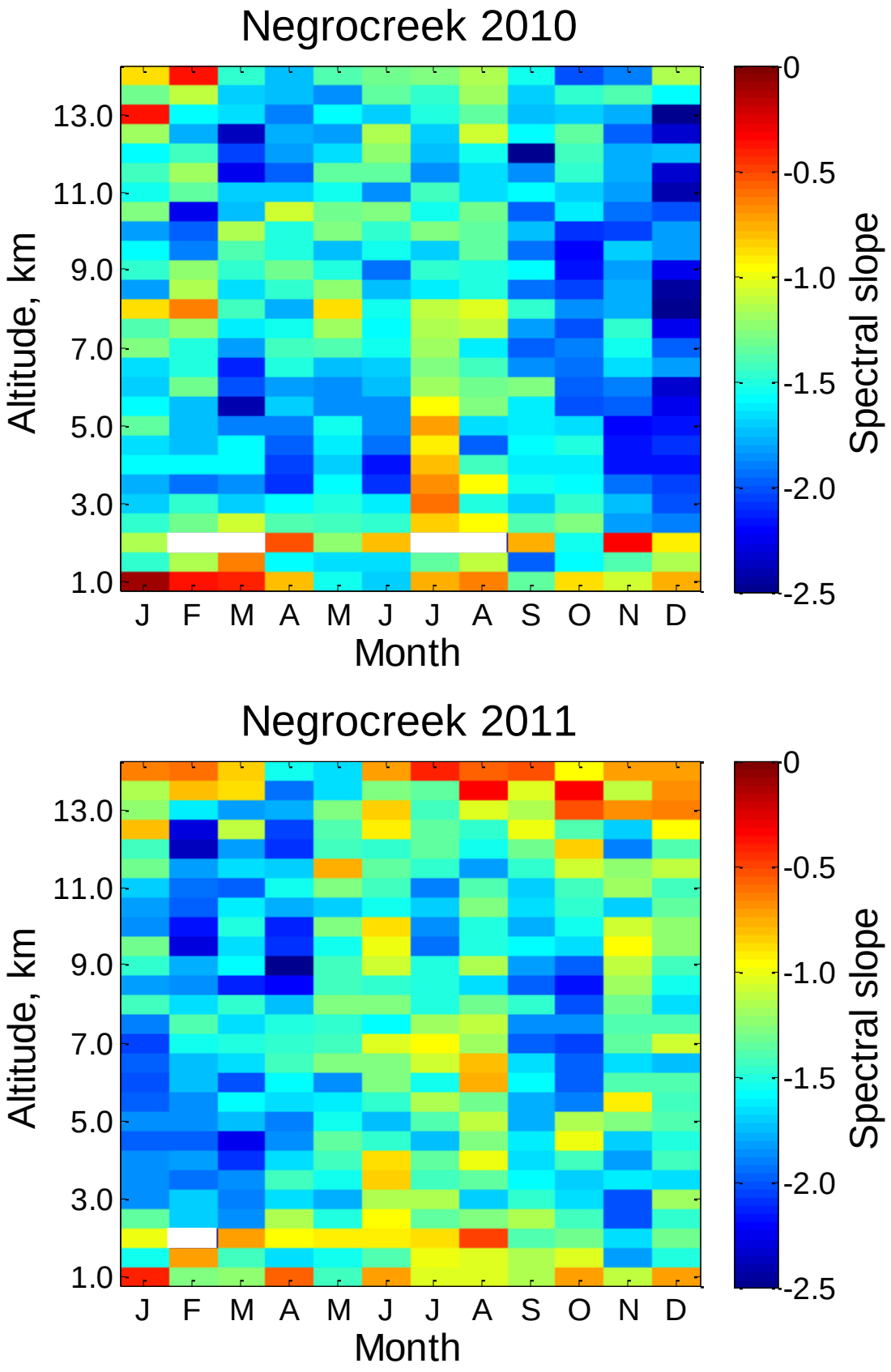

Eureka 2007
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Eureka 2008
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope

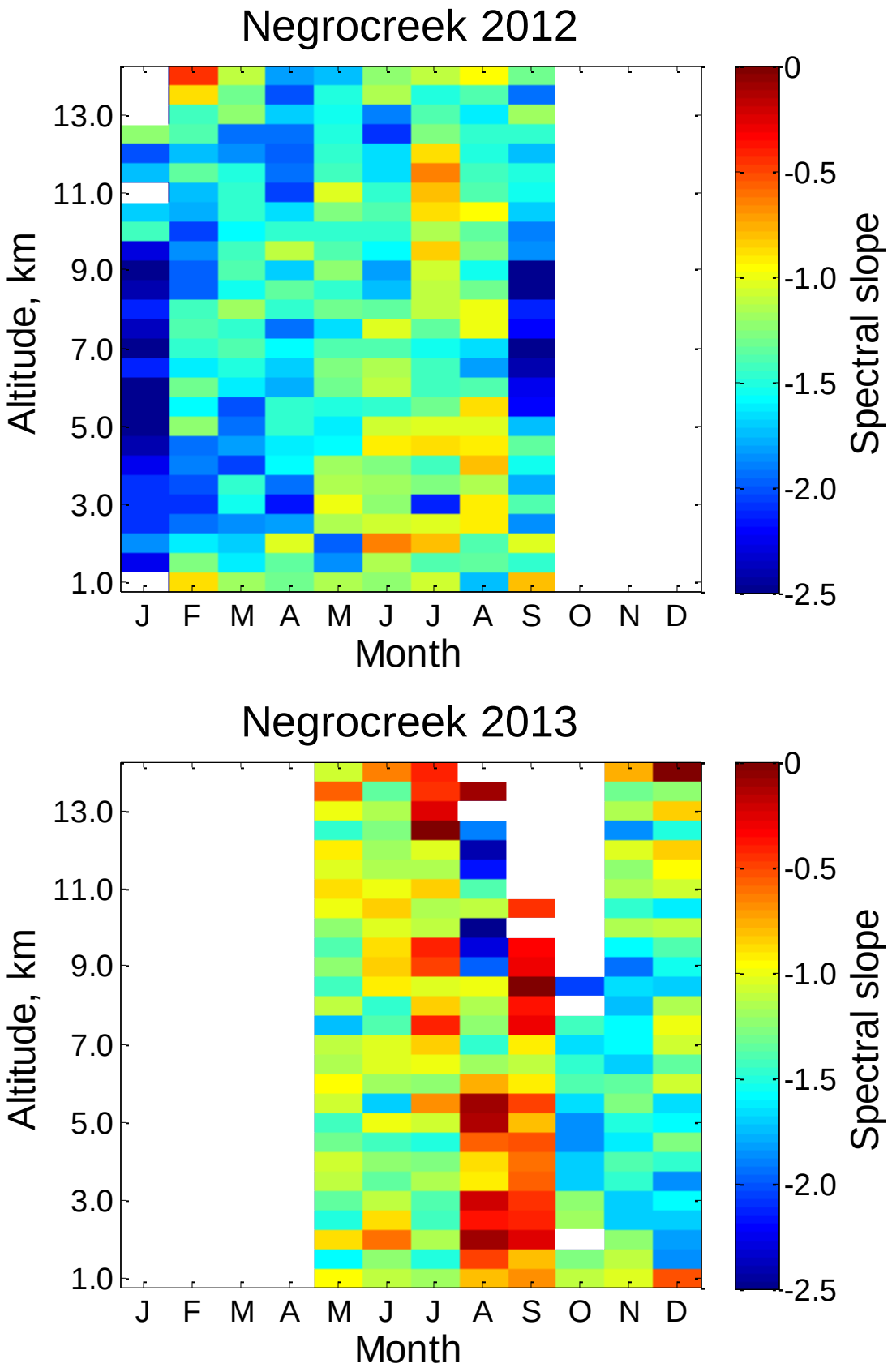

Eureka 2009
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Eureka 2010
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope

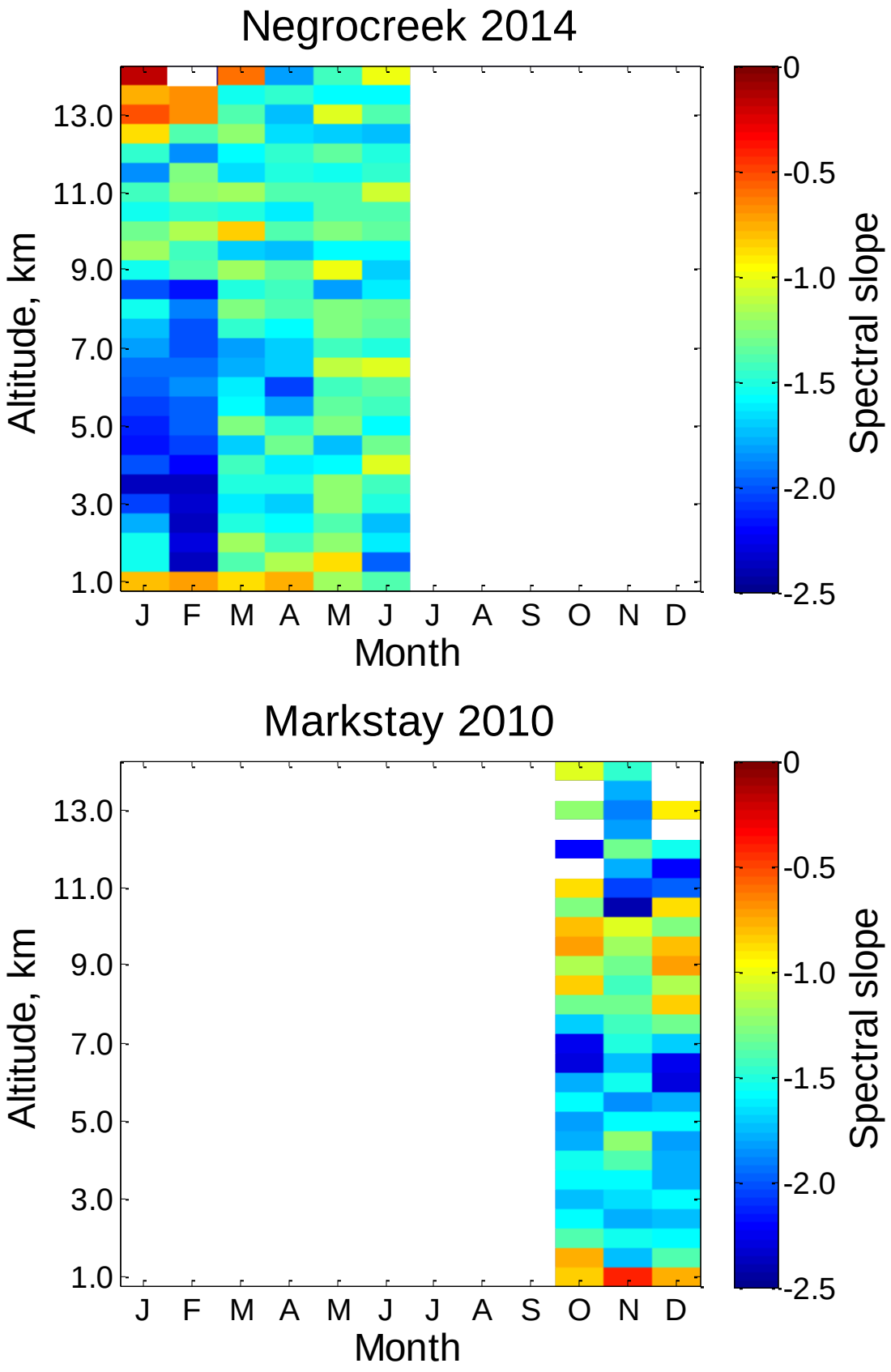

Eureka 2011
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Eureka 2012
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope

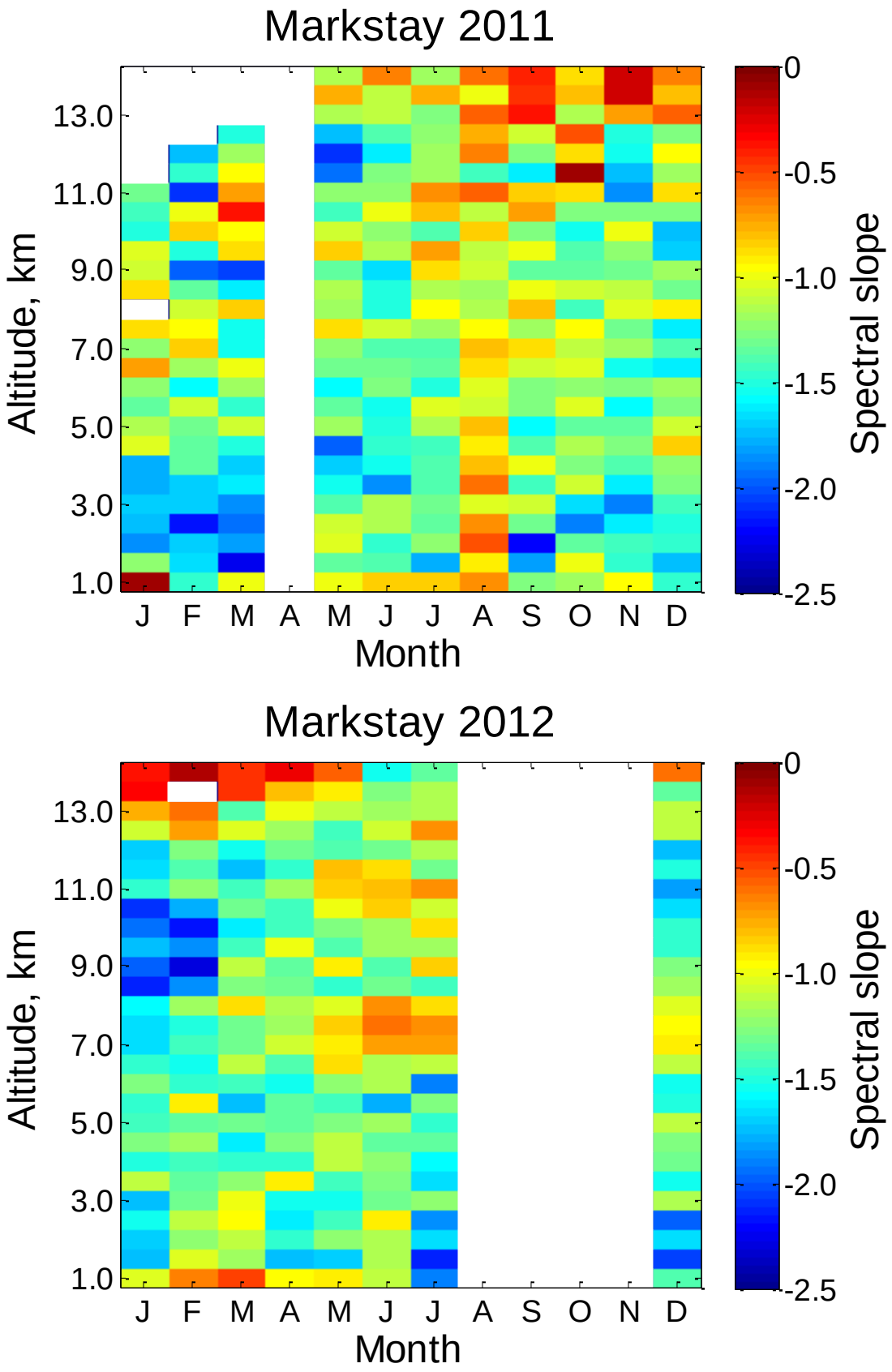

Eureka 2013
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Eureka 2014
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope

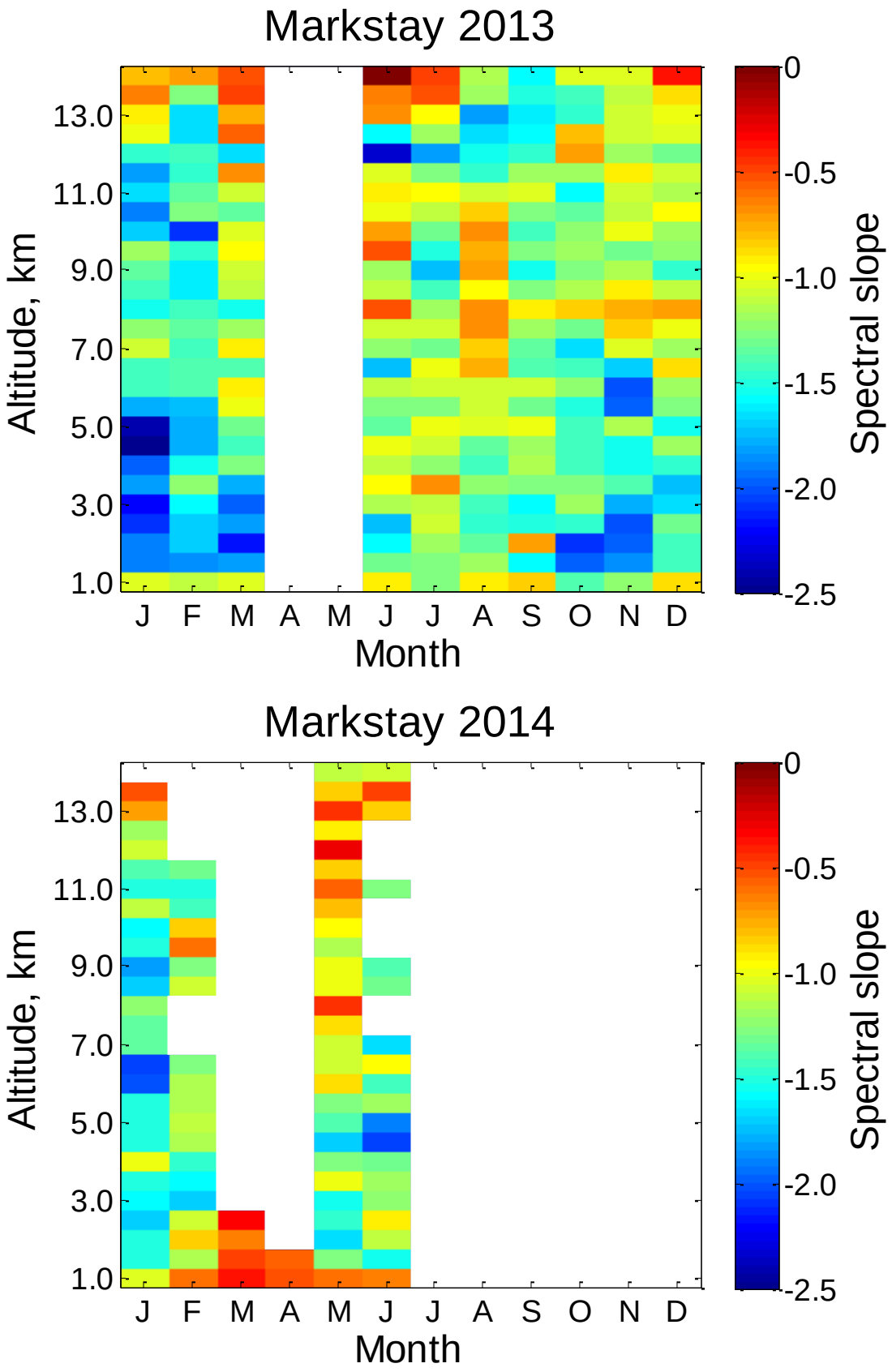

Negrocreek 2008
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Negrocreek 2009
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope


Negrocreek 2010
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Negrocreek 2011
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope


Negrocreek 2012
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Negrocreek 2013
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope


Negrocreek 2014
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Markstay 2010
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope


Markstay 2011
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Markstay 2012
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope


Markstay 2013
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope
Markstay 2014
Altitude, km
13.0
11.0
9.0
7.0
5.0
3.0
1.0
J F M A M J J A S O N D
Month
0
-0.5
-1.0
-1.5
-2.0
-2.5
Spectral slope

6.2.2 Altitude comparison

The slopes tend to flatten out above 11.0 km and below 2.0 km or so, as shown in Figure 6.1. This is likely due to instrument sensitivities, as discussed in Sections 3.4.4 and 3.4.3, respectively. As a result, the following analyses rely more heavily on data between 2.0 – 11.0 km altitude. Otherwise, there is no apparent dependence on altitude.

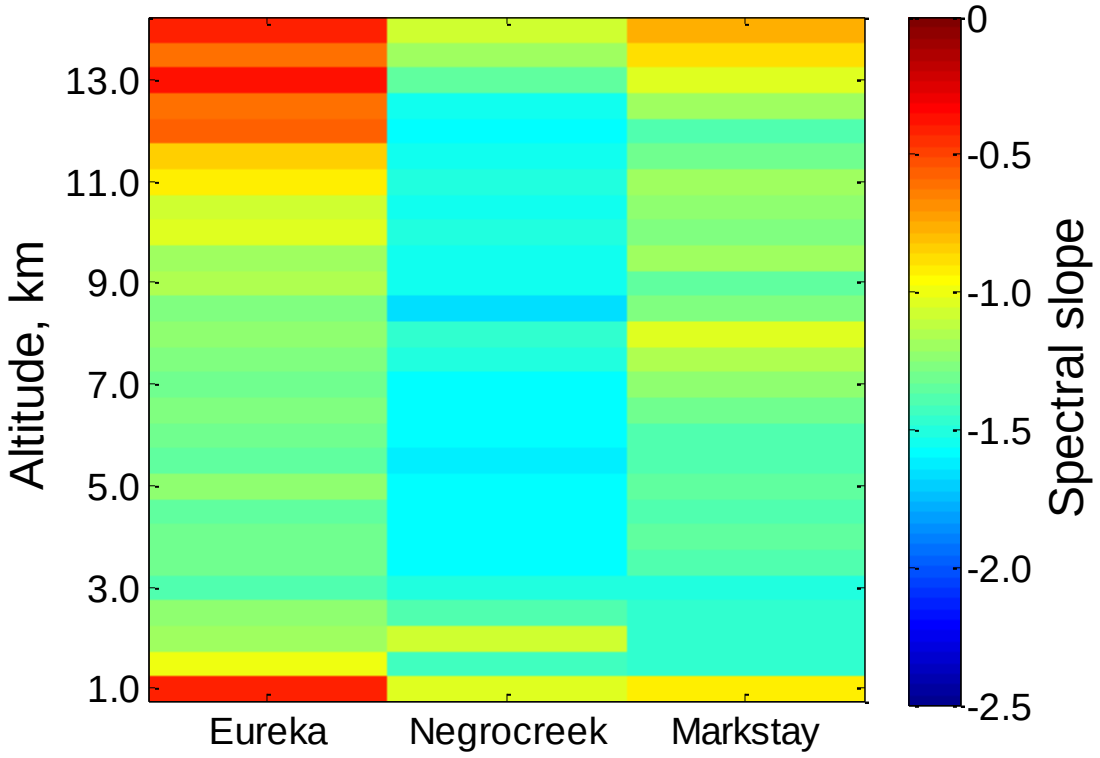


Figure 6.1: Spectral slope values, as a function of altitude, averaged over all operational years for the three radar sites.

6.2.3 Seasonal comparison

The spectral slopes for Eureka tend to be flatter than anticipated by the theoretical value, and further flatten in recent years, as shown in Figure 6.2. Despite the fluctuations in spectral slope value at Eureka over seven years, there is no apparent seasonal dependence (as shown in Figure 6.3).

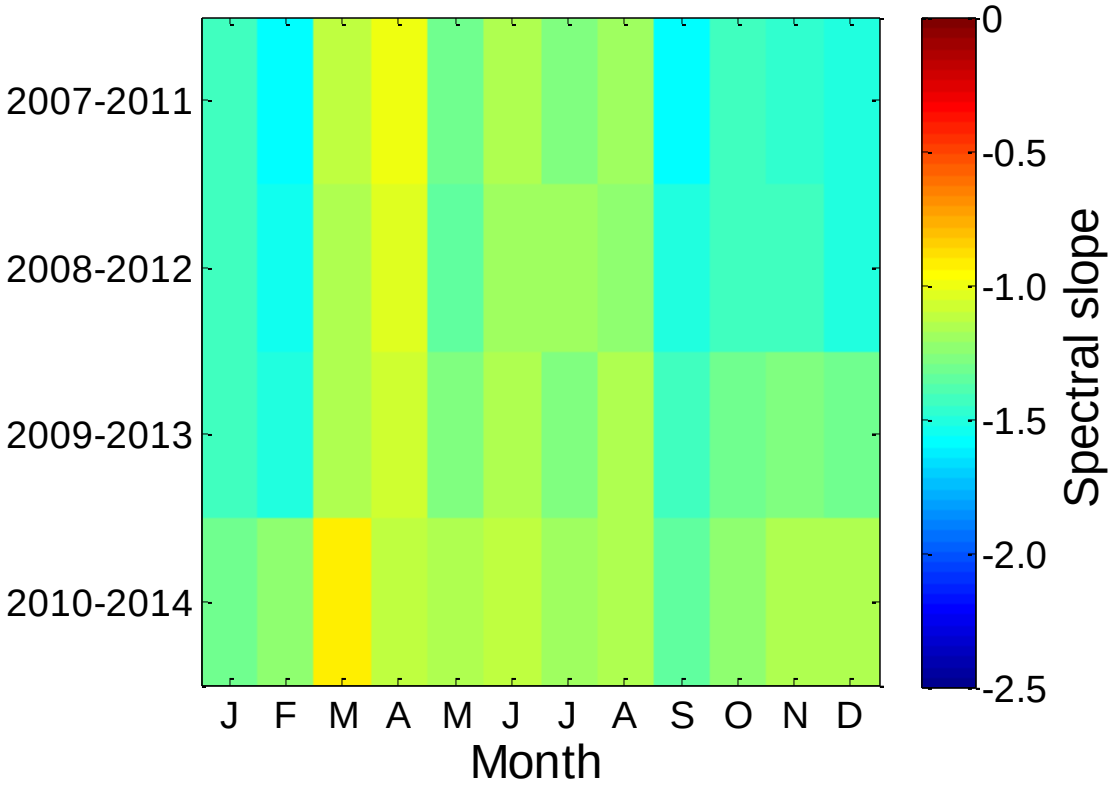


Figure 6.2: Spectral slope values for Eureka, averaged over 2.0-11.0 km altitude and four year intervals (indicated on the ordinate). The spectral slopes tend to flatten out in recent years.

The spectral slopes for Negrocreek hover around the theoretical value of $-5/3$, with temporal fluctuations of steeper slopes. The steeper slopes tend to occur in the winter months and there appears to be a slight seasonal dependence, as shown in Figure 6.3. At Markstay, steeper slopes also tend to occur in the winter months, though the slopes are (in general) shallower than the theoretical value and, like Negrocreek, there is a slight seasonal dependence, as shown in Figure 6.3.

Results from Negrocreek and Markstay appear to have a slight seasonal dependence, but results from Eureka hint at a longer time scale dependence (e.g., maybe linked to the solar cycle).

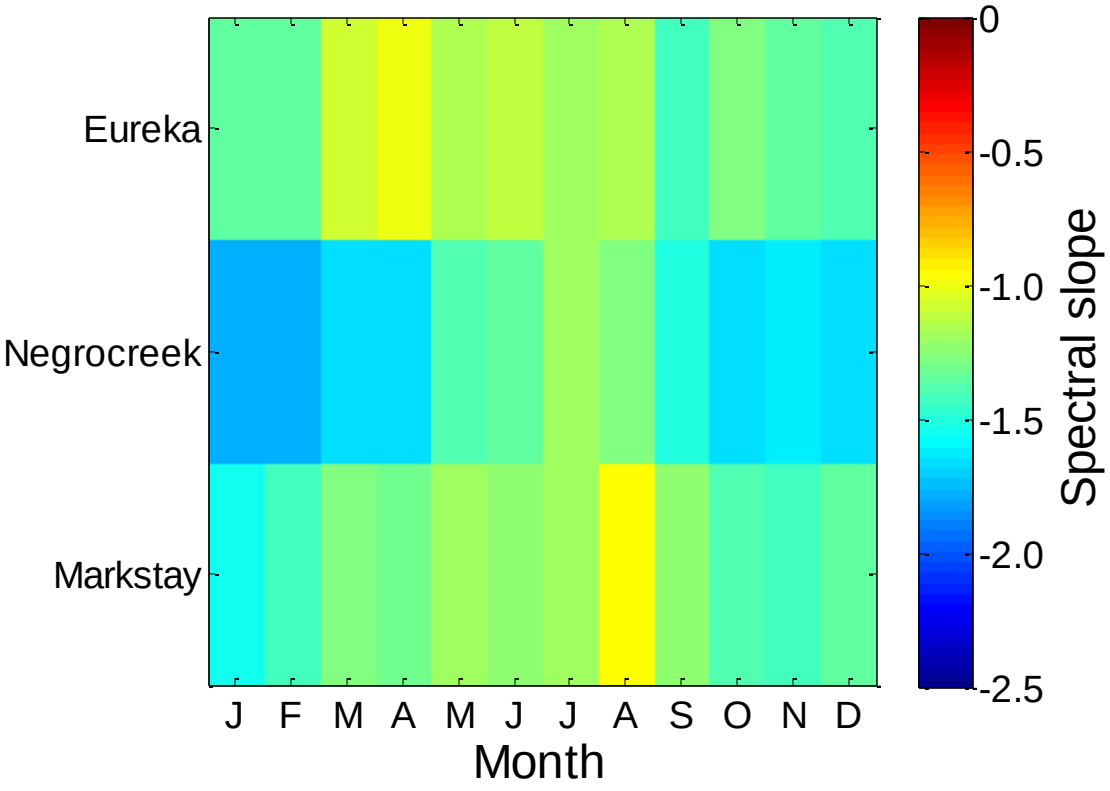


Figure 6.3: Spectral slope values for all three radar sites, averaged over 2.0-11.0 km altitude and all operational years.

6.2.4 Geographical comparison

The spectral slopes at Negrocreek tend to be consistent between roughly -1.2 to -2.0. The spectral slopes at Markstay are in general shallower, ranging from roughly -0.5 to -1.7.

In 2009, the spectral slopes at Eureka tend to be on par with those at Negrocreek, though more variable (consisting of shallower and steeper slopes than Negrocreek). In 2010 the spectral slopes at Eureka generally become flatter than Negrocreek (around September), but are still steeper than Markstay (which became operational in October of 2010). Throughout 2012, the slopes at Eureka continue to flatten and eventually become shallower than the slopes at Markstay.

If the gravity wave spectrum depended on latitude, one would expect to see very similar slopes at Markstay and Negrocreek (two fairly geographically close, mid-latitude radar sites) and different slopes at Eureka (an Arctic site). The slopes at Markstay and Negrocreek are fairly different and Eureka fluctuates between the two, as summarized in Table 6.1. Therefore, it is unlikely that the spectral slope value depends on latitude in any significant way.

While there is a systematic difference in spectral slopes between sites, it is not clear what the geographical dependency is. The noted differences could be terrain dependent. Negrocreek is close to the Great Lakes area (specifically, both west and south of Lake Huron), which may be a source of gravity waves. Markstay is surrounded by forest north of the Great Lakes region of Ontario. Eureka is surrounded by barren and mountainous tundra. The noted differences could also have some other dependence that appears geographical, such as whether a frontal system is near, close, or not present (as suggested by Belu (1999)).

Radar site	Average spectral slope	Standard deviation	Number of slopes used
Eureka	-1.23	0.53	1824
Negrocreek	-1.51	0.46	1718
Markstay	-1.29	0.40	941

Table 6.1: Spectral slope values and standard deviations for each radar site averaged over all months and heights 2.0 km to 11.0 km. The number of slopes (month-altitude pairs) used to calculate the spectral slope and standard deviation are also included.

6.2.5 Interpretation of the slopes

Despite minor difference discussed in the previous sections, overall, there is little to no seasonal or geographical trend in the spectral slopes (which somewhat supports the idea of a universal spectrum), but mean slopes are around -1.2 to -1.5 (see Table 6.1for values) and do not reach -1.6 or -1.7 as predicted. There are several reasons why this may not be the case.

6.2.5.1 Off vertical beam

Since we are using an off-vertical beam, the radial velocities measured are given by:

$$v_{rad} = u \sin\theta + w \cos\theta \tag{6.1}$$

where u is the horizontal gravity wave component, w is the vertical component, and θ is the vertical off-set of the radar beam, approximately 10°. This means:

$$v_{rad} \cong 0.18u + w \tag{6.2}$$

However, in a gravity wave, u and w are correlated, either in-phase or 180° out of phase (depending on the wave propagation direction). In addition, the relation between u and w is well known from the polarization relations. For cases where the period is more than an hour, we can approximately use:

$$\frac{w}{u} = \pm\frac{T_B}{T} \tag{6.3}$$

where T_B is the Brunt-Vaisala period and T is the wave period. Combining these and taking 0.18 as approximately 0.2:

$$v_{rad} \cong \left(0.2 \pm \frac{T_B}{T}\right) u = \left(1 \pm 5\frac{T_B}{T}\right) 0.2u \tag{6.4}$$

Thus the radial-velocity spectrum we measure differs from the horizontal velocity spectrum (which is the one used in theoretical discussions). Since we are only interested in spectral form, the relevant correction term is:

$$\left(1 + \frac{5T_B}{T}\right) \tag{6.5}$$

Consider an example where $T = 100$ min, $T_B = 10$ min, and consider the positive sign. Then the correction term is:

$$\left(1 + 5 \times \frac{10}{100}\right) = 1.5 \tag{6.6}$$

This is a large correction term and means that our measured v_{rad} values are 50% too large. However, if $T = 1000$ min, then the correction is only 1.05. Therefore, high frequency waves are over-represented and push the spectra to be higher at the high-frequency end, flattening the spectral slope.

To further complicate matters, this effect is more dominant because of our use of log-log plots. The frequencies between 10 and 100 min cover one decade, while the frequencies between 100 and 1000 min also cover one decade. So the character between 10 and 100 mins has a disproportionate effect on the slope determination.

Our calculations suggest that the flattening changes the slope by about 0.1. To see this in a crude sense, consider the 100 min case: v_{rad} is magnified 1.5 times (in a relative sense) so the powers are increased 1.5^2 times (or about twice). This error is about 0.3 in log coordinates. Taking the abscissa to cover 3 decades, the error in slope is about 0.3/3 (or about 0.1). So if the ideal slope is -1.6, our measured one will be closer to -1.5.

6.2.5.2 Noise

The second cause of error in the slope is the noise. It does not disappear at the break-point, and is still playing a role even as we move to lower frequencies from the break-point (i.e. to the left). The role becomes less important at the lowest frequencies, but in a

log-log plot, it has a disproportionate effect at the high-frequency end. This has a similar flattening effect to the off-vertical beam, possibly producing a further flattening of about 0.1 or more for noisier data. So this changes our original slope of -1.6 to 1.5 via the off-vertical beam, and possibly to -1.4 due to noise.

6.2.5.3 Presence of local gravity wave sources

As shown by Belu (1999) the presence of local gravity wave sources (e.g. frontal systems, orography) has a particularly strong impact at the higher frequencies, while the universal spectrum is based on gravity waves that have propagated from afar. Enhancement of the high frequencies even further flattens the spectrum. As such, our spectral slope values are probably fairly consistent with the universal spectrum, with some addition of high frequencies due to local gravity wave generation.

Chapter 7

7 Conclusions and future work

There appears to be some validity to the universal spectrum and the measured slopes are not inconsistent, provided that the impact of an off-vertical beam and noise are properly considered. However, the universal spectrum does not explain the whole measured spectrum. Some portion of the real spectra must be ascribed to local wave sources, although, the universal spectral component seems to provide a major part of the spectrum.

The gravity wave spectral slopes approximately above 11.0 km and below 2.0 km altitude are often not reliable. Mid-latitude radar sites suggest a slight seasonal dependence while an Arctic radar site suggests a longer time dependence. While the spectral slopes do not appear to depend on altitude or latitude, there is geographic variability. Other potential sources of the geographical variability (e.g., terrain, presence of frontal systems) should be examined.

This study could benefit from a longer data acquisition time and more radar sites (both at different latitudes and near similar terrain). Since the spectral slopes from Eureka suggest a longer time cycle, longer data acquisition would enable us to see (1) if there is a longer cycle to the gravity wave spectrum and (2) if roughly constant mid-latitude sites (such as Negrocreek and Markstay) also vary on a longer time scale, in addition to the slight seasonal dependence.

Each of the spectral slopes use a month of data. This study may benefit from using shorter timescale averages, such as averaging over a week or a day.

This study may also benefit from comparison to changes in the mean winds. Recently (over the past couple years), there have been more upper level jets above Eureka, as seen in Figure 7.1. These jets are often a source of gravity waves. When they are more common, there are more local wave-sources and we expect the universal spectrum to be most accurately followed when the sources are more distant. Hence we

suspect that a universal spectrum will be more common when there are no such local sources.

While the spectral slopes do not appear to depend on latitude, they may depend on terrain. More radar sites in physically similar regions to Eureka, Negrocreek, and Markstay may enable us to determine if the spectral slope depends on nearby terrain, jet streams, and other local gravity wave sources. This would enable us to determine whether proximity to gravity wave sources (such as at Negrocreek) affects the spectrum.

This study could also benefit from reanalysis, examining dependencies other than altitudinal, geographical, and latitudinal. One potential dependency of interest may be the effect frontal systems have on the gravity waves spectral slope.

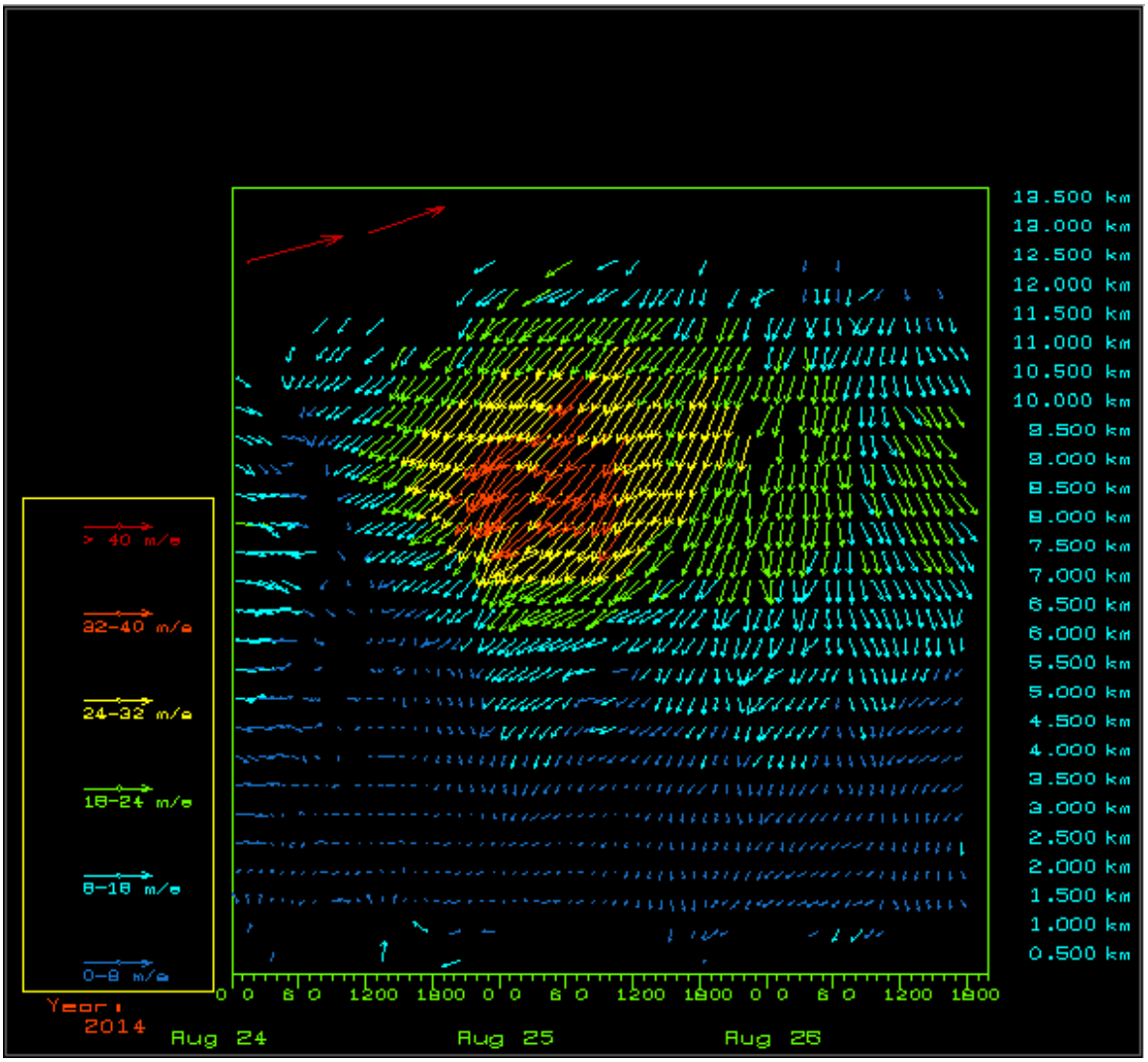


Figure 7.1: Mean wind values above Eureka as a function of height (between 0.5 and 12.5 km in 0.5 km increments) over a three day span (August 24-26, 2014). The direction of the arrows indicate the direction of the wind: pointing up on the graph represents a northward wind, pointing to the right on the graph represents an eastward wind, and so on. The colour and magnitude of the arrows indicates wind speed: dark blue is 0-8 m/s, cyan is 8-18 m/s, green is 18-24 m/s, yellow is 24-32 m/s, orangey red is 22-40 m/s, and dark red is greater than 40 m/s.

References

Belu, R. *Gravity waves sources and propagation characteristics in the lower and middle atmosphere, determined by Clovar radar and other ground-based methods*. London, Ontario, Canada: University of Western Ontario, 1999. Ph. D. thesis.

Cooley, J. W. and J. W. Tukey. "An algorithm for the machine computation of the complex Fourier series." *Mathematics of Computation* 19 (1965): 297-301.

Duhamel, P. and M. Vetterli. "Fast Fourier transforms: A tutorial review and a state of the art." *Signal Processing* 19 (1990): 259-299.

Eckermann, S., I. Hirota and W. Hocking. "Gravity wave and equatorial morphology of the stratosphere derived from long-term rocket soundings." *Quarterly Journal of the Royal Meteorlogical Society* 121 (1995): 149-186.

Ferraz-Mello, S. "Estimation of periods from unequally spaced observations." *The Astronomical Journal* 86.4 (1981): 619-624.

Gage, K. S. and J. L. Green. "Evidence for specular reflection from monostatic VHF radar observations of the stratosphere." *Radio Science* 13 (1978): 991-1001.

Garrett, Christopher and Walter Munk. "Space-time scales of internal waves." *Geophys. Fluid Dynamics* 2 (1972): 225-264.

—. "Space-time scales of internal waves: a progress report." *J. Geophys. Res.* 80 (1975): 291-297.

Gossard, Earl E. and William H. Hooke. *Waves in the atmosphere: atmospheric infrasound and gravity waves, their generation and propagation*. Netherlands: Elsevier Scientific Pub. Co., 1975.

Hines, C. O. "Internal atmospheric gravity waves of ionospheric height." *Canadian Journal of Physics* 38 (1960): 1441-1481.

Hines, Colin O. "The saturation of gravity waves in the middle atmosphere. Part I: Critique of linear-instability theory." *Journal of the Atmospheric Sciences* 48.11 (1991): 1348-1359.

—. "The saturation of gravity waves in the middle atmosphere. Part II: Development of Doppler-spread theory." *Journal of the Atmospheric Sciences* 48 (1991): 1360-1379.

Hocking, Wayne K. *Radar Methods in Mesosphere-Stratosphere-Troposphere (MST) and Windprofiler Studies* . In Preparation.

Hodges, R. "Generation of turbulence in the upper atmosphere by internal gravity waves." *Journal of Geophysics Res.* 72 (1967): 3455-3458.

Holton, J. "The influence of gravity wave breaking on the general circulation of the middle atmosphere." *J. Atmos. Sci.* 40 (1983): 2497-2507.

Lindzen, R. "Turbulence and stress owing to gravity wave and tidal breakdown." *J. Geophys. Res.* 86 (1981): 9707-9714.

Lomb, N. R. "Least-squares frequency analysis of unequally spaced data." *Astrophysics and Space Science* 39 (1976): 447-462.

Marple, S. Lawrence, Jr. *Digital Spectral Analysis with Applications*. New Jersey, USA: Prentice-Hall, 1987.

Medvedev, A. S. and G. P. Klaassen. "Parameterization of gravity wave momentum deposition based on nonlinear wave interactions: basic formulation and sensitivity tests." *Journal of Atmospheric and Solar-Terrestrial Physics* 62 (2000): 1015-1033.

Nappo, C. J. *An introduction to atmospheric gravity waves*. Vol. 1. San Diego, California: Academic Press, 2002.

Oppenheim, A. V. and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, 1989.

Rader, C. M. "Discrete Fourier transforms when the number of data samples is prime." *Proceedings of the IEEE* 56 (1968): 1107-1108.

Rottger, J. "Investigationg of lower and middle atmosphere dynamics with spaced antenna drift radars." *Journal of Atmospheric Terr. Physics* 43 (1981): 277-292.

—. "ST radar observations of atmospheric waves over mountainous areas: a review." *Ann. Geophysicae* 18 (2000): 750-765.

Scargle, Jeffrey D. "Studies in astronomical time series analysis. II. Statistical aspects of spectral analysis of unevenly spaced data." *The Astrophysical Journal* 263 (1982): 835-853.

Scorer, R. S. *Dynamics of Meteorology and Climate*. Chichester, England: John Wiley, 1997.

Smith, S. A., D. C. Fritts and T. E. Van Zandt. "Evidence for a saturated spectrum of atmospheric gravity waves." *J. Atmos. Sci.* 44 (1987): 1404-1410.

Stiller, B., et al. *New generation compact pulsed infrared coherent Doppler Lidars validation against Wind Profiler Radar and Radiosonde measurements at the Lindenberg GRUAN site*. Lidenberg Meteorological Observatory and Richard ABmann Observatory, 2012. 08 2014.

Tatarski, V. *Wave propagation in a turbulent medium*. New York: McGraw-Hill, 1961.

VanZandt, T. E. "A universal spectrum of buoyancy waves in the atmosphere." *Geophysical Research Letters* 9.5 (1982): 575-578.

Weinstock, J. "Nonlinear theory of acoustic-gravity waves: I. Saturation and enhanced diffusion." *Journal of Geophysical Research* 81.4 (1976): 633-652.

Appendix A: Mathematical derivations

This appendix contains mathematical derivations for the normalization constant and Fourier transform of the filtering windows used in this thesis.

A.1 Boxcar window

The boxcar window function is a non-zero constant on a finite range, with the value of zero everywhere else, viz.:

$$h(t) = \begin{cases} A & \text{if } |t| \leq T/2 \\ 0 & \text{otherwise} \end{cases}$$

The constant A should be normalized, such that the integral of the boxcar window function over all t is 1.

$$\int_{-\infty}^{\infty} h(t)dt = 1$$

$$\int_{-\infty}^{-T/2} 0dt + \int_{-T/2}^{T/2} Adt + \int_{T/2}^{\infty} 0dt = 1$$

$$A\int_{-T/2}^{T/2} dt = 1$$

$$A[t]_{-T/2}^{T/2} = 1$$

$$A\left(\frac{T}{2} - \left(-\frac{T}{2}\right)\right) = 1$$

$$AT = 1$$

$$A = \frac{1}{T}$$

The normalized boxcar window function is therefore:

$$\boxed{h(t) = \begin{cases} 1/T & \text{if } |t| \leq T/2 \\ 0 & \text{otherwise} \end{cases}}$$

The Fourier transform of the normalized boxcar window function is:

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t) \mathrm{e}^{-2\pi i f t} dt \\ &= \int_{-\infty}^{-T/2} 0 \mathrm{e}^{-2\pi i f t} dt + \int_{-T/2}^{T/2} \frac{1}{T} \mathrm{e}^{-2\pi i f t} dt + \int_{T/2}^{\infty} 0 \mathrm{e}^{-2\pi i f t} dt \\ &= \frac{1}{T} \int_{-T/2}^{T/2} \mathrm{e}^{-2\pi i f t} dt \\ &= \frac{1}{T} \left[\frac{i}{2\pi f} \mathrm{e}^{-2\pi i f t} \right]_{-T/2}^{T/2} \\ &= \frac{i}{2\pi f T} \left(\mathrm{e}^{-\pi i f T} - \mathrm{e}^{\pi i f T} \right) \end{aligned}$$

Recalling $\sin\theta = \frac{1}{2i}\left(e^{i\theta} - e^{-i\theta}\right)$:

$$\begin{aligned} H(f) &= \frac{i}{2\pi f T} (-2i \sin \pi f T) \\ &= \frac{\sin \pi f T}{\pi f T} \end{aligned}$$

Therefore, the Fourier transform of the boxcar window is:

$$\boxed{H(f) = \operatorname{sinc} \pi f T}$$

A.2 Boxcar with 10% cosine taper window

In attempt to reduce the ringing of a boxcar window function, this window function smooths the sharp edges of the boxcar window function. The boxcar with 10% cosine taper is defined as:

$$h(t) = \begin{cases} \frac{A}{2}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right) & \text{if } -\frac{T}{2} \le t < -\frac{2T}{5} \\ A & \text{if } |t| \le \frac{2T}{5} \\ \frac{A}{2}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right) & \text{if } \frac{2T}{5} < t \le \frac{T}{2} \\ 0 & \text{otherwise} \end{cases}$$

The constant A should be normalized such that the integral of the window over all t is 1.

$$\int_{-\infty}^{\infty} h(t)dt = 1$$

$$\int_{\infty}^{-T/2} 0dt + \int_{-T/2}^{-2T/5} \frac{A}{2}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)dt + \int_{-2T/5}^{2T/5} Adt + \int_{2T/5}^{T/2} \frac{A}{2}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)dt + \int_{T/2}^{\infty} 0dt = 1$$

$$A\left(\frac{1}{2}\int_{-T/2}^{-2T/5}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)dt + \int_{-2T/5}^{2T/5} dt + \frac{1}{2}\int_{2T/5}^{T/2}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)dt\right) = 1$$

$$A\left(\frac{1}{2}\left[t + \frac{T}{10\pi}\sin\frac{10\pi t}{T}\right]_{-\frac{T}{2}}^{-\frac{2T}{5}} + [t]_{-\frac{2T}{5}}^{\frac{2T}{5}} + \frac{1}{2}\left[1 + \frac{T}{10\pi}\sin\frac{10\pi t}{T}\right]_{\frac{2T}{5}}^{\frac{T}{2}}\right) = 1$$

$$A\left(\frac{1}{2}\left(-\frac{2T}{5} + \frac{T}{2}\right) + \left(\frac{2T}{5} - \left(-\frac{2T}{5}\right)\right) + \frac{1}{2}\left(\frac{T}{2} - \frac{2T}{5}\right)\right) = 1$$

$$\frac{9AT}{10} = 1$$

Therefore,

$$A = \frac{10}{9T}$$

The normalized boxcar with 10% cosine taper window function is:

$$h(t) = \begin{cases} \frac{5}{9T}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right) & \text{if } -\frac{T}{2} \le t < -\frac{2T}{5} \\ \frac{10}{9T} & \text{if } |t| \le \frac{2T}{5} \\ \frac{5}{9T}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right) & \text{if } \frac{2T}{5} < t \le \frac{T}{2} \\ 0 & \text{otherwise} \end{cases}$$

The Fourier transform of this window is:

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t)\mathrm{e}^{-2\pi i f t}\,dt \\ &= \int_{\infty}^{-T/2} 0\mathrm{e}^{-2\pi i f t}\,dt + \int_{-T/2}^{-2T/5} \frac{5}{9T}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)\mathrm{e}^{-2\pi i f t}\,dt \\ &\quad + \int_{-2T/5}^{2T/5} \frac{10}{9T}\mathrm{e}^{-2\pi i f t}\,dt + \int_{2T/5}^{T/2} \frac{5}{9T}\left(1 + \cos\left(\frac{10\pi}{T}t\right)\right)\mathrm{e}^{-2\pi i f t}\,dt \\ &\quad + \int_{T/2}^{\infty} 0\mathrm{e}^{-2\pi i f t}\,dt \\ &= \frac{5}{9T}\left(\int_{-\frac{T}{2}}^{-\frac{2T}{5}} \cos\left(\frac{10\pi}{T}t\right)\mathrm{e}^{-2\pi i f t}\,dt + \int_{-\frac{T}{2}}^{-\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt + 2\int_{-\frac{2T}{5}}^{\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt \right. \\ &\quad \left. + \int_{\frac{2T}{5}}^{\frac{T}{2}} \mathrm{e}^{-2\pi i f t}\,dt + \int_{\frac{2T}{5}}^{\frac{T}{2}} \cos\left(\frac{10\pi}{T}t\right)\mathrm{e}^{-2\pi i f t}\,dt \right) \end{aligned}$$

Splitting up the middle integral, then combining one half with the second and fourth integrals yields two integrals that resemble the Fourier transform of the boxcar window, viz.:

$$H(f) = \frac{5}{9T}\left(\int_{-\frac{T}{2}}^{-\frac{2T}{5}} \cos\left(\frac{10\pi}{T}t\right) \mathrm{e}^{-2\pi i f t}\,dt + \int_{-\frac{2T}{5}}^{\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt \right.$$
$$+ \left(\int_{-\frac{T}{2}}^{-\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt + \int_{-\frac{2T}{5}}^{\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt + \int_{\frac{2T}{5}}^{\frac{T}{2}} \mathrm{e}^{-2\pi i f t}\,dt\right)$$
$$\left. + \int_{\frac{2T}{5}}^{\frac{T}{2}} \cos\left(\frac{10\pi}{T}t\right) \mathrm{e}^{-2\pi i f t}\,dt\right)$$
$$= \frac{5}{9T}\left(\int_{-\frac{T}{2}}^{-\frac{2T}{5}} \cos\left(\frac{10\pi}{T}t\right) \mathrm{e}^{-2\pi i f t}\,dt + \int_{-\frac{2T}{5}}^{\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt + \int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{e}^{-2\pi i f t}\,dt \right.$$
$$\left. + \int_{\frac{2T}{5}}^{\frac{T}{2}} \cos\left(\frac{10\pi}{T}t\right) \mathrm{e}^{-2\pi i f t}\,dt\right)$$

The third integral is the same as the boxcar window Fourier transform:

$$\int_{-\frac{T}{2}}^{\frac{T}{2}} \mathrm{e}^{-2\pi i f t}\,dt = \frac{1}{\pi f}\sin \pi f T$$

The second integral is merely a boxcar window function with boundaries of $\pm\frac{2T}{5}$ instead of $\pm\frac{T}{2}$, therefore, from the boxcar window function Fourier transform:

$$\int_{-\frac{2T}{5}}^{\frac{2T}{5}} \mathrm{e}^{-2\pi i f t}\,dt = \frac{1}{\pi f}\sin \frac{4\pi f T}{5}$$

The other two integrals have the same integrand, which, for clarity's sake, will be denoted as $H_1(f)$:

$$H_1(f) = \int \cos\left(\frac{10\pi}{T}t\right) \mathrm{e}^{-2\pi i f t} dt$$

For ease of reading, the following constants are defined:

$$a = \frac{10\pi}{T}$$

$$b = 2\pi f$$

such that:

$$H_1(f) = \int \cos at \ \mathrm{e}^{-ibt} dt$$

Using the complex exponential form of cosine,

$$\cos\theta = \frac{1}{2}\left(\mathrm{e}^{i\theta} + \mathrm{e}^{-i\theta}\right)$$

$H_1(f)$ becomes:

$$\begin{aligned} H_1(f) &= \int \frac{1}{2}\left(\mathrm{e}^{iat} + \mathrm{e}^{-iat}\right)\mathrm{e}^{-ibt} dt \\ &= \frac{1}{2}\left(\int \mathrm{e}^{it(a-b)} dt + \int \mathrm{e}^{-it(a+b)} dt\right) \\ &= \frac{1}{2}\left(\frac{1}{i(a-b)}\mathrm{e}^{it(a-b)} - \frac{1}{i(a+b)}\mathrm{e}^{-it(a+b)}\right) \end{aligned}$$

Where the constant of integration has been dropped as this will later be used in a finite integral. This simplifies to:

$$H_1(f) = \frac{1}{2i}\frac{\mathrm{e}^{-ibt}}{(a-b)(a+b)}\left((a+b)\mathrm{e}^{ita} - (a-b)\mathrm{e}^{-ita}\right)$$

Grouping the a terms together and the b terms together:

$$H_1(f) = \frac{1}{2i}\frac{e^{-ibt}}{(a-b)(a+b)}\left(a\left(e^{ita} - e^{-ita}\right) + b\left(e^{ita} + e^{-ita}\right)\right)$$

Recalling the complex exponential forms of sine and cosine:

$$\frac{1}{2i}\left(\mathrm{e}^{i\theta} - \mathrm{e}^{-i\theta}\right) = \sin\theta$$

$$\frac{1}{2}\left(\mathrm{e}^{i\theta} + \mathrm{e}^{-i\theta}\right) = \cos\theta$$

$H_1(f)$ simplifies further to:

$$H_1(f) = \frac{\mathrm{e}^{-ibt}(a\sin at - ib\cos at)}{(a-b)(a+b)}$$

Returning to the Fourier transform of the boxcar with 10% cosine taper window function:

$$\begin{aligned} H(f) = \frac{5}{9T}\Bigg(&H_1(f)_{t=-\frac{2T}{5}} - H_1(f)_{t=-\frac{T}{2}} + \frac{1}{\pi f}\sin\pi fT + \frac{1}{\pi f}\sin\frac{4\pi fT}{5} + H_1(f)_{t=\frac{T}{2}} \\ &- H_1(f)_{t=\frac{2T}{5}}\Bigg) \\ = \frac{5}{9T}\Bigg(&H_1(f)_{t=\frac{T}{2}} - H_1(f)_{t=-\frac{T}{2}} - \left(H_1(f)_{t=\frac{2T}{5}} - H_1(f)_{t=-\frac{2T}{5}}\right) \\ &+ \frac{1}{\pi f}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)\Bigg) \end{aligned}$$

In this evaluation, we have two cases of

$$H_1(f)_{t=t^*} - H_1(f)_{t=-t^*}$$

where t^* is a constant. To avoid overly messy equations and repeating calculations, the following variable is used:

$$\Delta H_1(f)_{t=\pm t^*} \equiv H_1(f)_{t=t^*} - H_1(f)_{t=-t^*}$$

which is simplified below and then substituted into the following equation for $H(f)$.

$$H(f) = \frac{5}{9T}\left(\Delta H_1(f)_{t=\pm\frac{T}{2}} - \Delta H_1(f)_{t=\pm\frac{2T}{5}} + \frac{1}{\pi f}\left(\sin \pi f T + \sin\frac{4\pi f T}{5}\right)\right)$$

$$\begin{aligned}\Delta H_1(f)_{t=\pm t^*} &= \frac{\mathrm{e}^{-ibt^*}(a\sin at^* - ib\cos at^*)}{(a-b)(a+b)} - \frac{\mathrm{e}^{ibt^*}(-a\sin at^* - ib\cos at^*)}{(a-b)(a+b)} \\ &= \frac{\mathrm{e}^{-ibt^*}(a\sin at^* - ib\cos at^*) - \mathrm{e}^{ibt^*}(-a\sin at^* - ib\cos at^*)}{(a-b)(a+b)} \\ &= \frac{a\sin at^*\left(\mathrm{e}^{ibt^*} + \mathrm{e}^{-ibt^*}\right) + ib\cos at^*\left(\mathrm{e}^{ibt^*} - \mathrm{e}^{-ibt^*}\right)}{(a-b)(a+b)}\end{aligned}$$

Using the complex exponential forms of sine and cosine:

$$\Delta H_1(f)_{t=\pm t^*} = \frac{2}{(a-b)(a+b)}(a\sin at^*\cos bt^* - b\cos at^*\sin bt^*)$$

Which bears a striking resemblance to $\sin(a-b)t^*$. Putting this back into the definition of $H(f)$ yields:

$$\begin{aligned}H(f) = \frac{5}{9T}\Bigg(&\frac{2}{(a-b)(a+b)}\left(a\sin\frac{aT}{2}\cos\frac{bT}{2} - b\cos\frac{aT}{2}\sin\frac{bT}{2}\right) \\ &- \frac{2}{(a-b)(a+b)}\left(a\sin\frac{2aT}{5}\cos\frac{2bT}{5} - b\cos\frac{2aT}{5}\sin\frac{2bT}{5}\right) \\ &+ \frac{1}{\pi f}\left(\sin\pi f T + \sin\frac{4\pi f T}{5}\right)\Bigg) \\ = \frac{5}{9T}\Bigg(&\frac{2}{(a-b)(a+b)}\left(a\sin\frac{aT}{2}\cos\frac{bT}{2} - b\cos\frac{aT}{2}\sin\frac{bT}{2} - a\sin\frac{2aT}{5}\cos\frac{2bT}{5}\right. \\ &\left. + b\cos\frac{2aT}{5}\sin\frac{2bT}{5}\right) + \frac{1}{\pi f}\left(\sin\pi f T + \sin\frac{4\pi f T}{5}\right)\Bigg)\end{aligned}$$

Recalling $a = \frac{10\pi}{T}$ and $b = 2\pi f$,

$$H(f) = \frac{5}{9}\left(\frac{1}{\pi(5-fT)(5+fT)}\left(5\sin 5\pi\cos\pi fT - fT\cos 5\pi\sin\pi fT - 5\sin 4\pi\cos\frac{4\pi fT}{5} + fT\cos 4\pi\sin\frac{4\pi fT}{5}\right) + \frac{1}{\pi fT}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)\right)$$

$$= \frac{5}{9}\left(\frac{fT}{\pi(5-fT)(5+fT)}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right) + \frac{1}{\pi fT}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)\right)$$

$$= \frac{5}{9\pi}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)\left(\frac{fT}{(5-fT)(5+fT)} + \frac{1}{fT}\right)$$

$$= \frac{5}{9\pi}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)\left(\frac{5^2}{fT(5-fT)(5+fT)}\right)$$

Therefore, the Fourier transform of the boxcar with 10% cosine taper window is:

$$\boxed{H(f) = \frac{5^3}{9\pi fT(5-fT)(5+fT)}\left(\sin\pi fT + \sin\frac{4\pi fT}{5}\right)}$$

To compare this to the Fourier transform of the regular box car and see the effects of the cosine tapering, $H(f)$ is written as a function of $x = \pi fT$:

$$H(x) = \frac{5(5\pi)^2}{9x(5\pi - x)(5\pi + x)}\left(\sin x + \sin\frac{4x}{5}\right)$$

Recall that the Fourier transform of the boxcar window function (with the same definition of x) is:

$$H_{boxcar}(x) = \operatorname{sinc} x$$

A.3 Hann window

The Hann window, is usually defined for N discrete points as:

$$h(n) = \sin^2\left(\frac{\pi n}{N-1}\right)$$

for $\{n \in \mathrm{Z} \mid 0 \leq n \leq N-1\}$. In the continuous case of length T, the Hann window is:

$$h(t) = \begin{cases} A\sin^2\left(\frac{\pi t}{T}\right) & \text{for } 0 \leq \mathrm{t} \leq \mathrm{T} \\ 0 & \text{otherwise} \end{cases}$$

To centre the window around $t = 0$, like the boxcar and the boxcar with a 10% cosine taper, the Hann window is represented as:

$$h(t) = \begin{cases} A\cos^2\left(\frac{\pi t}{T}\right) & \text{for } |\mathrm{t}| \leq \frac{\mathrm{T}}{2} \\ 0 & \text{otherwise} \end{cases}$$

Using the trigonometry identity $\cos 2\theta = 2\cos^2\theta - 1$:

$$h(t) = \begin{cases} \frac{A}{2}\left(1 + \cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |\mathrm{t}| \leq \frac{\mathrm{T}}{2} \\ 0 & \text{otherwise} \end{cases}$$

To reduce the mathematics for the similarly shaped Hamming window (to be considered shortly), the normalization and Fourier transform for the Hann window is worked through using:

$$h(t) = \begin{cases} A\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |\mathrm{t}| \leq \frac{\mathrm{T}}{2} \\ 0 & \text{otherwise} \end{cases}$$

where $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{2}$ for the Hann window.

A is the normalization constant satisfying

$$\int_{-\infty}^{\infty} h(t)dt = 1$$

Then

$$\int_{-\infty}^{-T/2} 0dt + \int_{-T/2}^{T/2} A\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right)dt + \int_{T/2}^{\infty} 0dt = 1$$

$$A\int_{-T/2}^{T/2}\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right)dt = 1$$

$$A\left[\alpha t + \beta\frac{T}{2\pi}\sin\left(\frac{2\pi t}{T}\right)\right]_{-T/2}^{T/2} = 1$$

$$A\left(\frac{\alpha T}{2} - \left(-\frac{\alpha T}{2}\right) + \frac{\beta T}{2\pi}\sin\pi - \frac{\beta T}{2\pi}\sin -\pi\right) = 1$$

$$A\alpha T = 1$$

Hence

$$A = \frac{1}{\alpha T}$$

Recalling $\alpha = \frac{1}{2}$ for the Hann window, $A = \frac{2}{T}$ and the normalized Hann window is:

$$\boxed{h(t) = \begin{cases} \frac{2}{T}\cos^2\left(\frac{\pi t}{T}\right) & \text{for } |t| \le \frac{T}{2} \\ 0 & \text{otherwise} \end{cases}}$$

The Fourier transform of the Hann window (using α and β) is:

$$\begin{aligned} H(f) &= \int_{-\infty}^{\infty} h(t)\,\mathrm{e}^{-2\pi i f t}dt \\ &= \int_{-\infty}^{-T/2} 0\mathrm{e}^{-2\pi i f t}dt + \int_{-T/2}^{T/2}\frac{1}{\alpha T}\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right)\mathrm{e}^{-2\pi i f t}dt + \int_{T/2}^{\infty} 0\mathrm{e}^{-2\pi i f t}dt \\ &= \frac{1}{\alpha T}\int_{-T/2}^{T/2}\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right)\mathrm{e}^{-2\pi i f t}dt \end{aligned}$$

Using the complex exponential form of cosine:

$$H(f) = \frac{1}{\alpha T}\int_{-T/2}^{T/2}\left(\alpha + \frac{\beta}{2}\left(\mathrm{e}^{2\pi i t/T} + \mathrm{e}^{-2\pi i t/T}\right)\right)\mathrm{e}^{-2\pi i f t}\,dt$$

$$= \frac{1}{\alpha T}\int_{-T/2}^{T/2}\left(\alpha\mathrm{e}^{-2\pi i f t} + \frac{\beta}{2}\mathrm{e}^{2\pi i t(1-fT)/T} + \frac{\beta}{2}\mathrm{e}^{-2\pi i t(1+fT)/T}\right)dt$$

$$= \frac{1}{\alpha T}\left[\frac{i\alpha}{2\pi f}\mathrm{e}^{-2\pi i f t} - \frac{i\beta T}{4\pi(1-fT)}\mathrm{e}^{2\pi i t(1-fT)/T} + \frac{i\beta T}{4\pi(1+fT)}\mathrm{e}^{-2\pi i t(1+fT)/T}\right]_{-T/2}^{T/2}$$

$$= \frac{i}{2\pi fT}\left(\mathrm{e}^{-\pi i fT} - \mathrm{e}^{\pi i fT}\right) - \frac{i\beta}{4\pi\alpha(1-fT)}\left(\mathrm{e}^{\pi i(1-fT)} - \mathrm{e}^{-\pi i(1-fT)}\right)$$

$$- \frac{i\beta}{4\pi\alpha(1+fT)}\left(-\mathrm{e}^{-\pi i(1+fT)} + \mathrm{e}^{\pi i(1+fT)}\right)$$

Recalling $\mathrm{e}^{i\pi} = -1$ and $\mathrm{e}^{-i\pi} = -1$:

$$H(f) = \frac{-i}{2\pi fT}\left(\mathrm{e}^{\pi i fT} - \mathrm{e}^{-\pi i fT}\right) - \frac{i\beta}{4\pi\alpha(1-fT)}\left(\mathrm{e}^{\pi i fT} - \mathrm{e}^{-\pi i fT}\right)$$

$$+ \frac{i\beta}{4\pi\alpha(1+fT)}\left(\mathrm{e}^{\pi i fT} - \mathrm{e}^{-\pi i fT}\right)$$

$$= \frac{-i}{2}\left(\mathrm{e}^{\pi i fT} - \mathrm{e}^{-\pi i fT}\right)\left(\frac{1}{\pi fT} + \frac{\beta}{2\pi\alpha(1-fT)} - \frac{\beta}{2\pi\alpha(1+fT)}\right)$$

$$= \sin\pi fT\left(\frac{\alpha + (fT)^2(\beta-\alpha)}{\pi\alpha fT(1-fT)(1+fT)}\right)$$

For the Hann window with $\alpha = \frac{1}{2}$ and $\beta = \frac{1}{2}$, this yields:

$$H(f) = \sin\pi fT\left(\frac{1}{\pi fT(1-fT)(1+fT)}\right)$$

$$\boxed{H(f) = \frac{\operatorname{sinc}\pi fT}{(1-fT)(1+fT)}}$$

A.4 Hamming window

The Hamming window, is usually defined for N discrete points as:

$$h(n) = \frac{25}{46} - \frac{21}{46}\cos\left(\frac{2\pi n}{N-1}\right)$$

for $0 \leq n \leq N-1$. In the continuous case of length T, the Hamming window is:

$$h(t) = \begin{cases} A\left(\frac{25}{46} - \frac{21}{46}\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } 0 \leq \text{t} \leq \text{T} \\ 0 & \text{otherwise} \end{cases}$$

To centre the window around $t = 0$, as for the other windows, the Hamming window is represented as:

$$h(t) = \begin{cases} A\left(\frac{25}{46} + \frac{21}{46}\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |\text{t}| \leq \frac{\text{T}}{2} \\ 0 & \text{otherwise} \end{cases}$$

Note that this is the same form as the Hann window

$$h(t) = \begin{cases} A\left(\alpha + \beta\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |\text{t}| \leq \frac{\text{T}}{2} \\ 0 & \text{otherwise} \end{cases}$$

with $\alpha = \frac{25}{46}$ and $\beta = \frac{21}{46}$.

Using the equation from the Hann window, the normalization constant for the Hamming window is:

$$A = \frac{1}{\alpha T}$$

$$A = \frac{1}{\frac{25}{46}T}$$

$$A = \frac{46}{25T}$$

Therefore, the normalized Hamming window centred around $t = 0$ is:

$$\boxed{h(t) = \begin{cases} \frac{1}{T}\left(1 + \frac{21}{25}\cos\left(\frac{2\pi t}{T}\right)\right) & \text{for } |t| \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases}}$$

Using equation for the Fourier transform from the Hann window, the Fourier transform of the Hamming window is:

$$\begin{aligned} H(f) &= \sin \pi f T \left(\frac{\alpha + (fT)^2(\beta - \alpha)}{\pi \alpha f T(1 - fT)(1 + fT)}\right) \\ &= \sin \pi f T \left(\frac{\frac{25}{46} + (fT)^2\left(\frac{21}{46} - \frac{25}{46}\right)}{\pi \frac{25}{46} f T(1 - fT)(1 + fT)}\right) \\ &= \sin \pi f T \left(\frac{25 - 4(fT)^2}{25\pi f T(1 - fT)(1 + fT)}\right) \end{aligned}$$

$$\boxed{H(f) = \operatorname{sinc} \pi f T \left(\frac{(5 - 2fT)(5 + 2fT)}{25(1 - fT)(1 + fT)}\right)}$$

Appendix B: Matlab Code

This appendix contains the Matlab scripts and functions used for analysis.

B.1 FFT Script

```
%% Run first
% Makes all interpolated variable files.
% Change site, beamdir, mon, start, and stop as necessary.
site='negrocreek';      % radar site name
beamdir='N';            % beam direction
mon=200901:1:200912;    % months of interest, format YYYYMM
start=mon*100+ones(1,12);   % days to start interpolation at, format YYMMDD
stop=mon*100+[31,28,31,30,31,30,31,31,30,31,30,31]; % days to stop...
    % interpolation at, format YYYYMMDD

for i=1:length(start)
    for a=1.0:0.5:14.0;
        alt=num2str(a,'%2.1f');
        m=int2str(mon(i));
        close all
        setReadInt(site,alt,beamdir,start(i),m,stop(i),1);
    end
end

%% Run second
% Performs the FFT on the interpolated data for all 4 window types.
% Change site and beamdir as necessary.
site='eureka';  % radar site
beamdir='N';    % beam direction

for a=1.0:0.5:14.0;
    alt=num2str(a,'%2.1f');
    fileend=['.',site,'.',alt,beamdir,'_15min.mat'];
    for ym=200901:200912
        yyyymm=int2str(ym);
        filename=['FFT/',site,'/',yyyymm,'/',yyyymm,fileend];
        for wintype=1:4;
            FFTrun(filename,site,alt,beamdir,yyyymm,wintype)
            wt=int2str(wintype);
            filename2=['FFT/',site,'/',yyyymm,'/',yyyymm,'.win',wt,'.',...
                site,'.',alt,beamdir,'_15min.mat'];
            load(filename2,'w','I')

            % Get slope
            % Change f to be between 2 days and 6 hours
            startf=1/(2*24*60*60);  % 2 days
            endf=1/(6*60*60);       % 6 hours
            f=w(w>=startf & w<=endf);
            PSD=I(w>=startf & w<=endf);
            % Want the least squares fit line to
            %       log(PSD) =  n log(f) (+c)
            y=log10(PSD);
            x=log10(f);
```

```
            c=polyfit(x,y,1);
            u=10^c(2)*f.^c(1);
            slope=c(1);
            amp=c(2);
            fPSD=f;
            save(filename2,'fPSD','PSD','slope','amp','-append')
        end
    end
end

%% Run third
% Gather the slopes into one annual file.
% Change site, beamdir, y, and alt as necessary.
site='eureka';  % radar site
beamdir='N';    % beam direction
y=2009;         % year
yyyy=int2str(y);
alt=1:0.5:14;   % altitudes
for win=1:4;    % all windows
    wt=int2str(win);
    slopes=zeros(length(alt),12);
    amps=slopes;
    for j=1:length(alt)
        altstr=num2str(alt(j),'%2.1f');
        for m=1:12
            yyyymm=int2str(y*100+m);
            filename=['FFT/',site,'/',yyyymm,'/',yyyymm,'.win',wt,'.',...
                site,'.',altstr,beamdir,'_15min.mat'];
            load(filename,'slope','amp')
            slopes(j,m)=slope;
            amps(j,m)=amp;
        end
    end
    savefile=['FFT/',site,'/win',wt,'.',site,'.',beamdir,'_15min.mat'];
    save(savefile,'slopes','amps','-append')
end

%% Run fourth
% Plots spectral slopes for a year on an altitude vs. month plot.
% Change site, beamdir, and yyyy as necessary.
site='Eureka';  % radar site
beamdir='N';    % beam direction
yyyy='2009';    % year (string)
for win=1:4
    wt=int2str(win);
    file=['FFT/',site,'/win',wt,'.',site,'.',beamdir,'_15min.mat'];
    load(file,'slopes','npoints')
    x=slopes;
    x(x==-999)=NaN;

    figure
    imagescnanEmily(x)
    set(gca,'YDir','normal')
    % Make the plot look pretty:
    axis tight
    set(gca,'fontsize',14)
    set(gca,'YTick',[1 5 9 13 17 21 25])
```

```
    set(gca,'YTickLabel',{'1.0','3.0','5.0','7.0','9.0','11.0','13.0'})
    set(gca,'XTick',1:12)
    set(gca,'XTickLabel',{'J','F','M','A','M','J','J','A','S','O','N','D'})
    ylabel('Altitude, km','FontSize',18)
    xlabel('Month','FontSize',18)

    % Title, change as necessary:
    text=[site,' FFT Window',wt];
    title(text,'FontSize',20)

    % Colour bar and labels:
    h=colorbar('location','EastOutside');
    ylabel(h,'Spectral slope','FontSize',18)
    set(h,'fontsize',14)
    set(h,'YTickLabel',{'-2.5','-2.0','-1.5','-1.0','-0.5','0'})
    caxis([-2.5 0])
end
```

B.2 DCDFT Script

```
%% Run first.
% Calculates DCDFT values, slopes, and amps with window applied
% varfileList.txt is a list of file names in the format:
%   site/varfile/YYYY/YYYYMM/YYYYMM.site.altN_1min.mat
% where 'site' is the radar site name, 'YYYY' is the year, 'YYYYMM' is the
% year and month, 'alt' is the altitude in the form %2.1f, and 'N' can be
% replaced by another beam direction (E, S, or W).
fID=fopen('varfileList.txt');
file=strtrim(fgetl(fID));
win=1;      % window type, see winSel.m for the windows
wt=int2str(win);
while strcmp(file,'end')==0
    DCDFT1(file,win)
    newfile=[file(1:length(file)-3),'win',wt,'.mat'];
    load(newfile)
    I=2*(length(F)^(-1)).*(F.*conj(F));
    save(newfile,'I','-append')
    % Change f to be between 2 days and 6 hours
    startf=1/(2*24*60*60);  % 2 days
    endf=1/(6*60*60);       % 6 hours
    f=w(w>=startf & w<=endf);
    PSD=I(w>=startf & w<=endf);

    % Want the least squares fit line to
    %       log(PSD) =  n log(f) (+c)
    y=log10(PSD);
    x=log10(f);
    c=polyfit(x,y,1);

    u=10^c(2)*f.^c(1);
    slope=c(1);
    amp=c(2);
    fPSD=f;
    save(newfile,'fPSD','PSD','slope','amp','-append')
    file=strtrim(fgetl(fID));
end
```

```
%% Run second
% With DCDFT window applied, gather multiple slopes and amps into one file.
% Change site, years, beamdir, win, and alt as necessary.
site='markstay';    % radar site
years=2010:2014;    % years as integers
beamdir='N';        % beam direction
win=1;              % window type, see winSel.m for the windows
wt=int2str(win);
alt=1.0:0.5:14.0;   % altitudes

for i=1:length(years)
    y=years(i);
    yyyy=int2str(y);

    slopes=ones(length(alt),12)*-999;
    amps=slopes;
    npoints=zeros(length(alt),12);
    for j=1:length(alt)
        altstr=num2str(alt(j),'%2.1f');
        for m=1:12;
            yyyymm=int2str(y*100+m);

varfile=[site,'/varfile/',yyyy,'/',yyyymm,'/',yyyymm,'.',site,...
                '.',altstr,'N_1min.win',wt,'.mat'];
            if exist(varfile,'file')~=0;
                load(varfile,'slope','amp','w')
                slopes(j,m)=slope;
                amps(j,m)=amp;
                npoints(j,m)=length(w);
            else
                fprintf('No varfile for %s\r\n',varfile)
            end
        end
    end

savefile=['slopesandamps/',site,'/win',wt,'.',yyyy,'.',site,'.1minN.mat'];
    save(savefile,'slopes','amps','npoints')
end

%% Run third
% Plots spectral slopes for all years on an altitude vs. month plot.
% Change site, wt, and y as necessary.
site='Markstay';    % radar site
wt='1';             % window type, see winSel.m for numbers
% Set minimum number to base spectral estimates off of:
minnp=240;  % Corresponds to once every 3 hours (so that it can measure
            % at least 6 hour frequencies.

for y=2014:-1:2010  % years
    yyyy=int2str(y);
    file=['slopesandamps/',site,'/win',wt,'.',yyyy,'.',site,'.1minN.mat'];
    load(file,'slopes','npoints')
    x=slopes;
    x(x==-999)=NaN;
    x(npoints<minnp)=NaN;
```

```
    x(isnan(npoints)==1)=NaN;

    figure
    imagescnanEmily(x)
    set(gca,'YDir','normal')
    set(gca,'fontsize',14)
    axis tight
    set(gca,'Color',[0 0 0]);
    set(gca,'YTick',[1.5  5.5 9.5 13.5 17.5 21.5 25.5])
    set(gca,'YTickLabel',{'1.0','3.0','5.0','7.0','9.0','11.0','13.0'})
    set(gca,'XTick',1.5:12.5)
    set(gca,'XTickLabel',{'J','F','M','A','M','J','J','A','S','O','N','D'})
    ylabel('Altitude, km','FontSize',18)
    xlabel('Month','FontSize',18)

    text=[site,' ',yyyy];
    title(text,'FontSize',20)

    h=colorbar('location','EastOutside');
    ylabel(h,'Spectral slope','FontSize',18)
    set(h,'fontsize',14)
    set(h,'YTickLabel',{'-2.5','-2.0','-1.5','-1.0','-0.5','0'})
    caxis([-2.5 0])
end

%% Run fourth
% Calculates means, standard deviations, and the standard deviation of the
% mean.
% Change site, wt, and y as necessary.
site='Markstay';    % radar site
wt='1';             % window type, see winSel.m for numbers
alls=[];
allnp=[];
for y=2010:2014     % years
    yyyy=int2str(y);
    file=['slopesandamps/',site,'/win',wt,'.',yyyy,'.',site,'.1minN.mat'];
    load(file,'slopes','npoints')

    alls=[alls,slopes];
    allnp=[allnp,npoints];
end

% See how many points are used to calculate the average.
n=ones(size(alls));
n(isnan(alls)==1)=0;
n(alls==-999)=0;
n(allnp<240)=0;
size(n(1:2,:))
n(1:2,:)=zeros(size(n(1:2,:)));
n(22:27,:)=zeros(size(n(22:27,:)));
N=sum(sum(n));

% Format data so that no/not enough data is zero slope
x=alls;
x(isnan(x)==1)=0;
```

```
x=x.*n;

% Calculate overall mean
mean=sum(sum(x))/N;
diff=abs(-5/3-mean);

% Calculate std
std=(sum(sum(((x-mean).^2.*n)))/(N-1)).^1/2;
stdofmean=std/N.^(1/2);
```

B.3 Breakpoint script

```
% Hand select the break point frequency for all windows.
% Change the site, beamdir, and a as necessary.
site='negrocreek';  % radar site
beamdir='N';        % beam direction
a=1.0:0.5:14.0;     % altitude
i=0;
for win=1:4         % window type, see winSel.m for values.
    wt=int2str(win);
    bp=ones(length(a),12)*NaN;
    for j=1:length(a)
        alt=num2str(a(j),'%2.1f');
        for m=1:12
            i=i+1;
            yyyymm=int2str(200900+m);
            file=['FFT/',site,'/',yyyymm,'/',yyyymm,'.win',wt,'.',...
                site,'.',alt,beamdir,'_15min.mat'];
            %file=['DCDFT/',site,'/',yyyymm,'/',yyyymm,'.win',wt,'.',...
            %    site,'.',alt,beamdir,'_1min.mat'];
            load(file,'I','w')
            f=figure;
            loglog(w,I,'k')
            text=int2str(i);
            title(text)

            [x,y]=ginput(1);
            close(f)
            % Store breakpoint value:
            bp(j,m)=x;
        end
    end
    savefile=['FFT/',site,'/breakpoints2009.win',wt,'.',...
        site,'.',beamdir,'.mat'];
    %     savefile=['DCDFT/',site,'/breakpoints2009.win',wt,'.',...
    %         site,'.',beamdir,'.mat'];
    save(savefile,'bp')
end
```

B.4 createtf

```
function varfile=createtf(site,alt,beamdir,tint,start,stop)
%{
CREATETF  Reads in raw data and makes a new file with the values of f and
t.
    CREATETF(SITE,ALT,BEAMDIR,TINT,START) reads in the raw data and creates
```

```
a file of f and t for DCDFT.  The site, altitude, beam direction, and time
interval are given by the chracter strings SITE, ALT, BEAMDIR, and TINT
respectively.  ALT and TINT may be numbers.  The data spans the yyyymmdd
day START.

    CREATETF(...,STOP) saves f and t for days between START and STOP
inclusively.
%}

if nargin==5
    % Only use one day
    stop=start;
elseif nargin~=6
    error('SRI:argChk','Wrong number of input arguments.')
elseif isa(alt,'char')==0
    alt=num2str(alt,'%2.1f');
elseif isa(tint,'char')==0
    tint=num2str(tint,'%2.0i');
end

%**************************************************************************%
%           SET DAYS AS A VECTOR FROM START TO STOP
%**************************************************************************%

% Set a matrix for the days of the month for leap and non-leap years
n=[31,28,31,30,31,30,31,31,30,31,30,31];
% First row is for leap years, second row is for non-leap years, columns
% are representative days in the month.
numofdays=[n;n];
numofdays(1,2)=29;
clear('n')

% Break start and stop into year, month, day
yyyy=floor(start/10000);
mm=floor((start-yyyy*10000)/100);
dd=mod(start,100);
stopy=floor(stop/10000);
stopm=floor((stop-stopy*10000)/100);
stopd=mod(stop,100);

% See if start and stop are the same month.  If so, just gather days
% between.
if yyyy==stopy && mm==stopm
    days=yyyy*10000+mm*100+(dd:stopd);
else
    % Spans multiple months.
    % Determine if it is a leap year (use row 1) or not (use row 2).
    leap=ceil(mod(yyyy,4)/4)+1;
    % Put the remaining days of the current month into days.
    days=yyyy*10000+mm*100+(dd:numofdays(leap,mm));

    % Determine days between the start and stop months:
    while yyyy~=stopy && (mm+1)~=stopm
        mm=mm+1;
        if mm==13
```

```
            % If we reach "month 13", roll the clock over a year.
            yyyy=yyyy+1;
            % Determine if it is a leap year:
            leap=ceil(mod(yyyy,4)/4)+1;
            % Reset mm to zero since we add 1 to it before doing
            % anthing else (otherwise we would skip January every roll
            % over).
            mm=0;
        else
            % Include the days of this month in days.
            days=[days,yyyy*10000+mm*100+(1:numofdays(leap,mm))];
        end
    end

    % Include the first days of the stop month.
    days=[days, yyyy*10000+(mm+1)*100+(1:stopd)];
end
clear('numofdays','yyyy','mm','dd','stopy','stopm','stopd','leap')

%************************************************************************%
%           READ AND SORT DATA
%************************************************************************%

% Create filepath
fileend=['.',site,'.',alt,beamdir,'_',tint,'min.txt'];

vel=[];
npoints=[];
t=[];
for i=1:length(days)
    % Read each file and put into variables
    day=int2str(days(i));
    mon=day(1:6);
    yr=day(1:4);
    filepath=[site,'/',yr,'/',mon,'/',day,'/RadarData/',day,fileend];
    raw=dlmread(filepath,',',0,1);

    % Convert time into seconds
    time=raw(:,6)+60*(raw(:,5)+60*raw(:,4));
    % Sort data chronologically
    [time,order]=sort(time);
    vel=[vel;raw(order,8)];
    npoints=[npoints;raw(order,9)];
    % Convert add the appropriate number of days (in seconds) to time and
    % combine it with our t vector
    t=[t;time+60*60*24*(i-1)];
end

%************************************************************************%
%           GET RID OF -999.00 DATA AND ALL UNNEEDED VARIABLES
%************************************************************************%
t=t(vel>-999);
f=vel(vel>-999);

% For variables at the end
```

```
varfile=[site,'/varfile/',yr,'/',mon,'/',mon,'.',site,'.',alt,beamdir,'_',ti
nt,'min.mat'];
save(varfile,'t','f')
end
```

B.5 DCDFT1

```
function F=DCDFT1(varfile,wintype)
%{
DCDFT1    Calculates the power spectral density using the date-compensated
discrete Fourier transform as described by Ferraz-Mello (1981).
    DCDFT1(VARFILE,WINTYPE) saves the power spectral density (F) and
corresponding frequencies (w) to a file by the same name as VARFILE with
the window code at the end.  VARFILE should be the name of the file
containing the velocities (f) and time stamps (t) in the format:
    SITE/varfile/YYYY/YYYYMM/YYYYMM.SITE.ALTN_1min.mat
where SITE is the name of the radar site, YYYY is the year, YYYYMM is the
year and month, ALT is the altitude in the format %2.1f, and N is the beam
direction (N, E, S, or W).  WINTYPE is the window type, as specified by
winSel.m
%}

load(varfile)
wt=int2str(wintype);
newfile=[varfile(1:length(varfile)-3),'win',wt,'.mat'];
%**************************************************************************%
%            DETERMINE W AND CHECK DIMENSIONS
%**************************************************************************%
N=length(f);
% Find if spectra is odd or even in length as it slightly changes the math.
if floor(N/2)~=N/2
    % Then the signal length is odd
    offset=1;
else
    % The signal length is even
    offset=0;
end
% Sample frequency is one sample/unit time
ws=N/(max(t)-min(t));      % Hz
% Nyguist frequency (maxiumum measurable frequency) is half of the sample
% frequency
wN=ws/2;          % Hz
% The Fourier transform has points equally spaced between zero and the
% Nyguist frequency.  This corresponds to a frequency resolution of:
dw=ws/N;
% The FFT is at the frequencies:
w=(-wN+dw/2*offset:dw:wN-dw/2*offset)';
clear('offset','ws','wN','dw')
save(newfile,'w')

%**************************************************************************%
%            FERRAZ MELLO (1981)
%**************************************************************************%
% (3) Add constant so that sum(f) over all t = 0
f=f-sum(f)/N;
% Apply window
```

```
win=winSel(length(f),wintype);
f=f.*win;

F=zeros(size(w));
H0=ones(size(t));
a0=N^(-1/2);
h0=a0*H0;
clear('N')

for j=1:length(w)
    H1=cos(2*pi*w(j)*t);
    H2=sin(2*pi*w(j)*t);

    a1=(sum(H1.*H1)-a0^2*sum(H0.*H1)^2)^(-1/2);
    a2=(sum(H2.*H2)-a0^2*sum(H0.*H2)^2-a1^2*sum(H1.*H2)^2-...
        a1^2*a0^4*sum(H0.*H1)^2*sum(H0.*H2)^2+...
        2*a0^2*a1^2*sum(H0.*H1)*sum(H0.*H2)*sum(H1.*H2))^(-1/2);

    h1=a1*H1-a1*h0*sum(h0.*H1);
    clear('a1','H1')
    h2=a2*H2-a2*h0*sum(h0.*H2)-a2*h1*sum(h1.*H2);
    clear('a2','H2')

    F(j)=sum(f.*(h1+sqrt(-1)*h2))/(a0*sqrt(2));
    clear('h1','h2')
end
save(newfile,'F','-append')
end
```

B.6 FFTrun

```
function FFTrun(filename,site,alt,beamdir,yyyymm,wintype)
%{
FFTRUN    Calculates and saves the power spectral density and slope
estimations.
    FFTRUN(FILENAME,SITE,ALT,BEAMDIR,YYYYMM,WINTYPE) save the power
spectral density and slopes for the variables in FILENAME to the file:
FFT/SITE/YYYYMM/YYYYMM.winWT.SITE.ALTBEAMDIR_15min.mat
where SITE is teh radar site name, ALT is teh altitude in format %2.1f,
BEAMDIR is the beam direction (N, E, S, or W), YYYYMM is the year and
month, and WINTYPE is the window code (see winSel.m for the codes).
%}

load(filename)
% Get FFT spectra for various windows
%************************************************************************%
%           COMPUTE FFT AND PLOT POWER SPECTRA
%************************************************************************%
% Subtract the mean from the data
len=length(vel);
y=vel-(sum(vel)/len);
% Multiply by window
win=winSel(len,wintype);
y=y.*win;
Y=fft(y)/len;
```

```
PSD=Y.*conj(Y);
% Arrange the PSD from most negative to most positive frequency.  Double
% counts the Nyguist frequncy.
PSDpos=PSD(1:floor(len/2)+1);
PSDneg=PSD(ceil(len/2)+1:len);
PSD=[PSDneg;PSDpos];
clear('win','y','Y','PSDneg');

% Find if spectra is odd or even in length as it impacts the frequencies.
if floor(len/2)~=len/2
    % Then the signal length is odd
    offset=1;
else
    % The signal length is even
    offset=0;
end

% The Fourier transform frequency spacing is
df=1/max(t);    % Hz
% Sample frequency is one sample/unit time (or equivalently the number of
% samples divided by the total time)
sf=len*df;      % Hz
% Nyguist frequency (maxiumum measurable frequency) is half of the sample
% frequency
Nyg=sf/2;       % Hz
% The FFT is at the frequencies:
fpos=(0:df:Nyg-df/2*offset)';
f=(-Nyg+df/2*offset:df:Nyg-df/2*offset)';
clear('offset','Nyg','sf','df','offset')

% CHECK: length of PSD and f is the same
if size(f)~=size(PSD)
    display('CHECK: Size of f and PSD do not match!')
end
% CHECK: centre of f is 0
if f(ceil(length(f)/2))~=0
    display('CHECK: Middle f value is not 0!')
end
w=f;
wpos=fpos;
I=PSD;
Ipos=PSDpos;
wt=int2str(wintype);
filename=['FFT/',site,'/',yyyymm,'/',yyyymm,'.win',wt,'.',...
    site,'.',alt,beamdir,'_15min.mat'];
save(filename,'I','Ipos','w','wpos')
end
```

B.7 imagescnanEmily

```
function [H,HNAN] = imagescnanEmily(varargin)
%IMAGESCNAN   Scale data and display as image with uncolored NaNs.
%Edited by E.M. 20130516 to change the colourmap.
%Now: Uses a Moreland colormap (red to blue, through white) which is better
%for colour deficient folks (red/green). See http://www.sandia.gov/~kmorel/
%documents/ColorMaps/ and his article Diverging Color Maps for Scientific
```

```
%Visualization (Expanded) for details: "Diverging Color Maps for Scientific
%Visualization." Kenneth Moreland. In Proceedings of the 5th International
%Symposium on Visual Computing, December 2009.
%DOI 10.1007/978-3-642-10520-3_9.
%
%   SYNTAX:
%                imagescnan(U)
%                imagescnan(U,...,'NanColor',CNAN)
%                imagescnan(U,...,'NanMask',MNAN)
%                imagescnan(U,...,IOPT)
%                imagescnan(X,Y,U,...)
%     [H,HNAN] = imagescnan(...);
%
%   INPUT:
%     U    - 2 dimensional N-by-M image or N-by-M-by-3 RGB image.
%     X    - 2 extrema X-axis data; or the M values; or the N-by-M values
%            as obtained from MESHGRID (see DESCRIPTION below).
%            DEFAULT: [1 N]
%     Y    - 2 extrema X-axis data; or the N values; or the N-by-M values
%            as obtained from MESHGRID (see DESCRIPTION below).
%            DEFAULT: [1 M]
%     CNAN - Color for the NaNs elements. May be a char specifier or an [R
%            G B] triplet specifying the color.
%            DEFAULT: invisible (axes background color)
%     MNAN - Elements to be ignored besides not finite values. May be an
%            scalar or a logical M-by-N matrix indicating the elements to
%            be ignored.
%            DEFAULT: []
%     IOPT - IMAGE function normal optional pair arguments like
%            ('Parent',H) or/and CLIM like optional last argument as in
%            IMAGESC.
%            DEFAULT: none
%     map = the name of a colourmap to use. Default is Moreland (cool to
%     warm) but you could also put in "jet" or "gray".
%
%   OUTPUT (all optional):
%     H    - Image handle
%     HNAN - Handle of every ignored (NaN) value colored patch.
%
%   DESCRIPTION:
%     MATLAB function IMAGESC does not work properly with NaNs. This
%     programs deals with this problem by including colored patches over
%     this elements and maybe others specyfied by the user with MNAN.
%
%     Besides, those functions does not work properly with X,Y values
%     variable interval, but this functions does it by generating a whole
%     new image of several rectangular patches, but whose centers may not
%     lay in the specified coordinate (see NOTE below). This functionality
%     is experimental and not recommended (see ADDITIONAL NOTES inside this
%     program).
%
%     In previous release, 2-dim input images were transformed into a
%     3-dim RGB image. This is not used anymore (see ADDITIONAL NOTES
%     inside this file).
%
%   NOTE:
%     * Optional inputs use its DEFAULT value when not given or [].
```

```
%     * Optional outputs may or not be called.
%     * If X is a two element vector, min(X) will be the coordinate of the
%       first column and max(X) of the last column.
%     * If Y is a two element vector, min(Y) will be the coordinate of the
%       first row and max(Y) of the last row.
%     * If vector X-axis is decreasing U=fliplr(U) will be used.
%     * If vector Y-axis is decreasing U=flipud(U) will be used.
%     * When X or Y do not have a constant increasing/decreasing step, the
%       vertices of the color rectangules are set in the middle of each
%       pair of coordinates. For this reason its center may not lay on the
%       specified coordinate, except on the coordinates at the edges where
%       it always lays on the center.
%     * To get a non-scaled image (IMAGE instead of IMAGESC) use:
%         >> H = imagescnan(...);
%         >> set(H,'CDataMapping','direct')
%     * ADDITIONAL NOTES are included inside this file.
%
%   EXAMPLE:
%     % Compares with normal IMAGESC:
%      N     = 100;
%      PNaNs = 0.10;
%      U     = peaks(N);
%      U(round(1 + (N^2-1).*rand(N^2*PNaNs,1))) = NaN;         % Adds NaNs
%      subplot(221), imagesc(U)
%       title('With IMAGESC: ugly NaNs')
%      subplot(222), imagescnan(U)
%       title('With IMAGESCNAN: uncolored NaNs')
%     % Compares with SPY:
%      subplot(223), spy(isnan(U))
%       title('SPY(isnan(U))')
%      subplot(224), imagescnan(isnan(U),'NaNMask',0), axis equal tight
%       title('SPY with IMAGESCNAN')
%
%   SEE ALSO:
%     IMAGE, IMAGESC, COLORBAR, IMREAD, IMWRITE
%     and
%     CMAPPING, CBFREEZE by Carlos Vargas
%     at http://www.mathworks.com/matlabcentral/fileexchange
%
%
%   ---
%   MFILE:   imagescnan.m
%   VERSION: 2.1 (Aug 20, 2009) (<a
href="matlab:web('http://www.mathworks.com/matlabcentral/fileexchange/author
s/11258')">download</a>)
%   MATLAB:  7.7.0.471 (R2008b)
%   AUTHOR:  Carlos Adrian Vargas Aguilera (MEXICO)
%   CONTACT: nubeobscura@hotmail.com

%   ADDITIONAL NOTES:
%     * I keep getting a kind of BUG with the edges of the patched NaNs. I
%       added two NOTE inside this program that may fix this problem.
%       Another way is to convert the intensity matrix U into RGB colors by
%       using the CMAPPING function, as used by the first version of this
%       program.
%     * Besides, if the matrix is too large, sometimes there is an
%       undocumented failure while drawing the patch NaNs. Is recommended
```

```
%       to use U = cmapping(U,[],'k','discrete') instead, and change the
%       CLIM to [min(U(:)) max(U(:))].
%     * The use of not homogeneous step interval X,Y axes is not
%       recommended because the program tries to put its value in the
%       middle of the colored rectangle (as IMAGESC does) and soetimes the
%       result may not be what the user wants. So this is for experimental
%       use only.

%   REVISIONS:
%   1.0      Released. (Jun 30, 2008)
%   1.1      Fixed bug when CAXIS used. Colorbar freezed colormap. Fixed
%            bug in color vector input (Found by Greg King) and now
%            accets RGB image as input. (Jul 14, 2008)
%   2.0      Totally rewritten code. Do not converts to RGB anymore. Do not
%            freezes the colormap anymore. Do not output any colorbar. New
%            X and Y variable steps accepted input. Now uses patches. (Jun
%            08, 2009)
%   2.1      Fixed bug with RGB input. Added a NOTE about the use of
%            CMAPPING. (Aug 20, 2009)

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

%   Copyright (c) 2008,2009 Carlos Adrian Vargas Aguilera


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

% Initializes:
X    = [];
Y    = [];
CNAN = [];
MNAN = [];
ha   = [];
chooseCMAP = 'jet';
% chooseCMAP = 'MorelandColormapB';

% chooseCMAP = 'reversegray';
% map = gray; %EM added 20130614
% map = MorelandColormapB;

% Checks number of inputs:
if     nargin<1
 error('CVARGAS:imagescnan:notEnoughInputs',...
  'At least 1 input is required.')
elseif nargout>2
 error('CVARGAS:imagescnan:tooManyOutputs',...
  'At most 2 outputs are allowed.')
end

% Gets X,Y,U:
if ((nargin==1) || (nargin==2))
 U = varargin{1};
```

```
 varargin(1) = [];
else
 if (isnumeric(varargin{1}) && isnumeric(varargin{2}) && ...
   isnumeric(varargin{3}))
  X = varargin{1};
  Y = varargin{2};
  U = varargin{3};
  varargin(1:3) = [];
 else
  U = varargin{1};
  varargin(1) = [];
 end
end

% Check U:
ndim = ndims(U);
if      (ndim==2)
 [M,N]   = size(U);
 O = 1;
elseif (ndim==3)
 [M,N,O] = size(U);
 if (O~=3)
  error('CVARGAS:imagescnan:incorrectRgbImage',...
   'RGB image must be of size M-by-N-by-3.')
 end
else
 error('CVARGAS:imagescnan:incorrectImageSize',...
  'Image must be 2-dimensional or a 3-dim RGB image.')
end

% Check X:
aequal = true;    % Equal intervals on x-axis?
dX     = [];
if isempty(X)
 X = [1 N];
else
 if (ndims(X)>2)
  error('CVARGAS:imagescnan:incorrectXDims',...
   'X must be a vector or a matrix as a result of MESHGRID.')
 end
 if any(~isfinite(X(:)))
  error('CVARGAS:imagescnan:incorrectXValue',...
   'X elements must be numeric and finite.')
 end
 [Mx,Nx] = size(X);
 if ((Mx*Nx)==2)
  if X(2)<X(1)
   X = X([2 1]);
   for k = 1:O % Fixed bug Aug 2009
    U(:,:,k) = fliplr(U(:,:,k));
   end
  end
 else
  if      ((Mx==M) && (Nx==N))
   % Checks if generated with MESHGRID:
   dX    = abs(X(2:M,:)-repmat(X(1,:),M-1,1));
```

```
    if any(abs(dX(:))>(eps*max(abs(dX(:)))*1000))
     error('CVARGAS:imagescnan:incorrectXMatrix',...
      'X matrix must be as generated by MESHGRID.')
    end
    X = X(1,:);
   elseif (~any([Mx Nx]==1) && ~((Mx*Nx)==N))
    error('CVARGAS:imagescnan:incorrectXSize',...
      'X must be an scalar or a matrix.')
   end
   % Forces ascending x-axis:
   [X,I] = sort(X(:).');
   for k = 1:O % Fixed bug Aug 2009
    U(:,:,k) = U(:,I,k);
   end
   clear I
   % Checks equal intervals:
   dX = diff(X);
   if any(abs(dX(1)-dX(2:end))>(eps*max(dX)*1000))
    if aequal
     aequal = false;
    end
   else
    X  = [X(1) X(end)];
    dX = [];
   end
  end
end

% Check Y:
dY = [];
if isempty(Y)
 Y = [1 M];
else
 if (ndims(Y)>2)
  error('CVARGAS:imagescnan:incorrectYDims',...
   'Y must be a vector or a matrix as a result of MESHGRID.')
 end
 if any(~isfinite(Y(:)))
  error('CVARGAS:imagescnan:incorrectYValue',...
   'Y elements must be numeric and finite.')
 end
 [My,Ny] = size(Y);
 if ((My*Ny)==2)
  if Y(2)<Y(1)
   Y = Y([2 1]);
   for k = 1:O % Fixed bug Aug 2009
    U(:,:,k) = flipud(U(:,:,k));
   end
  end
 else
  if     ((My==M) && (Ny==N))
   % Checks if generated with MESHGRID:
   dY = abs(Y(:,2:N)-repmat(Y(:,1),1,N-1));
   if any(abs(dY(:))>(eps*max(abs(dY(:)))*1000))
    error('CVARGAS:imagescnan:incorrectYMatrix',...
     'Y matrix must be as generated by MESHGRID.')
   end
```

```
   Y = Y(:,1);
  elseif (~any([My Ny]==1) && ~((My*Ny)==M))
   error('CVARGAS:imagescnan:incorrectYSize',...
     'Y must be an scalar or a matrix.')
  end
  % Forces ascending y-axis:
  [Y,I] = sort(Y(:).');
  for k = 1:O % Fixed bug Aug 2009
   U(:,:,k) = U(I,:,k);
  end
  clear I
  % Checks equal intervals:
  dY = diff(Y);
  if any(abs(dY(1)-dY(2:end))>(eps*max(dY)*1000))
   if aequal
    aequal = false;
   end
  else
   Y  = [Y(1) Y(end)];
   dY = [];
  end
 end
end

% Checks varargin:
ind  = [];
Nopt = length(varargin);
for k = 1:Nopt-1
 if (~isempty(varargin{k}) && ischar(varargin{k}))
  if     strncmpi(varargin{k},'NanColor',4)
   CNAN = varargin{k+1};
   ind  = [ind k k+1];
  elseif strncmpi(varargin{k},'NanMask',4)
   MNAN = varargin{k+1};
   ind  = [ind k k+1];
  elseif (strncmpi(varargin{k},'Parent',2) && isempty(CNAN))
   try
    CNAN = get(varargin{k+1},'Color');
    ha   = varargin{k+1};
   catch
    error('CVARGAS:imagescnan:incorrectParentHandle',...
     '''Parent'' must be a valid axes handle.')
   end
  end
 end
end
varargin(ind) = [];
Nargin = length(varargin);

% Check ha:
if isempty(ha)
 ha = gca;
end

% Check CNAN:
if     isempty(CNAN)
```

```
 CNAN = get(ha,'Color');
elseif ischar(CNAN)
 switch lower(CNAN)
  case 'y', CNAN = [1 1 0];
  case 'm', CNAN = [1 0 0];
  case 'c', CNAN = [0 1 1];
  case 'r', CNAN = [1 0 0];
  case 'g', CNAN = [0 1 0];
  case 'b', CNAN = [0 0 1];
  case 'w', CNAN = [1 1 1];
  case 'k', CNAN = [0 0 0];
  otherwise
   error('CVARGAS:imagescnan:incorrectNancString',...
    'Color string must be a valid color identifier. One of ''ymcrgbwk''.')
 end
elseif isnumeric(CNAN) && (length(CNAN)==3)
 CNAN = CNAN(:).'; % Forces row vector.
else
 error('CVARGAS:imagescnan:incorrectNancInput',...
  'Not recognized CNAN input.')
end

% Check MNAN:
if isempty(MNAN)
 MNAN = any(~isfinite(U),3);
else
 if (ndims(MNAN)==2)
  [Mm,Nm] = size(MNAN);
  if     ((Mm*Nm)==1)
   MNAN = (any(~isfinite(U),3) | any(U==MNAN,3));
  elseif ((Mm==M) && (Nm==N) && islogical(MNAN))
   MNAN = (any(~isfinite(U),3) | MNAN);
  else
   error('CVARGAS:imagescnan:incorrectNanmSize',...
   'MNAN must be an scalar or a logical matrix of size M-by-N.')
  end
 else
  error('CVARGAS:imagescnan:incorrectNanmDims',...
   'MNAN must be an scalar or a matrix.')
 end
end


% EXTRA COLOUR MAPS ADDED BY EMILY
% --------------------------------------------------------------------

reversegray =
[1,1,1;0.984126984126984,0.984126984126984,0.984126984126984;0.9682539682539
68,0.968253968253968,0.968253968253968;0.952380952380952,0.952380952380952,0
.952380952380952;0.936507936507937,0.936507936507937,0.936507936507937;0.920
634920634921,0.920634920634921,0.920634920634921;0.904761904761905,0.9047619
04761905,0.904761904761905;0.888888888888889,0.888888888888889,0.88888888888
8889;0.873015873015873,0.873015873015873,0.873015873015873;0.857142857142857
,0.857142857142857,0.857142857142857;0.841269841269841,0.841269841269841,0.8
41269841269841;0.825396825396825,0.825396825396825,0.825396825396825;0.80952
3809523810,0.809523809523810,0.809523809523810;0.793650793650794,0.793650793
```

```
650794,0.793650793650794;0.777777777777778,0.777777777777778,0.7777777777777
78;0.761904761904762,0.761904761904762,0.761904761904762;0.746031746031746,0
.746031746031746,0.746031746031746;0.730158730158730,0.730158730158730,0.730
158730158730;0.714285714285714,0.714285714285714,0.714285714285714;0.6984126
98412698,0.698412698412698,0.698412698412698;0.682539682539683,0.68253968253
9683,0.682539682539683;0.666666666666667,0.666666666666667,0.666666666666667
;0.650793650793651,0.650793650793651,0.650793650793651;0.634920634920635,0.6
34920634920635,0.634920634920635;0.619047619047619,0.619047619047619,0.61904
7619047619;0.603174603174603,0.603174603174603,0.603174603174603;0.587301587
301587,0.587301587301587,0.587301587301587;0.571428571428571,0.5714285714285
71,0.571428571428571;0.555555555555556,0.555555555555556,0.555555555555556;0
.539682539682540,0.539682539682540,0.539682539682540;0.523809523809524,0.523
809523809524,0.523809523809524;0.507936507936508,0.507936507936508,0.5079365
07936508;0.492063492063492,0.492063492063492,0.492063492063492;0.47619047619
0476,0.476190476190476,0.476190476190476;0.460317460317460,0.460317460317460
,0.460317460317460;0.444444444444444,0.444444444444444,0.444444444444444;0.4
28571428571429,0.428571428571429,0.428571428571429;0.412698412698413,0.41269
8412698413,0.412698412698413;0.396825396825397,0.396825396825397,0.396825396
825397;0.380952380952381,0.380952380952381,0.380952380952381;0.3650793650793
65,0.365079365079365,0.365079365079365;0.349206349206349,0.349206349206349,0
.349206349206349;0.333333333333333,0.333333333333333,0.333333333333333;0.317
460317460317,0.317460317460317,0.317460317460317;0.301587301587302,0.3015873
01587302,0.301587301587302;0.285714285714286,0.285714285714286,0.28571428571
4286;0.269841269841270,0.269841269841270,0.269841269841270;0.253968253968254
,0.253968253968254,0.253968253968254;0.238095238095238,0.238095238095238,0.2
38095238095238;0.222222222222222,0.222222222222222,0.222222222222222;0.20634
9206349206,0.206349206349206,0.206349206349206;0.190476190476190,0.190476190
476190,0.190476190476190;0.174603174603175,0.174603174603175,0.1746031746031
75;0.158730158730159,0.158730158730159,0.158730158730159;0.142857142857143,0
.142857142857143,0.142857142857143;0.126984126984127,0.126984126984127,0.126
984126984127;0.111111111111111,0.111111111111111,0.111111111111111;0.0952380
952380952,0.0952380952380952,0.0952380952380952;0.0793650793650794,0.0793650
793650794,0.0793650793650794;0.0634920634920635,0.0634920634920635,0.0634920
634920635;0.0476190476190476,0.0476190476190476,0.0476190476190476;0.0317460
317460317,0.0317460317460317,0.0317460317460317;0.0158730158730159,0.0158730
158730159,0.0158730158730159;0,0,0;];


[MorelandColormapA] = [0.0 59 76 192
0.03125 68 90 204
0.0625 77 104 215
0.09375 87 117 225
0.125 98 130 234
0.15625 108 142 241
0.1875 119 154 247
0.21875 130 165 251
0.25 141 176 254
0.28125 152 185 255
0.3125 163 194 255
0.34375 174 201 253
0.375 184 208 249
0.40625 194 213 244
0.4375 204 217 238
0.46875 213 219 230
0.5 221 221 221
0.53125 229 216 209
```

```
0.5625 236 211 197
0.59375 241 204 185
0.625 245 196 173
0.65625 247 187 160
0.6875 247 177 148
0.71875 247 166 135
0.75 244 154 123
0.78125 241 141 111
0.8125 236 127 99
0.84375 229 112 88
0.875 222 96 77
0.90625 213 80 66
0.9375 203 62 56
0.96875 192 40 47
1.0 180 4 38];

% This line is correct:
% MorelandColormapB = MorelandColormapA(:,2:4) ./
max(max(MorelandColormapA(:,2:4))); %make all 0 to 1
% It outputs the values below, which I've hardcoded in here to save
% computation time when plotting stuff.
MorelandColormapB =
[[0.231372549019608;0.266666666666667;0.301960784313725;0.341176470588235;0.
384313725490196;0.423529411764706;0.466666666666667;0.509803921568627;0.5529
41176470588;0.596078431372549;0.639215686274510;0.682352941176471;0.72156862
7450980;0.760784313725490;0.800000000000000;0.835294117647059;0.866666666666
667;0.898039215686275;0.925490196078431;0.945098039215686;0.960784313725490;
0.968627450980392;0.968627450980392;0.968627450980392;0.956862745098039;0.94
5098039215686;0.925490196078431;0.898039215686275;0.870588235294118;0.835294
117647059;0.796078431372549;0.752941176470588;0.705882352941177;]
[0.298039215686275;0.352941176470588;0.407843137254902;0.458823529411765;0.5
09803921568627;0.556862745098039;0.603921568627451;0.647058823529412;0.69019
6078431373;0.725490196078431;0.760784313725490;0.788235294117647;0.815686274
509804;0.835294117647059;0.850980392156863;0.858823529411765;0.8666666666666
67;0.847058823529412;0.827450980392157;0.800000000000000;0.768627450980392;0
.733333333333333;0.694117647058824;0.650980392156863;0.603921568627451;0.552
941176470588;0.498039215686275;0.439215686274510;0.376470588235294;0.3137254
90196078;0.243137254901961;0.156862745098039;0.0156862745098039;]
[0.752941176470588;0.800000000000000;0.843137254901961;0.882352941176471;0.9
17647058823529;0.945098039215686;0.968627450980392;0.984313725490196;0.99607
8431372549;1;1;0.992156862745098;0.976470588235294;0.956862745098039;0.93333
3333333333;0.901960784313726;0.866666666666667;0.819607843137255;0.772549019
607843;0.725490196078431;0.678431372549020;0.627450980392157;0.5803921568627
45;0.529411764705882;0.482352941176471;0.435294117647059;0.388235294117647;0
.345098039215686;0.301960784313725;0.258823529411765;0.219607843137255;0.184
313725490196;0.149019607843137;]];


% COLOUR MAP TO ACTUALLY USE:
% ------------------------------------------------------------------------

if strcmp(chooseCMAP,'reversegray') == 1
 map = reversegray;
elseif strcmp(chooseCMAP,'MorelandColormapB') == 1
 map = MorelandColormapB;
elseif strcmp(chooseCMAP,'gray') == 1
```

```
 map = gray;
else
 map = jet;
end




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

% Generates the image:
if aequal
 % IMAGESC way.
 H = imagesc(X,Y,U,varargin{:});
%  MorelandColormap = colormap(MorelandColormapB);
colormap(map)
%  if strcmp(map,'gray') == 1
%      colormap(gray);
%  elseif strcmp(map,'jet') == 1
%      colormap(jet);
%  end


else
 % PATCH way.

 % Check clim:
 if (rem(Nargin,2)==1)
  clim          = varargin{end};
  varargin(end) = [];
  if ((length(clim)~=2) || (clim(1)>clim(2)))
   error('CVARGAS:imagescnan:incorrectClimInput',...
    'clim must be a 2 element increasing vector.')
  end
 else
  clim = [];
 end

 % Generates vertices between coordinates (coordinates may not be at the
 % center of these vertices):
 if (length(X)~=N)
  X  = (0:N-1)*((X(2)-X(1))/(N-1)) + X(1);
 end
 if (length(Y)~=M)
  Y  = (0:M-1)*((Y(2)-Y(1))/(M-1)) + Y(1);
 end
 if isempty(dX)
  dX = diff(X);
 end
 if isempty(dY)
  dY = diff(Y);
 end
 [X,Y] = meshgrid([X(1)-dX(1)/2 X+dX([1:N-1 N-1])/2],...
```

```
                    [Y(1)-dY(1)/2 Y+dY([1:M-1 M-1])/2]);

 % Generates faces:
 ind               = (1:(M+1)*N)';
 ind(M+1:M+1:end) = [];

 % Generates patches:
 H = patch(...
  'Vertices'       ,[X(:) Y(:)],...
  'Faces'          ,[ind ind+1 ind+M+2 ind+M+1],...
  'FaceVertexCData',U(:),...
  'FaceColor'      ,'flat',...
  'EdgeColor'      ,'none',... % NOTE: Sometimes this is not required.
  varargin{:});
 set(ha,...
  'YDir' ,'reverse',...
  'View' ,[0 90],...
  'Box'  ,'on',...
  'Layer','top')
 axis(ha,'tight')

 % Sets clim:
 if ~isempty(clim)
  set(ha,'CLim',clim)
 else
  set(ha,'CLimMode','auto')
 end

colormap(map)
end

% Adds NaNs patches:
if any(MNAN(:))
 if aequal
  % dX and dY is constant:
  [MNAN,NNAN] = ind2sub([M,N],find(MNAN));
  Nnan        = length(MNAN);
  dX   = (X(2)-X(1))/(N-1)/2;
  dY   = (Y(2)-Y(1))/(M-1)/2;
  HNAN = patch(repmat((X(1)+(NNAN(:)'-1)*(2*dX)),4,1) + ...
                                    (repmat([-1 1 1 -1]'*dX,1,Nnan)),...
               repmat((Y(1)+(MNAN(:)'-1)*(2*dY)),4,1) + ...
                                    (repmat([1 1 -1 -1]'*dY,1,Nnan)),...
               CNAN,...
               'EdgeColor',CNAN,... 'EdgeColor','none',...
               varargin{1:Nargin-rem(Nargin,2)});
 else
  % dX and/or dY is not constant:
  MNAN = find(MNAN);
  HNAN = patch(...
  'Vertices'       ,[X(:) Y(:)],...
  'Faces'          ,[ind(MNAN) ind(MNAN)+1 ind(MNAN)+M+2 ind(MNAN)+M+1],...
  'FaceColor'      ,CNAN,...
  'EdgeColor'      ,'none',... 'EdgeColor',CNAN,... % NOTE: may be better?
  varargin{:});
 end
```

```
else
 HNAN = [];
end


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

% Clears outputs?:
if (nargout==0)
 clear H
end

% [EOF]   imagescnan.m
```

B.8 setReadInt

```
function newfile=setReadInt(site,alt,beamdir,start,q,stop,pass)
%{
SETREADINT    Sets days as a vector, reads the data, and interpolates.
    SETREADINT(SITE,ALT,BEAMDIR,START,Q) saves the interpolated data to a
.mat file to be loaded by a spectral algorithm.  All data is in 15 minute
intervals.  The site, altitude, and beam direction are given by the
character strings SITE, ALT, and BEAMDIR respectively.  ALT may be a
number.  The data spans the yyyymmdd day START.  Q is the YYYYMM format in
a string.

    SETREADINT(...,STOP) saves the interpolated data to a .mat file for
days between START and STOP inclusively.

    SETREADINT(...,PASS) allows user to pass on clicking the initial and
values to interpolate over (0), otherwise, the computer chooses the first,
longest string of data.  Default is set to 1.

    NEWFILE=SETREADINT(...) returns the name of the .mat file the data is
stored in.
%}

if nargin==5
    % Only use one day
    stop=start;
    pass=1;
elseif nargin==6
    pass=1;
elseif nargin~=7
    error('SRI:argChk','Wrong number of input arguments.')
end
if isa(alt,'char')==0
    alt=num2str(alt,'%2.1f');
end

% Set the maximum number of missing data points.
miss=20;
%************************************************************************%
%           SET DAYS AS A VECTOR FROM START TO STOP
```

```
%************************************************************************%

% Set a matrix for the days of the month for leap and non-leap years
n=[31,28,31,30,31,30,31,31,30,31,30,31];
% First row is for leap years, second row is for non-leap years, columns
% are representative days in the month.
numofdays=[n;n];
numofdays(1,2)=29;
clear('n')

% Break start and stop into year, month, day
yyyy=floor(start/10000);
mm=floor((start-yyyy*10000)/100);
dd=mod(start,100);
stopy=floor(stop/10000);
stopm=floor((stop-stopy*10000)/100);
stopd=mod(stop,100);

% See if start and stop are the same month.  If so, just gather days
% between.
if yyyy==stopy && mm==stopm
    days=yyyy*10000+mm*100+(dd:stopd);
else
    % Spans multiple months.
    % Determine if it is a leap year (use row 1) or not (use row 2).
    leap=ceil(mod(yyyy,4)/4)+1;
    % Put the remaining days of the current month into days.
    days=yyyy*10000+mm*100+(dd:numofdays(leap,mm));

    % Determine days between the start and stop months:
    while yyyy~=stopy && (mm+1)~=stopm
        mm=mm+1;
        if mm==13
            % If we reach "month 13", roll the clock over a year.
            yyyy=yyyy+1;
            % Determine if it is a leap year:
            leap=ceil(mod(yyyy,4)/4)+1;
            % Reset mm to zero since we add 1 to it before doing
            % anthing else (otherwise we would skip January every roll
            % over).
            mm=0;
        else
            % Include the days of this month in days.
            days=[days,yyyy*10000+mm*100+(1:numofdays(leap,mm))];
        end
    end

    % Include the first days of the stop month.
    days=[days, yyyy*10000+(mm+1)*100+(1:stopd)];
end
clear('numofdays','yyyy','mm','dd','stopy','stopm','stopd','leap')

%************************************************************************%
%           READ AND SORT DATA
%************************************************************************%
```

```
% Create filepath
fileend=['.',site,'.',alt,beamdir,'_15min.txt'];

vel=[];
npoints=[];
t=[];
for i=1:length(days)
    % Read each file and put into variables
    day=int2str(days(i));
    mon=day(1:6);
    filepath=[site,'/',mon,'/',day,'/RadarData/',day,fileend];
    raw=dlmread(filepath,',',0,1);

    % Convert time into seconds
    time=raw(:,6)+60*(raw(:,5)+60*raw(:,4));
    % Sort data chronologically
    [time,order]=sort(time);
    vel=[vel;raw(order,8)];
    npoints=[npoints;raw(order,9)];
    % Convert add the appropriate number of days (in seconds) to time and
    % combine it with our t vector
    t=[t;time+60*60*24*(i-1)];
end
clear('i','day','filepath','raw','time','order')

% CHECK: data is equally spaced
dt=zeros(1,length(t)-1);
for i=2:length(t)
    dt(i-1)=t(i)-t(i-1);
end
if max(dt)~=min(dt)
    warning('FFT:dtChk','dt not equal!')
end
clear('dt','i')

% Keep a copy of the raw data if necessary to look at
velraw=vel;

%*************************************************************************%
%            INTERPOLATE DATA
%*************************************************************************%

% Find missing values in radar data.  Plot real values.
len=length(vel);
data=ones(len,1);   % Keeps track of which data points are real.

if pass~=0
    i1=1;
    lold=0;
    i=1;
    while i<=length(vel)
        if vel(i)==-999.0;
            % Check to see if next miss are also -999.0 (no data)
            if (i+miss-1)<=length(vel)
```

```
                x=vel(i:(i+miss-1));
            else
                x=vel(i:length(vel));
            end
            if x==(-999.0*ones(size(x)))
                lnew=i1:(i-1);
                % Set the longest strand to the new one
                if length(lnew)>length(lold)
                    lold=lnew;
                end
                i=i+miss;
                % Check if there are any subsequent missing data points
                while i<length(vel) && vel(i)==-999.0
                    i=i+1;
                end
                i1=i;
            end
        end
        i=i+1;
    end
    if lold==0;
        lold=1:length(vel);
    end
    vel=vel(lold);
    data=data(lold);
    t=t(lold);
    npoints=npoints(lold);
    len=length(vel);
    clear('lold','lnew','i','i1','x')
end

f1=figure;
m=int2str(floor(start/100));
text=[alt,beamdir,' ',m,];
title(text)
xlabel('Time (s)')
ylabel('Radial velocity (m/s)')
hold on
for i=1:len
    if vel(i)==-999
        vel(i)=0;
        data(i)=0;
        plot(t(i), 0, 'r^')
    else
        plot(t(i), vel(i), 'bo')
    end
end
hold off
clear('i','text')

% Select where to start/stop in the data here.
if pass==0;
    figure(f1)
    msg=char('On the figure, click where you want interpolation to',...
        'start and stop respectively.');
    h=msgbox(msg,'Select start and stop','help');
```

```
    [ts,~]=ginput(2);
    close(h)

    % Find starting index
    diff1=abs(t-ts(1));
    % Find where difference is minimum and choose as starting point
    [~,i1]=min(diff1);

    % Find ending index
    diff2=abs(t-ts(2));
    [~,i2]=min(diff2);

    % Set all data vectors to just data between indices i1 and i2
    t=t(i1:i2);
    vel=vel(i1:i2);
    data=data(i1:i2);
    len=length(vel);

    clear('h','i1','i2','istart','istop','msg','ts')
end

% If the first point selected needs to be interpolated, drop it instead.
while data(1)==0;
    t=t(2:len);
    vel=vel(2:len);
    npoints=npoints(2:len);
    data=data(2:len);
    len=len-1;
end

% If the last data point selected needs to be interpolated, drop it
% instead.
last=len;
rem=0; % Keeps track of how many data points at the end are removed.
while data(last)==0;
    last=last-1;
end
t=t(1:last);
vel=vel(1:last);
npoints=npoints(1:last);
data=data(1:last);
len=last;
clear('last','rem')
hold on
plot(t(vel~=0),vel(vel~=0),'g-');
hold off
% Shift the data so that the first data point occurs at t=0 (if necessary).
% Assumed that data is in chronological order.
if t(1)~=0;
    t=t-t(1);
end

% Go through missing data and interpolate:
skip=0;
% Set the number of points to use before and after for interpolation.
```

```
group=5;

% New method, don't use null
for i=1:length(data)
    if skip~=0
        % This point has already been interpolated.
        skip=skip-1;
    elseif data(i)==0
        % Need to interpolate
        j=i;
        while data(j+1)==0
            j=j+1;
        end
        % Need to interpolate points on the span [i,j]
        if i==j || i+1==j
            % 1 or 2 data point missing.  Interpolate linearly between two
            % closest.
            m=(vel(j+1)-vel(i-1))/(t(j+1)-t(i-1));
            b=vel(i-1)-m*t(i-1);
            vel(i:j)=m*t(i:j)+b;
            skip=j-i;
        elseif j-i+1>miss
            % Too many data points missing
            t=t(1:i-1);
            vel=vel(1:i-1);
            npoints=npoints(1:i-1);
            data=data(1:i-1);
            len=length(vel);
            fprintf('Too many missing data points.  Ended at point
%g.\n\n',i-1)
            break
        else
            % Can interpolate the number of data points with interpol
            % Check there are enough points before:
            if i-group>0
                % Then we have enough data points prior to the gap to
                % interpolate using GROUP points at the beginning.
                group1=group;
            else
                % We do not have enough points before the interpolation.
                % Use the ones available.
                group1=i-1;
            end
            % Check there are enough points after:
            if j+group<=length(data)
                % There are enough points after the gap to use GROUP
                % points.
                group2=group;
            else
                % There are not enough points after, use those available.
                group2=length(data)-j;
            end
            % Interpolate:
            vel(i:j)=interpol(vel(i-group1:j+group2),...
                t(i-group1:j+group2),data(i-group1:j+group2),group1,...
                group2);
            skip=j-i;
```

```
        end
    end
end

hold on
plot(t,vel,'m-');
hold off
clear('skip','group','miss','group1','group2','i','j','m','b','days',...
    'diff1','diff2','mon')

% Save variables to a .mat file so that I can use them in forming the
% spectra and only need to interpolate once for multiple windows.
start=num2str(start);
stop=num2str(stop);
if pass==0
    newfile=['FFT/',site,'/',start,'to',stop,'mod',...
        fileend(1:length(fileend)-3),'mat'];
else
    newfile=['FFT/',site,'/',q,'/',q,fileend(1:length(fileend)-3),'mat'];
end
clear('start','stop')
fprintf('%s\r\n',newfile)
save(newfile)
end
```

B.9 winSel

```
function win=winSel(N,type)
%{
WINSEL    Creates specified window.
    WINSEL(N) is the specified discrete window with N elements.

    WINSEL(...,TYPE) specifies the type of window without opening a
dialogue box where TYPE is an integer:
        0 -- cancel
        1 -- Hamming
        2 -- Hann (Hanning)
        3 -- boxcar with 10% cosine taper
        4 -- boxcar
%}

if nargin==1
    % Need to choose window type
    choices={'Hamming','Hann','Welch','Butterworth',...
        'Box car with 10% cosine taper','Box car'};
    [type,ok]=listdlg('SelectionMode','single', 'Name','Window type',...
        'ListSize',[175 300],'PromptString','Choose a window to use:',...
        'ListString',choices);
    % If the user pressed "cancel", pass value of zero.
    if ok==0
        type=0;
    end
elseif nargin==2
    % Check that type is an integer
    if isa(type,'numeric')==0
        error('winsel:argChk','TYPE is an integer.')
```

```
    end
else
    error('winsel:argChk','Wrong number of input arguments.')
end

% If the user pressed "cancel", pass value of 0.
if type==0
    win=zeros(N,1);
elseif type==1
    % Hamming window
    n=(0:N-1)';
    A=1/(25/46*N);
    win=A*(25/46-21/46*cos(2*pi*n/(N-1)));
elseif type==2
    % Hann window
    n=(0:N-1)';
    A=2/N;
    win=A*(1/2-1/2*cos(2*pi*n/(N-1)));
elseif type==3
    % Box car with 10% cosine taper
    % Create cosine taper shape for the first 10% and last 10% of the data.
    n=(0:N-1)';
    tenperc=floor(N/10);
    win=ones(N,1);
    win(n<=tenperc)=(1-cos(10*pi/N*n(n<=tenperc)))/2;
    win(n>=(N-tenperc))=(1-cos(10*pi/N*n(n>=(N-tenperc))))/2;
    win=win*10/(9*N);
elseif type==4
    % Box car
    win=ones(N,1)/N;
else
    error('winsel:argChk','TYPE is between 0 and 4 inclusive.')
end
end
```

Curriculum Vitae

Name:	Melanie Wright
Post-secondary Education and Degrees:	The University of Western Ontario London, Ontario, Canada 2007-2011 B.Sc.
	The University of Western Ontario London, Ontario, Canada 2013-2014 B.Ed.
	The University of Western Ontario London, Ontario, Canada 2011-2014 M.Sc.
Honours and Awards:	Great Ideas in Teaching Award 2013
	Northern Scientific Training Program (Award) 2011-2012
	TA Award of Excellence for Tutoring in Physics and Astronomy 2012
	Dean's Honor List 2007-2011
	Natural Sciences and Engineering Research Council (NSERC) Undergraduate Student Research Award (USRA) 2010
	The Western Scholarship of Excellence 2007
Related Work Experience	Teaching Assistant The University of Western Ontario 2011-2013