Multi-frequency Clutter-rejection Algorithms for Acoustic Radars

Thesis · June 2007

DOI: 10.13140/RG.2.1.4757.7449

Mathew Legg

Massey University, Albany, Auckland, New Zealand

The Department of Physics
The University of Auckland
New Zealand

Multi-frequency Clutter-rejection Algorithms for Acoustic Radars

Mathew Legg

June 2007

Supervisor: Professor Bradley

A thesis submitted in fulfillment of the requirements of Master of Physics

Abstract

This work investigates the removal of rain clutter from multi-frequency acoustic sounding. A computer program was written to simulate the backscattering from turbulence and rain from multi-frequency acoustic sounding. The strength of backscattering from turbulence has been shown to have a $\lambda^{-1/3}$ wavelength dependence while that from rain a λ^{-4} dependence. Based on this differential wavelength dependence, an algorithm was developed to fit the turbulence and rain components of simulated backscattered signal.

A multi-frequency sodar was developed to trial the algorithm. Since the algorithm relies on the differential backscattering strength on frequency, it was essential that all factors that might cause the sodar's gain to vary with frequency be allowed for. Measurements were made of the characteristics of the speakers, which were used in the sodar. The variation of gain with frequency and angle was obtained for the individual speakers and the speakers in an array. The sodar's variation of gain with angle was also studied. A transmit and receive gain calibration for the sodar was obtained for a range of frequencies from 2 to 11 kHz . This was included into the simulation and fitting algorithm. The optimal transmit frequencies were then obtained. A GUI program was written to run the sodar and a transmit/receive switch was designed and made. The sodar was tested in clear air and rain conditions and requirements for a pre-amplifier and bandpass filters established. A circuit board was designed and built containing a preamplifier and a wide bandwidth bandpass filter. Field tests were carried in clear air and in rain conditions and noise analysis carried out. Recommendation for further development and field testing are discussed.

Acknowledgements

I would like to thank my supervisor Professor Stuart Bradley for his help, encouragement, and understanding throughout this study and for his assistance in the designed of the sodar's circuit boards. I would also like thank everyone in the Physic Department workshop especially Mat Hogg and Steve Warrington, for their help and for the construction of the sodar case and turntable. Thanks also to Bryan Davis of the Physic Department electronics workshop for his help in the electronic component of this work and assistance in the design of the trasmit/receive switch of the sodar. I would also like to thank Professor Gary Bold for his help and encouragement throughout my time at university and also Professor Geoff Austin for his help. Finally, thanks to my mother for her encouragement and help.

Contents

List of Figures

List of Key Symbols

α	Atmospheric absorption coefficient [m^{-1}],[dB /m]
Δr	Height resolution [m]
θ	Sodar half beam width [rad]
λ	Transmit wavelength [m]
Λ	Marshal and Palmer drop size distribution exponential parameter [m^{-1}]
μ	Gamma drop size distribution power [m^{-1}]
σ	Total scattering cross section per unit volume of air per steradian [$\mathrm{m}^2\,\mathrm{m}^{-3}$]
σ_f	Gaussian width parameter [Hz]
σ_R	Rain scattering cross section per unit volume of air per steradian [$\mathrm{m}^2\,\mathrm{m}^{-3}$]
σ_T	Turbulence scattering cross section per unit volume of air per steradian [$\mathrm{m}^2\,\mathrm{m}^{-3}$]
τ	Transmitted pulse duration [s]

ϕ	Angle between original direction of propagation and direction of scattering [rad]
φ	Sodar beam tilt angle [rad]
A_e	Receiving effective area of the sodar [dB]
c	Speed of sound [m/s]
C_T	Thermal structure parameter [$\mathrm{Km}^{-2/3}$]
C_V	Velocity structure parameter [$\mathrm{m}^{4/3}\ \ \mathrm{s}^{-2}$]
D	Drop diameter [m]
f	Doppler shifted frequency [Hz]
F_s	Sampling frequency [Hz]
f_T	Transmission frequency [Hz]
G	Transmission antenna gain of the sodar [dB]
I	Rainfall Intensity [mm/h]
k	Wave number [m^{-1}]
N	Noise component of the backscattered spectral power [W]
$n(D)$	Rain drop size distribution [m^{-4}]
$N_{f\,b}$	Number of Doppler shift frequency bins
N_o	Coefficient of Marshal and Palmer drop size distribution [m^{-4}]
P	Total backscatter spectral power [W]
p_a	Atmospheric pressure [Pa]

P_T	Power transmitted by the sodar [W]
r	Range from sodar [m]
R	Rain component of the backscattered spectral power [W]
r_h	Relative humidity [%]
t	Time [s]
T	Turbulence component of the backscatter spectral power [W]
T_a	Temperature [K]
V	Scattering volume of a transmitted pulse [m^3]
V_T	Drop terminal velocity [m/s]
w	Velocity [m/s]
x	Inverse of the transmit wavelength λ^{-1} [m^{-1}]

Introduction

The acoustic RADAR or SODAR (SOund Detection And Ranging) has increasingly been used as a tool for studies of the atmospheric boundary layer. One application has been the measurement of wind and turbulence profiles for wind energy applications. A problem encountered when using sodars to measure wind and turbulence has been the corruption of data due to clutter in the form of backscattering from precipitation. Rain clutter is one of the main forms of noise in sodar measurements of wind and can cause significant wind measurement errors. Overestimates of the measured wind speed by as much as 30 m/s are possible (Bradley et al., 2005; Crescenti, 1997).

This issue has been addressed by others by attempting to detect when precipitation has occurred and deleting the measured data during this period. A problem with this technique is that it can lead to biasing of the data. For example, if rain tends to occur during periods of higher wind speeds, deleting all wind data during rain events could lead to biasing of the mean wind speed towards lower values. In addition, there are also issues surrounding the efficiency of detecting precipitation events. Problems such as these have led to the desire to develop methods of being able to make accurate sodar wind measurements during precipitation (de Noord et al., 2005; Bradley et al., 2005; Peters et al., 1998).

This work investigates the removal of rain clutter from sodar data by using multiple transmission frequencies to separate the turbulence component of the backscattered power spectra from that of rain. This method, proposed by Little (1972), uses the fact that the echo strength from turbulence has a $\lambda^{-1/3}$ dependence while that from rain has a λ^{-4} dependance. It is hoped this may enable wind measurements to be made in rain, or alternatively to make rain measurements in windy or turbulent conditions. To our knowledge, this method has not been investigated for sodars, though it has been used for electromagnetic radar studies (Sangren et al., 1984; Rajopadhyaya et al., 1998; Gage et al., 1999).

Chapter 1 gives a brief description of the theory of backscattering and Doppler shifting from turbulence and rain and how it relates to the Acoustic Radar Equation. In Chapter

2, a computer simulation of the backscattered power spectra from turbulence and rain is outlined. The algorithm developed to fit the simulated echoes is also discussed. A outline of the sodar's instrumentation is provided in Chapter 3. Chapter 4 gives the measurements made to enable the sodar to be calibrated. Measured characteristics of the sodar and its speaker array and speakers are also included. Field trials and the results obtained are discussed in Chapter 5. Recommendations for further research and field trials are given in Chapter 6.

1

Theory

1.1 Literature Review

1.1.1 Literature - Acoustic Sounding of the Atmosphere

The use of echoes from acoustic sounding is one of the oldest remote sensing techniques. Tyndall (1875) used a foghorn to investigate backscattering of sound due to temperature and wind structure in the atmosphere. Acoustic backscattering was used by Gilman et al. (1946) to investigate how temperature inversions affected microwave communication links. Kallistratova (1959a,b) and Kallistratova and Tatarski (1960) studied acoustic scattering from the atmosphere. In the late sixties, both McAllister and Little separately developed the theory of acoustic radars. McAllister (1968) showed that backscattering could be measured to heights of several hundred meters. Acoustic radars have since become an accepted tool in studies of the lower atmosphere and have been used for a range of purposes, including measuring wind and temperature profiles and for measurements of precipitation properties.

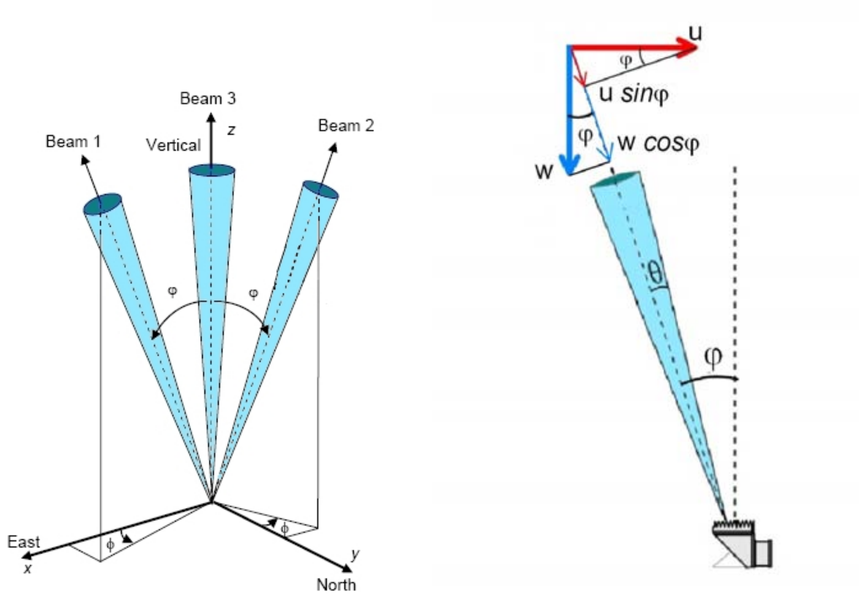
1.1.2 Literature - Acoustic Sounding of Precipitation

Acoustic radars have been used to measure backscattering from precipitation. Kelton and Bricout (1964) obtained anomalous wind velocities during heavy rain when studying wind velocities using Doppler scattering of a 10 kHz sound source. It was concluded that this was the result of backscattering from rain. Little (1972) investigated the theory for Rayleigh scattering from fog, cloud, rain, and snow and proposed several methods of using acoustic radars to deduce rainfall profile properties, including the use of multi-frequency transmission. Melling and List (1978) measured acoustic echoes scattered from snow flakes. Weill (1983) and Weill et al. (1986) used a 6 kHz minisodar to measure simultaneously the acoustic backscattering from rain and turbulence. Coulter et al. (1989) used a vertical-pointing minisodar as a rain radar to obtain rain rates which agreed closely with rain gauge measurements. A high frequency precipitation sodar was developed, calibrated, and used to obtain raindrop size distributions by Bradley et al. (1994), Bradley (1997), Webb (2000), and Bradley and Webb (2002). Sodar measurements of precipitation and methods of deducing drop size distributions were also investigated by Grassl et al. (2000), Pang et al. (2002), and Pang and Grassl (2005).

1.2 Sodar Wind Speed Measurements and Errors due to Rain

Figure 1.1: Photo which includes sodars used in wind energy studies by Bradley et al. (2005)

Wind speed profiles can be obtained by a sodar by emitting pulses of sound in three or more directions (see Fig. 1.2a). As the sound propagates away from the sodar, some of the sound is continuously being backscattered by refractive index variations due to turbulence. If there is a component of wind in the direction of the sodar beam, the echoes



(a) Sodar three beam configuration for wind measurements (Bradley et al., 2005)

(b) Wind u and rain w velocity vectors and the components in the direction of the sodar beam

Figure 1.2: Diagrams illustrating sodar wind measurement beam configurations and the wind and rain velocity components measured by the sodar

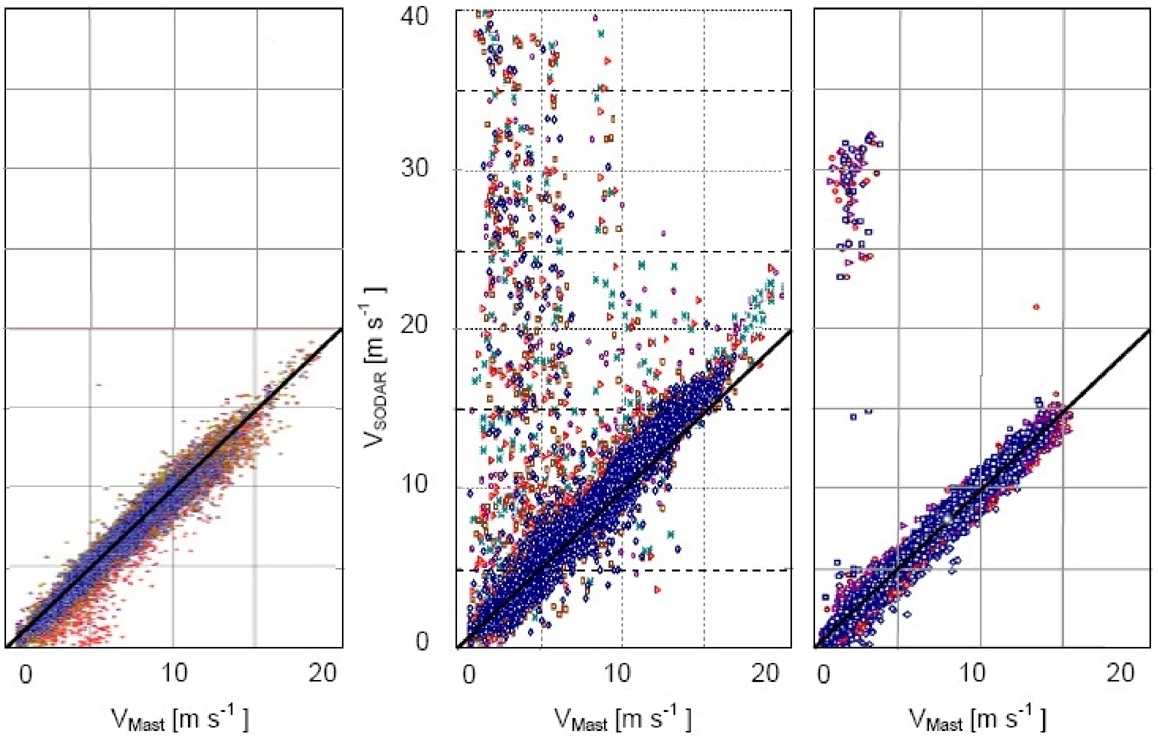
will be Doppler shifted by $f = 2\,v\,f_T/c$, where v is the component of the wind speed in the direction of the sodar beam, f_T is the transmission frequency, and c is the speed of sound. This Doppler shift is used to obtain the component of the wind velocity in the direction of the sodar beam. The wind speed and direction are obtained by combining the wind speed components from all beam directions. The height from which a wind speed measurement is made is calculated by $r = \Delta t\, c/2$, where Δt is the time delay between transmitted pulse and received signal and c is the speed of sound.

Significant errors in these wind speed measurements can be caused by rain. Consider rain falling vertically at velocity w. The component of w in the direction of a sodar beam, which is at an angle of φ to the vertical, will be $w \cos\varphi$ (see Fig. 1.2b). The resulting Doppler shift measured by the sodar will be identical to a horizontal wind with a wind speed component in the direction of the sodar beam of $\hat{u}\,\sin\varphi$. This means that rain can lead to an error in the estimated wind speed of

$$\hat{u} = \frac{w\,\cos\varphi}{\sin\varphi}. \tag{1.1}$$

Since the tilt angle of the sodar beam is relatively small, this wind speed error due to rain may be approximated as

$$\hat{u} \approx \frac{w}{\varphi}. \tag{1.2}$$



(a) Metek PCS2000-64 sodar operating at 1674 Hz (b) AeroVironment 4000 sodar operating at 4500 Hz (c) Scintec SFAS operating at 2540-4850 Hz

Figure 1.3: Mast wind speeds verses raw sodar wind speeds (Bradley et al., 2005)

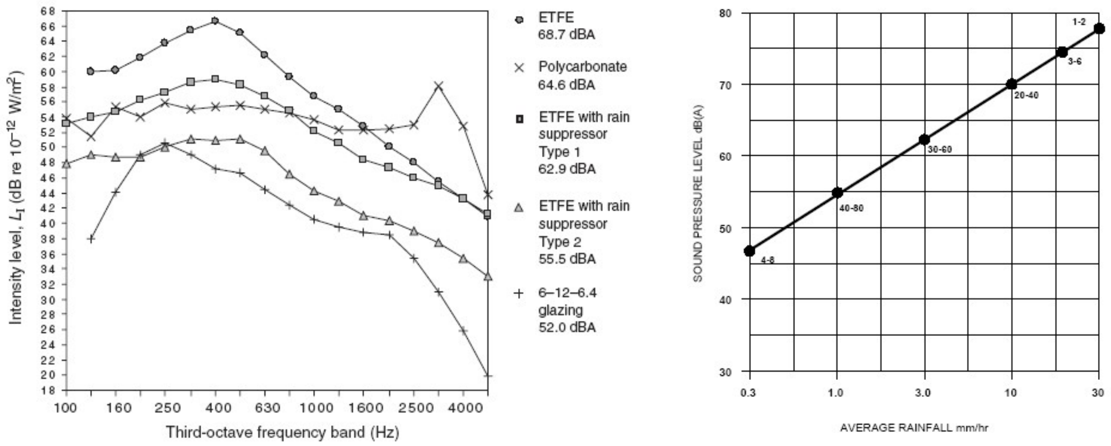
For the case of $\varphi = 17^o$ and a maximum rain fall speed of about 9 m/s, the error in estimated wind speed due to rain would be up to about 30 m/s. Echoes from rain are, therefore, a significant cause of error in wind measurements. In this work, an algorithm is developed which uses multi-frequency acoustic sounding to attempt to reduce these errors in wind speed measurements due to rain.

1.3 Why use Multi-frequency Acoustic Sounding?

1.3.1 Low Frequency Acoustic Wind Profiling - Advantages and Disadvantages

Acoustic radar wind profilers generally operate at relatively low frequencies, about 2 to 4 kHz. The main advantage of using these low frequencies is that atmospheric attenuation is relatively small for these frequencies. This means that measurements may be made to distances of up to several hundred meters, as compared to a few tens of meters for higher frequencies. Another factor making lower frequencies desirable is the fact that the strength of echoes from rain has a λ^{-4} dependence, meaning that the backscattering from rain is significantly less for lower frequencies. This is illustrated in a study by Bradley et al. (2005) where errors in wind speed measurements, due to echoes from rain, were observed for several commercial acoustic wind profilers (see Fig. 1.3). The wind speed errors due to rain for the Metek sodar operating at 1674Hz showed significantly less errors in wind speed the the other higher frequency sodars. Echoes from rain are, however, still a significant contributor of noise for acoustic wind profilers and can lead to significant errors in the measured wind speed.

The large size of low frequency sodars, compared to high frequency sodars, is a main reason why the use of higher transmission frequencies are often preferred. Another disadvantage of lower frequency acoustic radars is the fact that ambient noise is significantly greater at lower frequencies. An example of this is the noise of rain drops impacting on the acoustic radar. This is illustrated in a study of rain impact noise on roofing by Hopkins (2006), which shows increases in the noise level with decreased frequency (see Fig. 1.4a). This rain impact noise increases with rain intensity (see Fig. 1.4b).



(a) Measured sound intensity levels for the different roof elements as a function of frequency (Hopkins, 2006)

(b) Rain noise on a building roof as a function of rain fall intensity (Renzo Tonin & Associates PTY LTD, 2007)

Figure 1.4: Graphs of the sound intensity due to rain impacting on roofing.

1.3.2 High Frequency Acoustic Wind Profiling - Advantages and Disadvantages

The use of high frequency has been proposed as a solution to the problem associated with low frequency acoustic soundings. High frequency sodars have been shown to have the potential to allow both the turbulence and rain signal to be observed simultaneously as separate peaks of the power spectra (Coulter et al., 1989). Pang and Grassl (2005) describe a method of using high transmission frequencies, 6 to 20 kHz to obtain rain properties

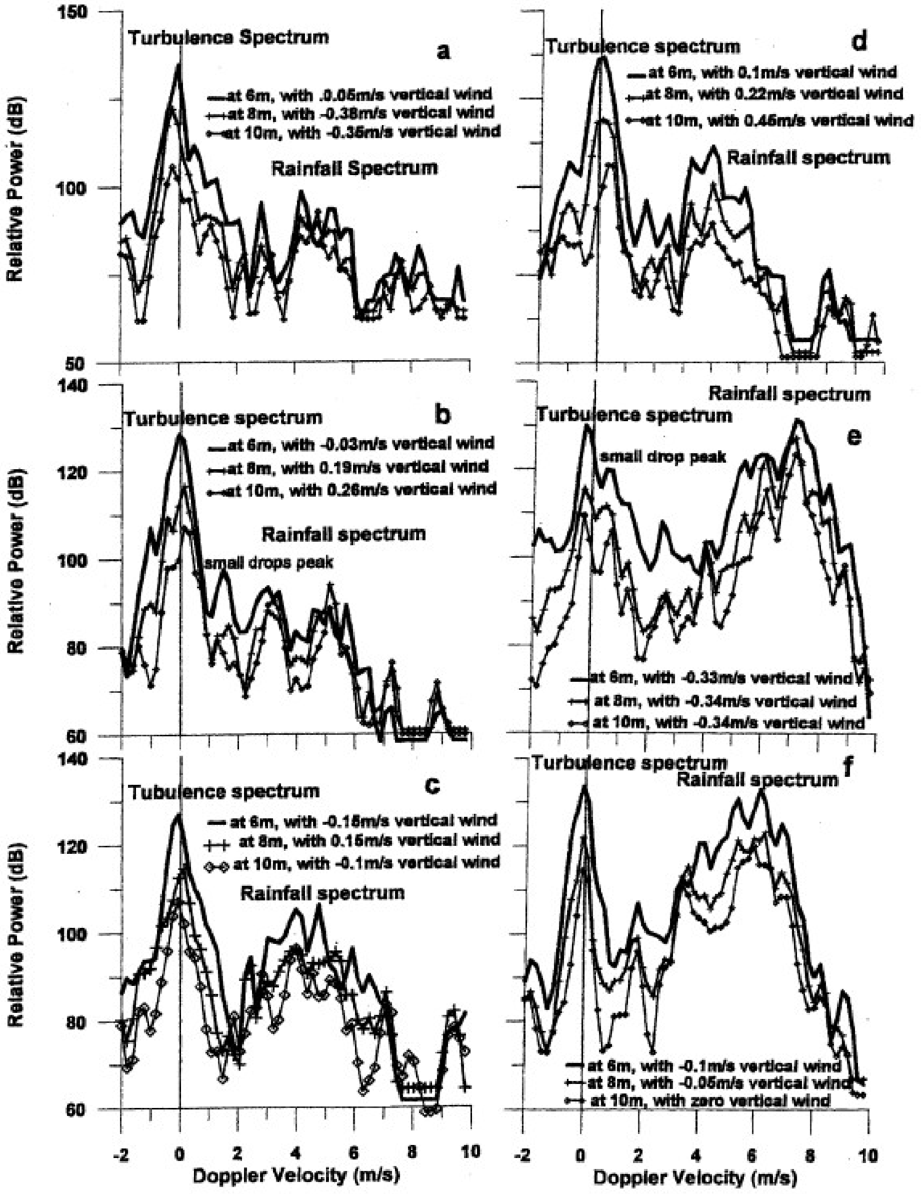


FIG. 1. Simultaneously measured rainfall and turbulence Doppler power spectra averaged over 1 min for 31 Jan 2004. (a) At 1331 UTC, rain rate $R = 1.03$ mm h^{-1}; (b) at 1512 UTC, $R = 1.52$ mm h^{-1}; (c) at 1337 UTC, $R = 4.31$ mm h^{-1}; (d) at 1519 UTC, $R = 5.34$ mm h^{-1}; (e) at 1537, $R = 44.58$ mm h^{-1}; and (f) at 1959 UTC, $R = 50.14$ mm h^{-1}.

Figure 1.5: Examples of high frequency acoustic radar measurements of rain and turbulence spectra taken from Pang and Grassl (2005)

by separating the rain and turbulence components of the backscattered power spectra. The atmospheric attenuation of sound, however, increases strongly with transmission frequency, meaning that these high frequency sodars are limited to making measurements to relatively low heights. This can be a significant problem in wind profile measurements where ranges of several hundred meters may be desired. This would appear to be a main reason why manufactures have shown a reluctance to produce high frequency sodars for wind profiling.

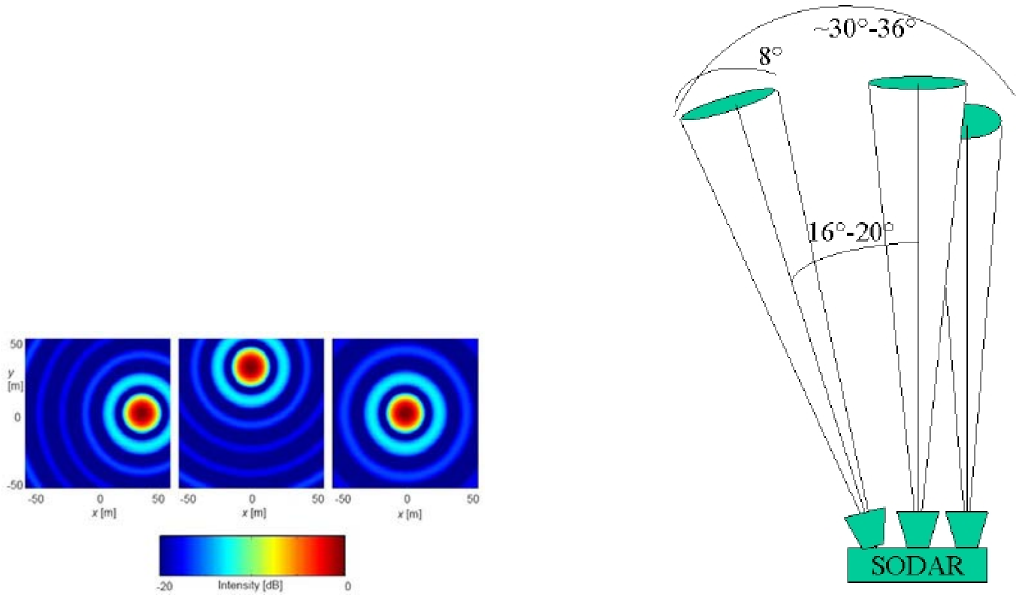
1.3.3 Multi-frequency Acoustic Wind Profiling - Advantages and Disadvantages

Multi-frequency acoustic sounding has the potential of enabling the rain and turbulence components of the backscattered power spectra to be separated, using turbulence and rain's respective backscattering strength dependence on frequency. This would have the advantage of allowing the echoes from rain to be rejected from wind measurements, while still allowing relatively low frequencies to be used. This it is hoped would avoid the strong atmospheric attenuation associated with the use of high frequency transmission. Another potential advantage is the ability to reduce the high ambient noise levels associated with the use of low frequencies. A problem expected to be associated with the use of multi-frequency sounding is the need to allow for frequency-dependent factors associated with acoustic sounding. These would include atmospheric attenuation, the transmission and receiving gain of the acoustic radar, and even perhaps the scattering volume of the pulses emitted by the acoustic radar.

1.4 Sodar Operating Principles

1.4.1 Comparison of the use of Sound verses Electromagnetic Waves for Remote Sensing of the Atmosphere

Refractive index variations of air are about 1000 times stronger for acoustic waves than for electromagnetic waves. While ice and water precipitation particles have the same acoustic refractive index at audio frequencies, they have different refractive indices for electromagnetic radiation. This makes measurement of mixed phase precipitation easier for acoustic waves. Other advantages of using sound over electromagnetic waves are the comparatively low attenuation of acoustic waves due to water, the ability to measure vertical air motion, and relative low cost of equipment. Disadvantages of using sound,



(a) Acoustic interference pattern from a speaker array (Bradley et al., 2005).

(b) Example of a three beam sodar (Antoniou et al., 2003).

Figure 1.6: Typical sodar beam configuration

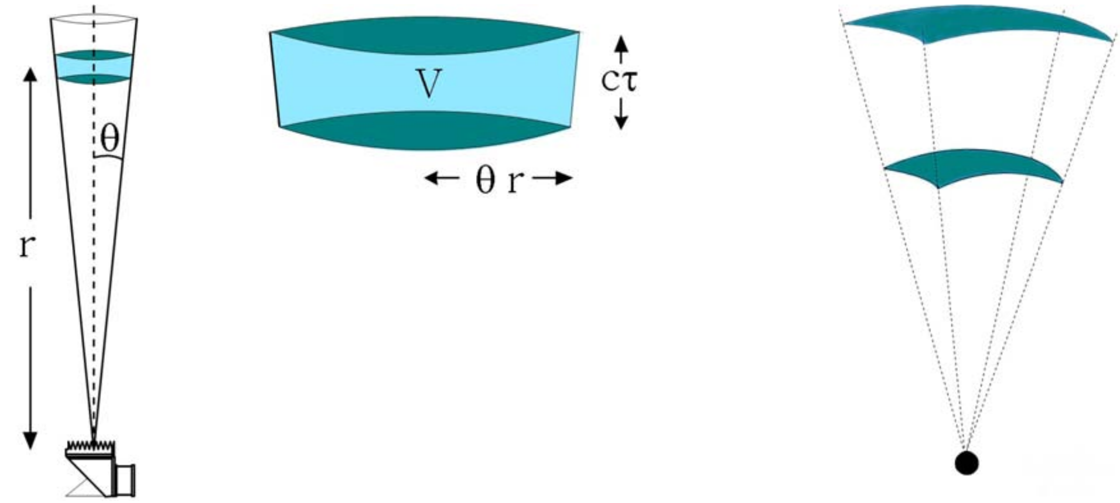
as apposed to using electromagnetic waves, include the high acoustic attenuation of air, which increases strongly with frequency, and sensitivity to ambient noise and vibrations (Little, 1969, 1972; Webb, 2000; Pang and Grassl, 2005).

1.4.2 Basic Operating Process

There are two forms of acoustic radars or sodars, monostatic and bistatic. A bistatic sodar uses continuous wave transmission, with transmitter and receiver at different locations. A monostatic sodar uses pulse transmission and has both the transmitter and receiver at the same location. This work uses a monostatic sodar configuration.

Sodars generally are composed of an array of speakers or a horn. The sound from the speakers cause an interference pattern that results in sound being transmitted as a main lobe or beam with secondary side lobes. Baffles are used to eliminate the side lobes and horizontally propagating noise. To stop rain from falling directly onto the speakers, a reflector is used sometimes to reflect the transmitted sound upwards. The resulting acoustic power transmitted from the sodar should have a narrow beam of sound. The angle that the acoustic beam is transmitted upwards can be varied by either tilting the sodar or using a phase delay to the individual speakers (see Fig. 1.6).

A monostatic sodar transmits pulses of sound which have a duration τ. Not all the electrical power, P_T, supplied to the sodar to create the pulse, is converted into acoustic



(a) Scattering volume

(b) Spherical spreading

Figure 1.7: Diagrams illustrating the scattering volume and spherical spreading

power. Also, the acoustic power produced is not transmitted evenly in all directions but in a beam-like intensity pattern. The acoustic power transmitted from the sodar may, therefore, be described as $P = G\,P_T$, where G is the directional transmission antenna gain of the sodar. The height resolution of the pulse is given by

$$\Delta r = c\,\tau, \tag{1.3}$$

where c is the speed of sound and τ is the pulse duration. The volume of atmosphere that the transmitted pulse takes up at any instant in time is

$$V = \pi\,(r\,\theta)^2\,c\,\tau, \tag{1.4}$$

where r is the distance propagated and θ is the half beam width (see Fig. 1.7a). As the acoustic pulse propagates, it experiences spherical spreading and atmospheric absorption. The resulting intensity at a height r is

$$P(r) = \frac{G\,P_T\,e^{-\alpha\,r}}{4\,\pi\,r^2}.$$

where P_T is the transmitted power and α is the atmospheric absorption coefficient.

The acoustic pulse experiences scattering due to refractive index variations in turbulence and from hydrometeors. This scattered energy can be equated to the acoustic energy being reflected from perfect reflectors of cross sectional area σ per unit volume, per unit solid angle of scattering. Some of the acoustic energy is backscattered towards the receiver, again experiencing spherical spreading and absorption. The sodar then acts as a receiver.

The amount of power the sodar detects can be equated to that falling on a perfect antenna of effective antenna area A_e. This takes into account the cross sectional area of the sodar and its efficiency in converting acoustic power into electrical power.

1.5 Acoustic Radar Equation for a Monostatic Sodar

The total backscattered power measured by the sodar, P_R, from a range r may be expressed as

$$P_R = \frac{P_T\, G\, A_e\, \sigma\, V\, e^{-2\,\alpha\, r}}{16\,\pi^2\, r^4},$$

where P_T is the transmitted power, G is the transmission antenna gain, A_e is the receiving antenna effective area, σ is the scattering cross-section for both turbulence and hydrometeors, V is the scattering volume, and α is the atmospheric absorption coefficient. Substituting Equation (1.4) for the scattering volume V gives

$$P_R = \frac{c\,\tau\,\theta^2\, P_T\, G\, A_e\, \sigma\, e^{-2\,\alpha\, r}}{32\,\pi\, r^2}, \tag{1.5}$$

where c is the speed of sound, τ is the pulse duration, and θ is the half beam width. There is an extra factor of 2 included in Equation (1.5) to allow for the fact that sound travels to r and back.

The scattering cross section σ is the sum of the scattering cross section due to turbulence, σ_T, and rain, σ_R. The turbulence scattering cross section has a $\lambda^{-1/3}$ dependence while that for rain has a λ^{-4} dependence. It is hoped that this wavelength dependence will enable rain and turbulence components of the backscattered power spectra to be separated. To achieve this, other frequency dependent terms in Equation (1.5) have to also be allowed for.

1.5.1 Turbulence Scattering Cross Section σ_T

Little (1969) showed that, for a Kolmogorov spectrum of turbulence, the scattering cross section was

$$\sigma_T = 0.03\, k^{1/3}\, \cos^2\phi \left[\frac{C_V^2}{c^2}\cos^2\left(\frac{\phi}{2}\right) + 0.13\,\frac{C_T^2}{T_a^2}\right] \sin^{11/3}\left(\frac{\phi}{2}\right), \tag{1.6}$$

where C_V is the velocity structure parameter, C_T is the thermal structure parameter, k is the wave number, c is the speed of sound, T_a is the temperature, and ϕ is the difference in angle between the original direction of propagation of the wave and the direction of scattering. For a monostatic sodar, where $\phi = 180^o$,

$$\sigma_T = 7 \times 10^{-3}\, \lambda^{1/3}\, \frac{C_T^2}{T_a^2}. \tag{1.7}$$

This is the turbulence scattering cross section per unit volume, per unit incident flux, per unit solid angle. Equation (1.7) shows that the scattering cross section for turbulence has a $\lambda^{-1/3}$ dependence.

1.5.2 Precipitation Scattering Cross Section $\sigma_\mathbf{R}$

The scattering of an acoustic wave from a particle can be described by Rayleigh scattering if the particle diameter D is small compared to the wavelength λ of the acoustic wave. Little (1972) has shown that, for Rayleigh scattering from a single sphere of water or ice, the scattering cross-section is given by

$$\sigma_R = \frac{\pi^5\, D^6}{9\, \lambda^4} [1 - 3/2\, \cos\phi], \tag{1.8}$$

where ϕ is the difference in angle between the original direction of propagation and the direction of scattering. For a monostatic sodar, where $\phi = 180^o$,

$$\sigma_R = \frac{\pi^5\, D^6}{36\, \lambda^4}. \tag{1.9}$$

If there are $n(D)dD$ drops of diameter D to $D + dD$ per unit volume,

$$\sigma_R = \frac{\pi^5}{36\, \lambda^4} \int D^6\, n(D)\, dD, \tag{1.10}$$

where $n(D)$ is the drop size distribution. This is the rain scattering cross section per unit volume, per unit incident flux, per unit solid angle. Equation (1.10) shows that, for Rayleigh scattering from rain, the scattering cross section for rain has a λ^{-4} dependence.

The maximum frequency range considered in this thesis is between 2 and 12 kHz . The question arises as to whether the scattering from raindrops is still Rayleigh scattering for the upper limit of this frequency range. Shamanaev (2003) has suggested that, for acoustic frequencies as low as 5 kHz, the scattering from a raindrop is better described by a Mie-like theory of scattering. Bradley, however, in studies of a 40 kHz sodar, found that Mie-like scattering from drops was minimal compared to Rayleigh scattering. From this research, it is assumed that the scattering cross section for rain is described by Rayleigh scattering in the frequency range considered.

Drop Size Distribution n(D)

Rain is composed of a range of drop sizes. As drops fall, collisions occur resulting in either rebound or coalescence. This process causes a reduction in the number of smaller drops and the increase in the size of the larger drops. Balancing this process is the tendency of larger drops to brake up into smaller drops. The resulting maximum expected drop diameter is about 6 mm (Doviak and Zrnic, 1993).

The number density of drop sizes is referred to as the drop size distribution, $n(D)$. The most commonly used drop size distribution is the exponential form proposed by Marshall and Palmer (1948). For a Marshal and Palmer drop size distribution,

$$n(D) = N_o\, e^{-\Lambda\, D} \tag{1.11}$$

in m^{-4}, where N_o is 8×10^6 m^{-4}, Λ is $4100\,/\,I^{0.21}$ in m^{-1}, and I is the rain intensity in mm/h. Other distributions have been proposed (Cerro et al., 1997). For example Ulbrich (1983) gave a gamma distribution,

$$n(D) = N_o\, D^{\mu}\, e^{-\Lambda\, D}, \tag{1.12}$$

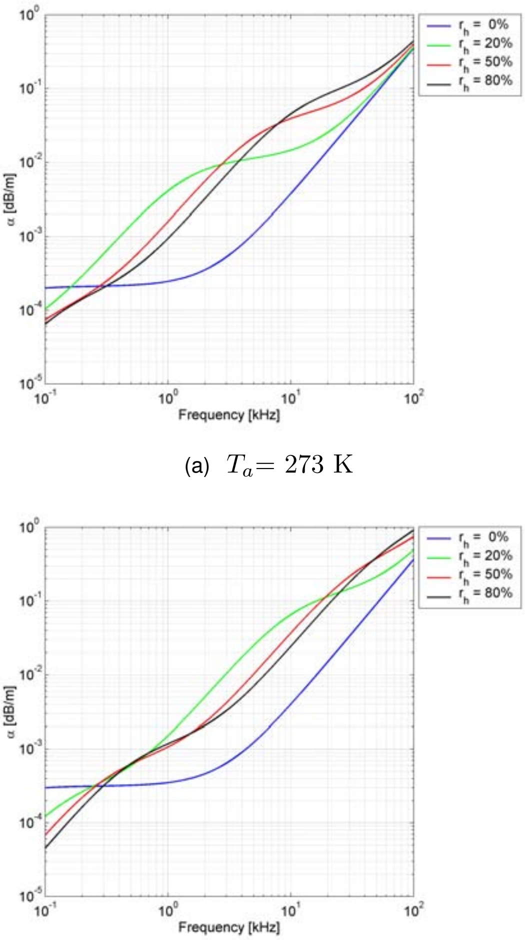
where μ may be between -3 and 8. It has also been been shown that the measured values of N_o, Λ, and μ are highly variable both with location and with time. This variability means that it is desirable not to make assumptions regarding the drop size distribution when interpreting radar or sodar data (Doviak and Zrnic, 1993; Sangren et al., 1984).

1.5.3 Atmospheric Absorption α

Atmospheric absorption is strongly dependant on frequency. High frequency acoustic waves are attenuated much more than lower frequency acoustic waves. This factor needs to be allowed for when using multiple frequencies.

Absorption can be be divided into two types, classical and non-classical absorption. In classical absorption, energy is lost as a result of the transfer of momentum and heat flow causing a reduction in viscosity and temperature gradients present in sound waves. This process is governed by the thermal conductivity and viscosity of air. Non-classical absorption is the result of energy losses due to oxygen and nitrogen molecules relaxing to their original state after being in a excited state following compression by a sound wave (Salomons, 2001).

The absorption of sound is dependent on the frequency of the sound and the pressure and the relative humidity of the atmosphere it is passing through. Rossing and Fletcher



(a) T_a= 273 K

(b) T_a= 293 K

Figure 1.8: Atmospheric absorption coefficient curves for a range of humidity values at two temperatures.

(1995) provided an approximation for the absorption coefficient, for frequencies f between 2 and 100 kHz

$$\alpha \approx \beta\, f^2, \tag{1.13}$$

where $\beta = 10^{-10}\ \ \mathrm{m}^{-1}\ \ \mathrm{Hz}^{-2}$. The International Standard ISO 9613-1:1993 is used in Equation (1.14) to calculate the absorption coefficient in dB m^{-1}, in terms of the frequency f in Hz, the atmospheric temperature T_a in K, the pressure p_a in Pa, and humidity r_h in percent (Salomons, 2001; Bass et al., 1995).

$$\alpha = 8.686\, f^2\, \tau_r^{1/2} \left(1.84 \times 10^{-11}\, \rho_r^{-1} + \tau_r^{-3} \left[\frac{0.1068\, e^{(-3352\,/\,T_a)}}{f_{r,N} + f^2/f_{r,N}} + \frac{0.01275\, e^{(-2239.1\,/\,T_a)}}{f_{r,O} + f^2/f_{r,O}}\right]\right), \tag{1.14}$$

where $\tau_r = T_a\,/\,T_{20}$ and $\rho_r = p_a\,/\,p_r$ and the constants T_{20} and p_r are 293.15 K and 101,325 Pa respectively.

The terms $f_{r,N}$ and $f_{r,O}$ are the relaxation frequencies for nitrogen and oxygen and are given by

$$f_{r,N} = \rho_r\, \tau_r^{-1/2} \left(9 + 280\, h\, e^{\left(-4.17\,\left[\tau_r^{-1/3}\,-\,1\right]\right)}\right) \tag{1.15}$$

$$f_{r,O} = \rho_r \left(24 + 40400\, h \left[\frac{0.02 + h}{0.391 + h}\right]\right). \tag{1.16}$$

The term h is the molar concentration of water vapor in the atmosphere in percent and is given by

$$h = \frac{r_h\, \rho_{sat}}{\rho_r}, \tag{1.17}$$

where ρ_{sat} is the saturation vapor pressure,

$$\rho_{sat} = 10^{\left(-6.8346\,[T_{01}\,/\,T_a\,]^{1.261}\,+\,4.6151\right)}, \tag{1.18}$$

with T_{01} being the triple point of water, 273.16 K.

1.5.4 Transmission Antenna Gain and Receiving Effective Area G and $\mathbf{A_e}$

The transmission antenna gain and the receiving effective area represent the sodar's efficiency at transmitting and receiving respectively. Both have the potential to be frequency dependent and must be allowed in any turbulence and rain backscatter fitting algorithm.

1.6 Backscattered Power due to Turbulence and Rain

The backscattering power from turbulence can be obtained by substituting equation Equation (1.7) into Equation (1.5),

$$P_{_{Turbulence}} \propto \left[\lambda^{-1/3}\ \frac{C_T^2}{T^2}\right] \left[\frac{c\,\tau\,\theta^2\, P_T\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\,r}}{r^2}\right]. \tag{1.19}$$

Substituting Equation (1.10) into Equation (1.5) gives the backscattering power due to rain,

$$P_{_{Rain}} \propto \left[\lambda^{-4} \int D^6\, n(D)\, dD\right] \left[\frac{c\,\tau\,\theta^2\, P_T\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\,r}}{r^2}\right]. \tag{1.20}$$

1.6.1 Turbulence Backscatter Power Spectra

Consider an acoustic pulse, with a narrow band width, that is transmitted into the atmosphere and backscattered by turbulence. The backscattered signal will experience some spectral broadening due to the turbulence. If there is no mean air motion in the direction of the the sodar beam, the measured spectra of the echo can be approximated to a Gaussian (Wakasugi et al., 1986) centered at the transmit frequency with a power given by Equation (1.19). This Gaussian can be expressed in terms of the Doppler shift frequency f_i, which has its zero at the transmit frequency and is related to the radial velocity w_i of the scatterer by

$$f_i = \frac{2\,w_i}{\lambda}, \tag{1.21}$$

where velocity in a downwards direction is positive. For the case of no mean air motion in the direction of transmission, the backscattered power spectra can be expressed as,

$$T(\lambda) = a\,G(\lambda)\,A_e(\lambda)\,e^{-2\,\alpha(\lambda)\,r}\,\lambda^{-1/3} \int e^{\left(-\frac{1}{2}\frac{f^2}{\sigma_f{}^2}\right)} df, \tag{1.22}$$

where σ_f is the width parameter, and a is a constant.

Equation (1.22) gives the backscatter power spectrum that would be expected for a continuous spectrum. The backscattered power spectrum, however, is generally obtained as discrete frequency values f_i, where $i = 0, 1, 2, ...N$. The signal power in each frequency bin may be calculated by

$$T_i(\lambda) = a\,A_e(\lambda)\,G(\lambda)\,e^{-2\,\alpha(\lambda)\,r}\,\lambda^{-1/3} \int_{f_i}^{f_{i+1}} e^{\left(-\frac{1}{2}\frac{f^2}{\sigma_f{}^2}\right)} df. \tag{1.23}$$

A component of the air motion in the direction of the sodar will result in a Doppler shifting of the turbulence peak. This Doppler shift may be used to obtain a profile of the wind's magnitude and direction by transmitting in three or more different angles, see Fig. 1.6b.

1.6.2 Rain Backscatter Power Spectra

Consider an acoustic pulse with a narrow bandwidth that is transmitted up into the atmosphere and backscattered from falling raindrops. Rain is composed of a range of drop sizes. Since each drop size falls at a different speed, each drop size will result in a different Doppler shift of the backscattered echo. Equation (1.26) gives a relationship between a drop's terminal velocity and its diameter. The relationship between the Doppler shift, f_i, and the drop's fall speed, w_i, is given by Equation (1.21). The frequencies of the backscattered echoes from rain may, therefore, be used to derive the range of drops sizes present in the rain. The magnitude of the backscattered power from drop size D is proportional to the number of drops of that size in the scattering volume and the sixth power of their diameter,

$$R(\lambda) = b\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\,r}\, \lambda^{-4} \int D^6\, n(D)\, dD, \tag{1.24}$$

where there are $n(D)\, dD$ drops of diameter D to $D + dD$ per unit volume of air. This means that the drop size distribution may be derived from the magnitude and Doppler shifting of the backscattered power spectra.

Equation (1.24) gives the backscattered power spectrum that would be expected of a continuous spectrum. The backscattered power spectrum, however, is measured as discrete frequencies or frequency 'bins' f_i, where i = 0, 1, 2, ... N. Each frequency bin f_i corresponds to a drop diameter range D_i. The backscattering power from rain for each frequency bin can thus be expressed as

$$R_i(\lambda) = b_i\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\,r}\, \lambda^{-4} \int_{D_i}^{D_{i+1}} D^6\, n(D)\, dD. \tag{1.25}$$

Raindrop Terminal Velocity

As a raindrop falls, the drop reaches a terminal velocity as the gravitational force on the drop is balanced by the drag force. The terminal velocity of a drop is related to its diameter. Gunn and Kinzer (1949) made accurate measurements of the terminal velocity of water drops in stagnant air at sea level. Using these measurements, many models of drop terminal velocity dependence on diameter have been proposed including those by Best (1950), Beard (1976), Dingle and Lee (1972), Atlas and Ulbrich, Uplinger (1981), and Foote and du Toit (1969). A common equation given by Atlas et al. (1973) is

$$V_T = 9.65 - 10.3\, e^{-600D}, \tag{1.26}$$

where V_T is in m/s and D is in m . This equation gives values which agree to within 2% with the measurements by Gunn and Kinzer (1949) for a drop diameter range of 0.6 to 5.8 cm (Doviak and Zrnic, 1993). Equation (1.26) has the advantage of being easily inverted to give the drop diameter as a function of drop fall speed,

$$D = -\frac{0.01}{6} \ln \left(\frac{965 - 100\, V_T}{1030} \right). \tag{1.27}$$

The dependence of a raindrop's terminal velocity on its diameter enables the drop size distribution to be deduced from the backscattered spectra measured by a sodar or radar (Coulter et al., 1989; Rogers and Pilie, 1962). If a drop is falling at its terminal velocity, the resulting Doppler shift will be

$$f = \frac{2\,V_T}{\lambda}, \tag{1.28}$$

where λ is the transmit wavelength of the sodar or radar. Note that velocity in a downwards direction and Doppler shifting due to a falling drop are both assumed to be positive. By rearranging Equation (1.28) for V_T and substituting into Equation (1.27), the corresponding drop size can be estimated.

Effect of Vertical Air Motion and Turbulence on Fall Speed

Vertical air motion will have the effect of changing a drop's fall speed from that of its terminal velocity. This results in a shift in the measured Doppler spectrum from what it would be in stagnant air. Not allowing for this vertical air motion can lead to errors in the estimated drop diameters from Doppler spectra (Atlas et al., 1973; Gossard, 1988).

Turbulence may also result in drops of the same size falling at different speeds within a given volume. This would result in broadening of the Doppler spectra. How much effect the turbulence would have on modifying the drops' fall speed would depend on the size of the drops and on the magnitude of the turbulence. Pang and Grassl (2005) and Wakasugi et al. (1986) have suggested that, for rain falling through turbulence, the observed backscattered spectra from rain can be described by the convolution of the turbulence spectra with the rain spectra that would have occurred if there had been no turbulence. Retrieval of the drop size distribution would then require deconvolution of the rain spectra with the turbulence spectra. This may be a good approximation for smaller drops, but larger drops will have more momentum and will be affected less by turbulence. This means that a pure convolution will be less likely to accurately describe the spectra from larger drops.

2

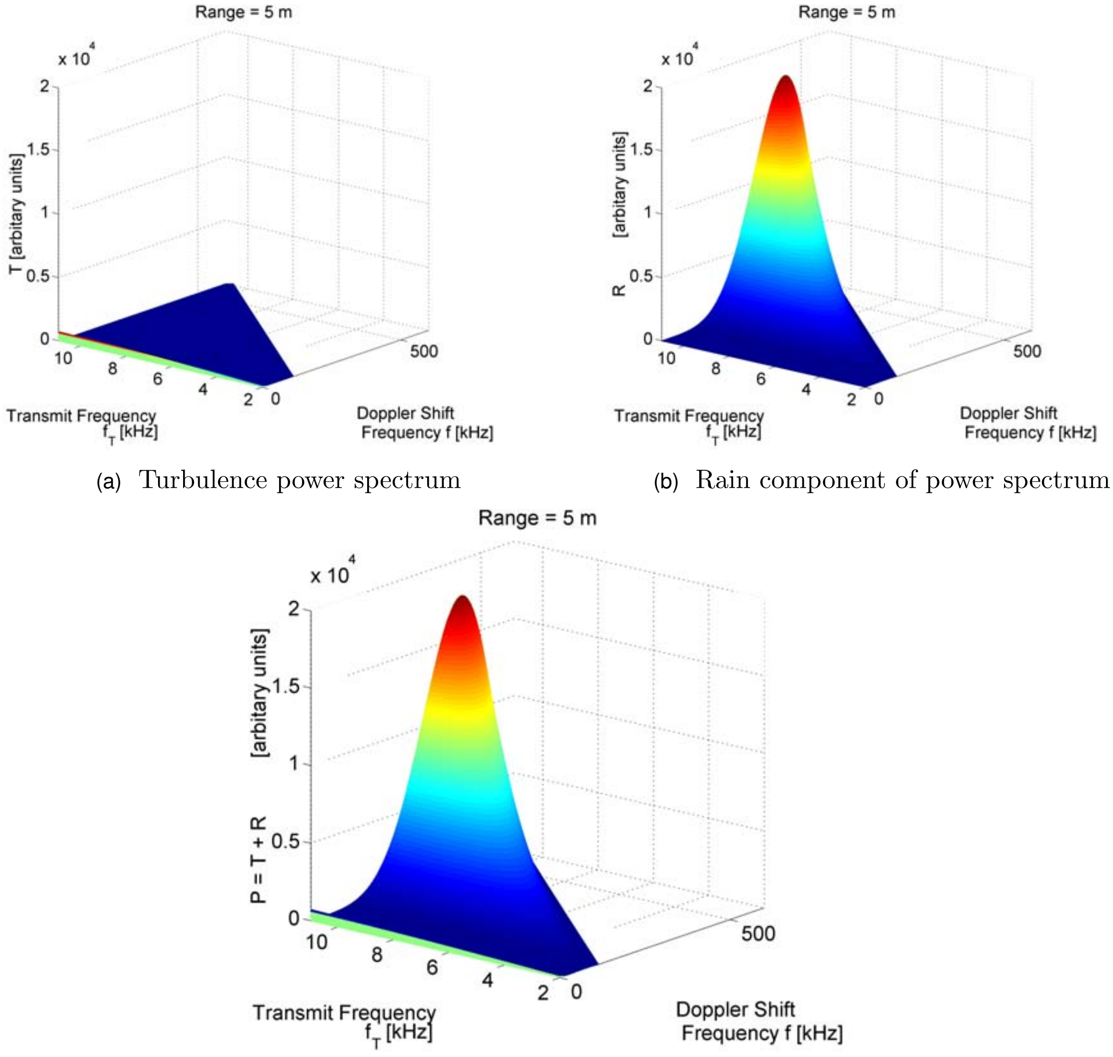
Simulation and Fitting of the Backscattered Power Spectra from Turbulence and Rain

This chapter describes a simulation of the power spectra of echoes from rain and turbulence due to multi-frequency sounding from a vertical pointing, monostatic sodar, for the case of no vertical air motion. An algorithm is then described which attempts to separate the rain and turbulence components of the measured backscattered echoes. It is hoped that this would enable accurate wind profiles to be measured during rain, or rain profiles to be measured during windy conditions.

2.1 Simulation

The backscattered power spectra is assumed to be the sum of the backscattering from turbulence and rain with noise added. For the case of the i_{th} Doppler shift frequency bin f_i this may be expressed as

$$P_i = T_i + R_i + N_i, \tag{2.1}$$



(a) Turbulence power spectrum

(b) Rain component of power spectrum

(c) Total backscattered power spectrum

Figure 2.1: Simulated backscattered power spectrum due to turbulence and rain using a transmit / receive gain of the sodar GA_e of unity.

where P is the total backscattered power spectrum, T and R are the backscattered power spectrum from turbulence and rain respectively, and N is noise power. Velocity in a downwards direction and Doppler shifting from the downward movement of a scatterer are assumed to be positive.

2.1.1 Backscattered Power Spectra from Turbulence

The backscattered power spectra from turbulence T_i in each frequency bin f_i is assumed to be described by Equation (1.23). This equation may be approximated as

$$T_i(\lambda) = a\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\, r}\, \lambda^{-1/3}\, e^{\left(-\frac{1}{2}\,\frac{f_i^2}{{\sigma_f}^2}\right)}\, (f_{i+1} - f_i)\,, \tag{2.2}$$

where $A_e(\lambda)$, $G(\lambda)$, and $\alpha(\lambda)$ are the wavelength dependant transmission and receiving gain of the sodar and the atmospheric absorption coefficient, r is the range, and σ_f is the Gaussian pulse width parameter. The Doppler shift frequencies f_i may be replaced by $2\,w_i\,/\,\lambda$, Equation (1.21), to give T_i in terms of the radial velocity of the scatterers. For multi-frequency transmission, Equation (2.2) may then be expressed as

$$T_{ij} = a\, G_j\, A_{e_j}\, e^{-2\,\alpha_j\, r}\, e^{\left(-\beta_i\, x_j^2\right)}\, v_i\, x_j^{4/3}, \tag{2.3}$$

where $x_j = \lambda_j^{-1}$, $v_i = 2\,(w_{i+1} - w_i)$, and $\beta_i = \left(2\,\frac{w_i^2}{\sigma_f^2}\right)$.

In this simulation, it is assumed that there is no vertical air motion and hence that the turbulence spectral peak is centered at the transmit frequency. If, however, the backscattering is from turbulence which is in air that has a mean vertical air motion $\overline{w}$, the velocity w_i should be replaced by $(w_i - \overline{w})$ (Wakasugi et al., 1986). This would cause a shift in the Doppler shift peak. It is this shift in the peak that is used in sodar measurements of wind magnitudes.

2.1.2 Backscattered Power Spectra from Rain

The backscattered power spectra from rain in each frequency bin is assumed to be described by Equation (1.25). This may be approximated by

$$R_i(\lambda) = b_i\, G(\lambda)\, A_e(\lambda)\, e^{-2\,\alpha(\lambda)\, r}\, \lambda^{-4}\, D_i^6\, n(D_i)\, (D_{i+1} - D_i), \tag{2.4}$$

where R_i is the rain backscattered power, D_i is the drop diameter, and $n(D_i)$ is the drop size distribution corresponding to the i_{th} frequency bin. For multi-frequency transmission and an assumed Marshall and Palmer drop size distribution, this may be expressed as

$$R_{ij} = b_i\, G_j\, A_{e_j}\, e^{-2\,\alpha_j\, r}\, D_i^6\, e^{-\Lambda D_i}\, dD_i\, x_j^4, \tag{2.5}$$

where $x_j = \lambda_j^{-1}$, $dD_i = (D_{i+1} - D_i)$, $\Lambda = 4100/I^{0.21}$, and I is the rain intensity. The drop diameters may be related to their terminal velocities V_{T_i}, using Equation (1.27), and to their corresponding Doppler shifts f_i, in still air, using Equation (1.28).

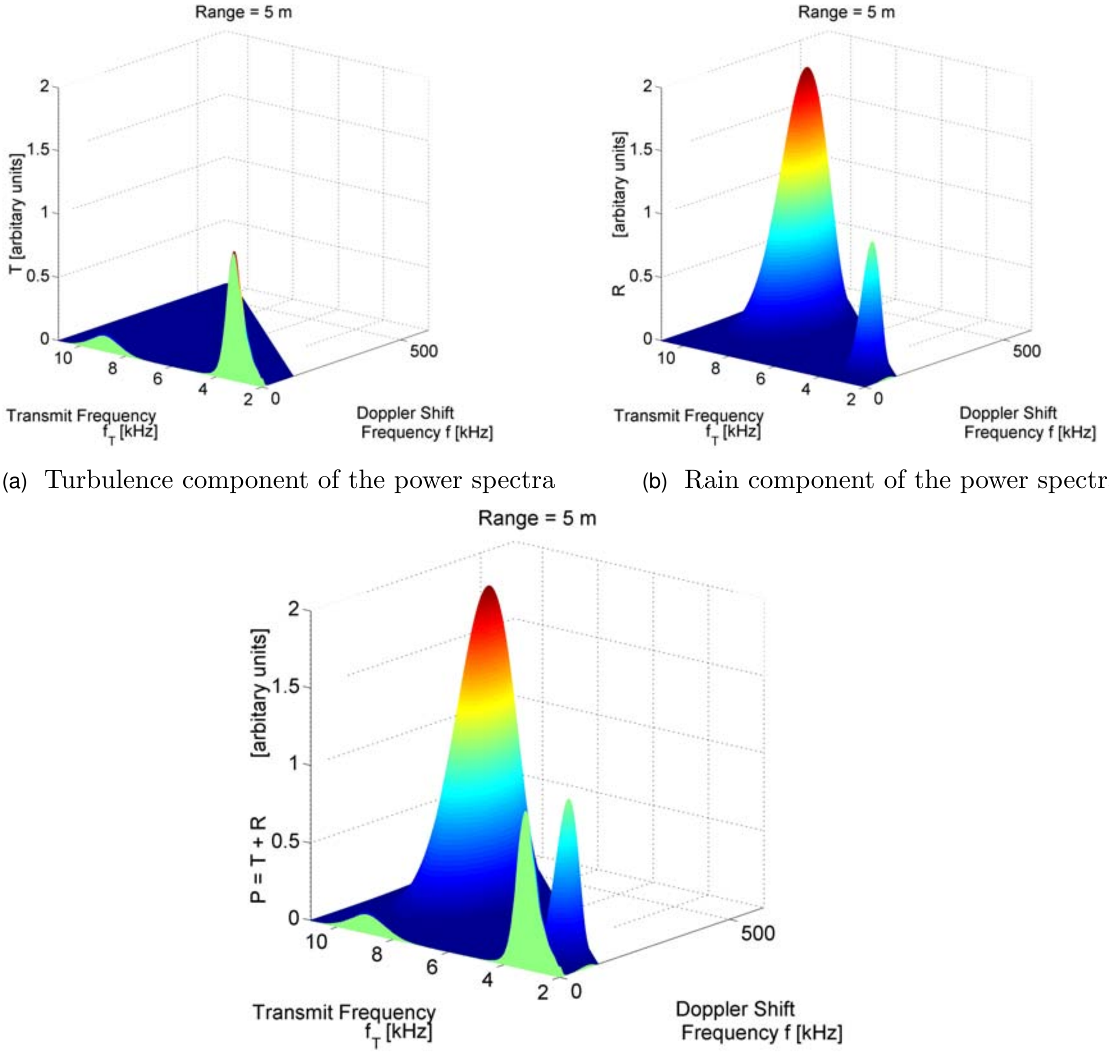
In this simulation, it is assumed that the drops fall at their terminal velocities. In reality, the fall speeds of rain drops may be modified by mean vertical air motions and turbulence. If the drops are falling through air which has a mean vertical air motion $\overline{w}$, the actual fall speed w_i of the drops will be $(V_{T_i} + \overline{w})$. This will cause a shift in Doppler spectra from the rain from what it would be for drops falling at their terminal velocities. Turbulence may also modify the fall speed of the drops depending on the strength of the turbulence and size of the drops. This might be expected to cause some broadening of the Doppler spectra from rain, particularly for smaller drop sizes. Pang and Grassl (2005) has suggested that the rain spectrum can be expressed as the convolution of the precipitation Doppler spectrum with the turbulence spectrum. Larger drops will have a higher momentum, however, than smaller drops and will, therefore, be less effected by turbulence. This may mean that the above simple convolution may not accurately describe the spectra for larger drops.

2.1.3 Noise

There will be a range of noise measured by the sodar that will contaminate the backscattered signal from turbulence and rain. This noise includes acoustic noise from rain impacting on the sodar case, wind, cars, aircraft, etc, electrical noise from the sodar circuitry, and from digitisation errors. The noise is expected to be frequency dependant, with higher noise levels at lower frequencies (Coulter and Martin, 1986; Hopkins, 2006).

2.1.4 Simulation Process

The backscattered power spectra from both rain and turbulence was simulated for the case of a multi-frequency, vertical pointing, monostatic sodar. These two simulated power spectra were then added to give a simulated total power spectrum. It was assumed that there was negligible vertical air motion and that the rain drops were falling at their terminal velocities. No allowance has been made in these simulations for the variation of gain with frequency of the sodar's bandpass filter circuit or of the sound card microphone



(a) Turbulence component of the power spectra

(b) Rain component of the power spectra

(c) Total backscattered power spectra

Figure 2.2: Simulated backscattered power spectra due to rain and turbulence using the measured transmit / receive gain of the sodar GA_e with no bandpass filters included.

input of the computer used to run the sodar (see Sections 4.3.3 and 4.3.4). Future work could include these gains into the simulations.

The first step in the simulation process was to create a function file `RT_Sim_functions.m`, see *MatLab* Code A.3, to generate terms given in Equation (2.3) and Equation (2.5). These included the Doppler shift frequency bins f_i, drop terminal velocities V_{T_i}, turbulence Gaussian exponential parameter β_i, and drop diameters D_i corresponding to the center of each bin. Also simulated were the drop terminal velocity difference v_i and drop size difference dD_i that would be expected across each frequency bin. The absorption coefficient α_j and the product of the sodar's transmit / receive gain $(G_j\, A_{e_j})$ for transmit wavelength λ_j was calculated using function files `RT_absorb.m` and `RT_G_Ae.m`, see *MatLab* Code A.1 and A.2. If a rain intensity is included as input argument to `RT_Sim_functions.m`, a Marshall and Palmer drops size distribution $n(D_i)$ was also calculated.

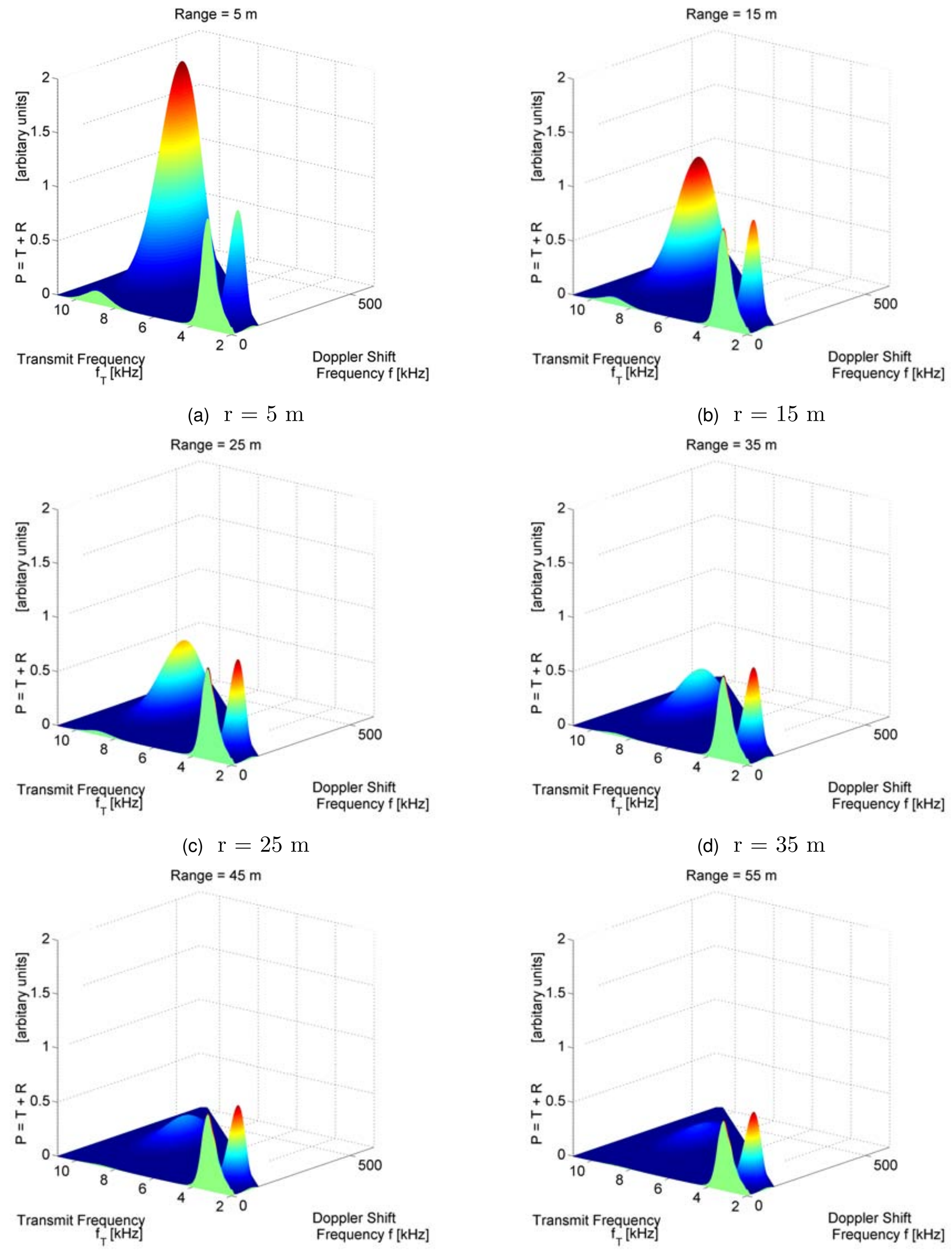
The terms calculated by `RT_Sim_functions.m` were then used by `RT_Sim_Echo.m`, see *MatLab* Code A.4, to simulate the backscattered power spectra from turbulence T_i and rain R_i, using Equation (2.3) and Equation (2.5), for a given height resolution interval centered at range r. These spectra are then added to give the total backscattered power spectra P_i in each frequency bin. The resulting values of T, R, and P are formatted as matrixes, with each row representing the simulated power spectrum for a given transmit wavelength and each column representing a Doppler shift frequency bin.

Simulation of Doppler Shift Frequency Bins, f_i

The Doppler shift frequency bins f_i, where $i = 0,\, 1,\, 2,\, \ldots N$, were made to have a band width equal to that which would be expected from rain falling at their terminal velocities. This was done by assuming a rain drop diameter range of between 0.1 and 6 mm. This range was then used to simulate a minimum and maximum terminal velocity using Equation (1.26). This was converted into a minimum and maximum Doppler shift for each transmit wavelength using Equation (1.28). Doppler shift bins were then generated by dividing this maximum and minimum frequency range into the same number of frequency bins. This process ensures that the i_{th} frequency bin f_i will correspond to the same drop size range regardless of the transmission wavelength.

Simulation of the Drop Terminal Velocities V_{T_i} and Related Terms v_i and β_i

The Doppler shift frequencies f_i where converted into terminal velocities $V_T(f_i)$ using Equation (1.28). The terminal velocities at the edges of the bins were then calculated as $V_T(f_i - bin/2)$ and $V_T(f_i + bin/2)$, where bin is the width of the frequency bins for



(a) $\mathrm{r} = 5\ \mathrm{m}$

(b) $\mathrm{r} = 15\ \mathrm{m}$

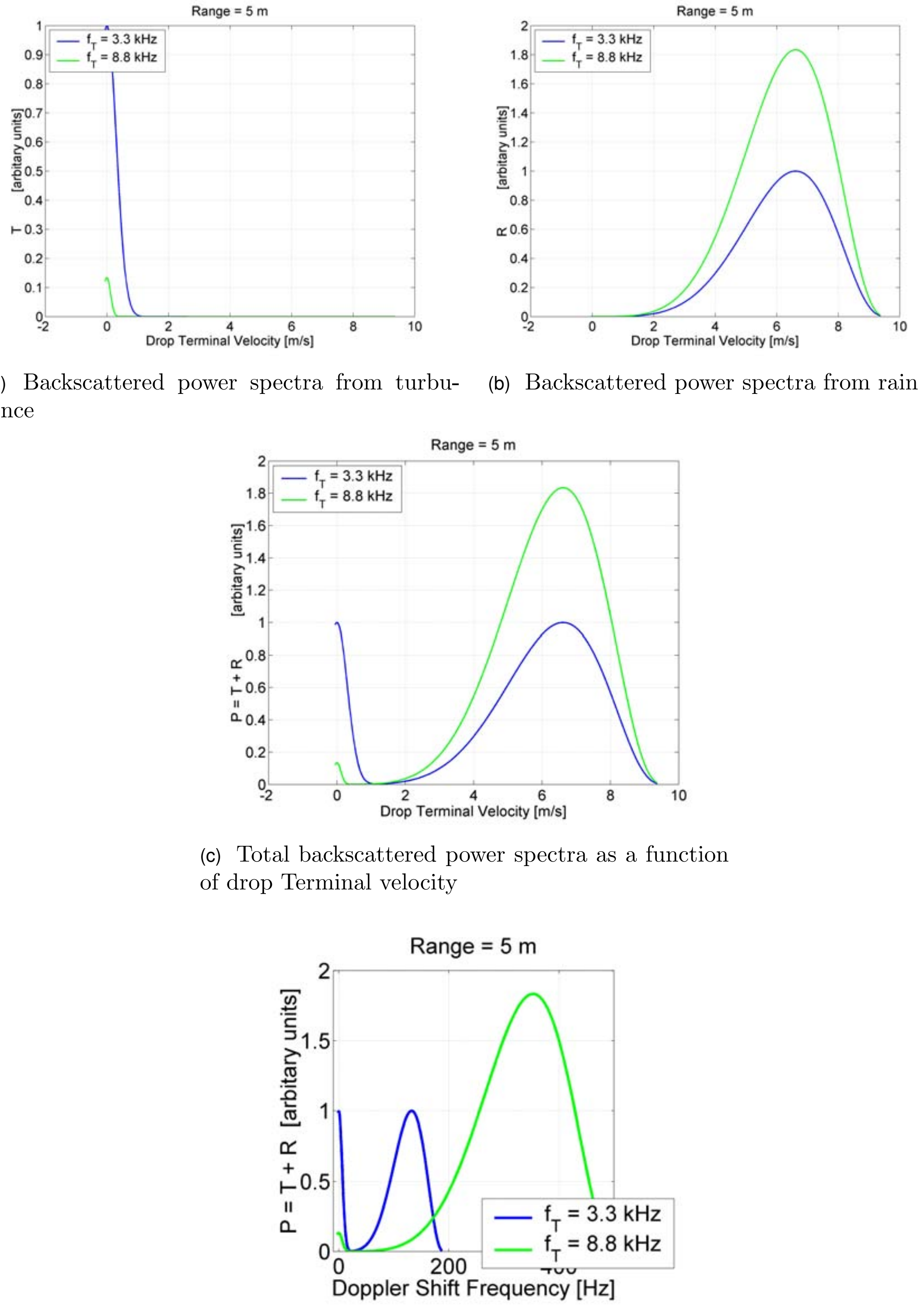
(c) $\mathrm{r} = 25\ \mathrm{m}$

(d) $\mathrm{r} = 35\ \mathrm{m}$

(e) $\mathrm{r} = 45\ \mathrm{m}$

(f) $\mathrm{r} = 55\ \mathrm{m}$

Figure 2.3: Simulated power spectrum showing the effect of atmospheric absorption for heights from 5 to 55 m in increments of 10 m.



(a) Backscattered power spectra from turbulence

(b) Backscattered power spectra from rain

(c) Total backscattered power spectra as a function of drop Terminal velocity

(d) Total backscattered power spectra as a function of Doppler shift frequency

Figure 2.4: Simulation of the total backscattered power spectra due to a combination of rain and turbulence for two transmit frequencies

transmission wavelength λ. Twice the difference v_i between these two frequencies was also calculated $v_i = 2\,[\,V_T(f_i + bin/2) - V_T(f_i - bin/2)\,]$ and the turbulence Gaussian exponential parameter $\beta_i = 2\,V_{T_i}^2 \,/\, \sigma_f^2$, where σ_f is the width parameter.

Simulation of the Drop Diameters D_i

The drop diameters D_i were then simulated using the drop terminal velocities $V_T(f_i)$ and Equation (1.27). The difference dD_i between the drop diameters corresponding to the edges of each bin was also calculated using $V_T(f_i - bin/2)$ and $V_T(f_i + bin/2)$.

Atmospheric Absorption Coefficient α

The atmospheric absorption coefficient is strongly dependent on the transmission frequencies. Higher transmit frequencies are much more strongly attenuated than lower frequencies. A *MatLab* function file, *MatLab* Code A.1, was written to calculate the atmospheric absorption coefficient `alpha` in m^{-1}, using the equations given in Section 1.5.3.

Transmission and Receiving Antenna Gain of the Sodar G and A_e

The transmission antenna gain of the sodar was investigated in Sections 4.2.1, 4.3.1, and 4.3.2. It was found that the transmit gain of the sodar was strongly frequency dependant due to the characteristics of the speakers used in the sodar's speaker array (see Fig. 4.13), and also due to the gain of the power amplifier, while loaded by the speaker array and amplifying the laptop's sound card headphone output (see Fig. 4.21). The receiving gain of the sodar was investigated in Sections 4.2.1, 4.3.1, and 4.3.3. A similar frequency dependance was found due to the gain characteristics of the speakers used in the speaker array (see Fig. 4.13). The measurements made in these sections were used to create a *MatLab* function file, *MatLab* Code A.2, which used a cubic spline to give an estimate of the product of sodar's transmission and receiving gain `G_Ae` (see Fig. 4.27).

2.1.5 Discussion

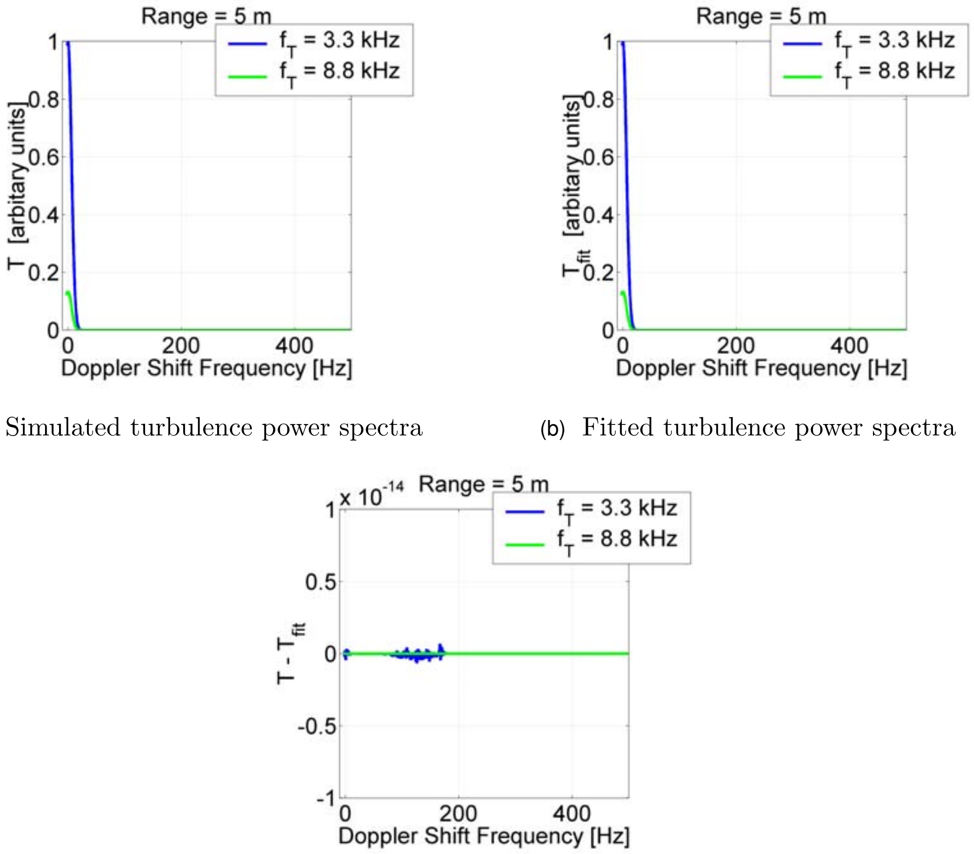
A simulation of the total backscattered power spectra from turbulence and rain is graphed in Fig. 2.1. This simulation has used a transmit/receive gain of the sodar, GA_e, equal to unity. The turbulence and rain terms have also been scaled so that their magnitudes are equal at 3.2 kHz. In Fig. 2.1a, the magnitude of the turbulence term can be seen to increase as the 4/3 power of the transmit frequency, $f_T^{4/3}$. In contrast, Fig. 2.1b shows a 4 power dependence on transmit frequency, f_T^4. It is this frequency dependence difference which is to be used to separate the turbulence and rain components of the backscattered power spectrum.

The measured transmit/receive gain of the sodar GA_e (see Fig. 4.27) with no bandpass filters and a microphone input sound card gain of unity, is included in the simulation, graphed in Fig. 2.2, of the backscattered power spectra from turbulence and rain. This simulation indicates that the gain of the sodar might impose limitations on the range of frequencies that would be suitable for transmission. This is caused by the frequency response of the speakers used in the sodar speaker array, see Fig. 4.8. The optimal transmit frequencies appear to be in two frequency ranges centered at 3.3 and 8.8 kHz.

The effect of atmospheric absorption is investigated in Fig. 2.3, where the backscattering from turbulence and rain is simulated for ranges from 5 to 55 m . This simulation illustrates how higher transmission frequencies experiences significantly more attenuation with range than lower frequencies. The 8.8 kHz turbulence and rain spectral peaks are attenuation significantly more with range than the 3.3 kHz peaks. If larger ranges are required, perhaps the 8.8 kHz range of transmit frequencies may be unsuitable. If only the 3.3 kHz range of suitable frequencies is used, the question arises as to whether this range has sufficient bandwidth to enable the algorithm to work successfully. In future designs of the sodar, the use of different speakers could be investigated, which have a flatter frequency response in the 2 to 5 kHz frequency range.

2.2 Least Squares Fitting to Separate the Rain and Turbulence Components from Simulated Backscattered Power Spectra

This section describes an algorithm which attempts to separate the rain and turbulence components of the measured backscattered echoes due to multi-frequency sounding from a vertical pointing, monostatic sodar, for the case of no vertical air motion. It is hoped that this may enable either wind measurements to be made during rain or rain measurements during windy conditions.



(a) Simulated turbulence power spectra

(b) Fitted turbulence power spectra

(c) Error in fitted turbulence power spectra

Figure 2.5: Simulated and fitted turbulence component of the backscattered power spectra for two transmit frequencies and the error in the fit.

2.2.1 Rain and Turbulence Functions to be Fitted

Consider the backscattered power in single Doppler shift frequency bin for the case of multi-frequency acoustic sounding. Let the power in this bin be P_1, P_2, $\ldots$ P_n, where there are n transmission frequencies used. It is assumed that the backscattered signal may be approximated by the simultaneous equations

$$\begin{aligned} b_1\, F_1(x_1) + b_2\, F_2(x_1) &= P_1, \\ b_1\, F_1(x_2) + b_2\, F_2(x_2) &= P_2, \\ &\vdots \\ b_1\, F_1(x_n) + b_2\, F_2(x_n) &= P_n. \end{aligned}$$

where $x = \lambda^{-1}$ the inverse of the transmission wavelength, F_1 is the turbulence backscattered power spectrum function

$$F_1(x_j) = G_j\, A_{e_j}\, e^{-2\,\alpha_j\, r}\, e^{\left(-\beta_i\, x_j^2\right)}\, v_i\, x_j^{4/3}, \tag{2.6}$$

F_2 is the rain backscattered power spectrum function

$$F_2(x_j) = G_j \, A_{e_j} \, e^{-2\,\alpha_j\,r} \, x_j^4, \tag{2.7}$$

and b_1 and b_2 are the respective coefficients of these function. Note that $v_i = 2\,(w_{i+1} - w_i)$, and $\beta_i = \left(2\,\frac{w_i^2}{\sigma_f^2}\right)$. The function F_1, given in Equation (2.6), is the same as the equation used to simulate the backscattered power spectra from turbulence, see Equation (2.3). The function F_2, given in Equation (2.7), differs from the equation used to simulate the backscattered power spectra from rain, see Equation (2.5), in that it omits the terms $D^6 n(D) dD$. Omitting these terms has the advantage of eliminating the need for making assumptions concerning the drop size distribution, which has the potential to cause errors. In the least squares fitting of the functions F_1 and F_2, it is assumed that the terms $D^6 n(D) dD$ will be contained in the fitted coefficient b_2.

The simultaneous equations above can be expressed as

$$\mathbf{A} b = P, \tag{2.8}$$

where

$$\mathbf{A} = \begin{bmatrix} F_1(x_1) & F_2(x_1) \\ F_1(x_2) & F_2(x_2) \\ \vdots & \vdots \\ F_1(x_n) & F_2(x_n) \end{bmatrix}, \tag{2.9}$$

$$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad \text{and} \quad P = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_n \end{bmatrix}.$$

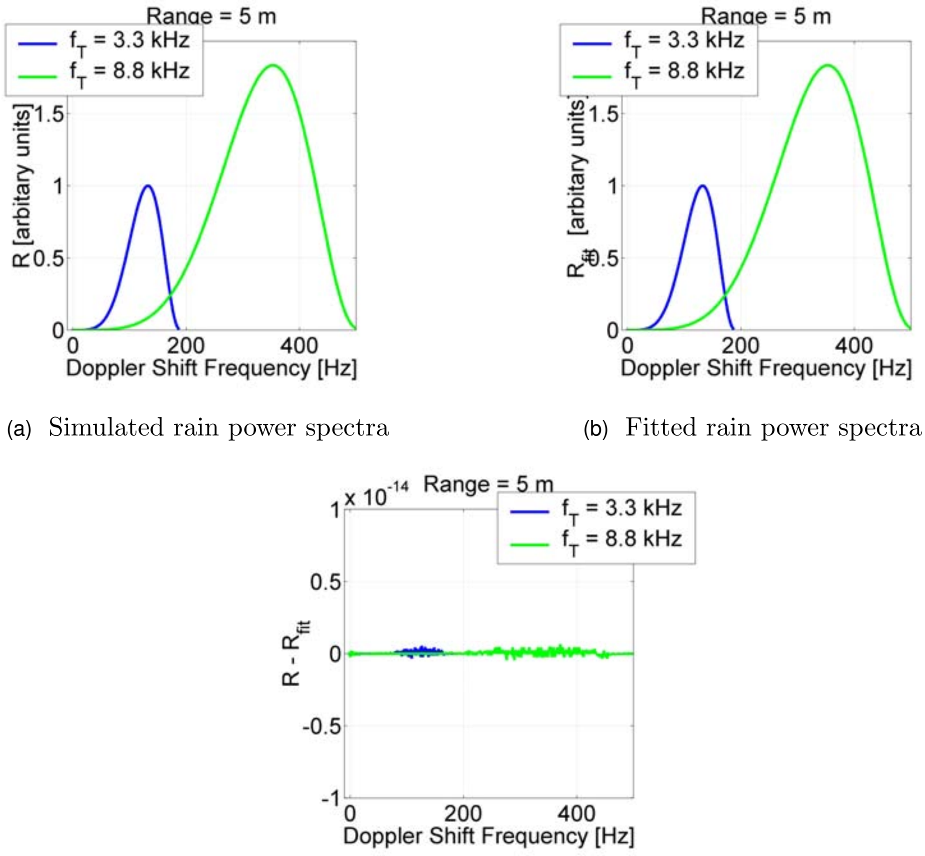
2.2.2 Least Squares Fitting Process

If an exact solution to the above simultaneous does not exist, a compromise solution is obtain by finding values of b_1 and b_2 which minimise the residuals

$$|r_j|^2 = |\, P_j - [\, b_1 \, F_1(x_j) + b_2 \, F_2(x_j) \,] \,|^2. \tag{2.10}$$

This is achieved by inverting Equation (2.8) to obtain the coefficients b (Recktenwald, 2001).

$$\mathbf{A} b = P \quad \Leftrightarrow \quad (\mathbf{A}^T \mathbf{A}) \, b = \mathbf{A}^T P \quad \Rightarrow$$



(a) Simulated rain power spectra

(b) Fitted rain power spectra

(c) Error in fitted rain power spectra

Figure 2.6: Simulated and fitted rain component of the backscattered power spectra for two transmit frequencies and the error in the fit.

.

$$b = (\mathbf{A}^T \mathbf{A})^{-1} \, (\mathbf{A}^T P). \tag{2.11}$$

The least squares fitted turbulence and rain components of the total backscattered power spectra for each Doppler shift frequency bin are then obtained by

$$T_{fit} \, (x_j) = b_1 \, F_1(x_j) \tag{2.12}$$

and

$$R_{fit} \, (x_j) = b_2 \, F_2(x_j). \tag{2.13}$$

2.2.3 Least Squares Fitting of Simulated Echoes

A *MatLab* function file `RT_Sim_Fit.m`, see *MatLab* Code A.5, was written to separate the turbulence and rain components of the backscattered power spectra using a least squares fitting of the backscattered power spectra. This function file calculates the turbulence and rain echo functions $F_1(x)$ and $F_2(x)$ using Equations (2.6) and (2.7). The terms used in these equations were calculated using the function file `RT_Sim_functions.m`, see *MatLab* Code A.3. Matrix $\mathbf{A}$ was then created using $F_1(x_j)$ as the j_{th} row, first column entry and $F_2(x_j)$ as the j_{th} row, second column entry, see Equation (2.9). A least squares fitting of the backscattered power spectra P is then preformed using Equation (2.11) to obtain the coefficients b of the functions F_1 and F_2. The fitted turbulence T_{fit} and rain R_{fit} components of the backscattered power spectra are given by Equation (2.12) and Equation (2.13).

Fig. 2.5, 2.6, and 2.8 contain plots of the simulation, fitting, and the difference between the simulation and fitting for the total backscattered power spectra and for the turbulence and rain components. The difference between the simulated and fitted values in these plots are of the order of $-300dB$, showing that the least squares fitting algorithm works for the simulated power spectra. Real acoustic sounding measurements are unlikely to be able to be fitted to the same accuracy. Factors causing this are due to noise, which is frequency dependant, and possibly to inadequacies in the theory.

2.2.4 Least Squares Fitting of Echoes from the Sodar's Raw Data

The sodar's raw data comprises time domain data points formatted as a *MatLab* vector for each hight resolution interval

$$M_h = [s_1,\, s_2,\, \ldots\, s_N].$$

The number of samples N recorded per height resolution interval is

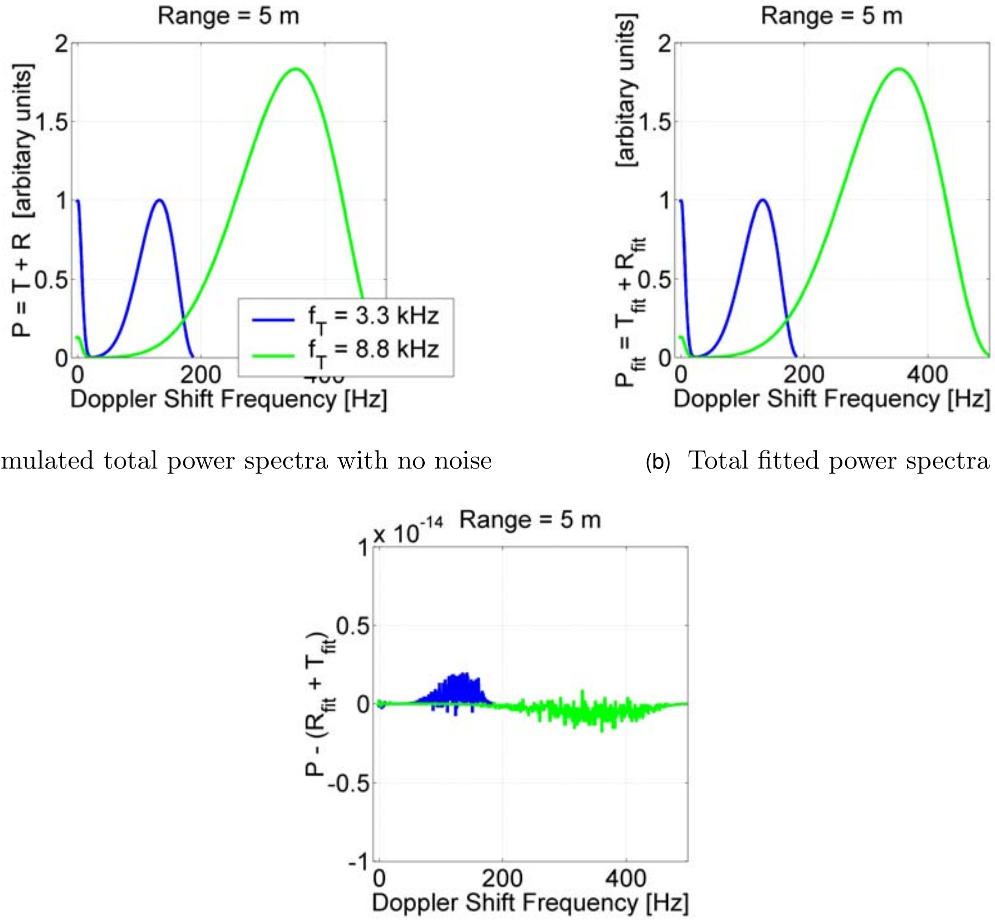
$$N = F_s \, \frac{2\,\Delta r}{c},$$

where F_s is the sampling frequency and Δr is the pulse height resolution interval. A discrete Fast Fourier Transform transform (FFT) is used to convert the time domain samples into frequency domain samples and the absolute values are then taken

$$Y_h = [S_1,\, S_2,\, \ldots\, S_N].$$

These values correspond to frequencies ranging from 0 to F_s in steps of $F_s\,/\,N$. This frequency separation between FFT values can also be expressed as $1/\tau$ or as $c/(2\,\Delta r)$.

Frequency bins are then created from the FFT values. These bins have a frequency range between f_T and $f_T\ +\ \max(f_{rain})$, where f_T is the transmission frequency and f_{rain} is the

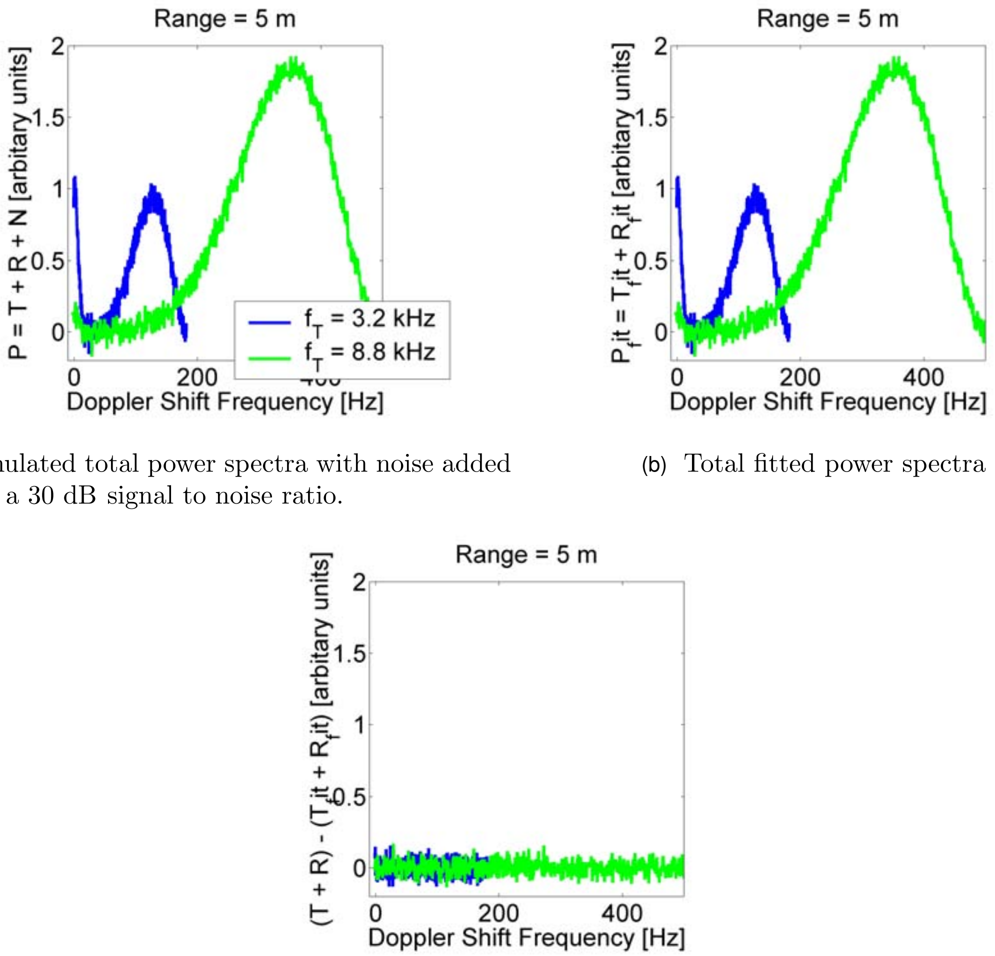


(a) Simulated total power spectra with no noise

(b) Total fitted power spectra

(c) Error in fitted power spectra

Figure 2.7: Simulated and fitted total backscattered power spectra due to a combination of rain and turbulence with no noise added for two transmit frequencies.



(a) Simulated total power spectra with noise added to give a 30 dB signal to noise ratio.

(b) Total fitted power spectra

(c) Error in fitted power spectra

Figure 2.8: Simulated and fitted total backscattered power spectra due to a combination of rain, turbulence, and noise and the error in the fit for two transmit frequencies.

expected Doppler shift frequencies from rain. The value of $\max(f_{rain})$ may be calculated using Equation (1.26) and (1.28) and by assuming a maximum drop size and that the drops fall at their terminal velocities. An approximation may be made that

$$\max(f_{rain}) \approx \frac{20\, f_{T}}{c}, \tag{2.14}$$

where c is the speed of sound. The frequency bins are created by dividing this frequency range up in $N_{f\,b}$ number of segments of equal width. The average is then taken of these FFT values within these segments. For the case of multi-frequency transmission, the number of frequency bins used needs to be kept constant for all transmission frequencies. If each Doppler shift frequency bin is to contain n FFT values, then using Equation (2.14)

$$\frac{1}{N_{f\,b}} \left(\frac{20\, f_{T_{min}}}{c} \right) \geq n \left(\frac{c}{2\,\Delta r} \right),$$

where $f_{T_{min}}$ is the minimum transmission frequency used. Rearranging for the number of frequency bins gives

$$N_{f\,b} \leq \frac{40\,\Delta r\, f_{T_{min}}}{n\, c^2}.$$

For the case of a single FFT value per bin, a minimum transmit frequency of 3 kHz, a height resolution interval of 20 m, and a speed of sound of 330 m/s, the maximum number of Doppler shift frequency bins would be 22 bins.

The *MatLab* function file `RT_Fit_RawData.m`, see *MatLab* Code A.8, was written to preform the least squares fitting of the echoes recorded in the sodar raw data file, see Section 3.5.1, using *MatLab* Code A.5. This function file averages the FFT vector over the entire file for a single height resolution interval. These FFT averaged vectors are then converted into frequency bins, formatted as a matrix with each row corresponding to a transmit frequency and each column corresponding to a Doppler shift frequency bin. *MatLab* Code A.5 is then used to preform the least squares fitting to separate the turbulence and rain components of the backscattered power spectra.

3
Sodar Design

3.1 Basic Operation of the Sodar

The sodar used in this work is a multi-frequency monostatic sodar with a vertical pointing antenna. It uses an array of 25 speakers which are mounted in the sodar case so as to point horizontally towards a reflector which is angled at 48 degrees from the vertical. Baffles and acoustic foam are used to reduce unwanted noise. Electrical pulses are generated using a *MatLab* computer code and a laptop's sound card headphone output. These pulses are amplified by a power amplifier. A transmit/receive switch is used to pass the electrical signal to the speaker array, which converts it into acoustic pulses. The sodar's reflector causes the pulse to be transmitted vertically up into the atmosphere. After each pulse, the transmit/receive switch connects the speaker array back to the laptop's microphone input*, via a preamplifier and bandpass filter.* The laptop is then able to use the sodar as a microphone to record echoes from turbulence and rain as a computer file. The GUI, which was written to run the sodar, allows the user to choose transmission parameters, such as the transmission frequencies, height resolution, and sampling frequency. To enable the rain measurements made by the sodar to be evaluated, a Doppler rain radar is run simultaneously with the sodar during precipitation events. The raw data recorded by the sodar is later processed and a least squares fitting algorithm used to separate the turbulence and rain components of the backscattered power spectra. The rain component is then compared with the drop size distribution profiles made using the Doppler rain radar while the sodar was being operated.

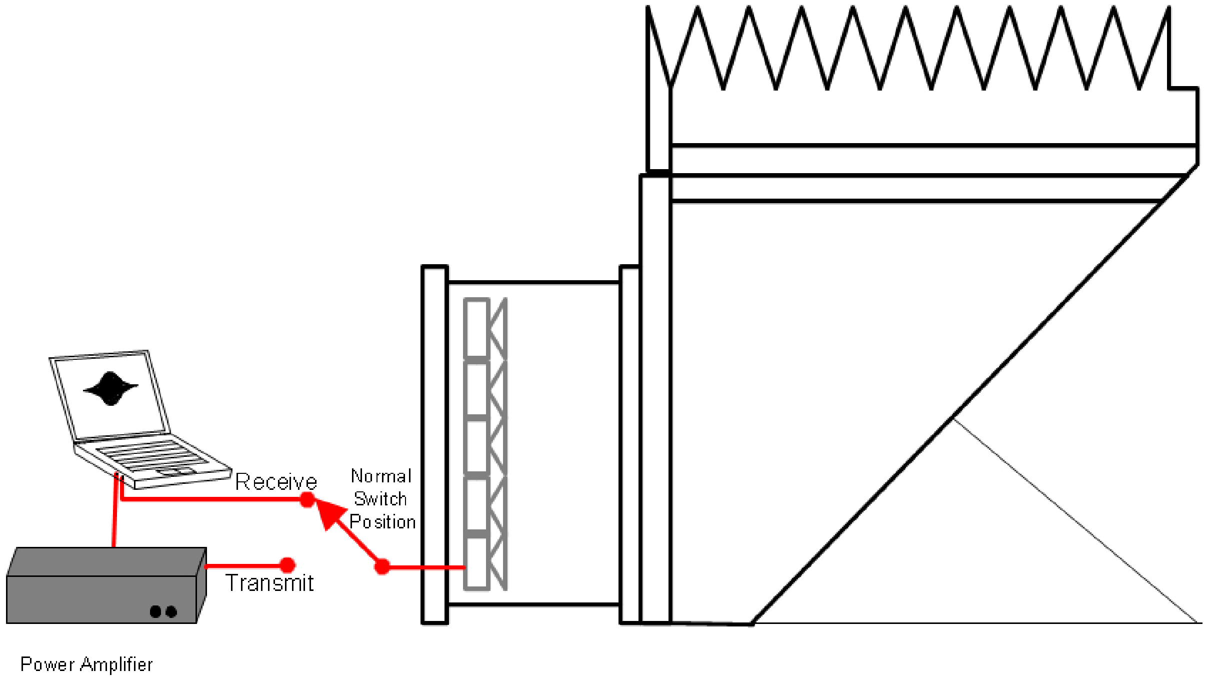


Figure 3.1: Diagram of sodar setup

3.2 Sodar Case

The sodar case houses the speaker array and prevents rain from impacting directly onto the speakers. A reflector is used to direct the sound vertically upwards. Baffles are used to reduce horizontally propagating noise. The case is composed of wood and plywood and is lined with acoustic foam to reduce external noise. See Fig. 3.2 for the dimensions of the case.

One aspect of the sodar's case that could be improved was identified during measurements of the gain of the sodar as a function of angle. It was found that, for some angles, sound was able to reach the speaker array without being first reflected from the sodar's reflector. This could be solved either by increasing the hight of the baffles or by extending the case so that the speaker array is further away from the reflector.

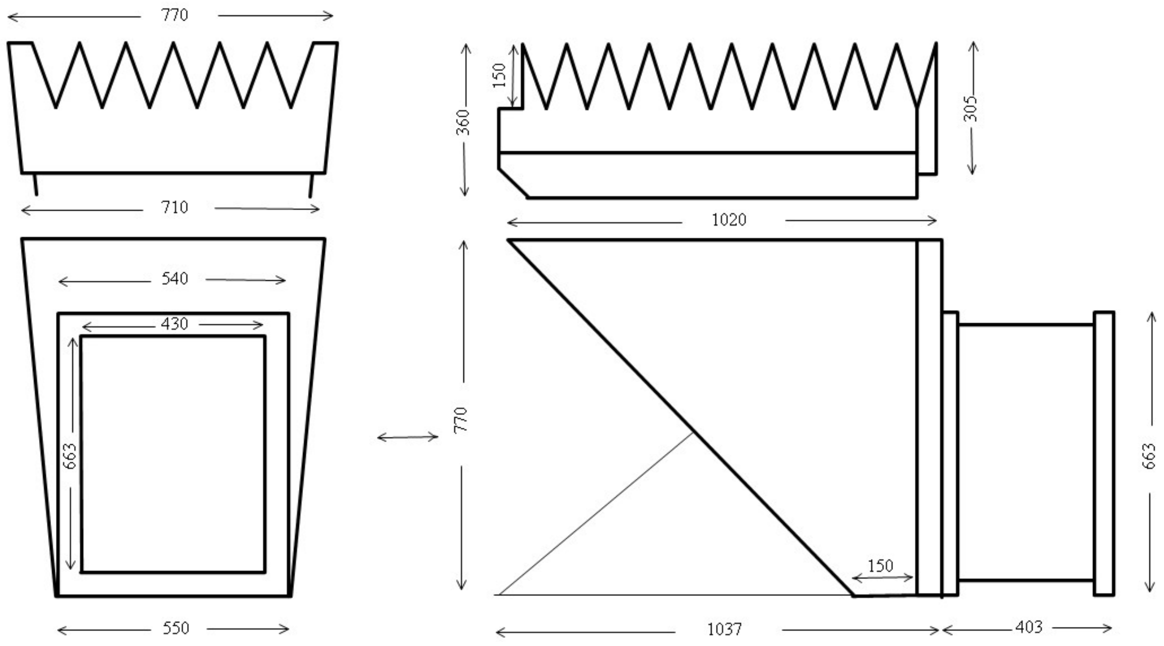


Figure 3.2: Diagram of the sodar's case in two planes where the dimensions are in millimeters

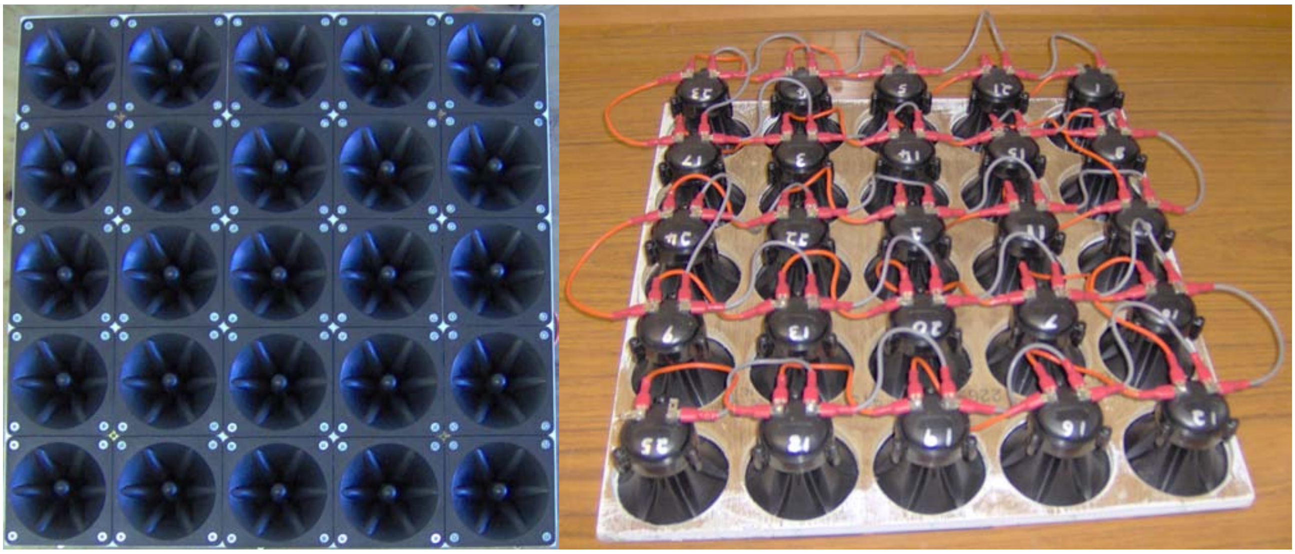
3.3 Sodar's Speaker Array

The sodar uses an array of twenty five speakers, wired in parallel, in a square grid of five rows and five columns. The speaker array is used to transmit pulses and also to act as a microphone. The speakers used were Motorola KSN1005 tweeters. These contain a piezoelectric crystal which changes shape when a voltage is applied across them or alternatively generates a voltage when they are compressed. There were no known manufacturers specifications or detailed studies of these speakers but previous work had indicated that they had high transmit / receive gains.

3.4 Sodar's Electronics

3.4.1 Transmit/Receive Switch

The transmit/receive switch was made using a double throw relay, an Omron G6K-2P 5Vdc. The relay normally connects the sodar's speaker array to the microphone input of the laptop's sound card. The sodar, therefore, normally acts as a microphone. While a pulse is being transmitted from the laptop's sound card, via the power amplifier, a dc voltage across the relay's coil rises above a threshold level. This causes the relay to disconnect the sodar's speaker array from the laptop's microphone input and connects it



(a) Array front

(b) Array back

Figure 3.3: Sodar's speaker array

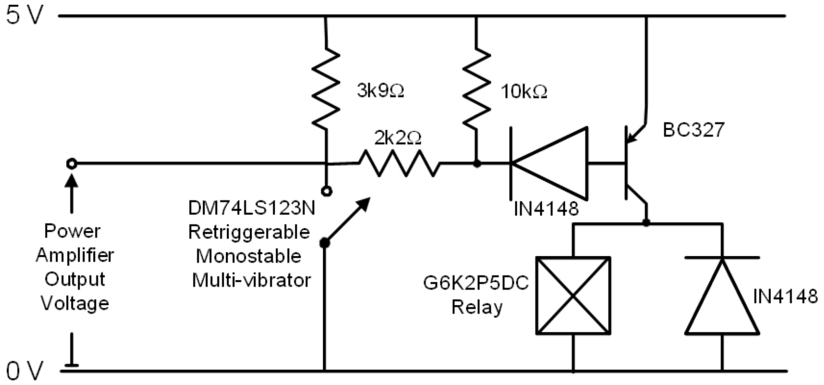


Figure 3.4: Circuit diagram of the sodar's transmit/receive switch

to the power amplifier output, allowing the the sodar to transmit the pulse. Once the pulse has ended, the voltage across the relay's coil drops below a threshold level causing the relay to disconnect the speaker array from the power amplifier and again connect it up to the laptop's sound card microphone input. This allows the laptop to use the sodar as a microphone again to record echoes caused by the transmitted pulse.

The dc voltage across the relay is controlled by a retriggerable monostable multi-vibrator, a Fairchild DM74LS123N. The power amplifier output is connected to the input of the retriggerable monostable oscillator. Each time the voltage from the power amplifier goes above a threshold level, the monostable output goes high for a set period of time. The period that monostable stays high was set by an appropriate choice of a resistor and capacitor. This period was chosen to be about a millisecond, which was longer than the period of any expected pulse transmit frequency. The result is that the retriggerable

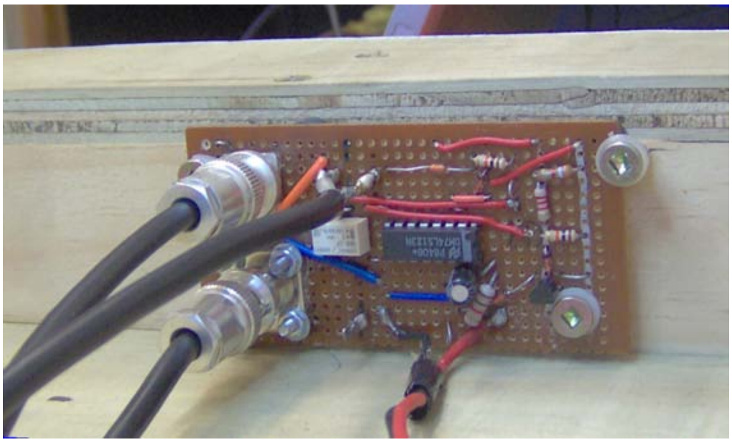


Figure 3.5: Transmit / receive switch circuit

monostable oscillator output stays high while a pulse is being transmitted by the the power amplifier. The monostable output is amplified by a transistor circuit and applied across the relay's coil. The circuit acts to apply the required dc voltage across the relays coil while the power amplifier is transmitting a pulse. This causes the sodar to transmit the pulse and then revert back to acting as a microphone.

3.4.2 Preamplifier and Bandpass Filter Circuit Board

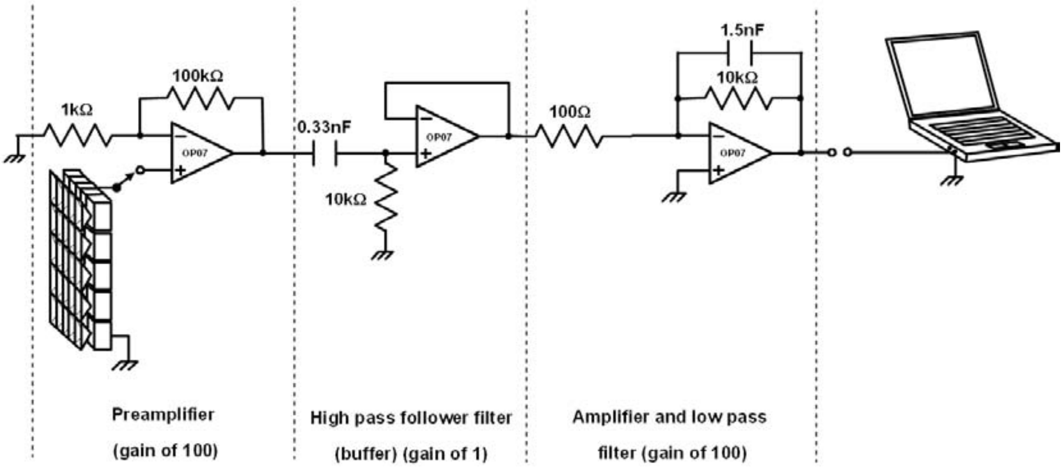


Figure 3.6: Diagram of the preamplifier and bandpass filter circuit

As a pulse propagates trough the atmosphere it undergoes spherical spreading. This results in the backscattered power received by the sodar to be reducing as the inverse square of the range, see Equation (1.5). In order to be able to detect echoes from a range of more than a few meters, it is generally necessary to use some form of preamplifier.

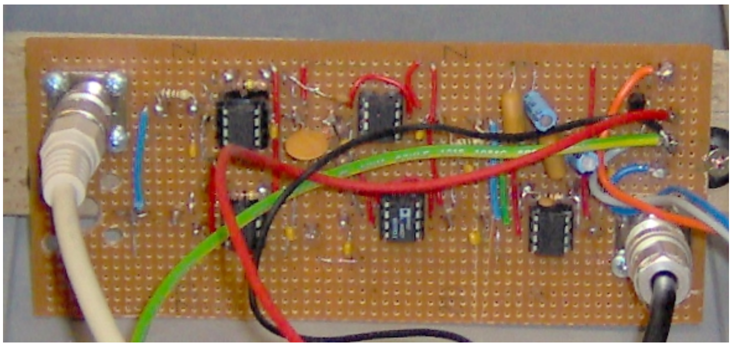


Figure 3.7: Preamplifier and bandpass filter circuit board

Bandpass filters are also generally required to reduce unwanted noise such as drop impacts on the sodar, noise from vehicles, wind, etc. The band width of the filters need to be as wide as the Doppler shift frequency range that would be expected from rain. Assuming no vertical air motion, Equation (1.26) gives a velocity range of about 0 to 9.65 m/s. Substituting this velocity range into Equation (1.28) gives a frequency bandwidth of about

$$\Delta f = \frac{20\, f_T}{c}, \tag{3.1}$$

where f_T is the transmit frequency and c is the speed of sound. The band pass filters should be centered at $f_T + \Delta f/2$. The bandwidth and central frequency for each bandpass filter, therefore, depends on the transmission frequency that they are designed for.

A circuit board has been designed and built for the sodar which contains the preamplifier and a single band pass filter (see Fig. 3.6). The preamplifier does not have any ramped gain. Initial two band pass filters were intended to be used but due to time limitations a single band pass filter was used which had a wide bandwidth (see Fig. 4.26).

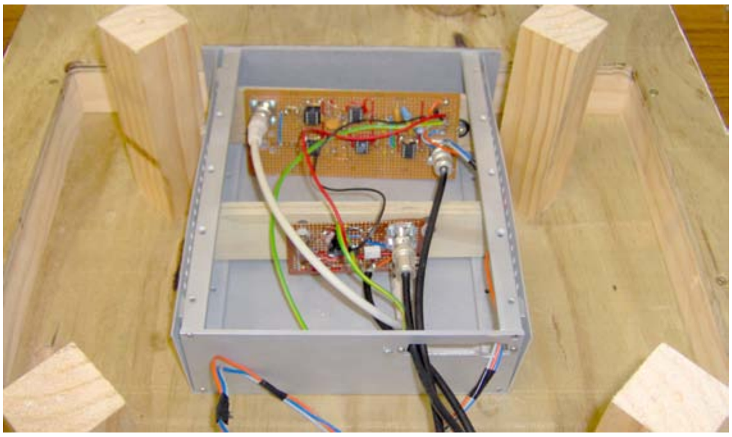


Figure 3.8: Transmit/receive switch and preamplifier/bandpass filter circuit boards installed in the sodar's case behind speaker array

3.5 Sodar Software

3.5.1 GUI Written to Run Software

A GUI (Graphics User Interface) was written using *MatLab* to generate multi-frequency transmission and recording by the sodar. The GUI initially allows the user to select how long the program runs, the transmission frequencies, maximum range, height resolution, sampling frequency, and the frequency range of the FFT (discrete Fast Fourier Transform) that is to be plotted on the GUI axes. After the 'Start' push button is selecting, a panel becomes visible allowing the user to select the recording type (trial, receive only, or transmit/receive), weather condition (no, light, medium, or heavy rain), and the wind condition (no, light, medium, or heavy wind). The temperature, atmospheric pressure, relative humidity, file path, comment, and location may also be entered.

Once the 'Ok' press button is selected, the program starts. For each transmit frequency, a pulse is generated, a recording made and saved, and FFTs plotted. A FFT is plotted for each height resolution interval, with frequency on the x-axes and height + FFT values on the y-axes. This process is repeated for the duration that the program runs.

Pulse Transmission and Recording

The pulses that are generated by the GUI are sine waves with a Gaussian envelop. A pulse is created by making a *MatLab* vector of numbers and then converting them into an analog voltage waveform by using the *MatLab* function `waveplay` and the computer's sound card as an digital to analog converter. The vector of numbers is created by using the equation

$$Pulse = e^{\left(-16\,\tau^{-2}\,[\,t-\tau/2\,]^2\right)}\,\sin(2\,\pi\,f_T\,t), \tag{3.2}$$

where t is the time, τ is the pulse duration, and f_T is the transmit frequency. The time t is a vector ranging from 0 to τ in steps of $1/F_s$, where F_s is the sampling frequency. The pulse duration τ is obtained by rearranging Equation (1.3) and using the height resolution interval entered into the GUI. The resulting vector of numbers is converted into an analog voltage waveform using the *MatLab* function `wavplay`. This function uses the computer's sound card headphone output as a digital to analog converter to produce an analog voltage from a vector of numbers. The resulting voltage waveform is amplified and then converted into an acoustic waveform by the sodar's speaker array.

After the GUI has finished transmitting the pulse, the GUI records for a period of time. The sodar's speaker array converts the acoustic signal into a analog voltage waveform. This waveform is converted into an vector of numbers using the *MatLab* function

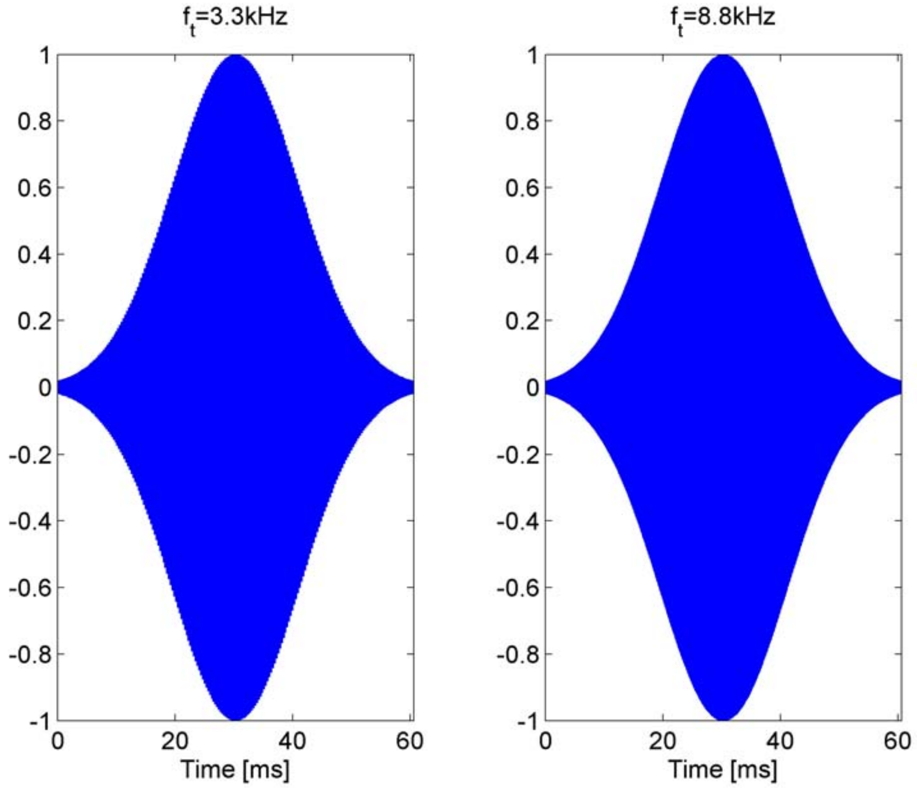


Figure 3.9: Two example of pulses generated by the sodar's GUI

`wavrecord` and the computer's sound card microphone input as an analog to digital converter. The period of time that the GUI records for, after transmitting the pulse, is determined by calculating the expected round trip travel time for the 'maximum range' that was entered into the GUI. A minimum time of 2 seconds is allowed between transmitted pulses to allow all backscattering to die away.

File Creation

A new file is created when the GUI is started. The file is a text file that has a name of the form 'RT_R_YYYYMMDD_hh_mm_pp.txt' where YYYY, MM, DD, hh, and mm stands for the zero padded year, month, day, and minuet when the program was started and pp stands for the selected weather condition (NR, LR, MR, or HR which means no, light, medium, or heavy rain respectively). If the hour changes while the program is running the existing file is closed and a new one is created with a file name that takes into account the time it was created.

Raw Data File Format

The file created by the GUI contains the recordings, in the time domain, made using the sodar. Also included are the parameters used to obtain the recordings and the observed weather and atmospheric conditions. The format used to write the values to file is as *MatLab* variables or vectors.

The first line of the raw data file contains the information on the transmit/receive parameter and the the observed atmospheric conditions. This information is the recording type, the observed rain and wind conditions when program was started, if the program was continued from a previous hour, program start time and date, program run time, maximum range, height resolution, pulse duration, transmit frequencies, sampling frequency, observed temperature, atmospheric pressure, and relative humidity, measurement location, and comment. Two blank lines follow.

For each pulse, there is a line containing the transmission time and transmit frequency. The next line contains a short recording made prior to transmission for noise analysis or noise reduction purposes. Following this are several lines contains the recording that followed the transmission of the pulse. Each of these lines contain the recording corresponding to a height resolution interval. These are formatted as a vector of the form ‘M_hnnn = [...];’, where ‘nnn’is a zero padded height which is located at the middle of the relevant height resolution interval. A blank line is place between each pulse segment. A truncated example file is given in Table 3.1, where ‘ ... ’ represents omitted data.

Table 3.1: Truncated Example File of Recording by GUI

```
'SODAR_RAW_DATA:Transmit_Receive,Light Rain,Medium Wind.';Hour=1;RunTime=3.000;MaxRange=200;HeightRes=20;r=[10,30,50,70,90,110, ... ];


T='070620155052'; f_T=1.9;
Noise =[+0.009338 +0.018005 +0.019775 +0.010620 -0.002686 -0.007751 +0.004517 +0.032715 +0.064026 +0.083252 +0.082520 +0.065735 ... ];
M_h010=[-0.068237 -0.090820 -0.110535 -0.120667 -0.122437 -0.121033 -0.117554 -0.106262 -0.078735 -0.032227 +0.024841 +0.075989 ... ];
M_h030=[+0.026733 +0.151733 +0.228760 +0.244568 +0.219421 +0.190552 +0.187500 +0.212097 +0.236694 +0.223572 +0.149597 +0.025513 ... ];
M_h050=[+0.046021 +0.065857 +0.077209 +0.075317 +0.059814 +0.035278 +0.008972 -0.012939 -0.027832 -0.036682 -0.043152 -0.049316 ... ];
M_h070=[-0.019348 -0.021729 -0.026489 -0.028503 -0.022766 -0.007996 +0.011292 +0.027405 +0.032898 +0.024963 +0.005920 -0.017578 ... ];
M_h090=[-0.004944 -0.033691 -0.057251 -0.058228 -0.027283 +0.029358 +0.091064 +0.132385 +0.136292 +0.102539 +0.046204 -0.010803 ... ];
M_h110=[-0.028992 -0.031677 -0.024902 -0.015259 -0.008606 -0.005493 -0.001831 +0.006653 +0.020813 +0.036377 +0.046997 +0.048584 ... ];
M_h130=[-0.006531 +0.004211 +0.011108 +0.011475 +0.007202 +0.003296 +0.004578 +0.012878 +0.025879 +0.039246 +0.049377 +0.054871 ... ];
M_h150=[+0.001526 -0.007202 -0.018433 -0.026855 -0.028015 -0.021667 -0.012329 -0.005554 -0.003967 -0.005066 -0.003906 +0.003174 ... ];
M_h170=[+0.149109 +0.163574 +0.169861 +0.169800 +0.157104 +0.121033 +0.056702 -0.025269 -0.098999 -0.135315 -0.118652 -0.057068 ... ];
M_h190=[-0.045227 -0.047302 -0.043701 -0.037170 -0.032837 -0.033813 -0.037964 -0.039490 -0.033447 -0.020813 -0.008240 -0.003845 ... ];

T='070620155054'; f_T=2.0;
Noise =[+0.012817 +0.028076 +0.031372 +0.025146 +0.017273 +0.015930 +0.024170 +0.038574 +0.051331 +0.056335 +0.052734 +0.045044 ... ];
M_h010=[+0.028748 +0.079895 +0.115906 +0.116516 +0.080627 +0.025574 -0.025452 -0.056335 -0.063477 -0.053833 -0.037415 -0.021545 ... ];
                                                               ...
```

3.6 Doppler Rain Radar used to Evaluate Sodar Rain Measurements

The Metek Micro Rain Radar MRR-2 is used to test if the sodar and the algorithm developed in Section 2.2 give realistic results. This is done by running the MMR simultaneously with multi-frequency sounding by the sodar. The drop size distribution obtained by the MRR is then compared with that derived algorithm.

The MMR-2 is a continuous wave 24 GHz electromagnetic radar that obtains profiles of the drop size distribution. From this, it calculates rain rates liquid water content, and characteristic fall velocities. It has a height resolution of 10 to 100 m and can make measurements from as low as 20 m up to 1000 /m1 above the ground.

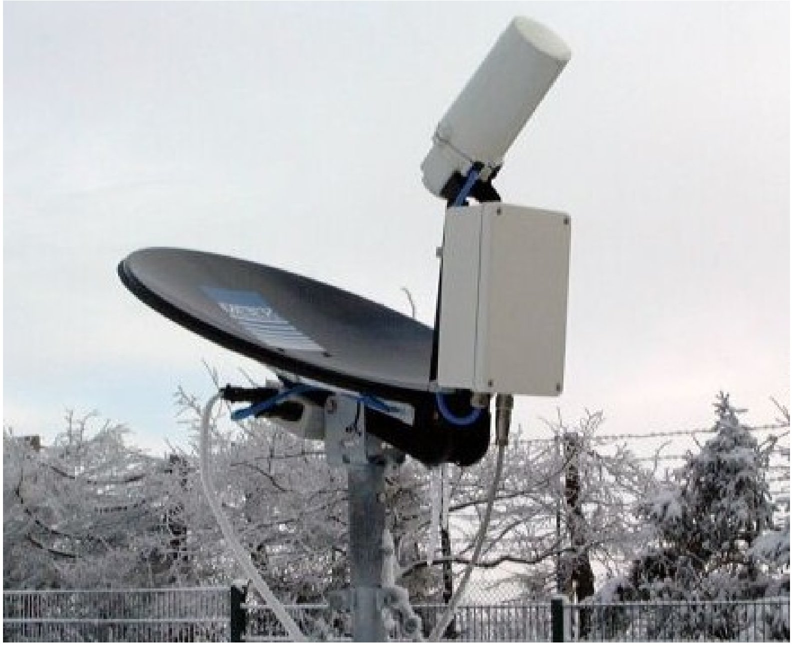


Figure 3.10: Doppler rain radar used in the evaluation of the sodarand fitting algorithm's ability to separate the rain component of the backscattered power spectra (Metek)

4

Laboratory Evaluation of the Sodar's System Performance

4.1 Speaker's Measured Characteristics

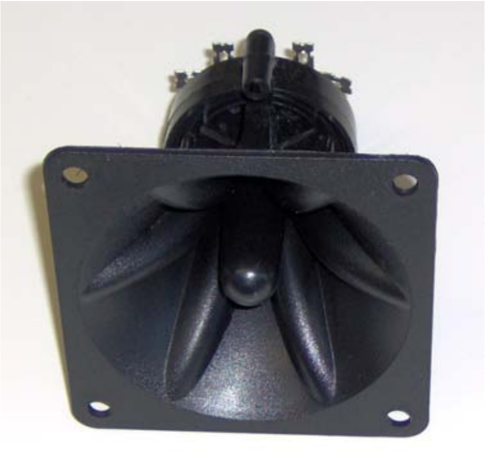


Figure 4.1: Motorola KSN1005 tweeter

The speakers used in the sodar, which is described in Chapter 3, are Motorola KSN1005 tweeters. They contain a piezoelectric crystal which changes shape if a voltage is applied across the crystal. Alternatively, if the crystal is compressed, a voltage difference is produced across the crystal. These characteristics allow the KSN1005 to be used either as a speaker or as a microphone. To our knowledge, there were no manufactures specifications or previous detailed studies on the characteristics of these speakers, though some limited studies had been carried out by the University of Auckland's Physics Department. As these speakers were also intended to be used for other sodars by the department, it was desirable that their characteristics be investigated. The key characteristics of the speakers which were investigated are the equivalent circuit of the speakers while transmitting and receiving, and the speakers' variation of gain with frequency and angle.

4.1.1 Equivalent Circuit of the Speakers

The KSN1005 speakers have resistive and capacitive characteristics. They are composed of two terminals with a piezoelectric crystal in between. This is similar to a parallel plate capacitor, as the crystal has a relatively high resistance. When a voltage is applied across the speaker, the speaker dissipates power, both in the form of sound and through internal resistance. The total power loss may be modeled as if it had been due to a an effective resistor. The speakers will also have a phase delay between the input voltage and the resulting sound, due to the inertia of the speakers and the internal capacitance. This phase delay can be modeled as if it were due to an effective capacitor. The speakers are, therefore, assumed to be able to be modeled as an capacitor and resistor in parallel, see Fig. 4.2.

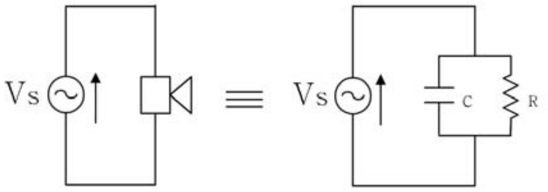
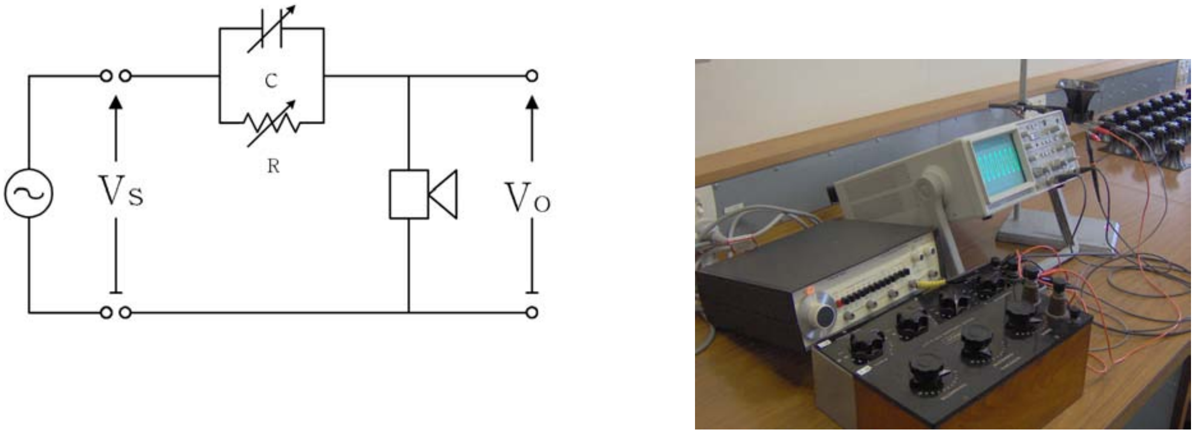


Figure 4.2: Assumed equivalent circuit of a speakers

Equivalent Circuit of the Speakers while Transmitting

The capacitance and resistance values of the speaker's equivalent circuit, while transmitting or acting as a speaker, were estimated by connecting the speaker in series with a variable capacitor and resistor, which were in parallel (see Fig. 4.3a). A continuous wave voltage, from a signal generator, was then applied across the circuit. The variable capacitor and resistor were adjusted until the voltage across the speaker was half of the input voltage and in phase with it. This was achieved by using an oscilloscope to set the voltage V_o to half of V_i. When both waveforms on the oscilloscope were in phase, it was estimated that the capacitance and resistance of the speaker were matched and were equal to that of the variable capacitor and resistor. These measurements were made for a range of frequencies for all 25 speakers that were to be used on the sodar.



(a) Circuit Diagram

(b) Experimental Setup

Figure 4.3: Experimental setup used to determine the equivalent circuit of the speakers while transmitting.

The measured capacitance and resistance values of the speakers' equivalent circuit, while transmitting, are graphed in Fig. 4.4, for a range of frequencies. This shows capacitance and resistance values of around 0.2 μF and 1 kΩ respectively. For a range of frequencies, particularly between 5.7kHz to 7.7kHz, it was found that increasing the variable resistance above an optimal value of about 1 to 2 kΩ did not appear to have a significant effect in moving the two voltage waveforms away from a matched condition. This is represented graphically in Fig. 4.4 as error bars. Could this indicate that the model used for the speakers is inadequate? It is interesting to note that this 5.7 kHz to 7.7 kHz frequency range mentioned above is in the frequency range where the speakers have a very poor gain, see Fig. 4.8.

Equivalent Circuit of the Speakers While Receiving

The capacitance and resistance values of a speaker's equivalent circuit, while receiving or acting as a microphone, was estimated by using a sound source and making measurements of the voltage and phase across the speaker, first with no load connecting across its terminals and then with a load (see Fig. 4.5a). A signal generator was used to supply a

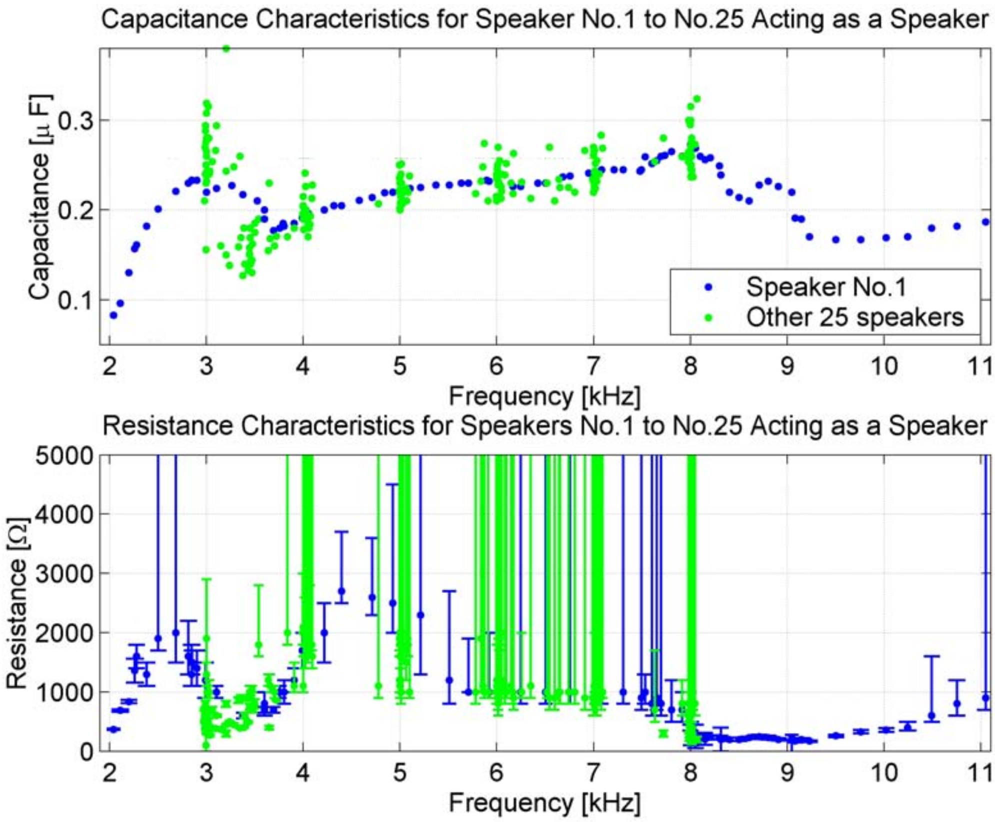
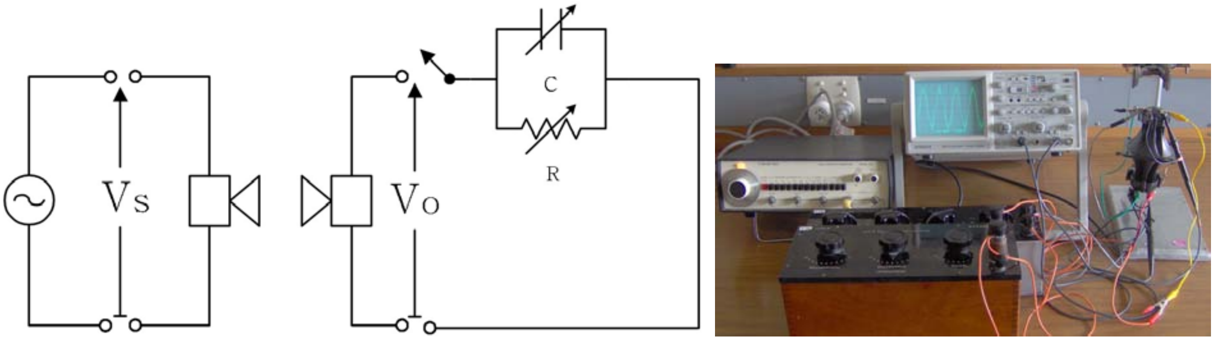


Figure 4.4: Capacitance and resistance characteristics of Speakers No.1 to No.25, while transmitting.

continuous sine wave voltage to a speaker which was to be used as a sound source. The speaker, whose characteristics as a microphone were to be measured, was place directly on top of the sound source speaker. An oscilloscope was used to observe the voltage, Vo, and phase across the speaker's terminals. A variable capacitor and resistor in parallel were then connected across the speaker's terminals. The variable capacitor and resistor were then varied until the voltage across the speaker's terminals was a half what it had been in open circuit conditions and in phase. At this point, it was estimated that the capacitance and resistance of the speaker were matched and were equal to that of the variable capacitor and resistor. These measurements were not able to be made as accurately as those made for the transmit equivalent circuit, where two wave forms were able to be observed spontaneously on the oscilloscope to aid in achieving a match. These measurements were made for a range of frequencies for 5 of the speakers that were to be used in the sodar. The measured capacitance and resistance values of the speakers' equivalent circuit, while receiving, are graphed in Fig. 4.4, for a range of frequencies. Results seem to be relatively similar to those obtained for the transmission equivalent circuit of the speaker, see Fig. 4.4.



(a) Circuit Diagram (b) Experimental Setup

Figure 4.5: Experimental setup used to determine the equivalent circuit of the speakers while receiving

4.1.2 Variation of the Speakers' Gain with Frequency

The speaker's variation of gain with frequency, or transfer function, was measured by applying a voltage across a speaker's terminals and measuring the voltage produced across another speaker. A signal generator was used to supply a continuous sine wave voltage to a speaker, which was used as a sound source. The speaker, whose transfer function was to be measured, was placed directly on top of the sound source speaker, see Fig. 4.1.2. An oscilloscope was used to observe the voltage across the speaker's terminals and the input voltage, for a range of frequencies. The transfer functions of all 25 speakers, which were to be used in the sodar, are graphed in Fig. 4.8. From the data plotted, it would appear that frequencies around 3.3 kHz and about 8 kHz have relatively high gains in comparison to the frequency range from about 4.5 kHz to 7.5 kHz.

An interesting phenomena that can be seen in Fig. 4.8, is that the data obtained using the continuous wave voltage has strong resonances or spikes and troughs. This was thought to be due to standing waves. To investigate this issue, pulses were used for one of the speakers. These pulses were made short enough, 3 ms, so that reflections of the pulse from the bench top, walls, etc. would not obscure the direct path arrival. A delay of a second was allowed between pulses to enable reverberations to die away. These pulses were generated by `RT_Pulses.m`, see *MatLab* Code A.9, using a laptop's sound card. They were composed of a sine wave with a Gaussian envelope, see Equation (3.2). Measurements were made on the oscilloscope of the amplitudes of the transmitted and received pulses. The transfer function obtained using the pulses showed a similar shape but it did not show the resonances seen in the continuous wave measurements (see Fig. 4.8). It was concluded that these resonances were possibly due to standing waves developed due to reflections from the wall, table, etc. All further measurements were made using pulses.

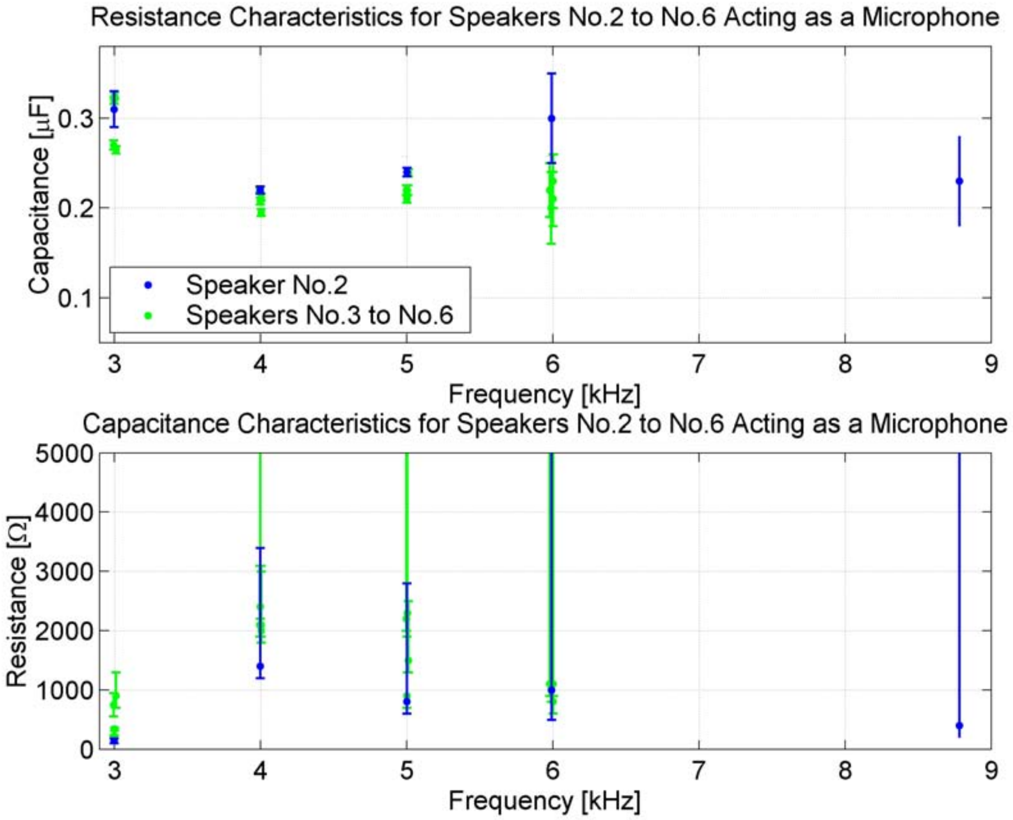
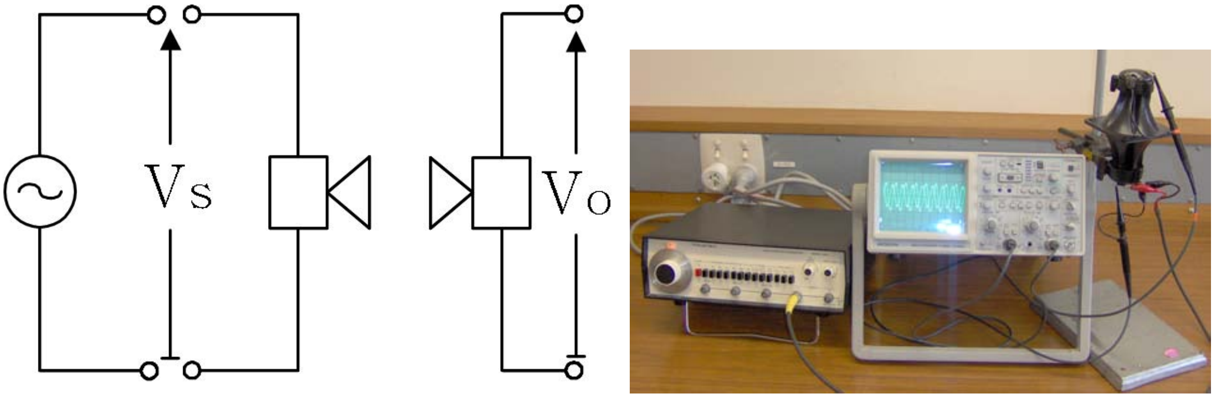


Figure 4.6: Capacitance and resistance characteristics of Speakers No.2 to No.6 while receiving.

4.1.3 Variation of the Speakers' Gain with Angle

The speakers' variation of gain with angle was measured by using one speaker as a sound source and measuring the voltage produced across another speaker for a range of angles, keeping the distance between the speakers constant. A board was marked out with a 600 mm radius semicircle and angles from 0 to 90 degrees. A speaker, which was used as a sound source, was positioned directly over the origin of the semicircle. This speaker was driven by pulses generated by *MatLab* Code A.9, via a laptop's sound card. The speaker, whose angular characteristics were to be measured, was used as a microphone. This speaker was placed, facing the sound source speaker, directly above the semicircle at a range of angles. To ensure accuracy, pointers were clamped to the stands on which the speakers were mounted so as to indicate a point directly below the front of each speaker (see Fig. 4.9). An oscilloscope was used to observe the voltage across both speakers' terminals for a range of angles and frequencies. The gain of a speaker as a function of angle is given in Fig. 4.10 for five frequencies. This shows that the speaker's gain falls off with increased angle from 0 to 90 degrees. The 8.8 kHz curve seemed to fall off at twice the rate to the other lower frequency curves.



(a) Circuit Diagram

(b) Experimental Setup

Figure 4.7: Experimental setup used to measure the transfer function of the speakers

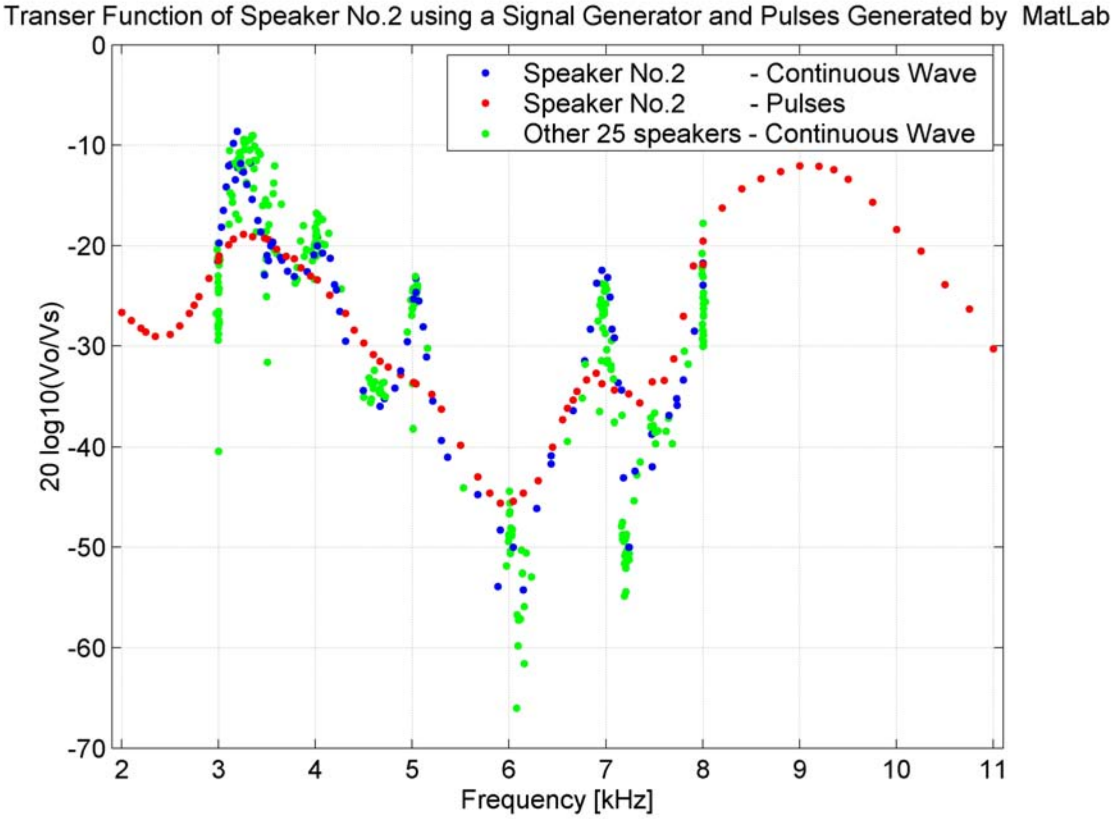
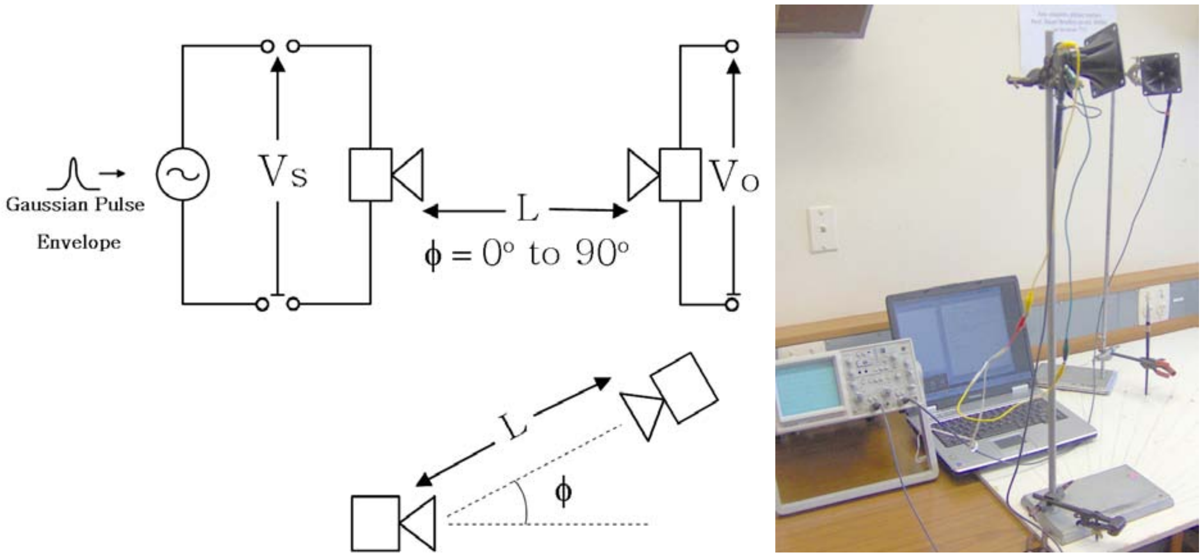


Figure 4.8: Transfer function of Speakers No.1 to No.25 measured individually using a signal generator and MatLab generated pulses.



(a) Circuit Diagram (b) Experimental Setup

Figure 4.9: Experimental setup used to measure the speaker's variation of gain with angle. Speaker separation, L, was 600mm. (Part 6)

]

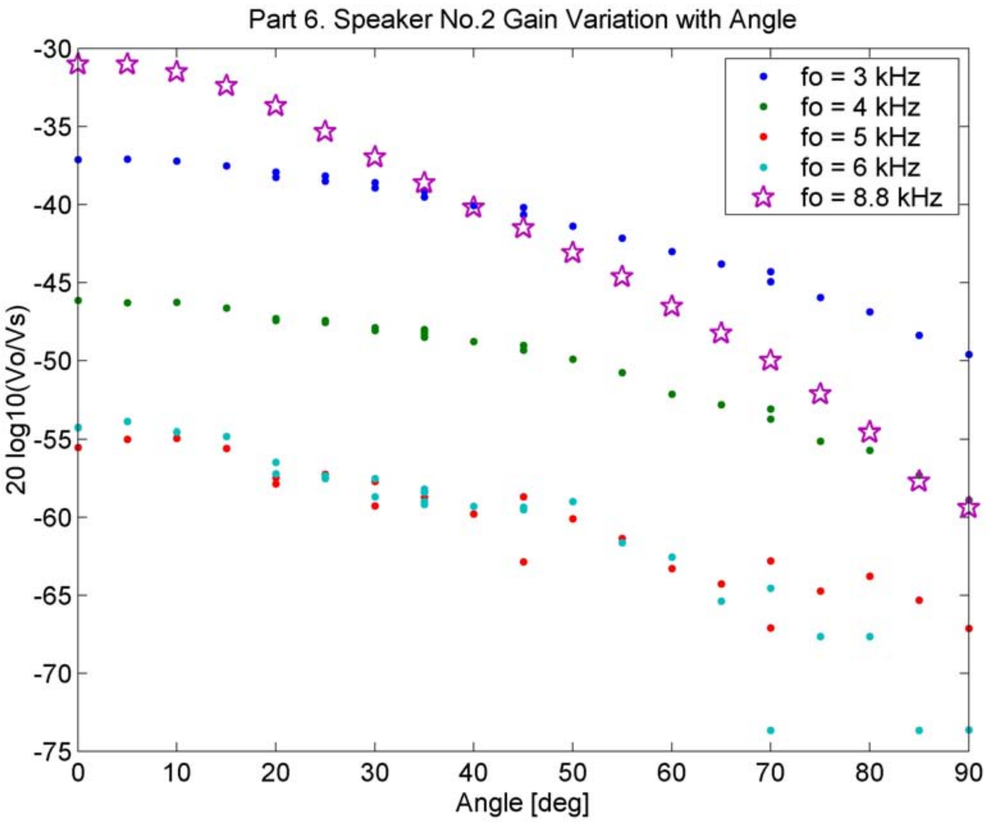


Figure 4.10: A speakers' variation of gain with angle

4.2 Speaker Array Measured Characteristics

The sodar's speaker array was created using the 25 speakers whose characteristics had been measured in Section 4.1. The speaker array consisted of five rows and five columns of speakers wired in parallel, see Fig. 4.11. The speakers which appeared to have poorer gain characteristics, were placed towards the edges of the array, while that with the best characteristics was placed at its center. It was hoped that this would reduce any diffraction effects. Measurements were then made to determine the transfer function and variation of gain with angle of the speaker array.

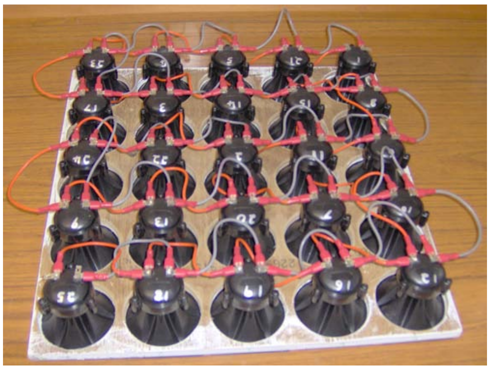
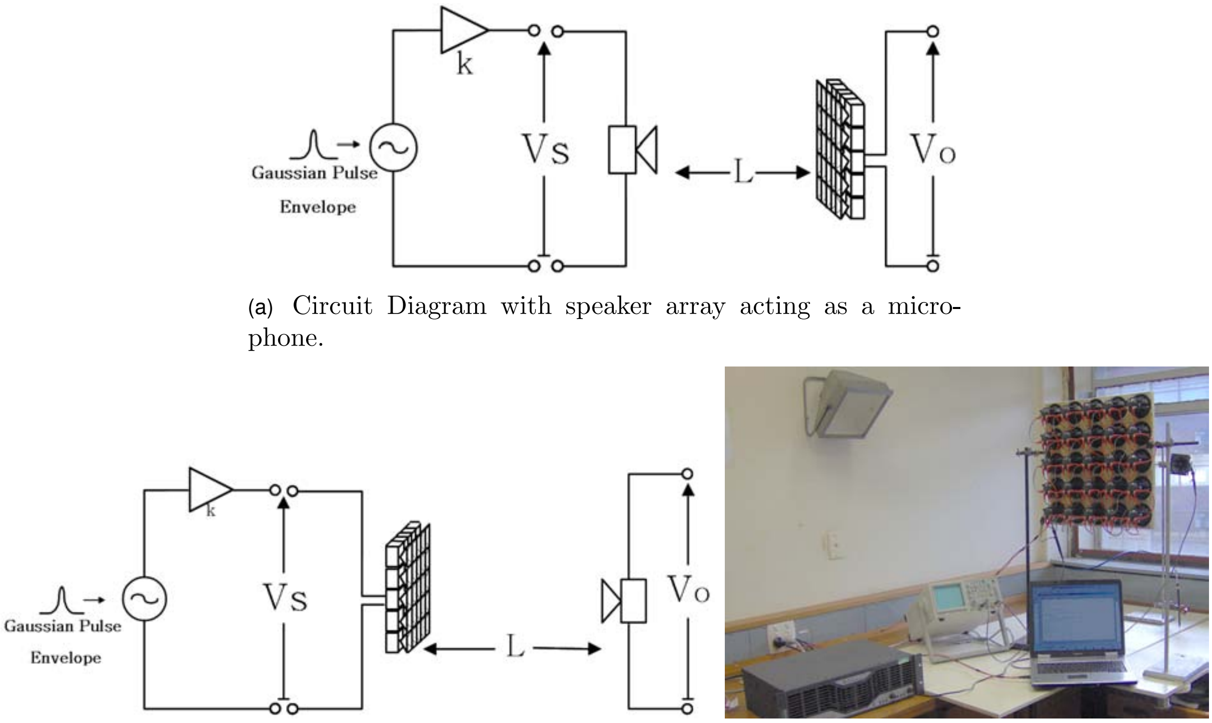


Figure 4.11: Speaker array

4.2.1 Transfer Function of the Speaker Array while Transmitting and Receiving

The transfer function of the speaker array, while transmitting, was obtained by using the speaker array as a sound source and measuring the voltage produce across a speaker placed opposite the speaker array. The speaker array and speaker were mounted facing each other on to stands 600 mm apart. The speaker array was driven by pulses generated by *MatLab* Code A.9, via a laptop's sound card and a power amplifier. The voltage across the speaker array and the speaker was observed for a range of frequencies. To determine transfer function of the speaker array, while receiving, the same process was repeated except that the speaker was used as the sound source (see Fig. 4.12). The transfer functions of the speaker array, while transmitting and receiving, is graphed in Fig. 4.13. This graph indicates that there are two peaks in the speaker array's gain at about 3.2 kHz and 8.9 kHz with a region of poor gain on either side of the peaks.

The apparent poor gain on either side of the speaker array gain peaks would seem to limit the optimal transmission frequencies of the sodar to frequencies around 3.2 kHz



(a) Circuit Diagram with speaker array acting as a microphone.

(b) Circuit Diagram with speaker array acting as a speaker.

(c) Experimental Setup

Figure 4.12: Experimental setup used to measure the transfer function of the speaker array, while transmitting and receiving.

and 8.9 kHz. This could be a problem if the frequency range around one of these peaks proved to be unsuitable and the frequency separation between transmission frequencies around the other peak was insufficient to enable a good least squares fitting of the rain and turbulence components of the backscattered power spectra.

4.2.2 Speaker Array's Variation of Gain with Angle

The speaker array's variation of gain with angle was investigated to see if the array's gain falls off sufficiently with increased angle, and to see if there are any undesirable secondary 'lobes' present. The speaker array's variation of gain with angle was measured by using a speaker as a sound source and measuring the voltage produced across the array for a range of angles. To achieve this, a turntable was made to mount the speaker array on. A pointer was added so that the angles on the board, used in Section 4.1.3, could be measured more accurately. A speaker, which was to be used as a microphone, was placed 2.5 m from the speaker array. The speaker array was driven by pulses generated by *MatLab* Code A.9, via a laptop's sound card and a power amplifier. An oscilloscope was used to observe the voltage across the speaker array and the speaker at 3.3 kHz and 8.8 kHz for a range of angles from 0 to 90 degrees (see Fig. 4.14).

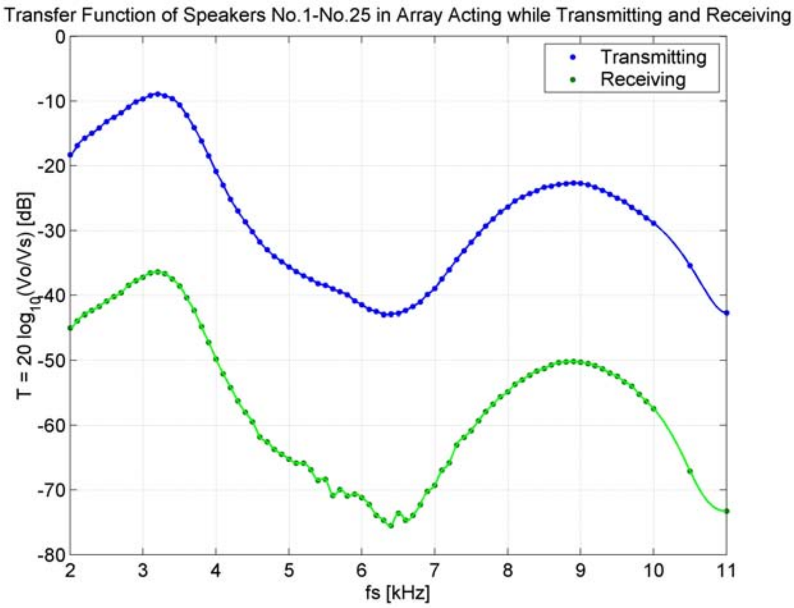
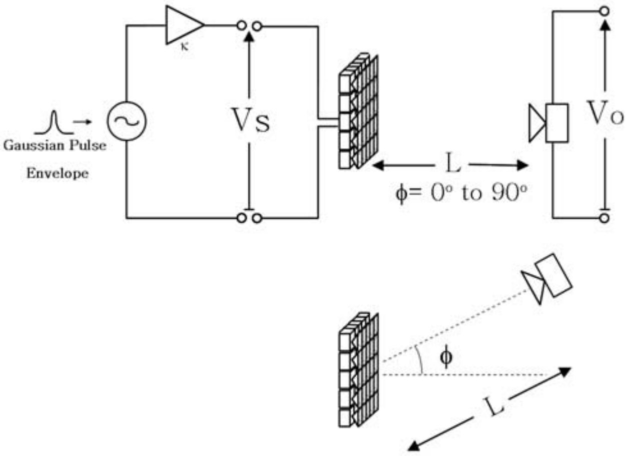
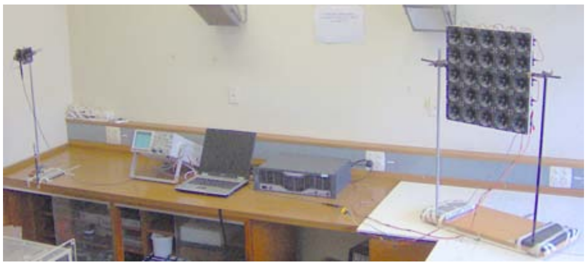


Figure 4.13: Transfer function of speaker array while transmitting and receiving.

The speaker array's measured variation of gain with angle for two frequencies is graphed in Fig. 4.15. The gain measured at 3.3 kHz shows an acceptable drop in gain with an increase in angle from zero and does not have any significant secondary lobes. The measured gain for 8.8 kHz, however, does seem to show some undesirable secondary lobes that could be of significants in transmission from the sodar. Section 4.3.1 measures the angular gain of the sodar, with the speaker array installed in it, to see what effect these side lobes have on the sodar.



(a) Circuit Diagram



(b) Experimental Setup

Figure 4.14: Experimental setup used to measure the speaker array's variation of gain with angle

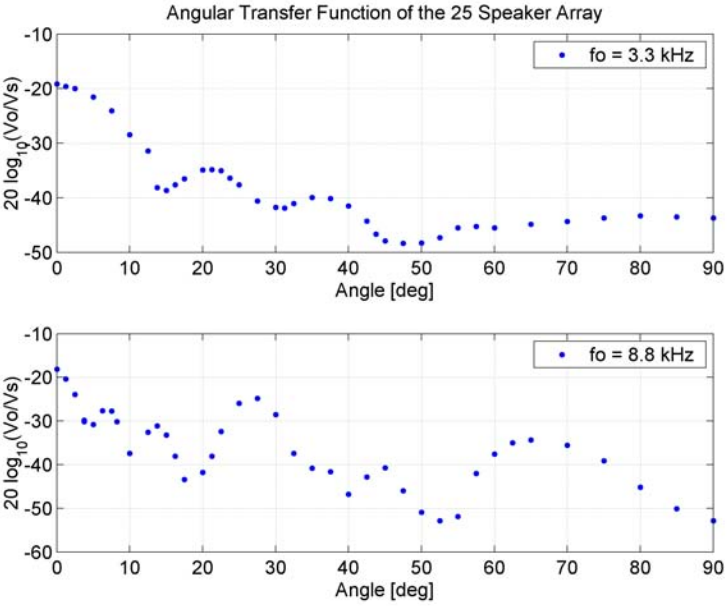


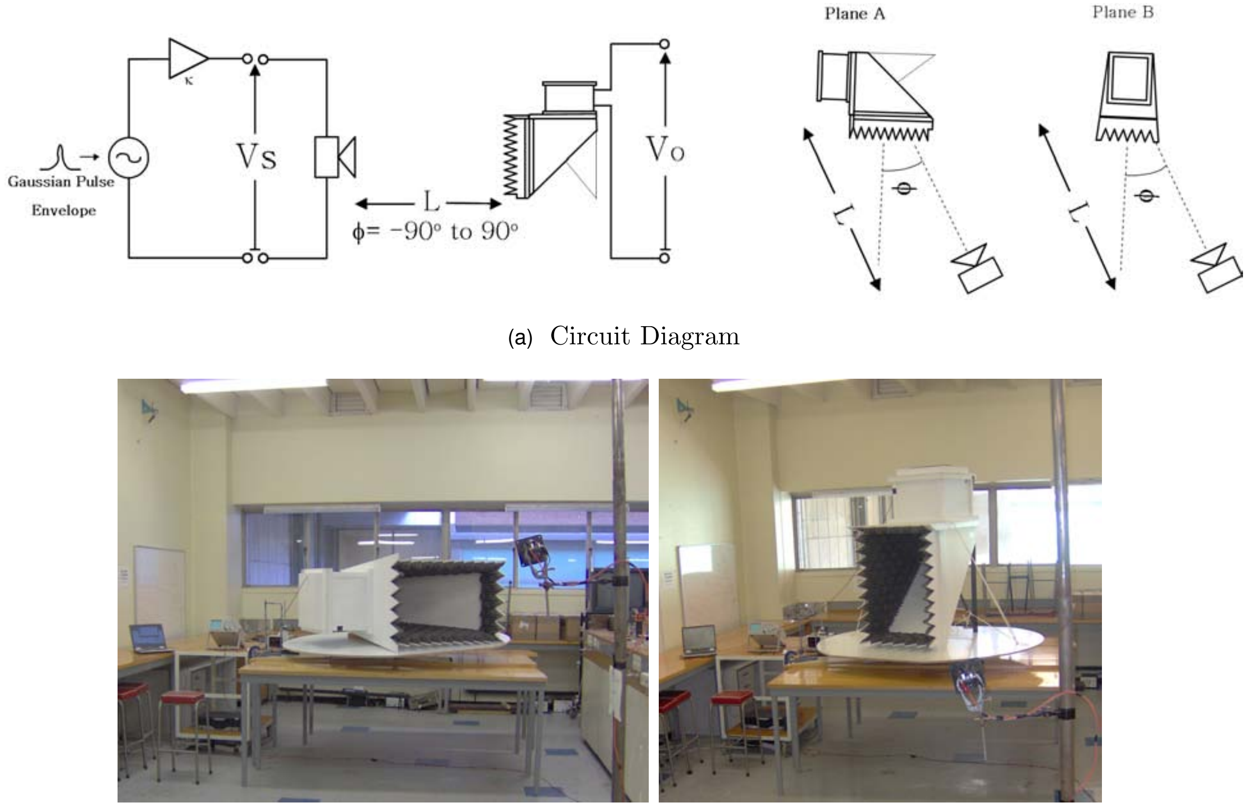
Figure 4.15: Speaker array's variation of gain with angle

4.3 Sodar's Measured Characteristics

4.3.1 Sodar's Variation of Gain with Angle

The sodar's variation of gain was studied, through two perpendicular planes, to determine if the sodar's gain falls off sufficiently with increased angle and to see if there are any undesirable secondary 'lobes' present. This variation of gain was measured by using a speaker as a sound source and measuring the voltage produced across the speaker array, which was mounted in the sodar, for a range of angles. In order to measure the angle between the sodar and the speaker, it was thought best to place the sodar on a turntable. A turntable had already been developed by the Physics Department workshop which tilted to enable the sodar to be leveled. The workshop modified this turntable by making and fitting a new stand and a larger circular top. Angles were then marked out on the board. Due to the tilt and shape of the turntable top, a solid pointer did not seem to be the optimal way to mark out the angles. Instead, a laser pointer, adapted to run off a DC power supply, was mounted so as to shine vertically down on the turntable and act as a pointer. See Fig. 4.17. The turntable and laser pointer were set up on a table and the the sodar was placed on it and leveled. A speaker was clamped to a stand five meters from the center of the sodar. *MatLab* Code A.9 was used to generate pulses, via a laptop's sound card. The pulses were 3.3kHz and 8.8kHz with a 3ms long Gaussian envelope and a second delay between pulses. These pulses were amplified and used to drive the speaker. The sodar was used as a microphone while the the angle between the sodar and the speaker was varied. Measurements of the amplitudes of the transmitted and received pulses were made on an oscilloscope in two planes though the sodar as indicated in Fig. 4.16a. To ensure that the direct path arrivals to the sodar were not confused with reflections from the walls, roof, etc., the arrival times of the pulses were measured.

The sodar's measured variation of gain with angle in two planes is graphed in Fig. 4.19. These measurements seem to indicate an acceptable fall in gain with increase of angle from a normal to the sodar. They also seems to indicate that there are no major side lobes present. An interesting phenomena can be seen in the results for plane A for angles from about 10 to 80 degrees. Within this angular range, the pulses either arrived earlier and with higher gain than was expected or the pulses arrived as two distinct pulses with the first of these two pulses arriving earlier and with a higher gain than expected. It was concluded that this was due to the sodar construction. Normally the pulses were reflected off the sodar reflector before reaching the speaker array. For plane A, however, in the range of angles mentioned above, direct arrivals appear to be able to reach the speaker array without first being reflected off the sodar reflector. This can be seen in Fig. 4.18 where the speaker array is able to be seen above the edge of the baffle. How much of an



(a) Circuit Diagram

(b) Experimental Setup - Plane A

(c) Experimental Setup - Plane B

Figure 4.16: Experimental setup used to measure the sodar's variation of gain with angle through two perpendicular planes.

effect this will have on the measurement made by this sodar is unknown. It is possibly a factor to be considered in any future designs of the sodar.

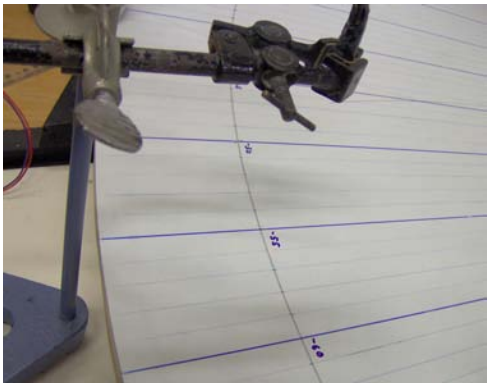


Figure 4.17: Laser pointer setup to measure the angle between the sodar and the speaker.

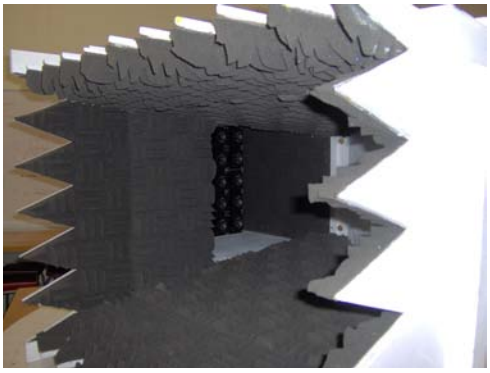


Figure 4.18: View of sodar's interior with speaker array visible at rear. This photo shows that, for a range of angles, sound may reach the speaker array by a direct path without being reflection from the reflector.

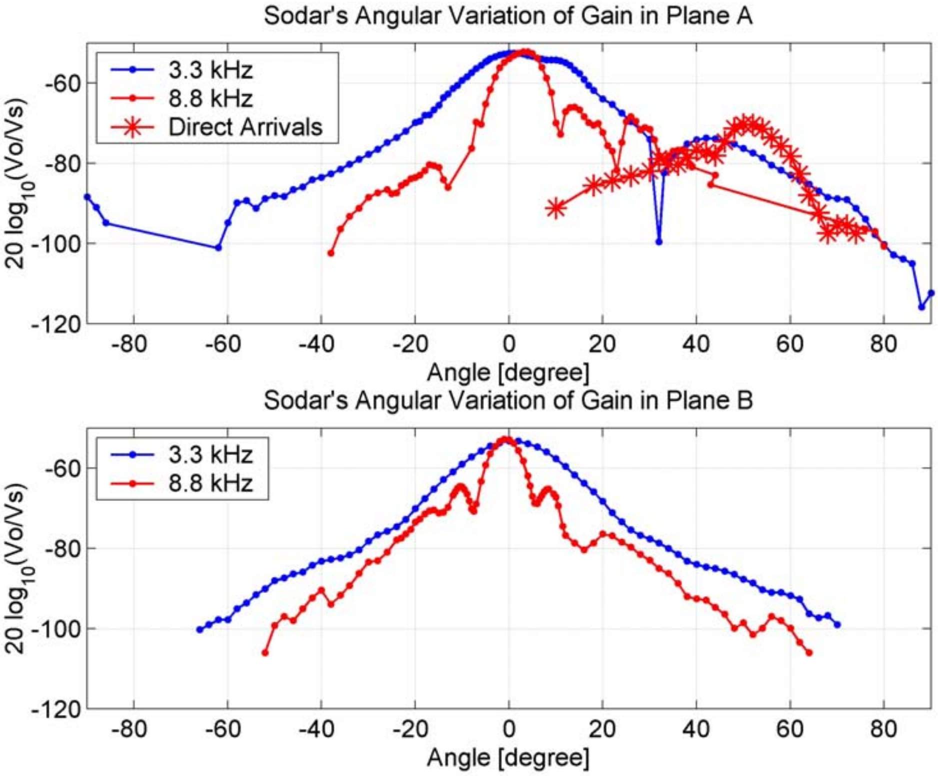


Figure 4.19: The sodar's variation of gain with angle, in two perpendicular planes, while receiving. The legend ”Direct Arrivals”, for plane A, refers to arrivals that appear to have been due to pulse arrivals directly to the sodar's speaker array instead of arrivals which were reflected from the sodar's reflector.

4.3.2 Transfer Function of the Computer's Sound Card Headphone Output and Power Amplifier

The transfer function of the output of the computer's sound card headphone output and power amplifier while loaded by the speaker array was measured. *MatLab* Code A.9 was used to generate pulses via the laptop's sound card headphone plug. These pulses had a duration of 60.6 ms duration and a Gaussian envelope. A power amplifier was used to amplify these pulses. The voltage output from the amplifier was observed on the oscilloscope for a range of frequencies with the amplifier unloaded and then loaded by the speaker array. See Fig. 4.20.

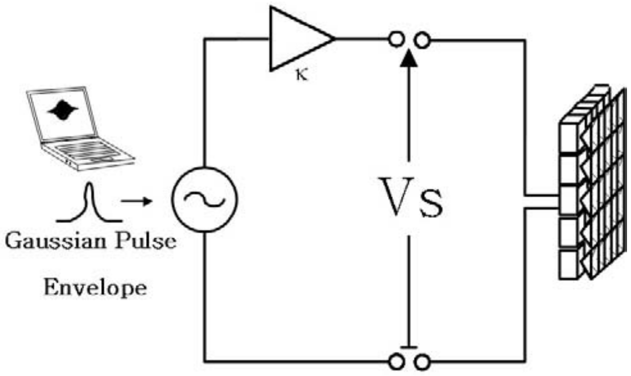


Figure 4.20: Circuit diagram of experimental setup used to measure the transfer function of the power amplifier, while loaded with the speaker array and amplifying the output computer's sound card headphone plug.

The voltage across the the amplifier, with no load connected across it, varied relatively negligibly variation with change of frequency. The voltage across the amplifier, with the speaker array connected across, however, increased as the frequency was increased. See Fig. 4.21. It was concluded that this was possibly due to the power amplifier struggling to drive the capacitance component of the speakers in the array. A cubic spline has been used to fit the data in this graph.

4.3.3 Transfer Function of the Microphone Input of Computer's Sound Card

The transfer function of the microphone input of the computer's sound card was measured. A signal generator was used to generate a sinusoidal waveform. This voltage was sampled via the microphone input of the computer's sound card using the *MatLab* function `wavrecord`, at a rate of 44,100 samples per second. See Fig. 4.22. The voltage amplitude of the signal generator was then varied through the frequency range to find

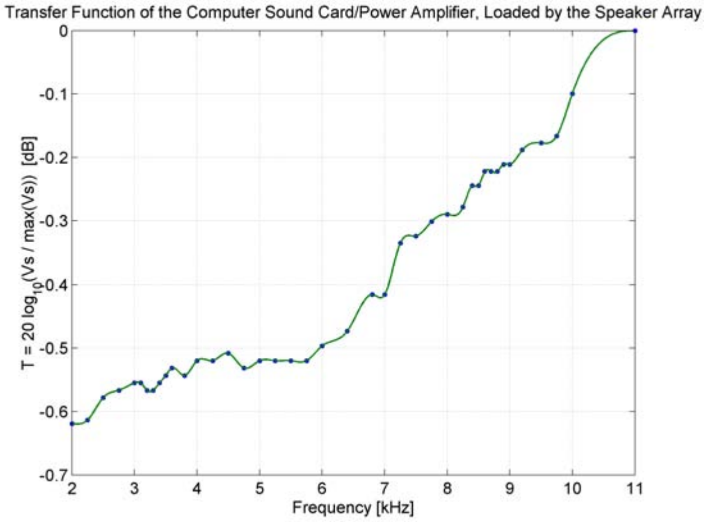


Figure 4.21: Measured transfer function of the power amplifier in series with computer sound card headphone plug while loaded by the speaker array.

a voltage level that would not cause clipping by the sound card. A 100 mV$_{pp}$ voltage amplitude was decided on. The amplitude of the signal generator voltage was then kept constant, while the frequency was varied. For each frequency that measurements were made, a hundred short recordings were then made. The maximum values were then selected from the log of the absolute values of the fast Fourier transform of these recordings. The corresponding frequencies were then calculated. An average and standard deviation were made of these maximum values and the corresponding frequencies. This process was repeated for a range of frequencies from 2 to 11 kHz.

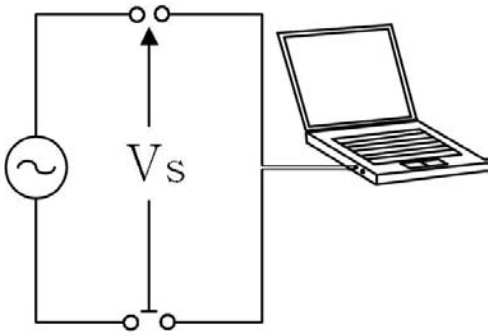


Figure 4.22: Circuit diagram of experimental setup used to measure the transfer function of the microphone input of the computer's sound card.

The laptop that was used for most of this work was a *Toshiba Satellite Pro L20*. The transfer function of the microphone input of this computer's sound card is given in Fig. 4.23. This graph seems to indicate a relatively flat gain over the frequency range that measurements were made. This indicates that, for this computer, corrections for variations of gain of the microphone input of the sound card with frequency is unnecessary. It was observed that the maximum input voltage that could be applied across the microphone input of this laptop's microphone input before clipping occurred was about 100 mV.

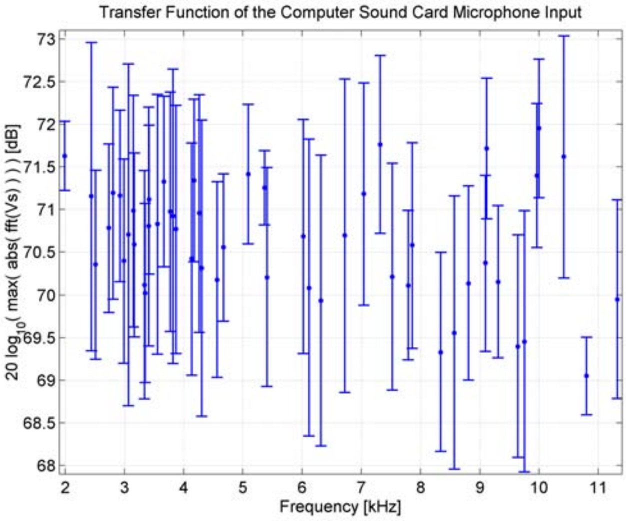


Figure 4.23: Measured transfer function of the microphone input of the *Toshiba Satellite Pro L20* laptop's sound card.

During field trials, it was found that the *Toshiba Satellite Pro L20* was not able to be used with the preamplifier circuit. Instead a *Toshiba Tecra M1* laptop was used. While this computer's sound card appeared to be able to cope with the sodar's preamplifier, its microphone input appeared to have a poor gain for frequencies greater then about 7 kHz. The characteristics of this sound card was measured and it was observed that its gain was flat up to about 7.1 kHz where it dropped off steeply. Frequencies beyond about 7.4 kHz were not able to be detected (see Fig. 4.24). This indicates that this laptop is unsuitable for running the sodar if higher frequencies such as around 8.8 kHz is desired to be used.

4.3.4 Preamplifier and Bandpass Filter Transfer Function

The gain of the sodar's preamplifier and bandpass filter circuit was measured by applying a voltage V_s across the input of the circuit and measuring the output voltage V_o (see Fig. 4.25). The input voltage was generated using a signal generator and voltage divider. The voltage divider was used to drop the signal generator voltage to about a millivolt to ensure that the circuit did not saturate. A oscilloscope was used to measure the input and output voltages for a range of frequencies. The measured gain of the circuit is graphed in Fig. 4.26. The maximum gain of the circuit is around 2.8 kHz.

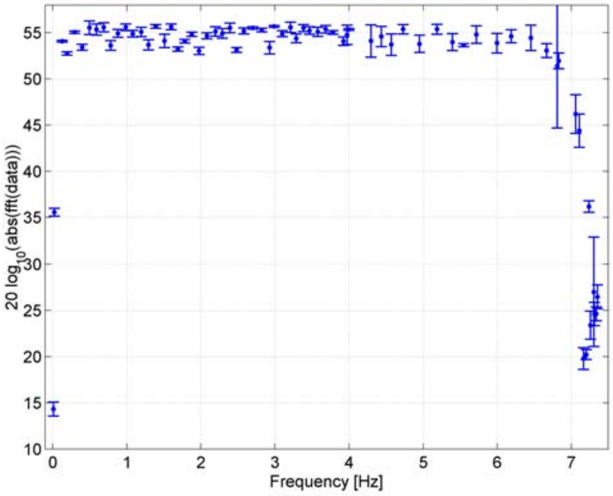


Figure 4.24: Measured transfer function of the microphone input of the *Toshiba Tecra M1* laptop's sound card.

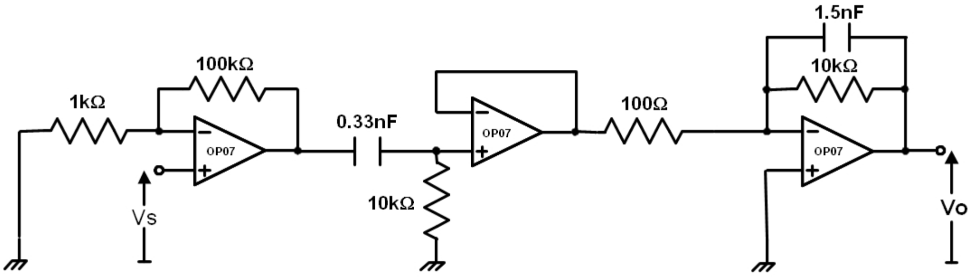


Figure 4.25: Circuit diagram of the sodar's preamplifier and bandpass filter

4.3.5 Transmission and Receiving Gain of the Sodar $G\,A_e$

The transmission antenna gain G and the receiving effective area A_e represent the sodar's efficiency at transmitting and receiving respectively. The transmission antenna gain of the sodar is assumed to be the product of the transmission gain of sodar's speaker array and that of the power amplifier while loaded by the speaker array and amplifying the computer sound card headphone output (see Fig. 4.13) and 4.21. The receiving gain of the sodar is assumed to be the receiving gain of the sodar's speaker array while receiving, the gain of the computer's sound card microphone input (see Fig. 4.13 and Fig. 4.23), and gain of the preamplifier and bandpass circuit. A *MatLab* function file `RT_G_Ae.m`, see *MatLab* Code A.2, was written using these gains to calculate the transmit/receive gains of the sodar for given transmission frequencies (see Fig. 4.27).

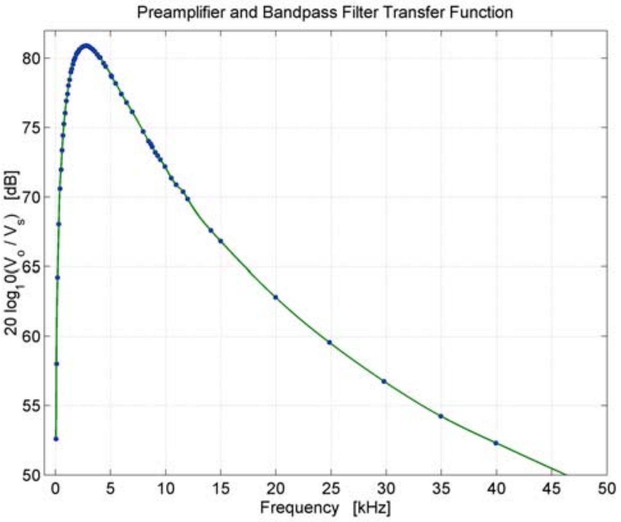


Figure 4.26: Sodar's preamplifier and bandpass filter transfer function

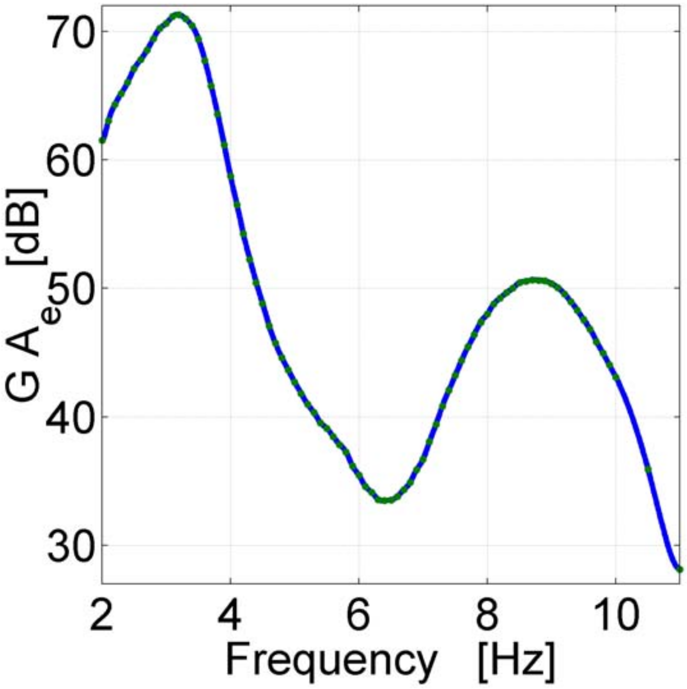


Figure 4.27: Transmission and receiving gain of the sodar, assuming a unity gain of the computer's sound card microphone input.

5

Field Trials and Results

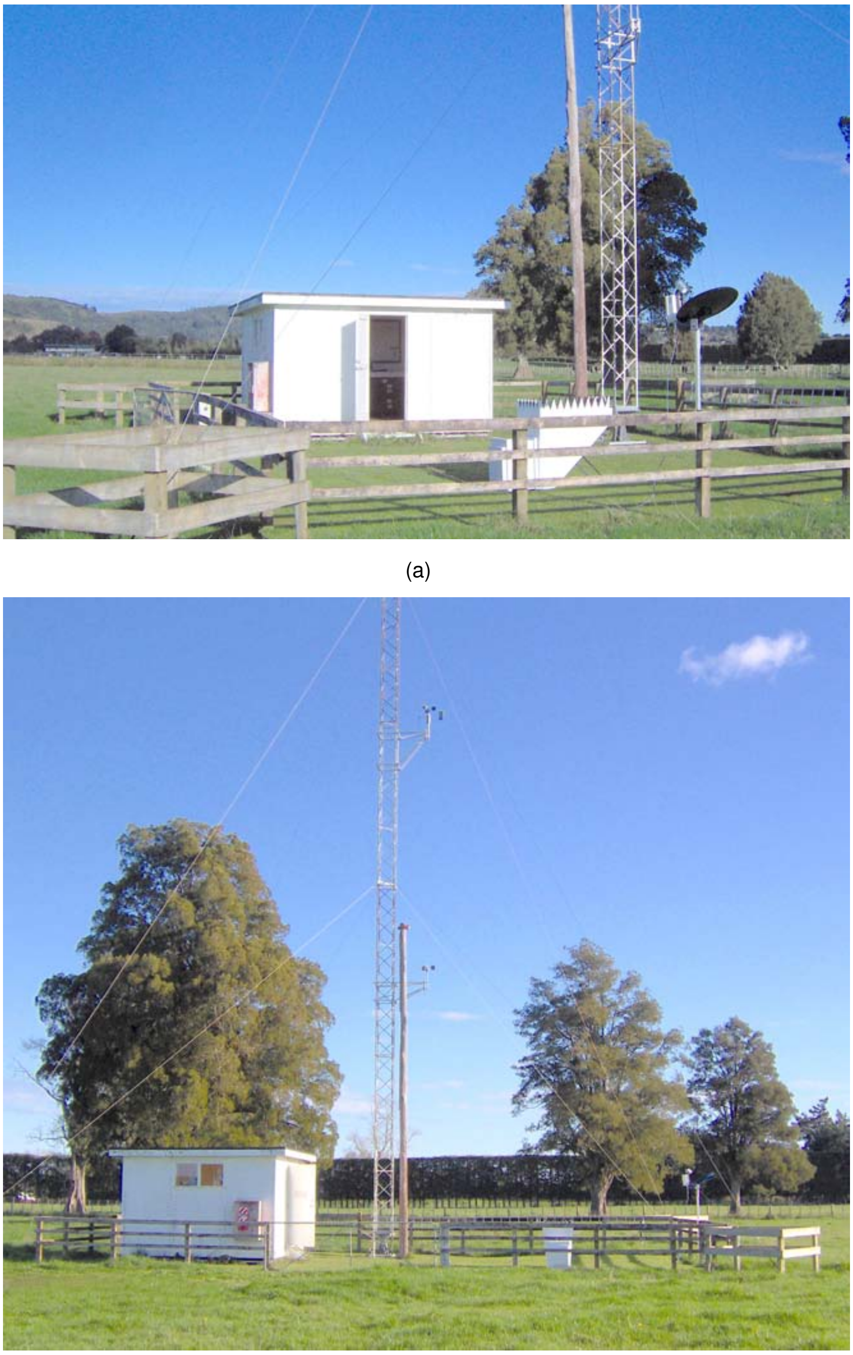
5.1 Field Site Description

The field measurements were carried out at the University of Auckland's field station at Ardmore, which is about 30 km south of Auckland's center (see Fig. B.1). The field station is situated in an area of flat farm land with a busy aerodrome located nearby. The field measurements were made in a fenced area, which contained a hut and a 30 meter tall met tower (see Fig. 5.1 and Fig. B.3). The sodar was positioned 6.5 meters in front of the hut and the MRR radar was mounted against the fence opposite the front of the hut (see Fig. 5.2).

5.2 Evaluation of Sodar Background Noise

The signal recorded by the sodar contains noise. This can include atmospheric pressure fluctuations, wind, leaf rustling, rain splash noise, electronics noise, digitization errors, fixed echoes, aeroplane and car noise, etc. Noise analysis is made in clear air and in rain conditions.

To investigate this noise, the sodar was run with the power amplifier turned off so that the sodar only recorded noise and did not transmit. Initial testing had been made without



(a)

(b)

Figure 5.1: Photos of field test location

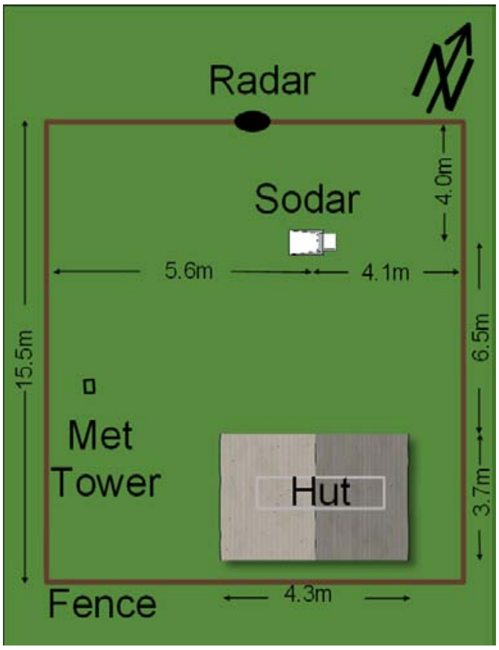


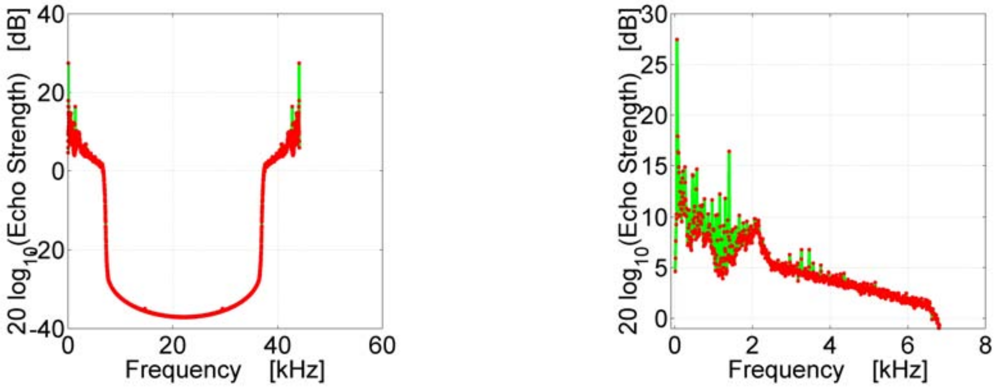
Figure 5.2: Diagram of the layout of the site where field measurements were made

the preamplifier and bandpass filter circuit. This analysis had had indicated significant ac mains noise. This appeared to be significantly reduced with the use of the preamplifier and bandpass filter circuit and by housing the sodar's circuit boards in grounded metal box. A problem encountered using the preamplifier circuit, however, was that the laptop, a *Toshiba Satellite Pro L20*, which had originally been intended to be used to run the sodar, could not be used due to sound card problems. Instead another laptop, a *Toshiba Tecra M1*, was used. While this laptop's sound card was able to cope with the preamplifier circuit, it proved to have a poor ability to detect sound at frequencies higher then about 7.1 kHz, see Section 4.3.3. This is seen in the following graphs as a sharp cut off in the spectra at around 7 kHz.

The noise recorded by the sodar was in the form of a text file containing short recordings of the time domain noise signal, taken every two seconds while the sodar program ran. This was analysis by taking the absolute value of the fast Fourier transform (FFT) of these recordings and averaging over the entire file using the *MatLab* code `RT_Average_RawData.m`, see *MatLab* Code A.7.

5.2.1 Noise Evaluation in Clear Air

Analysis of noise in clear air conditions was carried out on a day with no rain, minimal cloud cover, and light winds. There was almost continuous background noise from aircraft. It is possible that on this day the aircraft traffic may have been higher then normal due



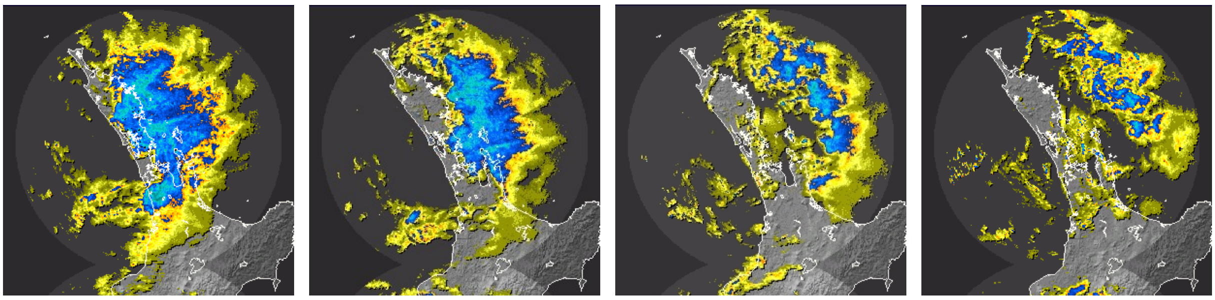
(a) Completed FFT

(b) Partial FFT

Figure 5.3: Noise analysis in clear air

to it being a fine weekend. The aircraft noise was variable depending on the number and type of aircraft and how close they were to the field site. Attempts were made to carry out acoustic sounding at times when aircraft traffic seemed to be minimal. An average of the fast Fourier transform of the recording is graphed in Fig. 5.3. The sharp cutoff in noise level above 7.1 kHz is believed to be due to the gain characteristics of the *Toshiba Tecra M1* laptop's sound card, which was used to run the sodar.

5.2.2 Noise Evaluation in Rain



(a) 12PM

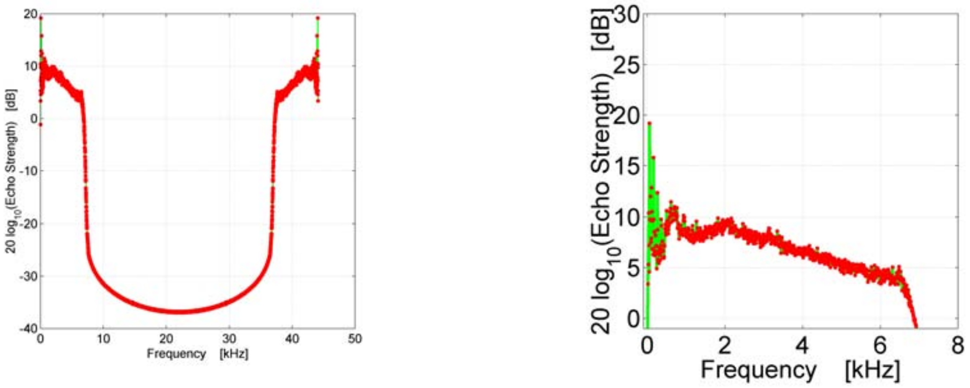
(b) 1PM

(c) 3PM

(d) 4PM

Figure 5.4: Rain rates for Auckland on the 20 Jun 2007 (MetService)

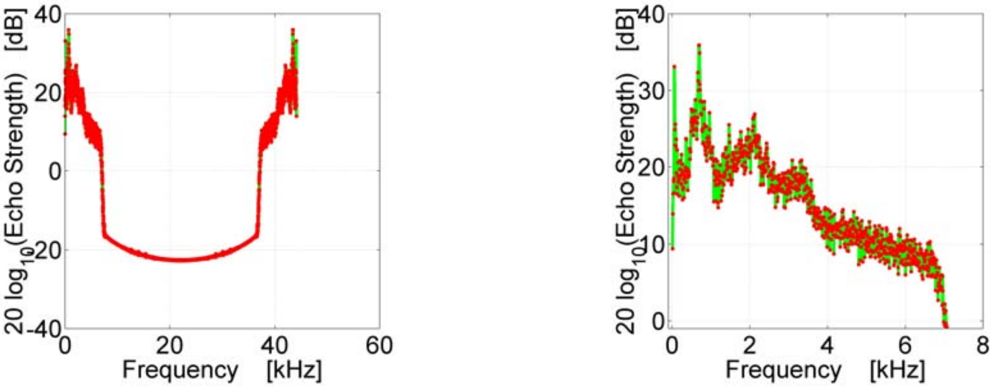
Analysis of noise during rain was carried out at 11am on the 20 June 2007 during moderately heavy rain, complete cloud cover, and light winds. Acoustic foam was placed on the sodar case above the speaker array to attempt to reduce impact noise from drops falling on the sodar's case. During most of the noise analysis recording, there was no aircraft noise. At the end of the recording a aircraft, possibly a helicopter, took off from Ardmore Aerodrome and headed towards the field station causing the noise levels to increase. The



(a) Completed FFT

(b) Partial FFT

Figure 5.5: Noise analysis in rain



(a) Completed FFT

(b) Partial FFT

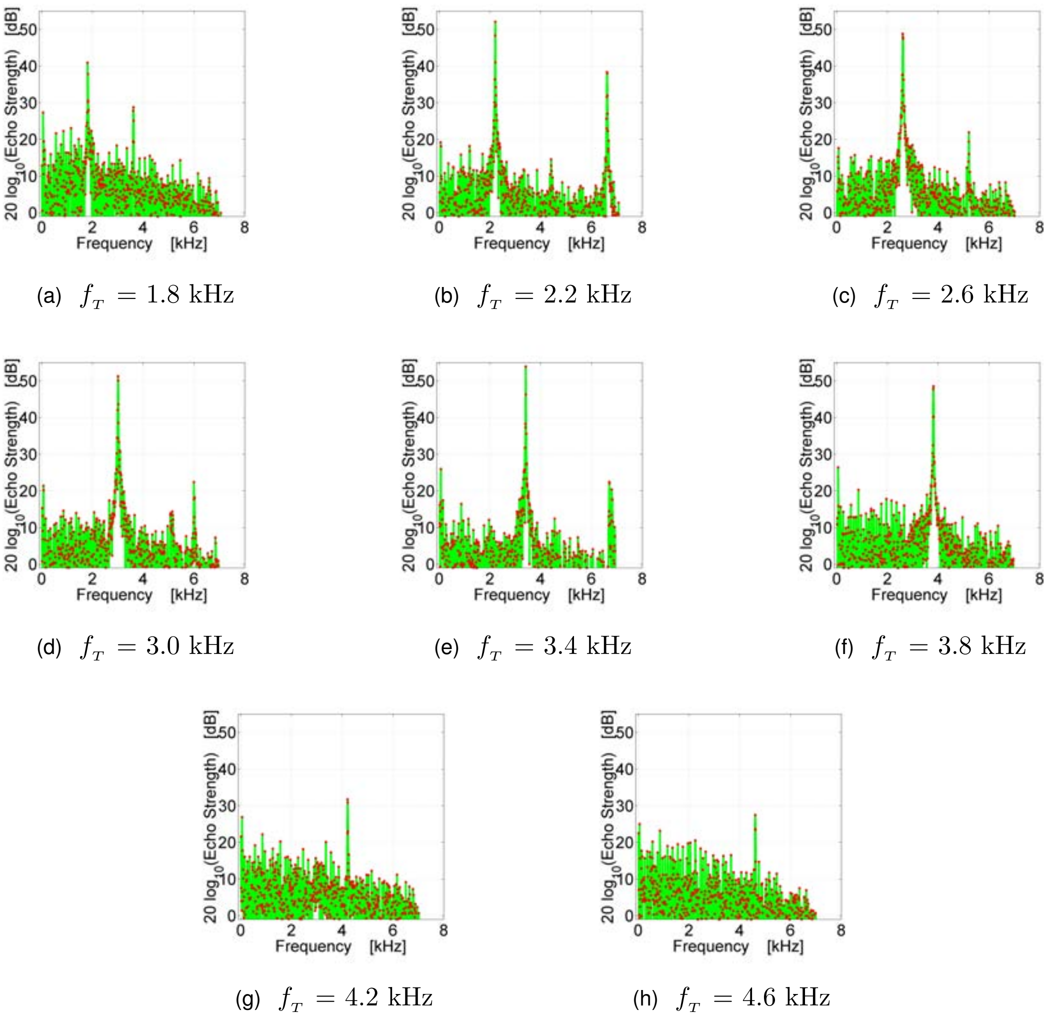
Figure 5.6: Noise analysis in rain with aircraft flying nearby

recording was stopped and a new recording started. After the aircraft had flown overhead the second recording was stopped.

An average of the fast Fourier transform of the first noise recording is graphed in Fig. 5.5. It is interesting to note that the noise level in this graph is actually lower then for that of the clear air recording. One possible explanation for this could be the low aircraft noise during the rain noise analysis as compared to during the clear air analysis. In Fig. 5.6, the noise spectra is plotted for the period when an aircraft was flying nearby.

5.3 Signal to Noise Ratio in Clear Air

The signal to noise ratio in clear air was investigated by running the sodar on 17 June 2007, when there was no rain, minimal cloud cover, and light winds. There was almost



(a) $f_T = 1.8$ kHz

(b) $f_T = 2.2$ kHz

(c) $f_T = 2.6$ kHz

(d) $f_T = 3.0$ kHz

(e) $f_T = 3.4$ kHz

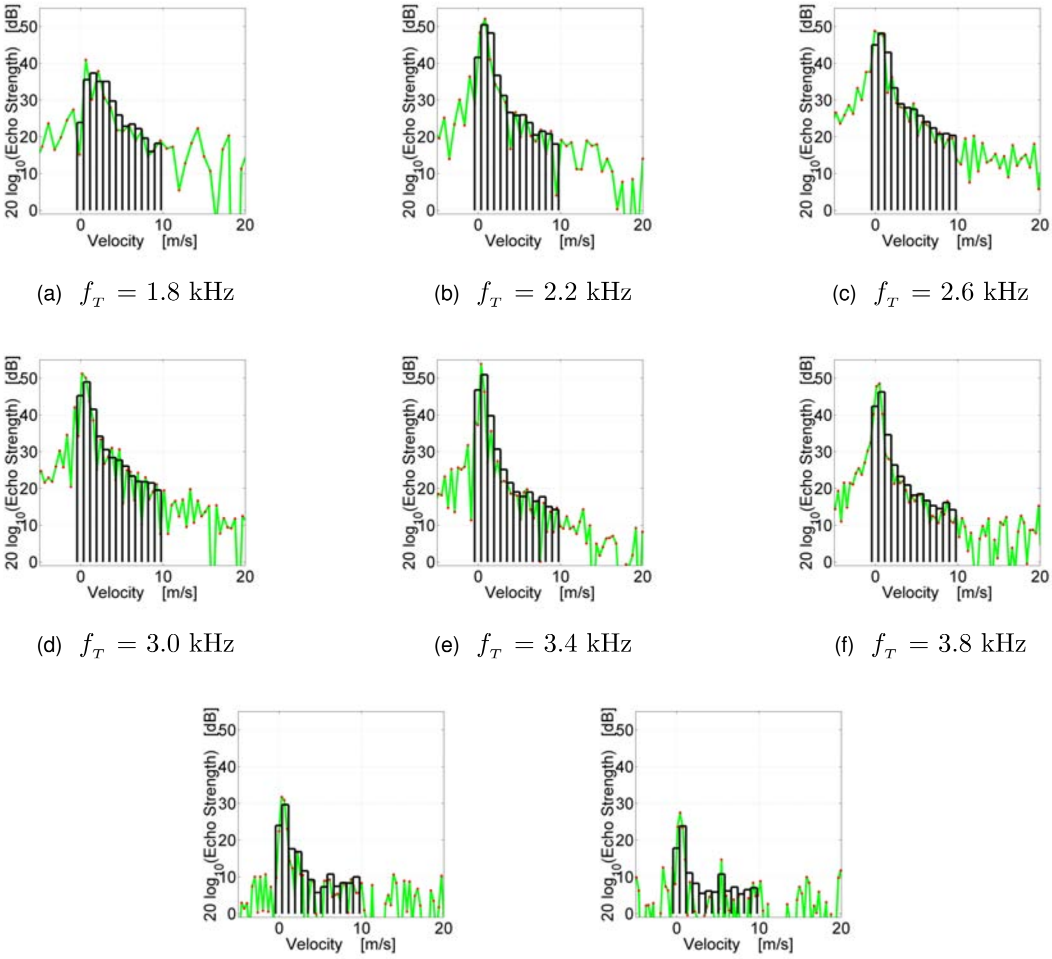
(f) $f_T = 3.8$ kHz

(g) $f_T = 4.2$ kHz

(h) $f_T = 4.6$ kHz

Figure 5.7: Echo signal in clear air for an range of transmit frequencies from a single height (30 m)

continuous background noise from aircraft during the recording. Transmit frequencies from 1 to 10 kHz in steps of 0.1 kHz and a height resolution of 20 meters was used. Relatively good signal to noise ratio was observed at frequencies between about 1.8 to about 4.6 kHz. Beyond this range, the amplitude of the signal appeared relatively comparable to that of the noise. Some harmonics or contributions to higher frequencies seem to be present in the data. In Fig. 5.7 averaged echo spectra for a range of transmission frequencies is plotted. The spectra immediately surrounding the turbulence peaks are plotted in Fig. 5.8 as a function scattering velocity. The maximum height from which a good signal to noise ratio was able to be achieved was about 60 m (see Fig. 5.9).

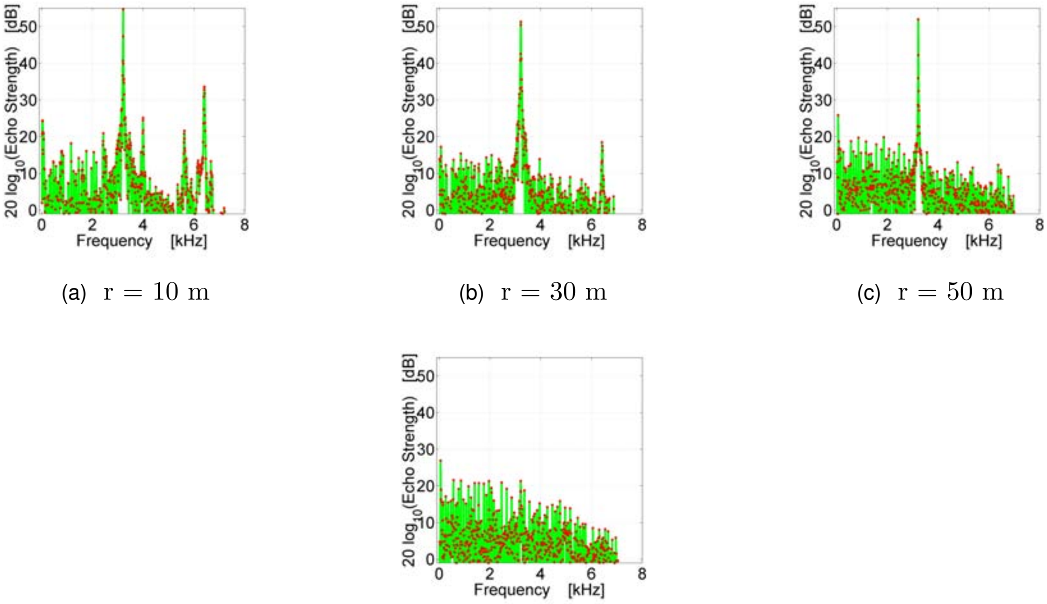


(a) $f_T = 1.8$ kHz (b) $f_T = 2.2$ kHz (c) $f_T = 2.6$ kHz

(d) $f_T = 3.0$ kHz (e) $f_T = 3.4$ kHz (f) $f_T = 3.8$ kHz

(g) $f_T = 4.2$ kHz (h) $f_T = 4.6$ kHz

Figure 5.8: Echo signal strength in clear air as a function of the velocity of the scatters for an range of transmit frequencies from a single height (30 m). Velocity bins which could be used in a the fitting algorithm are also plotted.



(a) r = 10 m

(b) r = 30 m

(c) r = 50 m

(d) r = 70 m

Figure 5.9: Echo signal in clear air from a range of heights for a single transmit frequency (3.2 kHz)

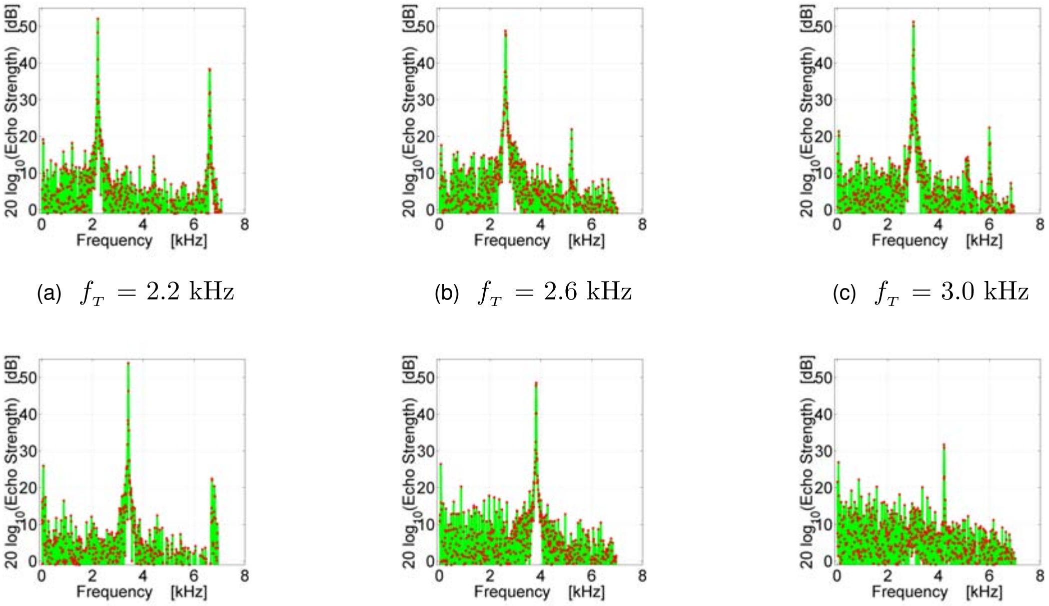
5.4 Acoustic Sounding during Rain

The acoustic soundings during rain was made on the 20 June 2007 and began at 11.20am and at 3.30pm. The sounding at 11.20am was made during moderately heavy rain and light winds, while the sounding at 3.30pm was made during light rain and moderate winds from the north east. There was minimal aircraft noise. The MRR Doppler rain radar had been intended to be used during the soundings. This was a new instrument, however, and, while it had been run on a previous day, it could not be got to work on the day the soundings were made, possibly due to a different computer being used. A failure of the transmit/receive switch circuit occurred after the sounding which began at 3.30pm. This meant that, while noise analysis was possible, no further soundings could be made.

The acoustic sounding was made using transmit frequencies from 1 to 10 kHz in steps of 0.1 kHz and a height resolution of 20 meters. Relatively good signal to noise ratio was observed for the turbulence peak at frequencies between about 1.8 to about 4.6 kHz (see Fig. 5.10). Except for some frequencies around 6 kHz, the amplitude of the signal outside this range appeared comparable to that of the noise. No signal was obtained above about 7 kHz. This is thought to be due to the characteristics of the sound card used to make the recordings, see Section 4.3.2.

In Fig. 5.11 and Fig. 5.12, average spectra is plotted as a function of the velocity of the scatterers for a single height resolution interval and for a range of transmission frequencies. Doppler shift frequency bins or velocity bins have also been plotted on these graphs. These bins cover the Doppler shift frequency range or velocity range that might be expected from rain falling at their terminal velocities. While some rain echo signal may be present in these plots, no clear echoes due to rain is thought to be observed. If the spectra given, for example, in Fig. 5.12h was due to echoes from rain, this would illustrate problem associated with sodar wind speed measurements during rain. A turbulence peak fitting algorithm might mistake the second peak, which could be due to rain, for the turbulence peak, resulting in a significant wind speed measurement error, see Section 1.2.

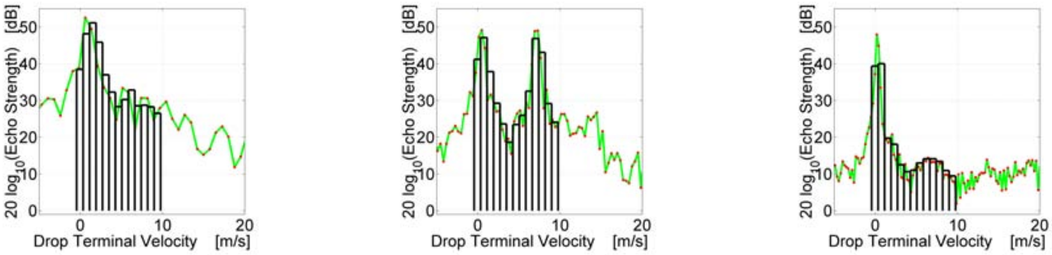
The failure to observe distinct rain echoes in the analyzed data may be due to the variable nature of rain with time. This variability means that different pulses, transmitted by the sodar, may result in echoes from rain which have different drop size distributions and different rain fall intensities. The rain spectra obtained by the sodar may, therefore, vary considerably with time. While averaging of spectra might serve to increase the signal to noise ratio of echoes from turbulence it may not have the same effect for echoes from rain. Future work could investigate the spectrum for each pulse instead of the average spectrum.



(a) $f_T = 2.2$ kHz (b) $f_T = 2.6$ kHz (c) $f_T = 3.0$ kHz

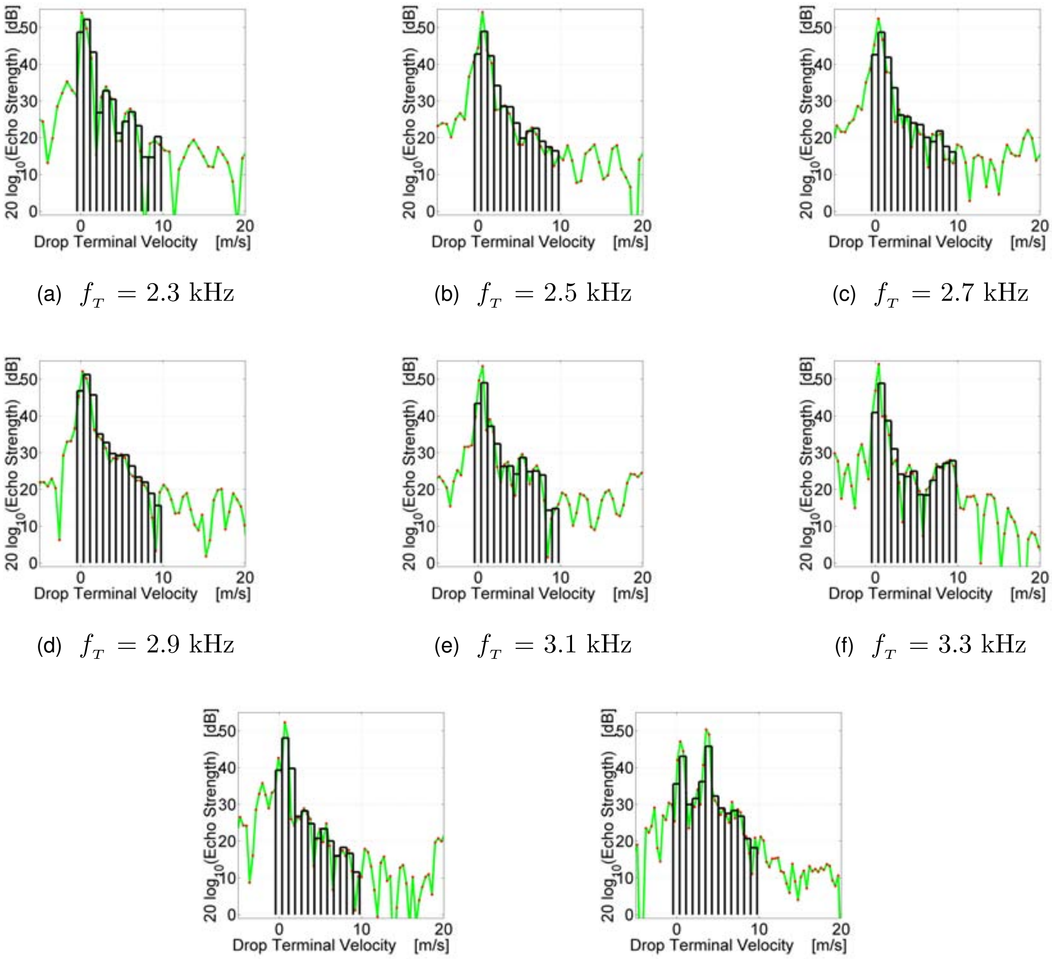
(d) $f_T = 3.4$ kHz (e) $f_T = 3.8$ kHz (f) $f_T = 4.2$ kHz

Figure 5.10: Echo signal in moderate rain for an range of transmit frequencies from a single height, 30 m (Sounding begun at at 11.20am)



(a) $f_T = 1.8$ kHz (b) $f_T = 3.8$ kHz (c) $f_T = 5.9$ kHz

Figure 5.11: Backscattered power spectra from turbulence and moderate rain (Sounding begun at at 11.20am)



(a) $f_T = 2.3$ kHz

(b) $f_T = 2.5$ kHz

(c) $f_T = 2.7$ kHz

(d) $f_T = 2.9$ kHz

(e) $f_T = 3.1$ kHz

(f) $f_T = 3.3$ kHz

(g) $f_T = 3.5$ kHz

(h) $f_T = 3.9$ kHz

Figure 5.12: Backscattered power spectra from turbulence and light rain (Sounding begun at 3.30pm)

6

Recommendations for Future Research and Improvements

6.1 Recommendation for Improvements to the Sodar

6.1.1 Computer Sound Card used to Transmit and Receive

The sodar uses a computer's sound card to transmit acoustic pulses and record the echoes. During field trials, problems were encountered with the two sound cards used. The *Toshiba Satellite Pro L20* laptop's sound card was unable to cope with preamplifier circuit. It is believed that this may have been due to the amplified signal being too large for this sound card. In studies of the characteristics of this sound card, it had been found that clipping of the recorded signal occurred if the input voltage to the microphone input was larger then about 100 mV, see Section 4.3.3. The output from the preamplifier, due to noise, was observed during field trials to exceed this voltage level. The *Toshiba Tecra M1* laptop's sound card did not appear to show this clipping and seemed to be able to run the sodar with the preamplifier. The problem encountered with this laptop's sound card was that it appeared to be unable to record sound which had a frequency greater then about 7.1 kHz. This meant no signal was able to obtained from frequencies around 8.8

kHz, which was a frequency range were the sodar's speaker array had optimal gain, see Section 4.2.1. It had been hoped that strong echoes from rain would be achieved in this region. Future work could investigate the use of other sound cards which have are able to cope with the voltages from the sodar's preamplifier and which have a flat gain to about 10 kHz. Alternatively a analog to digital converter (ADC) could be used instead. This could have the advantage of a higher bit level, say 18 bits as apposed to the 16 bits of the laptop's sound cards.

6.1.2 Sodar's Case

The current design of the sodar's case allows sound to reach the sodar's speaker array from a range of angles without first being reflected off its reflector (see Section 4.3.1). In a similar manner, when the sodar is being transmitting, a secondary lobe of sound will be emitted at the same range of angles. These undesirable characteristics may be addressed by either increasing the height of the case's baffles or extending the case so that the speaker array is further back from reflector.

6.1.3 Sodar's Speaker Array

The speakers used in the sodar's speaker array are strongly frequency dependant. This frequency dependence consisted of two peaks at 3.2 kHz and 8.8 kHz with a region of low gain between (see Fig. 4.13). This frequency response may be unsuitable for wind measurements using multi-frequency sounding, where a relatively flat gain between 2 and 5 kHz might be desired. The frequency response of other speakers could be investigated and trailed in the sodar.

6.1.4 Sodar's Electronics

During field trials of the sodar the transmit/recieve switch stopped transmitting. This needs to be fixed before further field testing can proceed. It is believed this may have been due to a failure of the voltage regulator, which was used to supply dc voltage to the transmit/recieve circuit. Alternatively the relay and monostable oscillator chips could be checked.

The sodar's current preamplifier and bandpass filter circuit board was designed for preliminary measurements only. It currently only has a single bandpass filter with a wide bandwidth. A circuit board with two channels, each containing a ramped gain preamplifier and a bandpass filter, has been designed and printed (see Fig. 6.1) but due to time limitations it was not able to be completed within period of this work. This circuit board could be completed and tested. Optimal bandpass characteristics and noise reduction could be investigated. A main form of noise believed to observed during field trials was ac mains noise. Reduction of this form of noise could be investigated. It may be desirable in the future to run the sodar continuously unattended for periods of time. This could be kept in mind in future work on the sodar.

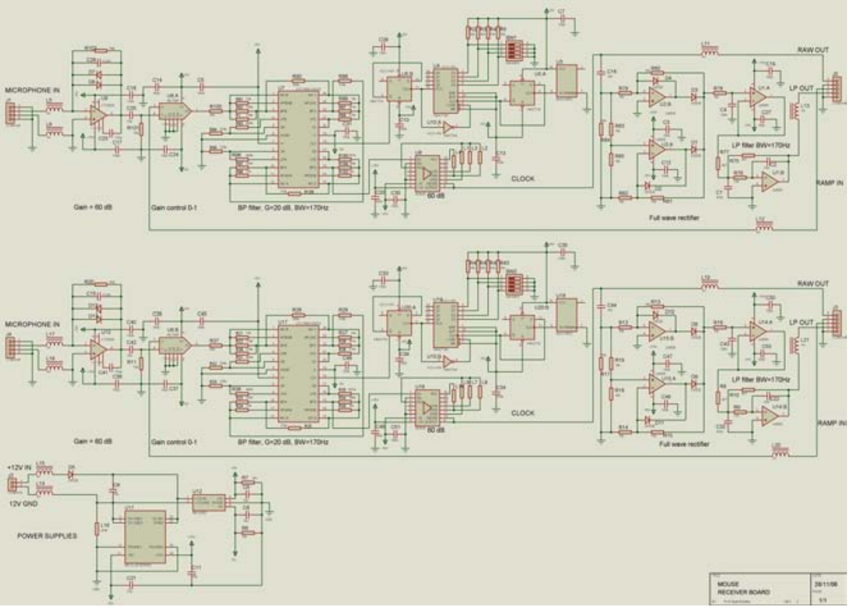


Figure 6.1: Diagram of the circuit board containing the pre-amplifier and bandpass filters which was designed and printed for the sodar (Bradley)

In wind measurements, it would be desirable to use a narrow bandpass filter to observe the turbulence peak. The bandwidth of the turbulence peak is significantly smaller then the width of the spectra from echoes from rain. This means that a narrow bandpass filter for turbulence would not be wide enough to allow through the backscattered power spectra from rain which would be needed needed for the turbulence and rain echo fitting algorithm. A possible solution that could be investigated is to use two band pass filters per transmission frequency, one for the turbulence peak and one for the rain spectra.

Wind measurements require tilting of the sodar beam in three or more directions. The sodar currently transmits vertically and tilting of the sodar beam requires manual tilting of the sodar. Future work could include creating a circuit board that allows the sodar beam to be tilted electronically using a phase delay across the speaker array.

6.1.5 Transmit/Receive GUI

The design of the sodar's GUI is currently focused on transmitting and receiving. Although it is possible to use the GUI to analysis noise while not transmitting, the GUI could be improved by making this process more routine. Also an average of the spectra recorded by the sodar could be displayed on the GUI. Perhaps the GUI could also include a graph of the fitted rain and turbulence components of the backscattered power spectra recorded by the sodar.

The sodar currently preforms multi-frequencies transmission by cycling through the transmission frequencies using a single pulse for each transmission frequency, with a pause of about two seconds between each pulse. This means that there is a time delay between pulses of different frequencies. The turbulence and rain echo fitting algorithm assumes that the the echoes from all transmission frequencies are from identical turbulence and rain condition. If, however, there are variations of these conditions with time, errors could be expected in the fitting. Averaging has been used to try to reduce this error. Future work could consider altering the GUI to transmit more then one frequency in each pulse, as in Coulter and Martin (1986). This would require that the separation between the transmission frequencies in each pulse was greater then the range of frequencies expected from rain and turbulence. An alternative that could be considered would be to transmit two or more pulses of different transmission frequencies immediately after each other and then correcting for the time delay. The same frequency separation condition as above would apply.

If tilting of the sodar beam is used, the GUI could be modified to allow for this tilting. Wind speed and direction calculations could be included. Also, if an analog to digital converter is used instead of a computer's sound card, and this GUI was desired to be used, it would need to be modified to allow it to capture the recorded data from the ADC. This GUI would also need to be modified if more then one channel were desired.

6.2 Suggested Research/Improvements for the Turbulence and Rain Fitting Algorithm

The turbulence and fitting algorithm needs to be invested further with field testing to see if it provides a valid and practical method of separating the rain and turbulence components. Noise is expected to be a significant issue concerning the viability of this method. Noise reduction, noise filtering or smoothing, and developing the gain at optimal transmission frequencies could be factors to be investigated.

In analysis of raw data presented in this work, spectra from an individual transmitted pulse has not been presented. Instead averaging over an entire file was preformed to obtain the average spectra from a given height resolution interval and a given transmission frequency. While this is likely to enhance the signal to noise ratio for echoes from turbulence, it may have the opposite effect for echoes from rain, which are likely to be highly variable with time. Future work could investigate the effect of using the turbulence and rain fitting algorithm on the backscattered spectra from each individual pulse instead of on a spectra which is averaged over may pulses.

Currently when fitting raw data, the noise level is subtracted from the turbulence and rain backscattered power spectra by manually inspecting the noise level at either end of the turbulence and rain echo spectra and subtracting the mean value of this noise. Future work would could include this or some similar process into the fitting algorithm.

This algorithm also currently assumes no air motion in the direction of the sodar's beam. Shifts in the backscattered spectra due to air motion in the direction of the sodar beam are currently corrected by manual inspection and shifting of the power spectra. Future development could include turbulence peak fitting to estimate the radial air motion and automatically shift the power spectra to correct for this radial air motion. Another improvement could be writing a piece of code to compare the drop size distribution obtained using the sodar to that obtained using the MRR Doppler rain radar.

6.3 Suggestions for Further Field Trials

The field trials were limited by time and the availability of rain. Extensive field testing needs to be made to investigate the validity and practicality of the of the turbulence and rain echo fitting algorithm described in Chapter 2.

6.3.1 Noise Identification and Reduction

A main factor determining the viability of the rain and turbulence fitting algorithm will be the signal to noise ratio for echoes from turbulence and rain. Field tests need to be carried out to investigate the causes of noise detected by the sodar. A main cause of noise detected during field trials was thought to be ac mains noise. How much this contributes to the total noise could be investigated and attempts to reduce it could be trialled. Another significant form of noise at Ardmore is aircraft noise. This could be investigated to see how much of a issue this is. The type and number of aircraft, the distance to the aircraft from the field station, and the velocity and direction of travel are possible factors effecting the noise detected by the sodar. An identification and rejection algorithm could be investigated.

The clear air measurements were made on a day with relatively high aircraft traffic. This noise analysis could be repeated during a period when there is no noise from aircraft. Further trials could be made to identify and reduce the noise due to rain impacts on the sodar's case. The variability of the rain echo spectra with time and the effects of averaging this spectra could also be studied. Smoothing and filtering of noise could be investigated.

6.3.2 Optimization of the Sodar's Ability to Detect Echoes from Turbulence and Rain

Field trials have indicated that the sodar could achieve a relatively good signal to noise ratio for echoes from turbulence for transmission frequencies between 1.8 and 3.6 kHz. Some echoes, particularly from rain, were thought to be likely for transmission frequencies around 8.8 kHz. This frequency range was not able to be investigated in field trials due to sound card problems. Field work could investigate both of the above frequency range to try to optimize echoes from turbulence and rain, while minimising noise. If the 8.8 kHz frequency range proves unsuitable, the 3.3 kHz range would need to be investigated further to see if these frequencies are able to generate echoes of sufficient strength from rain and if there is sufficient frequency separation in this frequency range to allow the fitting algorithm to work efficiently. Future work could investigate the use of different speakers in the sodar's speaker array and adjustment to any bandpass filters used to obtain transmission frequencies that are optimal for wind and rain measurements and which have sufficient frequency separation to allow the fitting algorithm to work.

6.3.3 Turbulence and Rain Fitting Algorithm Analysis

The fitting algorithm could be evaluated by comparing the rain drop size distribution obtained using the sodar and the fitting algorithm and that obtained using the MRR Doppler rain radar. Alternatively the sodar could be developed for wind speed measurements. The accuracy of these wind speed measurements could be evaluated in both in clear air and rain conditions using anemometers' wind speed measurements. This could be used to evaluate how efficiently the fitting algorithm is able to eliminate wind speed measurement errors due to rain.

Conclusion

The acoustic radar or sodar has increasingly been used as a tool for studies of the lower atmosphere. One application has been the measurement of wind speed profiles for wind energy purposes. A main cause of error in sodar wind speed measurements is contamination from rain leading to significant overestimates in measured wind speeds. This work develops a multi-frequency clutter rejection algorithm for reducing wind speed errors due to rain. This algorithm relies on the fact that the backscattered power spectra from turbulence has a $\lambda^{-1/3}$ wavelength dependence while that from rain has a λ^{-4} dependence.

Simulations were made of the total backscattered power spectra due to turbulence and rain. A least squares fitting algorithm was developed to separate the rain and turbulence components of the backscattered power spectra. A multi-frequency sodar was designed and a GUI program was written to run the sodar. Laboratory evaluations of the sodar were carried out. The speakers used in the sodar's array were found to have a gain which had peaks at 3.2 and 8.8 kHz with regions of poor gain surrounding these peaks. This characteristic of the speakers limits the frequencies that may practically be used for multi-frequency sounding.

Preliminary field testing was made in clear air and rain conditions. Echoes from turbulence were measured which had relatively good signal to noise ratios for frequencies between about 1.8 to 4.6 kHz . Sound card problems prevented frequencies above 7 kHz being measured. This meant that the 8.8 kHz frequency range, where the sodar's speaker array had relatively good gain, was not able to be investigated. No distinct echoes from rain were observed. This may have been due to the variable nature of rain with time and also due to the averaging processing used in analyzing the raw data. Further development and field testing needs to be carried out to optimize the sodar's ability for detection of echoes from turbulence and rain. Field tests then need to be performed to determine the viability and practicality of the turbulence and rain fitting algorithm and its ability to reduce errors in wind speed measurements.

In retrospect, the breadth of this project, involving as it did the entire sweep from concept, theory, simulation, instrument design and construction, laboratory evaluation, and field trials, was very ambitious. However the findings in this thesis provide a solid basis for ongoing research.

A

MatLab Codes

MatLab Code A.1: **RT_absorb.m**

```
function [alpha_dB, alpha] = RT_absorb(f,T,p_a,r_h,varargin)
% {
This function file calculates the absorption coefficient of sound in the atmosphere in dB/m alpha_dB and in m^-1
alpha, see Section 1.5.3. The input arguments are frequency f [Hz], atmospheric temperature T [K], pressure p_a
[Pa], and relative humidity r_h in percent, see.
% }
 pr = 101325; T_20 = 293.15; T_01 = 273.16;                % Pressure [Pa] and temperatures [K].
tau_r    = T ./ T_20;
rho_r    = p_a ./ pr;
C_sat    = -6.8346 .* (T_01 ./T ).^1.261 + 4.6151;
rho_sat  = 10 .^ C_sat;
h        = r_h .* rho_sat / rho_r;
% Relaxation frequencies for oxygen and nitrogen, see Equations (1.16) and (1.15).
f_rO     = rho_r .* ( 24 + 40400 .* h .* (0.02 + h ) ./ (0.391 + h) );

f_rN     = rho_r .* tau_r.^(-1/2) .* ( 9 + 280 .* h .* exp( -4.17 .*( tau_r.^(-1/3) -1) ) );

b_2      = 0.01275 .* exp( -2239.1 ./ T)  ./ ( f_rO + f.^2 ./ f_rO );

b_1      = 0.1068 .* exp(-3352 ./ T) ./ ( f_rN + f.^2 ./ f_rN );

% Absorption coefficient [dB/m], Equation (1.14). Absorption = alpha_dB * r, for a range r.
alpha_dB = 8.686 .* f.^2 .* tau_r.^0.5 .* (1.84e-11 ./ rho_r + tau_r^-3 .* (b_1 + b_2));

dBconvrt = 1/(10.*log10(exp(1)));                          % Coefficient for conversion from dB/m to m^-1.
% Absorption coefficient [m^-1]. Absorption = exp(-alpha * r).
alpha    = alpha_dB .* dBconvrt;
```

MatLab Code A.2: **RT_G_Ae.m**

```
function G_Ae = RT_G_Ae(f_T)
% {
This function file gives the product G_Ae of the estimated gains of the sodar, while transmitting and receiving,
for transmit frequencies f_T, base on measurements made in Chapter 4. The transmission antenna gain G of the
sodar is estimated from the measured transmit gain of the sodar's speaker array G_SpeakerArray and the gain of
the power amplifier G_PowerAmp, while loaded by the speaker array and amplifying the output of the computers
sound card's headphone output. The receiving gain of the sodar or antenna effective area A_e is estimated by
measuring the gain of the sodar's speaker array A_SpeakerArray, while receiving, and measuring the gain of the
computer sound card's microphone input A_compscd_mic. A cubic spline is then used for frequencies at which
measurements were not made.
% }

% Measure gain of the speaker array while transmitting for a range of transmit frequencies.
fs_p9_1to25a   = [2:0.1:10 10.5 11].*1e3;              %   Frequencies used to measure the transfer function
                                                            of the speaker array while transmitting. [Hz]
Vs_p9_1to25a   = [3.395 3.355 3.350 3.420 3.405 3.355 3.415 3.415 3.380 3.375 3.425 3.410 3.415 ...
                  3.425 3.430 3.415 3.435 3.445 3.425 3.440 3.455 3.425 3.445 3.445 3.450 3.430 ...
                  3.475 3.470 3.460 3.455 3.485 3.485 3.485 3.460 3.485 3.465 3.480 3.540 3.450 ...
                  3.540 3.500 3.545 3.490 3.560 3.510 3.557 3.500 3.570 3.525 3.525 3.545 3.525 ...
                  3.540 3.560 3.560 3.560 3.570 3.580 3.585 3.565 3.580 3.580 3.595 3.600 3.595 ...
                  3.595 3.610 3.615 3.600 3.590 3.590 3.590 3.590 3.595 3.600 3.600 3.600 3.600 ...
                  3.610 3.615 3.615 3.675 3.740];        %   Input voltage [V]
Vo_p9_1to25a   = [413 482 547 609 666 737 808 878 962 1054 1124 1194 1226 1192 1130 1004 842    ...
                  680 534 410 313.5 243.0 190.5 154.0 127.2 106.0 89.8 77.8 69.2 62.8 57.7 53.2 ...
                  49.1 46.0 42.9  41.2 39.0 37.6 34.8 32.0 29.6 27.5 26.1 25.30 25.1 25.90      ...
                  26.65 29.25 31.30 35.65 40.0 47.2 55.8 67.1 78.6 91.2 106.4 122.6 139.0 157.0 ...
                  171.5 191.0 205.5 219.0 230.5 245.5 252.0 259.5 262.5 264.0 262.0 256.0 245.0 ...
                  232.0 217.5 202.5 189.5 171.5 157.5 143.4 130.2 62.1 27.4].*1e-3;
                                                       %   Output voltage across a speaker [V]
G_SpeakerArray = spline(fs_p9_1to25a,[0 (Vo_p9_1to25a ./ Vs_p9_1to25a) 0], f_T);
                                                       %   Cubic spline.
% {
Measured gain of power amplifier while amplifying the computer sound card's headphone output and while loaded
by speaker array.
% }
fs_p17c_1to25a = [2.00 2.25 2.50 2.75 3.00 3.10 3.20 3.30 3.40 3.50 3.60 3.80 4.00 4.25 4.50    ...
                  4.75 5.00 5.25 5.50 5.75 6.00 6.40 6.80 7.00 7.25 7.50 7.75 8.00 8.25 8.40    ...
                  8.50 8.60 8.70 8.80 8.90 9.00 9.20 9.50 9.75 10.00 11.00].*1e3;
                                                       %   [Hz]
Vs_p17c_1to25a = [14.75 14.76 14.82 14.84 14.86 14.86 14.84 14.84 14.86 14.88 14.90 14.88 14.92 ...
                  14.92 14.94 14.90 14.92 14.92 14.92 14.92 14.96 15.00 15.10 15.10 15.24 15.26 ...
                  15.30 15.32 15.34 15.40 15.40 15.44 15.44 15.44 15.46 15.46 15.50 15.52 15.54 ...
                  15.66 15.84];                        %   [V]
G_PowerAmp     = spline( fs_p17c_1to25a, [0 (Vs_p17c_1to25a./max(Vs_p17c_1to25a)) 0], f_T );

G  = G_SpeakerArray .* G_PowerAmp;                     %   Estimate of the transmission antenna gain of the
                                                            sodar.
% Measured gain of the speaker array while receiving or acting as a microphone.
fs_p8_1to25a   = [2:0.1:10 10.5 11].*1e3;              %   Frequencies used to measure the transfer func-
                                                            tions of the speaker array while receiving. [Hz]
Vs_p8_1to25a   = [2.955 2.925 2.920 2.975 2.950 2.910 2.955 2.980 2.960 2.975 2.995 2.985 2.975 ...
                  2.995 3.000 2.975 2.995 3.005 2.995 3.005 3.005 3.005 2.985 3.015 3.020 3.005 ...
                  3.015 3.025 2.995 3.010 3.015 5.210 5.210 5.200 5.190 5.23 5.95 5.97 5.96     ...
                  5.95 5.97 5.94 5.96 5.95 5.97 5.96 5.98 5.97 5.97 6.00 5.96 5.97 5.97 5.99    ...
                  5.96 5.97 5.97 5.99 5.99 5.94 5.96 5.93 5.99 5.97 5.97 5.99 5.98 5.97 5.97    ...
```

MatLab Code A.2: (continued)

```
                  5.95 5.98 5.98 5.99 5.99 5.98 5.96 5.98 5.99 5.99 5.93 5.99 5.99 5.99];
                                                                      %   [V]
Vo_p8_1to25a   = [16.65 18.75 20.80 22.65 24.20 26.15 28.75 31.20 35.25 38.45 41.3 44.3 44.9   ...
                  43.9 40.0 35.1 28.6 22.9 17.3 13.15 9.75 7.50 5.85 4.65 3.80 3.20 2.45 2.25   ...
                  1.95 1.80 1.65 2.65 2.65 2.35 1.95 2.00 1.70 1.90 1.70 1.75 1.65 1.45 1.20    ...
                  1.10 1.00 1.25 1.10 1.20 1.45 1.85 2.05 2.68 3.06 4.22 4.82 5.44 6.48 7.62    ...
                  8.70 9.84 10.80 12.26 13.46 14.56 15.68 16.40 17.45 18.20 18.50 18.50 18.35   ...
                  17.95 17.33 16.35 15.10 14.20 12.95 12.00 10.40 9.10 8.05 2.65 1.30].*1e-3;
                                                                      %   [V]
A_SpeakerArray = spline(fs_p8_1to25a,[0 (Vo_p8_1to25a ./ Vs_p8_1to25a) 0], f_T);

A_compscd_mic  = ones(size(f_T));                                     %   Measurements of the gain of the computer sound
                                                                          card's microphone input were independent of
                                                                          transmit frequency.
% Measured gain of the sodar's preamplifier and bandpass filter
fs_p18         = [0.052 0.097 0.202 0.316 0.403 0.511 0.615 0.699 0.782 0.877 0.993 1.088 1.188 ...
                  1.292 1.402 1.491 1.587 1.684 1.775 1.887 1.980 2.081 2.169 2.392 2.586 2.788 ...
                  2.979 3.166 3.376 3.573 3.775 3.978 4.050 4.363 4.566 5.053 5.111 5.464 5.988 ...
                  6.443 6.961 7.963 8.458 8.634 8.781 9.047 9.265 9.512 9.916 10.51 10.93 11.57 ...
                  11.99 14.10 14.96 19.95 24.84 29.78 34.93 39.92 46.44 50.34 61.88 80.72].*1e3;
                                                                      %   [Hz]
Vo_p18         = [0.444 0.828 1.695 2.630 3.525 4.13  4.85  5.49  6.04  6.61  7.31  7.74  8.30  ...
                  8.70  9.26  9.54  9.90  10.22 10.44 10.74 10.92 11.00 11.18 11.40 11.50 11.56 ...
                  11.50 11.40 11.22 11.02 10.74 10.52 10.46 10.00 9.72  9.02  8.92  8.44  7.74  ...
                  7.22  6.67  5.66  5.23  5.11  4.98  4.77  4.63  4.49  4.23  3.85  3.64  3.440 ...
                  3.235 2.490 2.285 1.438 0.988 0.716 0.536 0.429 0.327 0.283 0.191 0.116];
                                                                      %   [V]
Vs_p18         = 1.04e-3.*ones(size(Vo_p18));                         %   [V]
A_PreAmpBndPas = spline(fs_p18, [0 (Vo_p18 ./ Vs_p18) 0], f_T);

Ae = A_SpeakerArray .* A_compscd_mic .* A_PreAmpBndPas;               %   Estimated receiving gain or receiving effective
                                                                          area of the sodar.
G_Ae   =   G .* Ae;                                                   %   Estimate of the product of the transmission and
                                                                          receiving gain of the sodar.
```

MatLab Code A.3: **RT_Sim_functions.m**

```
function [x,D,VT,dD,bin,Dsf,f_edg,G_Ae,alpha,v,beta,varargout] = RT_Sim_functions(...
    f_T,c,Nfb,HeightRes,Temp,pa,rh,varargin)
% {
This function calculates terms the the simulation and fitting functions MatLab Code A.4 and MatLab Code ??.
It simulates Nfb number of Doppler shift frequency bins Dsf for each transmission frequency f_T. These Doppler
shift vaues have a zero value at the transmission frequency. The bandwidth bin of these bins changes with each
transmission frequency to enable the same rain drops diameters to contribute to the same frequency bins regardless
of the transmission frequency. The combined bandwidth of the bins is made equal to the range of Doppler shift
values that one would expect from rain drops falling at their terminal velocities. For the central frequencies of each
of these bins, a terminal velocity VT, drop diameter D, and Marshal and Palmer drop size distribution n_D is
simulated, using the assumption that the drops are falling at their terminal velocities. The difference in terminal
velocities v and diameter dD for the largest and smallest drops that might be expected to contributing to each
frequency bin is also simulated, using the above assumption. Also, the inverse of the transmitted wavelength x and
the absorption coefficient alpha for each transmission frequency is also calculated. The input arguments are the
transmission frequencies f_T [Hz], the number of Doppler shift frequency bins Nfb, the rain intensity I [mm/h], the
speed of sound c [m/s], the temperature T [K], atmospheric pressure p_a [Pa], and relative humidity r_h [percent].
% }
lambda  = c./f_T;                               % Transmission wavelengths for given transmission
                                                  frequencies f_T and the speed of sound c.
x       = 1./lambda;                            % Inverse of wavelength [1/m].
D_min   = 0.1e-3;                               % Minimum expected drop diameter [m].
D_max   = 6e-3;                                 % Maximum expected drop diameter (Doviak and
                                                  Zrnic, 1993) [m].
VT_min  = 9.65-10.3*exp(-600*D_min);            % Minimum expected terminal velocity, Equation
                                                  (1.26) [m/s].
VT_max  = 9.65-10.3*exp(-600*D_max);            % Maximum expected terminal velocity [m/s].
f1_min  = 2 * (VT_min)*f_T(1)/ c;               % Minimum expected Doppler shift frequency given
                                                  transmission frequency f_T(1), Equation (1.28)
                                                  [Hz].
f1_max  = 2 * (VT_max)*f_T(1)/ c;               % Maximum expected Doppler shift frequency given
                                                  transmission frequency f_T(1) [Hz].
f_1     = linspace(f1_min, f1_max,Nfb);         % Doppler shifted frequency bins for tranmission
                                                  frequency f_T(1) [Hz].
bin_1   = f_1(2)-f_1(1);                        % Bin size for first transmission frequency [Hz].
VT      = f_1.*lambda(1)./2;                    % Drop terminal velocities corresponding to each
                                                  Doppler shift frequency bin [m/s].
VTp     = (f_1 + bin_1/2).*lambda(1)./2;        % Drop terminal velocities at frequency f+bin/2
                                                  [m/s].
VTm     = (f_1 - bin_1/2).*lambda(1)./2;        % Drop terminal velocities at frequency f-bin/2
                                                  [m/s].
v       = 2.*(VTp - VTm);                       % Terminal velocity differences across frequency
                                                  bins.
D       = -1./600.*log((9.65 - VT)./10.3);      % Drop diameters calculated from the terminal ve-
                                                  locities, see Equation (1.27) [m].
Dp      = -1./600.*log((9.65 - VTp)./10.3);     % Drop diameters calculated from the terminal ve-
                                                  locities [m].
Dm      = -1./600.*log((9.65 - VTm)./10.3);     % Drop diameters calculated from the terminal ve-
                                                  locities [m].
dD      = Dp - Dm;                              % Drop diameter difference [m].
for m   = 1:length(f_T)
    s          = lambda(1)/lambda(m);           % Factor by which fo has changed from the first
                                                  transmit frequency.
```

MatLab Code A.3: (continued)

```
    bin(m)      = bin_1*s;                             % This streaches/compresses Doppler Shift bin size as lambda in changed.

    Dsf(m,:)    = f_1.*s;                              % This is the simulated Doppler shift values, one row for each fo value.

    f           = f_T(m) + Dsf(m,:);                   % Changes centering of Doppler shift frequencies from zero to f_T.

    f_edg(m,:) = [(min(f)-bin(m)/2) (f + bin(m)/2)];   % Edges of the frequency bins.
end
tor     = 2 * HeightRes/ c;                            % Pulse duration as a function of the height resolution of the pulse and speed of sound [s].

sigma_f = 0.37/(tor);                                  % Gaussian spectrum width parameter.
beta    = 2.*VT.^2./sigma_f.^2;                        % Turbulence Gaussian exponential term.
G_Ae    = RT_G_Ae(f_T);                                % Function used to estimate the transmit and receive gain of the sodar from measurements described in Chapter 3.

[alpha_dB, alpha] = RT_absorb(f_T,Temp,pa,rh);         % Acoustic atmospheric absorption coefficient [dB/m]. Have used the transmit frequency as the input frequency and have ignored the Doppler shifts.

if ~isempty(varargin)                                  % This loop calculates a Marshal and Palmer drop size distribution n_D if a rain rate I is the last input argument .

    scal_T  = varargin{1}(1);                          % Turbulence term scaling constant
    scal_R  = varargin{1}(2);                          % Rain term scaling constant
    varargout{1}=scal_T;
    varargout{2}=scal_R;
    if length(varargin) > 1
        I   = varargin{2};                             % Rain rate [mm/hr]
        G   = 4100./(I.^0.21);                         % The Marshall and Palmer Drop size distribution exponential term [1/m].

        n_D = exp(-G.* D);                             % Marshall and Palmer Drop size distribution Equation (1.11).

        varargout{3}=n_D;
    end
end
```

MatLab Code A.4: **RT_Sim_Echo.m**

```
function [P,R,T,Y,VT,Dsf] = RT_Sim_Echo(r,eror,f_T,c,Nfb,HeightRes,Temp,pa,rh,Fs,scal,I)
% {
This function file simulates the total backscattered power spectra P from rain and turbulence in Doppler shift bins,
for a range of transmission frequencies f_T. This is done by simulating the backscattered power spectra from rain
R and turbulence T separately with noise and then adding the two spectrum, P = R + T. These spectrum are
in frequency bins have their zero at f_T but correspond to a frequency range of between f_T and (f_T+ maximum
Doppler Shift). A Fast Fourier Transform (FFT) of a time domain recording of the echoes from rain and turbulence
is also simulated from P for each transmission wavelength. The input arguments are the transmission frequencies
f_T [Hz], the number of Doppler shift frequency bins Nfb, the sampling frequency Fs [Hz], the distance r [m] that
the scattering volume is from the sodar, the height resolution HeightRes [m] of the transmitted pulse, the rain
intensity I [mm/h], the speed of sound c [m/s], the temperature Temp [K], atmospheric pressure p_a [Pa], relative
humidity r_h [percent], the noise eror, and a scaling scal of the rain and turbulence terms.
% }
[x,D,VT,dD,bin,Dsf,f_edg,G_Ae,alpha,v,beta,scal_T,scal_R,n_D] = RT_Sim_functions(...
    f_T,c,Nfb,HeightRes,Temp,pa,rh,scal,I);
T          = zeros(length(f_T),length(D));
R          = zeros(length(f_T),length(D));
for m      = 1:length(x)
    a      = scal_T .* G_Ae(m) .* exp(-2.*alpha(m).*r); %   Product of sodar transmit/receive gain, atmo-
                                                              spheric absorption, and Gaussian term.
    T(m,:) = a .* v .* exp(-beta .* x(m).^2) .* x(m)^(4/3);
                                                         %   Simulated rain backscattered power spectra as a
                                                              matrix, with each row being the spectra for a
                                                              f_T value.
    b      = scal_R .* G_Ae(m) .* exp(-2.*alpha(m).*r) .* D.^6 .* n_D .* dD;
                                                         %   Product of rain drop reflectivity integrand, sodar
                                                              gain, and atmospheric absorption.
    R(m,:) = b .* x(m).^4;                               %   Simulated rain backscattered power spectra as a
                                                              matrix, with each row being the spectra for a
                                                              f_T value.
end
N          = eror .* randn(size(T));                     %   Noise
P          = R + T + N;                                  %   Total backscatter spectral power is assumed to be
                                                              the sum of the rain, turbulence, and noise spectra.

Ns  = ceil(2*HeightRes*Fs/c);                            %   Number of samples per height resolution interval.

Y          = zeros(length(f_T),Ns);                      %   Generates a m by k matrix, where the m_th row
                                                              will corresponds to Ns simulated FFT values for
                                                              the m_th transmit frequency.
for m      = 1:length(f_T)
    N      = floor(f_edg(m,:) .* Ns./Fs);                %   Index number of the simulated FFT samples, of
                                                              the time domain recordings, corresponding to the
                                                              edges of the Doppler shift frequency bins.
    for k  = 1:length(N)-1
        n =  N(k):N(k+1);
        Y(m,n) = ones(1 , length(n)).*P(m,k);            %   Simulated backscattered power in frequency bins.
    end
   for k = 1:floor(Ns/2), Y(m,Ns-k-1) = Y(m,k); end      %   This simulates the upper frequency end of the
                                                              FFT values. Have ignored any complex compo-
                                                              nent of Y.
end
```

MatLab Code A.5: **RT_Sim_Fit.m**

```
function [R_fit,T_fit,RMSD,b] = RT_Sim_Fit(P,r,f_T,c,Nfb,HeightRes,Temp,pa,rh,scal)
scal_r=1; scal_t=1;
% {
This function file gives a fit to the rain R_fit and turbulence T_fit components of the backscattering power spectra
P. It uses a least squares fittiing of P to obtain coefficents of rain and turbulence functons. A root mean squared
sum of the differences RMSD between P and the sum of R_fit and T_fit is also calculated to check the accuracy
of the fitting. The input arguments are the backscattered power spectra P, the transmission frequencies f_T [Hz],
the number of Doppler shift frequency bins Nfb, the height resolution HeightRes [m] of the transmitted pulse,
the rain intensity I [mm/h], the speed of sound c [m/s], the temperature Temp [K], atmospheric pressure p_a [Pa],
relative humidity r_h [percent], the distance r [m] that the scattering volume is from the sodar, the noise eror, and
a scaling scal of the rain and turbulence terms.
% }
warning off                                         %   This turns off a scaling warning. No problems
                                                        have been detected relating to scaling but it may
                                                        an issue to be considered.
[x,D,VT,dD,bin,Dsf,f_edg,G_Ae,alpha,v,beta,scal_T,scal_R] = RT_Sim_functions(f_T,c,Nfb,HeightRes, ...
    Temp,pa,rh,scal);
for k=1:Nfb                                         %   Iterates through Doppler shift frequency bins.
    for m = 1:length(f_T)                           %   Iterates through transmit frequencies, f_T.
        f1(m)  = max(G_Ae(m) .* exp(-2.*alpha(m).*r) .* v(k) .* exp(-beta(k) .* x(m).^2) .* ...
                 x(m).^(4/3), 1e-5);                %   Turbulence backscattered power spectra function
                                                        which takes into account atmospheric absorption
                                                        and transmit and receive gain of the sodar. This
                                                        value is limited so that it cannot reach zero. This
                                                        is to prevent the fitting coefficient b_1 going to in-
                                                        finity.
        f2(m)  = G_Ae(m) .* exp(-2.*alpha(m).*r) .* x(m).^4;
                                                    %   Rain backscattered power spectra function which
                                                        takes into account atmospheric absorption and
                                                        transmit and receive gain of the sodar.
    end
    A       = [f1(:) f2(:)];                        %   A matrix whose first column is the rain functions
                                                        and whose second column is the turbulence func-
                                                        tions.
    b(:,k)  = (A'*A)\(A'*P(:,k));                   %   The coefficients of the rain and turbulence func-
                                                        tions from a least squares fitting of P.

    T_fit(:,k) = b(1,k) .* f1(:);                   %   Least squares fitted of the turbulence functions.
    R_fit(:,k) = b(2,k) .* f2(:);                   %   Least squares fitted of the rain functions.
 end
SqrDiff = (P - (R_fit + T_fit)).^2;                 %   Square of the sum of the differences between P
                                                        and the sum of the fitted rain and turbulence com-
                                                        ponents.
RMSD     = sqrt(sum(SqrDiff)./length(P(:,1)));      %   Root mean square of the sum of the differences
```

MatLab Code A.6: **RT_GUI_Pulse_Record_2.m**

```
function varargout = RT_GUI_Pulse_Record_2(varargin)
% {
This GUI function file was written to generate multi-frequency acoustic soundings by the sodar described in Chapter
3. It generates multi-frequency acoustic pulses, records after each pulse, and saves the recorded data to file.
% }

if nargin == 0                                              %    LAUNCH GUI
    fig = openfig(mfilename,'reuse');
    handles = guihandles(fig); guidata(fig, handles); gcbo;
        if nargout > 0, varargout{1} = fig;, end
% Settings made before user has access to GUI.
clc; c = RT_SoundSpeed; DefaultRunTime = 0.01;
DefaultMaxRange='100'; DefaultHeightRes = '10';
Defaultf_T = '3.2, 8.8'; DefaultFs = '100';
Defaulttor = sprintf(['%0.1f'], 2 * eval(DefaultHeightRes)/c*1e3);
DefaultFilePath = 'C:/Thesis/SodarRawData/Trial/';
set(handles.RunTime,'String',DefaultRunTime); set(handles.HeightRes,'String',DefaultHeightRes)
set(handles.OtherMaxRange,'String',DefaultMaxRange,'Visible','off');
set(handles.f_T,'String',Defaultf_T); set(handles.Fs,'String',DefaultFs)
set(handles.tor,'String',Defaulttor); set(handles.FilePath,'String',DefaultFilePath)
set(handles.PleaseWait,'Visible','off'); set(handles.Stop_pushbutton,'Visible','off')
set(handles.Start_pushbutton,'Visible','on'); set(handles.Question,'visible','off')
set(handles.Yes,'visible','off'); set(handles.No,'visible','off')
set(handles.InfoFrame,'visible','off'); set(handles.Ok_pushbutton,'visible','off')
set(handles.Info_Text,'visible','off'); set(handles.RecType,'visible','off')
set(handles.WeathCnd,'visible','off'); set(handles.WindCnd,'visible','off')
set(handles.Temp,'visible','off'); set(handles.pa,'visible','off')
set(handles.rh,'visible','off'); set(handles.FilePath,'visible','off')
set(handles.comment,'visible','off'); set(handles.location,'visible','off')
axes(handles.axesFFT);cla(handles.axesFFT);
axis([0 eval(DefaultFs) 0 eval(DefaultMaxRange)])
axes(handles.axesFit); cla(handles.axesFit);
axis([0 5 0 eval(DefaultMaxRange)])
% INVOKE NAMED SUBFUNCTION OR CALLBACK
elseif ischar(varargin{1}),
    try, [varargout{1:nargout}] = feval(varargin{:});   %    FEVAL switchyard
    catch, disp(lasterr); end, end
% ------------------------------------------------------------------
function varargout = Start_pushbutton_Callback(h,eventdata,handles,varargin)

global NOTSTOP; global OK; global RECTYPE;                  %    Start program
set(handles.Start_pushbutton,'Visible','off')
set(handles.PleaseWait, 'Visible','on'); pause(0.001)
NOTSTOP = 1; OK = 0;
disp('Program Running')
c         = RT_SoundSpeed;
RunTime   = eval(get(handles.RunTime,'String')) * 3600;  %    Time program runs for [s].
MaxRange  = MaxRange_Callback(h, eventdata, handles, varargin);
                                                            %    Maximum range program records from.
HeightRes = floor(eval(get(handles.HeightRes,'String')));%    Height resolution.
tor       = 2 * HeightRes/ c;                               %    Pulse duration [ms]
f_T        = eval([ '[' get(handles.f_T,  'String')      '].* %    Transmission frequencies [Hz]
1e3']);
PlotFR    = eval([ '[' get(handles.PlotFR, 'String') ']']); %    Axis range for FFT plot [kHz]
Fs        = eval(get(handles.Fs,        'String')) * 1e3;%    Sampling frequency [kHz]
```

MatLab Code A.6: (continued)

```
RecType   = RecType_Callback(h, eventdata, handles, varargin);

WeathCnd  = WeathCnd_Callback(h, eventdata, handles, varargin);

set(handles.FilePath,'string',['C:/Thesis/SodarRawData/' RecType '/'])
ShowInfoOn(handles)
while ~ OK, pause(0.00000001), end
clear global WEATHCND
RecType   = RecType_Callback(h, eventdata, handles, varargin);

WeathCnd  = WeathCnd_Callback(h, eventdata, handles, varargin);

WindCnd   = WindCnd_Callback(h, eventdata, handles, varargin);

Temp      = eval(get(handles.Temp, 'String')) + 273.16;  %  Atmospheric temperature [K].
rh        = eval(get(handles.rh, 'String'));             %  Relative humidity [percent]
pa        = eval(get(handles.pa, 'String')) * 1e3;       %  Atmospheric Pressure [Pa]
location  =      get(handles.location,   'String');      %  Location string
comment   =      get(handles.comment,   'String');       %  Comment string
FilePath  =      get(handles.FilePath, 'String');        %  File path to save recorded data to.
ShowInfoOff(handles); PausT = 2-(2*MaxRange/c);          %  Minimum time between successive pulses [s]
axes(handles.axesFFT); axis([PlotFR(1) PlotFR(2) 0 MaxRange]); cla(handles.axesFFT)


RT_Pulse_Record(RunTime, MaxRange, HeightRes, tor, f_T, PlotFR, Fs, c, location, comment, PausT, ...
                FilePath, Temp, pa, rh, RecType, WeathCnd, WindCnd, handles);


clear global NOTSTOP; clear global RECTYPE
disp('Program Finished')
% ----------------------------------------------------------------------
function RT_Pulse_Record(RunTime, MaxRange, HeightRes, tor, f_T, PlotFR, Fs, c, location, comment, ...
                         PausT, FilePath, Temp, pa, rh, RecType, WeathCnd, WindCnd, handles)

% {
This function file generates audio pulses of duration tor at different transmission frequencies. Following each pulse
a recorded is made for a period RecTime at sampling frequency Fs. The program runs for a time of FinishTim.
% }

global NOTSTOP
T = 25.*ones(1,4);                                       %  This is a dummy vector. It ensures that the con-
                                                            dition T(4)  = C(4) holds when program is first
                                                            started and hence that a new file is created.
start  = 1;                                              %  Dummy value to determine if program has just
                                                            stared.
f = RT_Fprintf_Format(MaxRange, HeightRes, Fs, c);       %  Generates format string f for fprintf command to
                                                            save data to file.
n = 1; N = 0; to = clock;                                %  Start time of program.
set(handles.PleaseWait,'Visible','off'); set(handles.Stop_pushbutton,'Visible','on')
pause(0.001)
while etime(clock,to) < RunTime & NOTSTOP;
    pause(0.001); axes(handles.axesFFT); cla(handles.axesFFT)
                                                         %  Clears axes.
    hold on; N = N + 1; C = clock;                       %  Current date and time
```

MatLab Code A.6: (continued)

```
    if T(4) ~= C(4);                              %  Compares the hour component of the two vectors.
        File_Name = RT_File_Create(RunTime, MaxRange, HeightRes, f_T, tor, Fs, Temp, pa, rh, ...
            location, comment, FilePath, RecType, WeathCnd, WindCnd, start);
                                                  %  Creates a file called File_Name when program is
                                                     first run and subsequently when the hour changes.
                                                     A header line is included.
    end; start = 0; T = C; X = [];
% The following piece of code is designed to generate a pulse at different frequencies and record after each pulse.

    for k       = 1:length(f_T)
        Pulse   = RT_Pulse_Gen(f_T(k),Fs,tor);    %  Generates a pulse and a vector.
        tpulse = clock;
        RecData = wavrecord(floor(2*Fs*MaxRange/c), Fs, 'double');
                                                  %  This records after the initial pulse was sent.
        fid     = fopen(File_Name,'at');
        strA=['\n\nT'  '="'];
        strB=[RT_DateTime('T:') '"; f_T='];
        strC=[num2str(f_T(k)/1e3,'%2.1f') ';' ];
        fprintf(fid, [strA strB strC]);
        fprintf(fid, f, RecData); fclose(fid); pause(0.001)
        RT_PlotFFT(RecData,MaxRange,HeightRes,c,Fs,PlotFR,k,handles)

        X(k,:) = RecData'; time = etime(clock,tpulse);
        if (PausT - time) > 0, pause(PausT-time), end  %  Attempts to prevent recording data from above a
                                                          certain height.
        pause(0.001)                              %  Helps to ensure plotting is processed before con-
                                                     tinuing.
    end; hold off
end
set(handles.Question, 'visible', 'off'); set(handles.Yes,'visible','off')
set(handles.No,'visible','off');  set(handles.Stop_pushbutton,'visible','off')
set(handles.Start_pushbutton, 'visible', 'on')
% ----------------------------------------------------------------------
function f = RT_Fprintf_Format(MaxRange, HeightRes, Fs, c)
% This sub function creates a format string f used by the fprintf command to write recorded data to file.
MaxRNs   = floor(2*Fs/c*MaxRange);                %  Maximum number of samples per pulse.
HResNs   = ceil(2*Fs/c*HeightRes);                %  Maximum number of samples per height resolu-
                                                     tion interval.
fstr = []; n = 0; R = HeightRes/2;
for k = 1:MaxRNs                                  %  One iteraton for each data point.
    if k > n

        if k == 1
            fstr = [fstr '\nM_h' sprintf(['%03.0f'], R) '=[' ];
        elseif k < MaxRNs
            fstr = [fstr '];\nM_h' sprintf(['%03.0f'],R) '=['];
        end; n = n + HResNs; R = R + HeightRes;
    end; fstr = [fstr '%+f '];
end; f = [fstr '];' ];
% ----------------------------------------------------------------------
function File_Name = RT_File_Create(RunTime, MaxRange, HeightRes, f_T, tor, Fs, Temp, pa, rh, ...
    location, comment, FilePath, RecType, WeathCnd, WindCnd, start)

% {
```

MatLab Code A.6: (continued)

```
This sub function creates a file called File_Name.  The function RT_DateTime.m is used in creating the file
name. It then adds a comment and other text to the top of the file.
% }

global WEATHCND; persistent N
if start, N = 1;, else, N = N + 1;, end                 %   File continued from previous hour
DateTime  = RT_DateTime('file');                        %   Creates a string of the date and time for file nam-
                                                            ing.
File_Name = [FilePath '/RT_R_' DateTime '_' WEATHCND '.txt'];
for k=1:ceil(MaxRange/HeightRes); r(k)=(k-0.5)*HeightRes; end;
                                                        %   Mid-heights of Height resolution intervals
fid   = fopen(File_Name,'w');                           %   Create and open a new file.
% The following lines are to produce the file's header line.
fprintf(fid,"'SODAR_RAW_DATA:%s,%s,%s.";Hour=%s',RecType,WeathCnd,WindCnd,num2str(N));
fprintf(fid,';T="%s";RunTime=%4.3f;MaxRange=%3.0f;',RT_DateTime('T:'),RunTime/3600,MaxRange);
fprintf(fid,'HeightRes=%2.0f;%stor=%2.1f;%s',HeightRes,range_str(r),tor*1e3,f_T_str(f_T))
fprintf(fid,'Fs=%3.1f;Temp=%3.1f;pa=%6.3f;rh=%2.0f;',Fs/1e3,Temp-273.16,pa/1e3,rh)
fprintf(fid,'Location=%sComment=%s',SetStr(location,100),SetStr(comment,200));
fprintf(fid,'%% NOTE:Format_of_T_is_"YYMMDDhhmmss"_and_the_units_of_RunTime,MaxRange,HeightRes,')
fprintf(fid,'r,tor,fo,Fs,Temp,pa,and_rh_are_in_hrs,m,m,m,ms,kHz,kHz,deg_C,kPa,and_%%_respectively.')
fclose(fid);
% ------------------------------------------------------------------------
function DateTime = RT_DateTime(varargin)
% This function file outputs the date as a vector, string, or file name.
C = clock;
if strcmp(varargin,'file')
    DateTime = sprintf('%0.2d%0.2d%0.2d_%0.2d_%0.2d',C(1),C(2),C(3),C(4),C(5));
elseif strcmp(varargin,'T:')
    DateTime = sprintf('%02.0f%02.0f%02.0f%02.0f%02.0f%02.0f',C(1)-2000,C(2),C(3),C(4),C(5),C(6) );
elseif strcmp(varargin,'str'),DateTime = DATESTR(now,0);,end
% ------------------------------------------------------------------------
function Pulse = RT_Pulse_Gen(f_T,Fs,tor,varargin)
% {
This function file generates a pulse of length tor.  It is composed of a sin wave of frequency f_T with a Gausian
envelope. It is sampled at a frequency Fs. Fs must be larger then the Nyquist frequency of f_T, ie. 2*f_T.
% }
Fs_p  = 44100; t = 0:(1/Fs):tor;                        %   Time [s]
A     = 1;                                              %   Maximum level of MatLab sound.
alpha = 10/tor^2;
% Gausian exponential term normalised by tor to attempt to make pulse shape independent of pulse length.
G     = A.* exp(- alpha*(t - tor/2).^2);                %   Gausian envelope.
Pulse = G .* sin(2.*pi.*f_T.*t);                        %   A vector consisting of a sin wave of frequency f_T
                                                            with a Gausian envelope.
wavplay(Pulse, Fs_p, 'sync');                           %   Plays vector Pulse at sampling frequency Fs.
% ------------------------------------------------------------------------
function RT_PlotFFT(RecData,MaxRange,HeightRes,c,Fs,PlotFR,kth_val,handles)

% {
This function file plots the FFT for heights separated by HeightRes. Note that each FFT point Y has a Doppler
shift frequency separation of FS/length(Y). This peice of code plots the FFT points that lie between frequencies
range of PlotFR(1) to PlotFR(2).
% }
NumSamp = ceil(2 * Fs * HeightRes /c);                  %   Number of samples within each height resolution
                                                            interval.
NH      = ceil(MaxRange/HeightRes);                     %   Number of height resolution intervals used.
```

MatLab Code A.6: (continued)

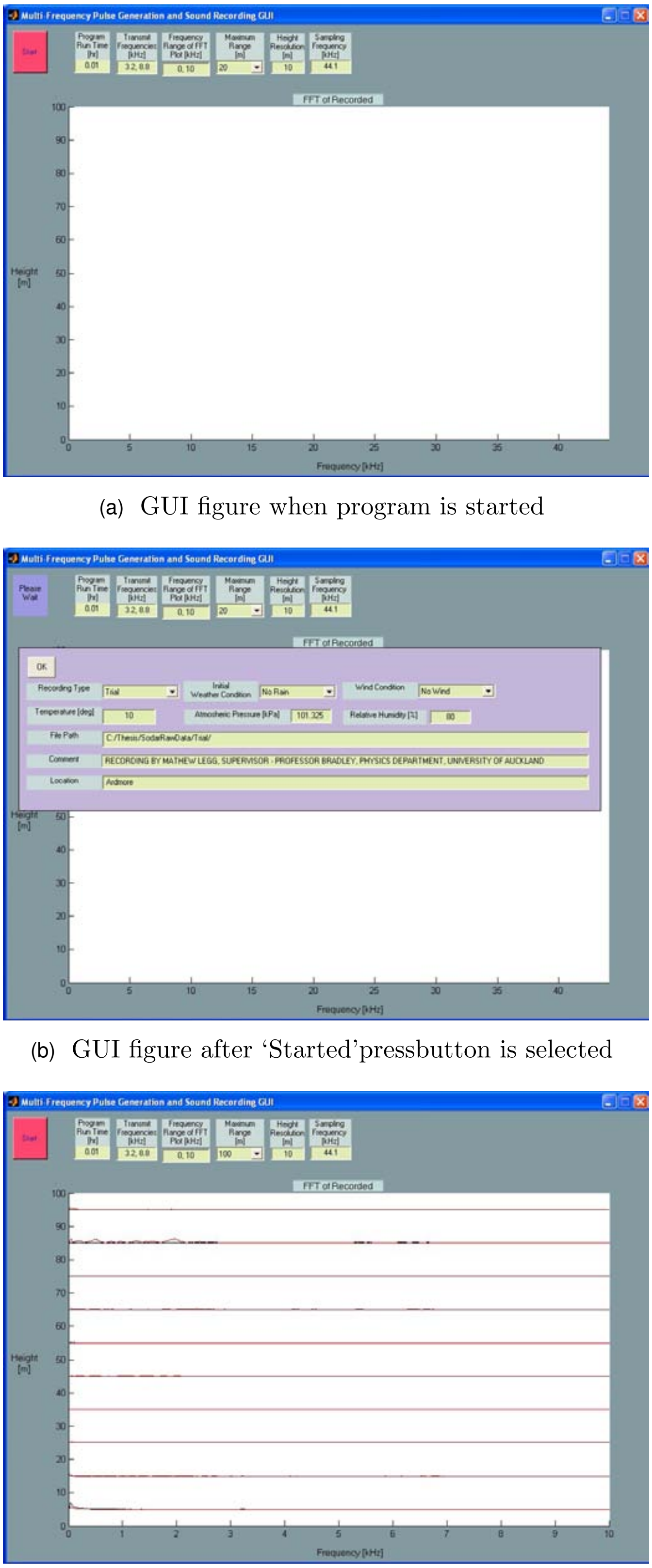
```
L      = length(RecData);                           % Number of samples recorded after the pulse.
fs     = zeros(NH,NumSamp); Ys = zeros(NumSamp,NH);
axes(handles.axesFFT)                               % Selects desired axes on GUI.
N1 = 0; hold on
for k = 1 : NH
    MidHeight = (k-0.5) * HeightRes;                % Mid-height if k_th height resolution interval.
    N2        = min(N1 + NumSamp, L);               % Number of samples recorded by upper limit of k_th
                                                      height resolution interval.
    X         = RecData(N1+1:N2);                   % Recorded data samples within this height resolu-
                                                      tion interval.
    Y         = abs(fft(X))’;
    f         = [Fs/length(Y):Fs/length(Y):Fs]./1e3;
    N1        = N2;                                 % Resets number of samples by lower limit of k_th
                                                      height resolution interval to upper limit for next
                                                      iteration.
    m=1; n=1;
    while f(m) < PlotFR(1), m = m+1;, end           % Minimum frequency index value.
    while f(n) < PlotFR(2), n = n+1;, end
    f = f(m:n);                                     % Selected Doppler shift frequencies.
    Ymn = Y(m:n);                                   % Selected FFT values.
    col = linecolour(kth_val); plot(f,Ymn+MidHeight,col)
end; hold off
% ----------------------------------------------------------------------
function Ok_pushbutton_Callback(h, eventdata, handles);
% This push button allows the program to proceed.
global OK; OK = 1;
% ----------------------------------------------------------------------
function NOTSTOP = Stop_pushbutton_Callback(h, eventdata, handles, varargin);

% This pushbuttom allows the user to stopped the program by making visable a YES or NO pushbuttom.
set(handles.Stop_pushbutton, ’visible’, ’off’); set(handles.Question, ’visible’, ’on’)
set(handles.Yes, ’visible’, ’on’); set(handles.No, ’visible’, ’on’)
% ----------------------------------------------------------------------
function yes = Yes_Callback(h, eventdata, handles, varargin)
% {
Allows user to answer ’yes’to the question ’Do you wish to stop program‘. This function cause the output to equal
one when its pushbutton is used. This will cause the pulse and record process to stop.
% }

global NOTSTOP; NOTSTOP = NOTSTOP*0;
pause(0.001)
set(handles.Question, ’visible’, ’off’); set(handles.Yes, ’visible’, ’off’)
set(handles.No, ’visible’, ’off’); set(handles.Start_pushbutton, ’visible’, ’on’)
% ----------------------------------------------------------------------
function n = No_Callback(h, eventdata, handles, varargin)
% Answer to question do you wish to stop program.
set(handles.Question,’visible’,’off’);set(handles.Yes,’visible’,’off’)
set(handles.No,’visible’,’off’);set(handles.Stop_pushbutton,’visible’,’on’)
% ----------------------------------------------------------------------
function RecType = RecType_Callback(h, eventdata, handles, varargin)

% This function allows selection of a RecType from list.
global RECTYPE
string_list = get(handles.RecType,’String’);
val = get(handles.RecType,’Value’);
```

MatLab Code A.6: (continued)

```
RecType = (string_list{val});                            %    Convert from cell array
RecType  = strrep(strrep(RecType, ' ','_'),'/','_');
RECTYPE = (RecType);
filepath = ['C:/Thesis/SodarRawData/' RECTYPE '/'];
set(handles.FilePath, 'string', filepath)
% -----------------------------------------------------------------
function WeathCnd = WeathCnd_Callback(h, eventdata, handles, varargin)

% This function allows selection of a Weather Condition from a list.
global WEATHCND
string_list = get(handles.WeathCnd,'String');
val = get(handles.WeathCnd,'Value');
WeathCnd = (string_list{val});                           %    Convert from cell array
Wstr={'NR' 'LR' 'MR' 'HR'}; WEATHCND = char(Wstr(val));
% -----------------------------------------------------------------
function WindCnd = WindCnd_Callback(h, eventdata, handles, varargin)

% This function allows selection of a Wind Condition from a list.
string_list = get(handles.WindCnd,'String');
val = get(handles.WindCnd,'Value');
WindCnd = string_list{val};                              %    Convert from cell array
% -----------------------------------------------------------------
function MaxRange = MaxRange_Callback(h, eventdata, handles, varargin)

% This function allows selection of a maximum range either from a list or to enter an "other" value. [m]
string_list = get(handles.MaxRange,'String');
val = get(handles.MaxRange,'Value');
if val ~= length(string_list)
    MaxRange = floor(eval(string_list{val}));            %    Convert from cell array
else
    set(handles.OtherMaxRange,'Visible','on')
    RecTime = (eval(get(handles.OtherMaxRange,'string')));
end
% -----------------------------------------------------------------
function varargout = HeightRes_Callback(h, eventdata, handles, varargin)

% This function calculates the required tor value given an entered height resolution.
c         = RT_SoundSpeed
HeightRes = floor(eval(get(handles.HeightRes,'String')));%    Height resolution.
tor       = 2 * HeightRes/ c;                            %    Pulse duration [ms]
set(handles.tor, 'String', sprintf('%0.1f', tor*1e3) )
% -----------------------------------------------------------------
function c = RT_SoundSpeed
% This function gives the speed of sound. This function is included incase at a later date a formula desired.
c = 330;                                                 %    [m/s]
% -----------------------------------------------------------------
function s = f_T_str(f_T)
% This sub function makes a string of the transmission frequencies of the form f_T=[..];.
s = 'f_T=[';
for k=1:length(f_T)
    if k < length(f_T), s = [s num2str( f_T(k)/1e3,'%1.1f') ',
'];
    else, s = [s num2str( f_T(k)/1e3,'%1.1f') '];'];, end
end
% -----------------------------------------------------------------
```

MatLab Code A.6: (continued)

```
function s = range_str(r)
% This sub function makes a string of the ranges r=[..];.
s = ’r=[’;
for k=1:length(r)
    if k < length(r), s = [s num2str( r(k),’%3.0f’) ’,
’];
    else, s = [s num2str( r(k),’%3.0f’) ’];’];, end
end
% ----------------------------------------------------------------------------
function s = SetStr(string, MaxNum)
% {
This sub-function makes a string of set length from the comment. If its length is longer it ignors the end of the
comment.
% }
s = [””];
for k = 1:MaxNum
    if k <= length(string), s = [s string(k)];, end
end
s = [s ”’;’]; s = upper(s);
% ----------------------------------------------------------------------------
function col = linecolour(n)
% This function gives the colour to plot a line.
m = num2str(n); n=eval(m(length(m)))+1;
colstr = [’k’,’r’,’b’,’g’,’y’,’m’,’c’,’b’,’y’,’g’];
col = colstr(n);
% ----------------------------------------------------------------------------
function ShowInfoOn(handles)
set(handles.InfoFrame, ’visible’,’on’); set(handles.Ok_pushbutton, ’visible’,’on’)
set(handles.Info_Text, ’visible’,’on’); set(handles.RecType, ’visible’,’on’)
set(handles.WeathCnd, ’visible’,’on’);  set(handles.WindCnd, ’visible’,’on’)
set(handles.Temp, ’visible’,’on’);      set(handles.pa, ’visible’,’on’)
set(handles.rh, ’visible’,’on’);        set(handles.FilePath, ’visible’,’on’)
set(handles.comment, ’visible’,’on’);   set(handles.location, ’visible’,’on’)
% ----------------------------------------------------------------------------
function ShowInfoOff(handles)
set(handles.InfoFrame, ’visible’,’off’); set(handles.Ok_pushbutton, ’visible’,’off’)
set(handles.Info_Text, ’visible’,’off’); set(handles.RecType, ’visible’,’off’)
set(handles.WeathCnd, ’visible’,’off’);  set(handles.WindCnd, ’visible’,’off’)
set(handles.Temp, ’visible’,’off’);      set(handles.pa, ’visible’,’off’)
set(handles.rh, ’visible’,’off’);        set(handles.FilePath, ’visible’,’off’)
set(handles.comment, ’visible’,’off’);   set(handles.location, ’visible’,’off’)
% ----------------------------------------------------------------------------
function varargout = f_T_Callback(h,eventdata,handles,varargin)
function varargout = tor_Callback(h,eventdata,handles,varargin)
function varargout = Fs_Callback(h,eventdata,handles,varargin)
function varargout = Temp_Callback(h,eventdata,handles,varargin)
function varargout = pa_Callback(h,eventdata,handles,varargin)
function varargout = rh_Callback(h,eventdata,handles,varargin)
function varargout = FilePath_Callback(h,eventdata,handles,varargin)
function varargout = location_Callback(h,eventdata,handles,varargin)
function varargout = comment_Callback(h,eventdata,handles,varargin)
function varargout = RunTime_Callback(h,eventdata,handles,varargin)
function varargout = PlotFR_Callback(h,eventdata,handles,varargin)
function varargout = OtherMaxRange_Callback(h,eventdata,handles,varargin)
```



Program Run Time [hr]: 0.01
Transmit Frequencies [kHz]: 3.2, 8.8
Frequency Range of FFT Plot [kHz]: 0, 10
Maximum Range [m]: 20
Height Resolution [m]: 10
Sampling Frequency [kHz]: 44.1

(a) GUI figure when program is started

Program Run Time [hr]: 0.01
Transmit Frequencies [kHz]: 3.2, 8.8
Frequency Range of FFT Plot [kHz]: 0, 10
Maximum Range [m]: 20
Height Resolution [m]: 10
Sampling Frequency [kHz]: 44.1

Recording Type: Trial
Initial Weather Condition: No Rain
Wind Condition: No Wind
Temperature [deg]: 10
Atmosheric Pressure [kPa]: 101.325
Relative Humidity [%]: 80
File Path: C:/Thesis/SodarRawData/Trial/
Comment: RECORDING BY MATHEW LEGG, SUPERVISOR - PROFESSOR BRADLEY, PHYSICS DEPARTMENT, UNIVERSITY OF AUCKLAND
Location: Ardmore

(b) GUI figure after ‘Started’pressbutton is selected

Program Run Time [hr]: 0.01
Transmit Frequencies [kHz]: 3.2, 8.8
Frequency Range of FFT Plot [kHz]: 0, 10
Maximum Range [m]: 100
Height Resolution [m]: 10
Sampling Frequency [kHz]: 44.1

(c) GUI figure while program running.

Figure A.1: Figure component of the GUI used to run the sodar

MatLab Code A.7: **RT_Average_RawData.m**

```
function [Y, YN] = RT_Average_RawData(filename,fT,r)
% {
This function file cycles though the sodar raw data file and finds the average of the FFT values for a given height
resolution interval and a given transmit frequency.
% }
No_Y = 0; f_T = ”; r_str = sprintf(’M_h%03.0f’,r);
fid      = fopen(filename); YN = 0; Ynoise = 0;
while 1                                           %   This cycles though the lines of the sodar raw data
                                                      file.

    tline = fgets(fid);
    if ~ischar(tline), break, end                 %   If end of file, break.
    if ~isempty(tline)
        if ~isempty(strmatch( ’T’,tline(1) ))     %   Checks if line contains the transmission time and
                                                      transmission frequency.
            eval(tline);                          %   Evaluates the string tline
            f_T = f_T.*1e3;                       %   Converts from kHz to Hz.
        end
        if f_T == fT                              %   Checks if the line is in a segment related to the
                                                      transmission frequency being considered.
            if ~isempty(strmatch(’Noise’,tline))
                eval(tline); Ynoise = abs(fft(Noise));
            end
            if ~isempty(strmatch(r_str,tline))    %   Checks if the current line is a vector correspond-
                                                      ing to a recording from the relevant height reso-
                                                      lution interval.
                eval(tline);
                y=eval(r_str);
                Y_FFT = abs(fft(y));              %   Preform fft of this vector
                if ~exist(’Y’), Y=zeros(size(Y_FFT)); end
                Y = Y + Y_FFT;
                YN = Y - Ynoise;
                No_Y = No_Y + 1;                  %   Number to be used for averaging
            end
        end
    end
end
Y = Y./No_Y; YN = YN./No_Y;                       %   This complets the averaging of the FFT values for
                                                      a single f_T values and a single hight resolution
                                                      interval.
```

MatLab Code A.8: **RT_Fit_RawData.m**

```
function [Y,YN,f,VT,P,R_fit,T_fit,RMSD] = RT_Fit_RawData(filename,HeightInt,c,Nfb,Min_fT,Max_fT,scal)
% {
This function file provides a fit to the turbulence T_fit and rain R_fit components of backscattered power spectra,
contained in the sodar's raw data for a given height resolution interval HeightInt. The input arguements are the
file name filename, the height resolution interval HeightInt, the speed of sound c, the number of frequency bins
Nfb, and the rain and turbulence scalling values scal.
% }
fid   = fopen(filename);                                 %   File is opened.
tline = fgets(fid);                                      %   Reads and evaluates header line containing pa-
                                                             rameters used in sampling, etc.
eval(tline);
if Nfb > 40 * HeightRes.*1e3 * min(f_T)/(1 * c^2)
    warning('Too many frequency bins');
end
f_T = f_T.*1e3; Temp = Temp + 273.15; pa = pa * 1e3; Fs = Fs * 1e3;
                                                         %   Converts the variables in the file which are not on
                                                             SI units to SI units.
r=r(HeightInt);
fclose(fid);
[x,D,VT,dD,bin,Dsf,f_edg,G_Ae,alpha,v,beta,scal_T,scal_R] = RT_Sim_functions(f_T,c,Nfb,HeightRes, ...
    Temp,pa,rh,scal);                                    %   This function file uses the information contained
                                                             in the header line of the raw data file to calculate
                                                             terms required to create frequency bins and also
                                                             to preform the least sqaures fit.
a=(Min_fT-1)/0.1+1;
b=(Max_fT-1)/0.1+1;
for m = a:1:b
    [Y, YN] = RT_Average_RawData(filename,f_T(m),r);     %   This function preforms an average over the file of
                                                             the FFT values for a single transmission frequency
                                                             f_T and a single height resolution interval r.
    f         = [Fs/length(Y):Fs/length(Y):Fs]./1e3;
    N = floor(f_edg(m,:) .* length(Y)/Fs);
    for k = 1:length(N)-1
        P(m,k) = mean(Y( N(k):N(k+1) ));                 %   Creates a matrix P which contains the Doppler
                                                             shift bins as columns and has a row for each trans-
                                                             mit frequency.
    end
    disp(sprintf('f_T=%3.1f',f_T(m)/1e3)); figure
    vt = (f.*1e3-f_T(m))./max(Dsf(m,:))*max(VT);
    vteg = (f_edg.*1e3-f_T(m))./max(Dsf(m,:))*max(VT);
    dVT=(VT(2)-VT(1))/2;
    close all; figure
    plot(f,20.*log10(YN),'g',f,20.*log10(YN),'.r')
    xlabel('Frequency [kHz]')
    ylabel(sprintf('20log_10(Echo Strength)[dB]'))
    grid; figure
    plot(vt,20.*log10(YN),'g',vt,20.*log10(YN),'.r')
    xlabel('Velocity [m/s]')
    ylabel(sprintf('20log_10(Echo Strength)[dB]'))
    axis([-5 20 -1 55]); grid; hold on
    for k=1:length(VT)
        plot([(VT(k)-dVT) (VT(k)+dVT)],[20.*log10(P(m,k)) 20.*log10(P(m,k))])

        plot([(VT(k)-dVT) (VT(k)-dVT)],[0 20.*log10(P(m,k))])
```

MatLab Code A.8: (continued)

```
        plot([(VT(k)+dVT) (VT(k)+dVT)],[0 20.*log10(P(m,k))])

    end
end
end
[R_fit,T_fit,RMSD] = RT_Sim_Fit(P,r,(f_T.*1e3),c,Nfb,HeightRes,Temp,pa,rh,scal);
                                                       %    This function file preforms a least squares fit to
                                                            separate the rain and turbulence components of
                                                            the backscattered power spectra.
```

MatLab Code A.9: **RT_Pulses.m**

```
function RT_Pulses(f_T, tau, N)
% {
This function file produces voltage waveform pulses via the computers sound card headphone plug. These pulses
are sine waves of frequency f_T [Hz] with a Gaussian envelope of duration tau [s]. The pulse is repeated N times
with a second pause between each pulses. This function file was used to measure the characteristics of the speakers,
speaker, array, and sodar.
% }
Fs    = 441000;                                        %    Sampling Frequency [Hz]
A     = 1;                                             %    Maximum 'int16' level of MatLab sound.
t     = 0:(1/Fs):tau;                                  %    Time [s]
alpha = 16/tau^2;                                      %    Gaussian exponential term
G     = A.*exp(- alpha.*(t - tau/2).^2);               %    Gaussian envelope
Pulse = G .* sin(2.*pi.*f_T.*t);                       %    Pulse - sine wave with a Gaussian envelope.
plot(t,Pulse)
for k = 1:N                                            %    Repeats pulse N times.
    wavplay(Pulse, Fs, 'sync');                        %    Converts vector into a voltage waveform via the
                                                            computer's sound card headphone output.

    pause(1)                                           %    Pauses one second between pulses
end
```

B

Location Maps for the Ardmore Field Site

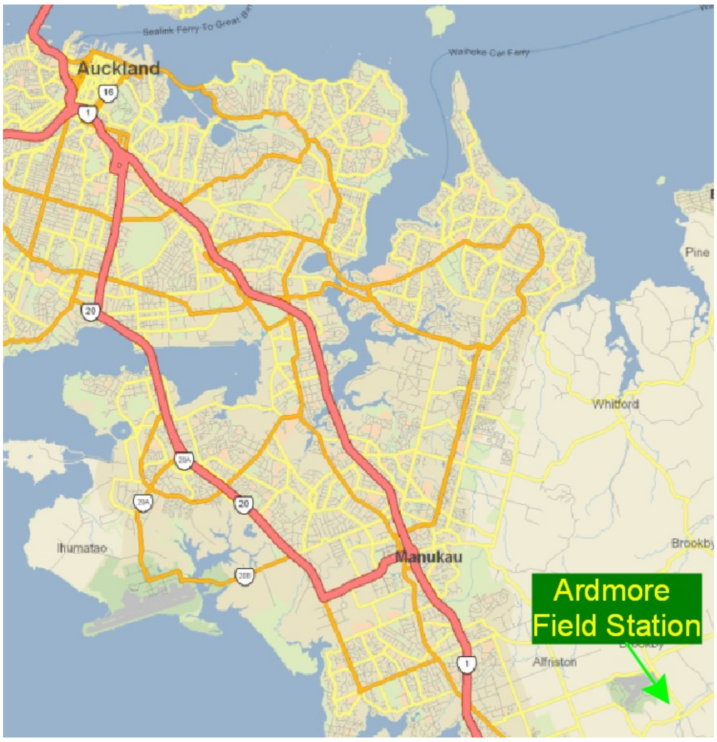


Figure B.1: Map of Auckland

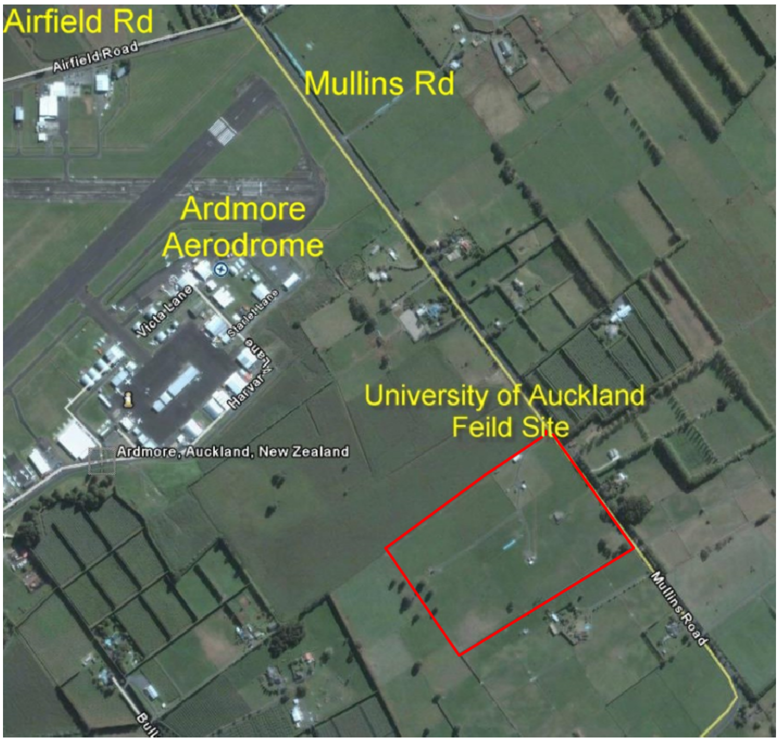


Figure B.2: Areal photograph of Ardmore showing the location of the University of Auckland's field station

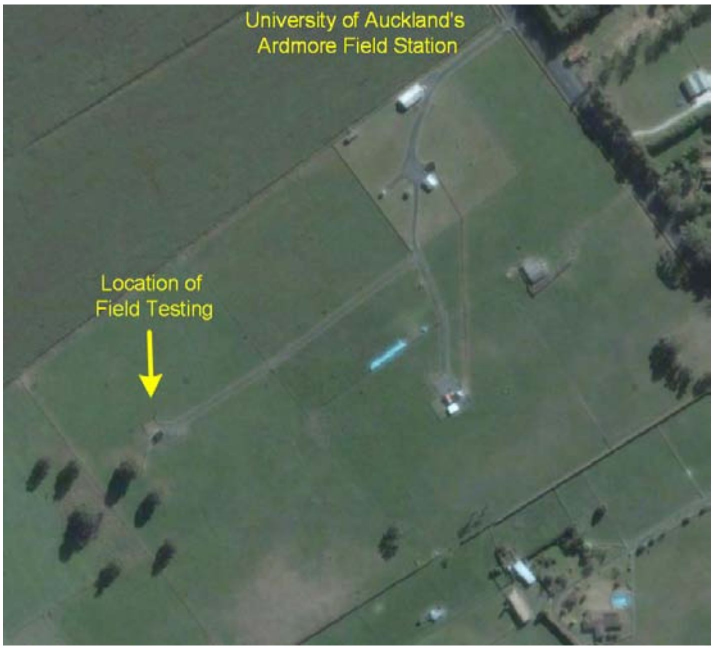


Figure B.3: Areal photo of the University of Auckland’s Ardmore field station

Bibliography

Antoniou, I., H. E. Jorgensen, F. Ormel, S. G. Bradley, S. von Hunerbein, S. Emeis, and G. Warmbier, 2003: On the theory of sodar measurement techniques (final reporting on wp1, eu wise project nne5-2001-297). Tech. rep.

Atlas, D., R. C. Srivasta, and R. S. Sekhon, 1973: Doppler radar characteristics of precipitation at vertical incidence. *Rev. Geophys.*, **11** (1), 1–35.

Atlas, D. and C. W. Ulbrich, : Path and area-integrated rainfall measurement by microwave attenuation in the 1-3 cm band. *J. Appl. Meteor.*.

Bass, H. E., L. C. Sutherland, A. J. Zuckerwar, D. T. Blackstock, and D. M. Hester, 1995: Atmospheric absorption of sound: further developments. *J. Acoust. Soc. Am.* , **97** (1), 680–683.

Beard, K. V., 1976: Terminal velocity and shape of cloud and precipitation drops aloft. *J. Atmos. Sci.*, **33** (5), 851–864.

Best, A. C., 1950: Empirical formulae for the terminal velocity of water drops falling through the atmosphere. *Quart. J. Roy. Meteor. Soc.*, **76**, 302–311.

Bradley, S. G., 1997: Acoustic radar studies of rain microphysics. *J. Atmos. Oceanic Technol.*, **14** (3), 547.

Bradley, S. G., I. Antoniou, S. von Hunerbein, D. Kindler, M. de Noord, and H. E. Jorgensen, 2005: Sodar calibration for wind energy applications (final reporting on WP3, EU WISE project NNE5-2001-297).

Bradley, S. G., K. R. George, M. J. T. Camilleri, G. L. Austin, and C. D. Stow, 1994: Acoustic radar studies of rain microphysics. *Proc. 7th ISARS 1994 (Addendum)*, Boulder, A4–A8.

Bradley, S. G. and T. Webb, 2002: Use of an ultrasonic sodar to sense raindrop size distributions. *J. Atmos. Oceanic Technol.*, **19** (8), 1203–1207.

Cerro, C., B. Codina, J. Bech, and J. Lorente, 1997: Modeling raindrop size distribution and Z(R) relations in the Western Mediterranean area. *J. Appl. Meteor.*, **36**, 1470–1479.

Coulter, R. L. and T. J. Martin, 1986: Investigations of the convective planetary boundary layer using a multiple frequency sodar. *Atmos. Res.*, **20** (2-4), 225–233.

Coulter, R. L., T. J. Martin, and T. M. Weckwerth, 1989: Minisodar measurements of rain. *J. Atmos. Oceanic Technol.*, **6** (3), 369–377.

Crescenti, G. H., 1997: A look back on two decades of Doppler sodar comparison studies. *Bull. Amer. Meteor. Soc.*, **78** (4), 651–673.

de Noord, M., A. Curvers, P. Eecen, I. Antoniou, H. E. Jorgensen, T. F. Pedersen, S. Bradley, S. von Hunerbein, and D. Kindler, 2005: Sodar power performance measurements, WISE WP5 (ECN-C-05-041).

Dingle, A. N. and Y. Lee, 1972: Terminal fallspeeds of raindrops. *J. Appl. Meteor.*, **11**, 877–879.

Doviak, R. J. and D. S. Zrnic, 1993: *Doppler Radar and Weather Observations.* Academic Press, Inc., San Diego, 2nd ed.

Foote, G. B. and P. S. du Toit, 1969: Terminal velocity of raindrops aloft. *J. Appl. Meteor.*.

Gage, K. S., C. R. Williams, W. L. Ecklund, and P. E. Johnston, 1999: Use of two profilers during MCTEX for unambiguous identification of Bragg scattering and Rayleigh scattering. *J. Atmos. Sci.*, **56** (21), 3679–3691.

Gilman, G. W., H. B. Coxhead, and F. H. Willis, 1946: Reflection of sound signals in the troposphere. *J. Acoust. Soc. Am.* , **18**, 274–283.

Gossard, E. E., 1988: Measuring drop-size distributions in clouds with a clear-air-sensing Doppler radar. *J. Atmos. Oceanic Technol.*, **5** (5), 640–649.

Grassl, H., S. Pang, and F. Mo, 2000: A mini-sodar on a single DSP board. *Proc. 10th ISARS 2000*, Auckland, 39–44.

Gunn, R. and G. D. Kinzer, 1949: The terminal velocity of fall for water droplets in stagnant air. *J. Atmos. Sci.*, **6** (4), 243–248.

Hopkins, C., 2006: Rain noise from glazed and lightweight roofing. Tech. rep., BRE Department for Education and Skills.

Kallistratova, M. A., 1959a: An experimental investigation into the scattering of sound in a turbulent atmosphere.

Kallistratova, M. A., 1959b: Procedure for investigating sound scattering in the atmosphere. *Soviet Physics - Acoustics*, **5**, 512–514.

Kallistratova, M. A. and V. I. Tatarski, 1960: Accounting for the wind turbulence in the calculation of sound scattering in the atmosphere. *Soviet Physics - Acoustics*, **6**, 503–505.

Kelton, G. and P. Bricout, 1964: Wind velocity measurements using sodar techniques. *Bull. Amer. Meteor. Soc.*, **45**, 571–580.

Little, C. G., 1969: Acoustic methods for the remote probing of the lower atmosphere. *Proc. of the IEEE*, **57** (4), 571–578.

Little, C. G., 1972: On the detectability of fog, cloud, rain and snow by acoustic echo-sounding methods. *J. Atmos. Sci.*, **29**, 748–755.

Marshall, J. S. and W. M. Palmer, 1948: The distribution of raindrops with size. *J. Atmos. Sci.*, **5** (4), 165–166.

McAllister, L. G., 1968: Acoustic sounding of the lower troposphere. *J. Atmos. Terr. Phys.*, **30**, 1439–1440.

Melling, H. and R. List, 1978: Acoustic Doppler sounding of falling snow. *J. Appl. Meteor.*, **17** (9), 1267–1273.

Pang, S. and H. Grassl, 2005: High-frequency single-board Doppler minisodar for precipitation measurements. Part I: Rainfall and hail. *J. Atmos. Oceanic Technol.*, **22** (4), 421–432.

Pang, S. X., H. Grassl, and F. Y. Mo, 2002: Special aspects of a high-frequency minisodar for precipitation measurements. *Proc. 11th ISARS 2002*, Rome, Italy, 445–448.

Peters, G., B. Fischer, and H. J. Kirtzel, 1998: One-year operational measurements with a sonic anemometer-thermometer and a Doppler sodar. *J. Atmos. Oceanic Technol.*, **15** (1), 18–28.

Rajopadhyaya, D. K., P. T. May, R. C. Cifelli, S. K. Avery, C. R. Williams, W. L. Ecklund, and K. S. Gage, 1998: The effect of vertical air motions on rain rates and median volume diameter determined from combined UHF and VHF wind profiler measurements and comparisons with rain gauge measurements. *J. Atmos. Oceanic Technol.*, **15**, 1306–1319.

Recktenwald, G., 2001: Least squares fitting of data to a curve. www.prenhall.com/recktenwald.

Renzo Tonin & Associates PTY LTD, 2007: Acoustic requirements to curb rain noise from metal deck roofs. www.rtagroup.com.au/pdfs/technote_1.pdf.

Rogers, R. R. and R. J. Pilie, 1962: Radar measurements of drop-size distribution. *J. Atmos. Sci.*, **19**, 503–506.

Rossing, T. D. and N. H. Fletcher, 1995: *Principles of Vibration and Sound*. Springer-Verlag.

Salomons, E. M., 2001: *Computational Atmospheric Acoustics*. Kluwer Academic Publishers, Dordrecht.

Sangren, K. L., P. S. Ray, and G. B. Walker, 1984: A comparison of techniques to estimate vertical air motions and raindrop size distributions. *J. Atmos. Oceanic Technol.*, **1**, 152–165.

Shamanaev, S. V., 2003: Acoustic sounding of raindrop size distribution. *J. Atmos. Oceanic Technol.*, **20** (1), 152–158.

Tyndall, J., 1875: *Sound*. Appleton Co., London, 3rd ed.

Ulbrich, C. W., 1983: Natural variations in the analytical form of the raindrop size distribution. *J. Appl. Meteor.*, **22** (10), 1764–1775.

Uplinger, C. W., 1981: A new formula for raindrop terminal velocity. *Preprints, 20th Conf. on Radar Meteorology*, Meteor. Soc., Boston, MA, Amer, 389–391.

Wakasugi, K., A. Mizutani, M. Matsuo, S. Fukao, and S. Kato, 1986: A direct method for deriving drop-size distribution and vertical air velocities from VHF Doppler radar spectra. *J. Atmos. Oceanic Technol.*, **3** (4), 623–629.

Webb, T., 2000: The design and testing of an ultrasonic disdrometer for the remote sensing of rainfall. Msc, University of Auckland.

Weill, A., 1983: Atmospheric applications of sodar. *Proc. 2nd ISARS 1983*, Rome, Italy, 1.1–1.41.

Weill, A., C. Klapisz, and F. Baudin, 1986: The CRPE minisodar: Applications in micrometeorology and in physics of precipitations. *Atmos. Res.*, **20** (2-4), 317–333.