UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

Temporal and Spatial Interference Mitigation Strategies to Improve Radar Data Quality

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

John Lannie Lake
Norman, Oklahoma
2019

Temporal and Spatial Interference Mitigation Strategies to Improve Radar Data Quality

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Acknowledgments

The path to getting my PhD has been a long one, and too many people to individually acknowledge have played a part in it. That said, there are several people in particular that I want to thank.

First, to Dr. Mark Yeary. Dr. Yeary has been the quintessential advisor and advocate throughout my grad school journey, and I would not be the scientist I am today without his guidance. Thank you.

Second, to my friends and fellow graduate students, both here at the University of Oklahoma and elsewhere. I couldn't possibly name you all, and, in the way of all relationships, friends have come and gone over the past six years. Nevertheless, having people with whom to celebrate and encourage, to eat and drink, to talk and be nerdy about research, to hike and play board and video games and play Dungeons and Dragons... it's all made my grad school experience so much better. Thank you.

Among my friends, however, I especially want to thank Katie Shoemaker. Katie, you've been my best friend since undergrad, and, while we went to different schools for our PhDs, you were still always just a phone call away. I wouldn't have made it through grad school and everything else in the past six years without you there to bolster me. Thank you.

I also want to thank Dr. Jessica Ruyle. Female faculty bear a disproportionate share of the emotional labor, and I'm sure that's exacerbated by the gender imbalance in STEM fields, and probably even further by the gender imbalance of professors at the RIL. Nevertheless, you were there for me those times I needed it. Without those talks, I would have quit, and I would have spent the rest of my life regretting it. Thank you.

And finally, to my family, both nuclear and extended. Despite me spending five and a half years getting two bachelors degrees, they didn't balk at my tentative plans to embark on an at-least-six-year commitment to getting my PhD. Rather, they encouraged me to do grad school if it meant I would have a job at which I would be fulfilled. Who I am today would not have been possible without their support. Thank you.

Table of Contents

List of Tables

List of Figures

Abstract

The microwave band is well suited to wireless applications, including radar, communications, and electronic warfare. While radar operations currently have priority in a portion of the microwave band, wireless companies are lobbying to change that; such a change would force current operators into a smaller total bandwidth. Interference would occur, and has already occurred at the former National Weather Radar Testbed Phased Array Radar.

The research in this dissertation was motivated by this interference — it occurred even without a change to radar's primacy in the microwave band. If microwave operations had to squeeze into a smaller overall bandwidth, such interference, whether originating from other radars or some other source, would only become more common. The radio frequency interference (RFI) present at the National Weather Radar Testbed Phased Array Radar altered the statistical properties at certain locations, causing targets to be erroneously detected. While harmless enough in clear air, it could affect National Weather Service decisions if it occurred during a weather event.

The initial experiments, covered in Chapter 2, used data comprised of a single channel of in-phase and quadrature (IQ) data, reflecting the resources available to the National Weather Service's weather radar surveillance network. A new algorithm, the Interference Spike Detection Algorithm, was developed with these restrictions in mind. This new algorithm outperforms several interference detection algorithms developed by industry. Tests on this data examined algorithm performance quantitatively, using real and simulated weather data and radio frequency interference. Additionally, machine learning classification algorithms were employed for the first time to the RFI classification problem and it was found that, given enough resources, machine learning had the potential to perform even better than the other temporal algorithms.

Subsequent experiments, covered in Chapter 3, used spatial data from phased arrays and looked at methods of interference mitigation that leveraged this spatial data. Specifically, adaptive beamforming techniques could be used to mitigate interference and improve data quality. A variety of adaptive digital beamforming techniques were evaluated in terms of their performance at interference mitigation for a communications task. Additionally, weather radar data contaminated with ground clutter was collected from the sidelobe canceller channels of the former National Weather Radar Testbed Phased Array Radar and, using the reasoning that ground clutter is simply interference from the ground, adaptive digital beamforming was successfully employed to mitigate the impact of ground clutter and restore the data to reflect the statistics of the underlying weather data.

Tests on digital equalization, covered in Chapter 4, used data from a prototype receiver for Horus, a digital phased array radar under development at the University of Oklahoma. The data suffered from significant channel mismatch, which can severely negatively impact the performance of phased arrays. Equalization, implemented both via older digital filter design methods and, for the first time, via newer machine learning regression methods, was able to improve channel matching. When used before adaptive digital beamforming, it was found that digital equalization always improved system performance.

Chapter 1

Radars and Radio Frequency Interference

1.1 Radar Background

In the early 1900s, an engineer in Germany proved that ships passing between his continuous wave radio transmitter and his receiver acted to interrupt the signal. Over the next twenty years, technology developed past the ability to simply detect the presence of objects, and in 1924 a transmitter using frequency modulations was able to determine the range at which the ionosphere lay. Ranging with pulsed radio waves soon followed, and in the mid 1930s British engineers successfully demonstrated radio detection and ranging using aircraft as a target. Similar development had taken place almost in parallel in other countries — the word "radar" is an acronym for "RAdio Detection And Ranging", and was initially used by the U.S. Navy before being adopted internationally — and the development of the magnetron, an efficient and powerful transmitter at microwave frequencies, made microwave radar feasible. This proved timely, as World War II was underway and the Battle of Britain was soon to come, during which this technology was used to great effect, providing early warning of approaching aircraft and giving British pilots time to mobilize for defense against German aircraft raids on Britain.

During the war, it was noted that clouds and precipitation scattered the signal of the microwave-wavelength radar. Wartime efforts focused on accommodating for this attenuation so that approaching German aircraft wouldn't be missed, but after the war interest in these meteorological echoes waxed again and radar began to be used for detection of weather echoes. In 1957, the National Weather Service began deploying a national network of weather surveillance radars, "WSR-57"s, which gave meteorologists reflectivity data. In the 1960s and 1970s, solid state technology combined with the development of high-gain klystron amplifiers (which have coherent phase) made the collection of Doppler data from weather signals feasible, and in 1988 the WSR-88D began deploying across the United States, giving meteorologists data about reflectivity, radial velocity, and spectrum width [5]. In 2010, dual polarization capability was deployed to WSR-88D sites [6], and phased arrays — systems that use multiple antennas acting in concert — are being explored for the next generation of weather radars [7, 8, 9, 3].

Weather radars function by sending out a series of electromagnetic pulses, spending time "listening" for the echoes from distant objects, and performing analysis on the echoes to derive information about the statistical properties of the targets hit over the course of the time spent in that "dwell", looking in that particular direction in a coherent processing interval (CPI). This CPI must be short enough that the statistical properties of the volume being measured do not change. The time it took for an echo to return after transmission of the pulse is used to determine the range of the target: $R = c \cdot t/2$, where R is range, c is the speed of light ($3 \cdot 10^8$m s^{-1}), and t is the time since the previous pulse transmission. The pulse repetition interval (PRI) is the time between subsequent pulses; shorter PRIs result in a shorter maximum unambiguous range. Following this convention, "fast time" is associated with range and "slow time" is associated with separate pulses.

Up until 2010, only one polarization of electromagnetic wave was transmitted and received by WSR-88Ds, so radar products were limited to a grid of reflectivity factor (colloquially just "reflectivity" in weather radar circles), radial velocity, and spectrum width, over the range of azimuths and elevations through which the radar scanned. The reflectivity factor is proportional to the power of the received signal:

$$\hat{P} = \frac{1}{M} \sum_{k=0}^{M-1} P_k \tag{1.1}$$

where M is the number of pulses and P_k is the power at sample k. Because meteorological targets are distributed targets rather than point targets, they show up in the velocity spectrum as a Gaussian distribution centered around some mean; this mean velocity is estimated using the equation:

$$\hat{v} = -\frac{\lambda}{4\pi T_s} arg\hat{R}(T_s) \tag{1.2}$$

where λ is the wavelength, $arg\{\cdot\}$ takes the phase angle of the argument, $R(\hat{T_s})$ is the autocorrelation of a signal with a time lag of T_s (typically 1 sample), and so $arg\hat{R}(T_s)$ is the phase angle of the T_s-lag autocorrelation. The negative is a convention applied so that targets moving toward the radar will have a negative radial velocity. The spectrum width of the velocity spectrum is estimated using the equation:

$$\hat{\sigma}_v = \left(\frac{\lambda}{2\pi T_s 6^{0.5}}\right) |\ln|\tfrac{\hat{R}_1}{\hat{R}_2}||^{0.5} \tag{1.3}$$

where T_s is the PRI, R_k is the autocovariance evaluated at lag k. Note that if the spectrum width ends up being calculated as negative, it is typically a sign of poor quality data — either bad low SNR or too-narrow spectrum widths — and is set to 0 [5].

1.2 Radio Frequency Interference

The echoes that are processed by the radar can come from many sources. For weather radar, water droplets and ice particles are the targets of interest; however, ground clutter is particularly common close to the radar and can act to obscure the underlying weather signal. Other sources interfere with the signal received as well. The sun, for example, is a natural emitter of electromagnetic energy across a broad spectrum, including microwave frequencies, and thus can show up on weather radar data as “sun spikes”, as in Figure 1.1 [1]. Additionally, other emitters can interfere with the radar signal as well. Unfortunately, the three single-polarization variables are sensitive to data quality issues, and the dual polarization variables and downstream algorithms that use these variables are even more sensitive [10]. If data that is not representative of the meteorological environment is received and alters the value of these variables, it could affect algorithms

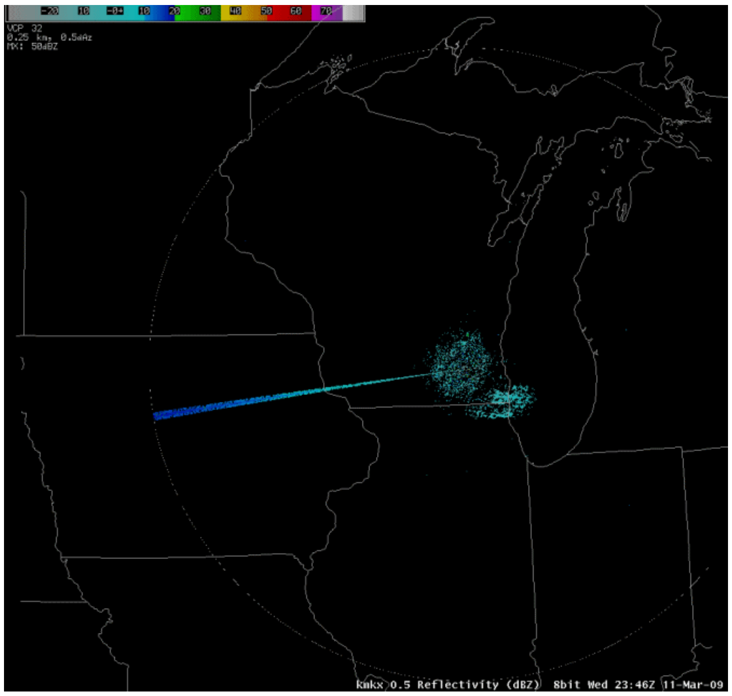


Figure 1.1: Sun spikes are simply the manifestations of RFI from a natural source [1].

and mislead forecasters about the environment, potentially altering decisions needed to protect lives and property.

The electromagnetic spectrum between 1 MHz and 100 GHz is a precious resource. It has applications in communications, radionavigation, broadcasting, and radar. The spectrum between 30 MHz and 3 GHz is especially useful, as the atmospheric opacity is very low in that band, meaning that loss from passage through the atmosphere is negligible (see Figure 1.2 [2]). Weather radar has a particular interest in the band around 3 GHz; the largest raindrops are around 8 mm in diameter, so having a wavelength of 10 cm allows the backscattering from the vast majority of meteorological targets to be approximated using the Rayleigh approximation [5].

Currently, radars have primacy in the 2 – 4 GHz band, meaning that other users can only operate in that frequency band if they do not interfere with radar operations [11]. However, the 2012 presidential mandate to sell 100 MHz of bandwidth combined with telecommunications lobbying for a downgrading of radar's primacy in the 3.4 –

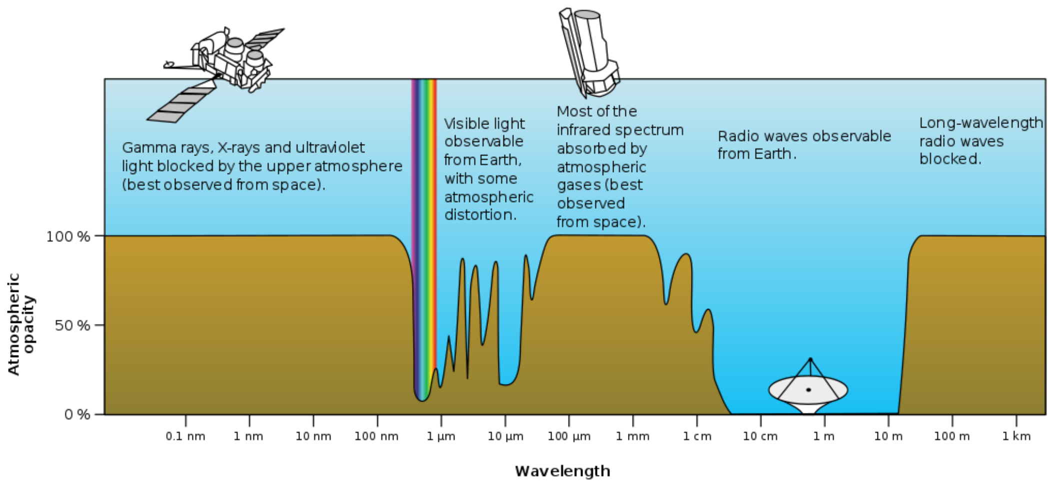


Figure 1.2: Electromagnetic waves are absorbed and scattered by the atmosphere; the amount of scattering and absorption is a function of the wavelength [2].

3.7 GHz band means that current radar systems would be forced to operate on a smaller bandwidth. Reduced available bandwidth for radar operation reduces the quality of data from radars, and, additionally, corrupt data caused by mutual interference between radars and other users is a problem that will only get worse with time [12, 11].

Even without a loss of radar's primacy in this band, radio frequency interference (RFI) has manifested at the National Weather Radar Testbed (NWRT) phased array radar (PAR), a Navy SPY-1A radar converted for weather radar applications; the NWRT PAR is shown in Figure 1.3, and the RFI can be seen in Figure 1.4 [8, 13]. The presence of RFI at the NWRT is due partly to the high density of radars nearby (see Figure 1.5 [3]) and partly to the fact that many of these radars operate on the same portion of the electromagnetic spectrum, as can be seen in Table 1.1 [14]. Even in other locations without such a high density of radar sites, if more users are forced into a smaller portion of the already-crowded S-band RFI will become an issue at more locations than just the NWRT. The data quality requirements for meteorological radar are strict. The algorithms that calculate reflectivity, velocity, and spectrum width are already sensitive

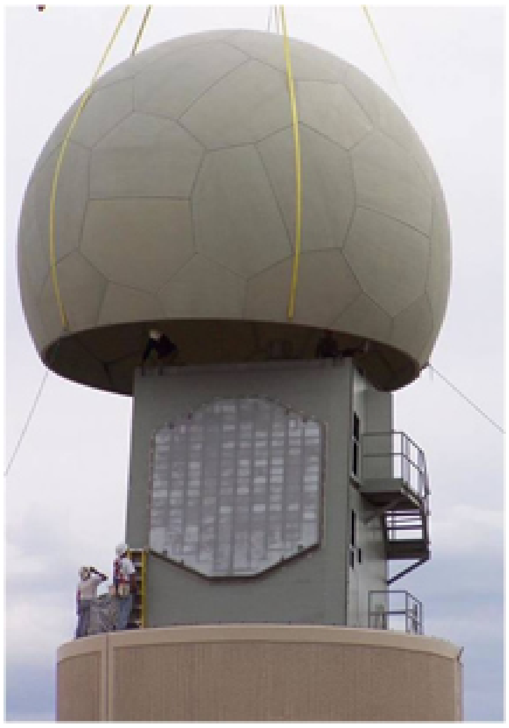


Figure 1.3: The NWRT PAR is a Navy SPY-1A converted for weather radar purposes.

Table 1.1: Several types of government radars have similar operating frequencies and pulse repetition intervals.

Radar	Operating Frequency	Approximate PRI
WSR-88D	2.7-3 GHz	1 ms
TDWR	5.5-5.65 GHz	0.5 ms
ASR-9	2.7-2.9 GHz	1 ms
ASR-11	2.5-2.9 GHz	1 ms

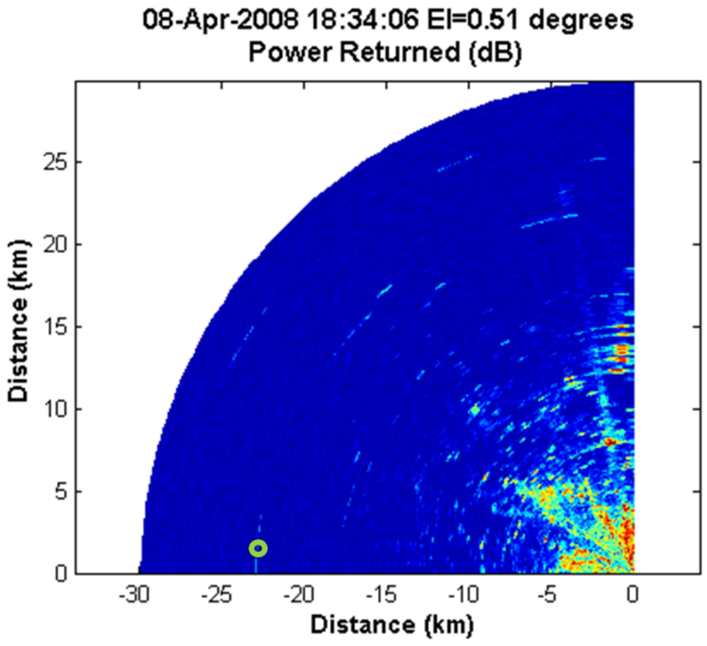


Figure 1.4: Co-channel transmitter interference manifests as spike in the magnitude of the returned power, as seen in a PPI (a display of constant elevation and varying azimuth) of clear air at the NWRT PAR.

to RFI, and the dual polarization product calculations are even more sensitive to interference; if RFI is present, it could potentially have a large effect on all downstream radar products. Methods to detect and mitigate the effects of RFI are needed.

Some methods of dealing with RFI already exist. For example, the Vaisala algorithms were developed to detect and mitigate RFI when only a single time series of IQ data is available. The Vaisala algorithms examine the data in a radar coherent processing interval and, at each range gate, look for large pulse-to-pulse variations in power [15]; these algorithms work reasonably well, but better performing alternatives are possible. For example, a cell-averaging CFAR-type method, ISDA, was developed as an alternative to the Vaisala algorithms, and machine learning algorithms can be quite potent at classification problems such as the detection of RFI. These temporal methods of detecting RFI are the subject of Chapter 2. Similarly, some techniques already exist to deal

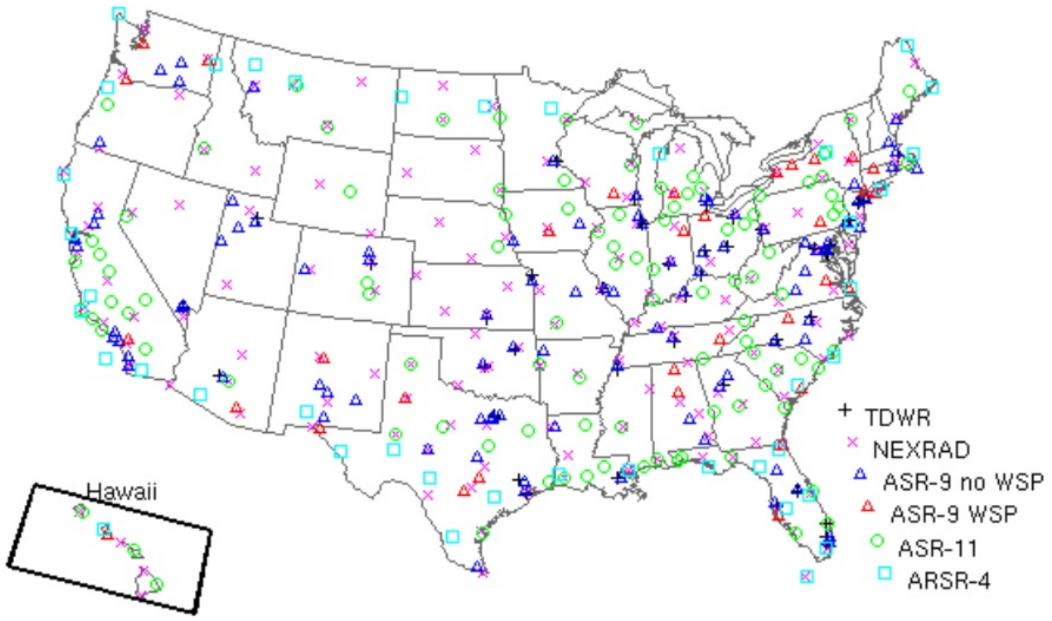


Figure 1.5: The location of government radar sites that operate in or near S-band [3].

with noise-like interference, but continuous wave interference is difficult for single-data-stream hardware to deal with without radar user intervention [16]. Fortunately, the increasing availability of digital phased arrays allows the collection of spatial data, which opens up the possibilities for spatial processing of radar data. Adaptive digital beamforming is a method of mitigating interference using such spatial data, and shows promise as a method of dealing with interference not only in weather radar applications but in applications ranging from communications to electronic warfare. These spatial processing methods are discussed in Chapter 3. These spatial processing techniques are powerful, but they require well-behaved and well-calibrated radar systems, which can be tricky if components do not behave consistently as their temperatures change. Digital equalization is a powerful technique that can be used to restore parity to these digital phased arrays and thus enhance the performance of these spatial processing algorithms, and is the subject of Chapter 4.

Chapter 2

Temporal Strategies

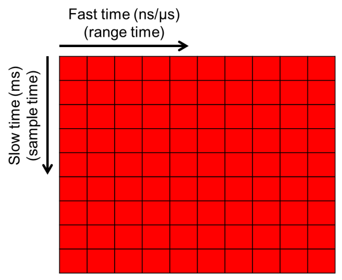
2.1 Background and Motivation

Three types of RFI affect radars: intermittent interference, continuous wave interference, and noise-like interference. Noise-like interference is non-coherent interference present in every sample of a CPI, acting to generally increase noise power; "sun spikes" are an example of noise-like interference, seen in Figure 1.1 [1]. Continuous wave interference is also present at every sample, but occurs at a particular frequency and thus manifests in the Doppler spectrum as a spike at a particular frequency bin. Intermittent interference, also called pulsed interference, does not affect every sample of a CPI but nonetheless can adversely affect data quality. Intermittent interference can be simulated using the equation

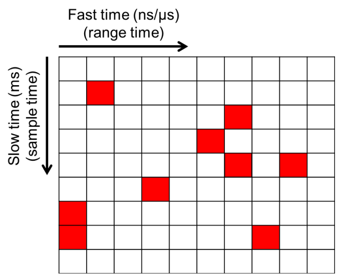
$$V_h(m) = \begin{cases} \sqrt{I_h}\exp\left[j\phi_P\right], & \text{if } m = k \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

where I_h is the interference power defined by the INR, ϕ_P is the random phase of the pulsed interference uniformly distributed from 0 to 2π, m is the sample index from 1 to M, and k is the randomly chosen integer that defines where the simulated RFI will be injected [16].

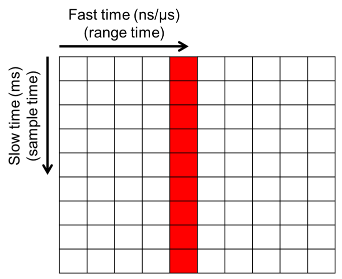
Example manifestations of the different types of interference are shown in Figure 2.1. Continuous wave RFI or noise-like RFI affects all range bins at all pulses (Figure 2.1a). Intermittent RFI affects random range bins from random pulses with no discernible pattern (Figure 2.1b). If the PRIs are identical, intermittent RFI from another radar can affect every pulse from a range bin in a particular CPI (Figure 2.1c). If the PRIs are identical but the pulse length of the interfering radar is longer than that of the



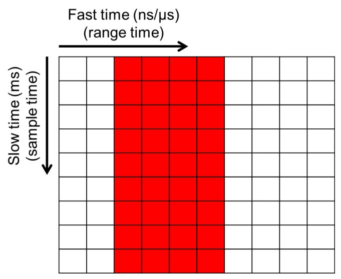
(a) Continuous wave & noise-like



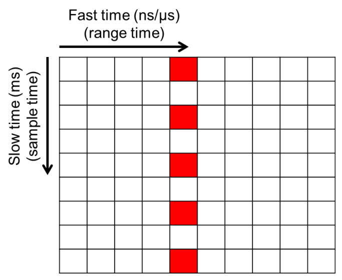
(b) Intermittent, random



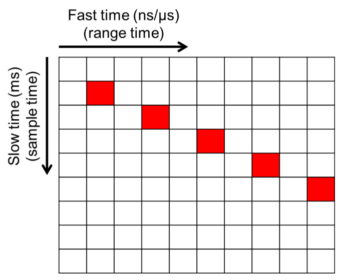
(c) Intermittent, identical PRI



(d) Intermittent, identical RFI, long pulse



(e) Intermittent, multiple PRI



(f) Intermittent, non-integer multiple PRI

Figure 2.1: Dwells illustrating how the different types of RFI can manifest, including several variations on intermittent interference. In all figures, red cells are those affected by RFI, while white cells are those that are unaffected by RFI. The X axis is fast time, representing range, while the Y axis is slow time, representing different pulses.

receiving radar, then such synced RFI can affect every pulse from multiple contiguous range bins (Figure 2.1d). If the PRI of the interfering radar is an integer multiple of the receiving radar's PRI, then only some of the pulses from a particular range bin in a CPI may be affected (Figure 2.1e). If the PRI of the interfering radar is not an integer multiple of the receiving radar's PRI, then the effect of the RFI will be spread out over multiple range bins (Figure 2.1f).

Temporal strategies of RFI mitigation use the information contained in the complex IQ data received by the radar during a CPI. If the statistics of the RFI are markedly different from the statistics of the weather, the RFI can be flagged as such and dealt with [15]. There are two steps in temporal RFI mitigation algorithms: detection and data recovery. The detection step is an important step and the more interesting one. Several algorithms were considered: the Vaisala algorithms, the Electromagnetic Interference Filter, and the Interference Spike Detection Algorithm [15, 13, 17].

This chapter is organized as follows: Section 2.2 gives an overview of several RFI mitigation algorithms and of different data recovery options, and Section 2.3 goes over the results of several experiments performed to explore the performance of these algorithms. The experiment in Subsection 2.3.1 first looks at how the probability of false alarm of ISDA varies with different parameters, and then examines the performance of all the temporal algorithms on white noise embedded with simulated RFI; the experiment in Subsection 2.3.2 looks at the performance of the algorithms on some data collected with RFI; the experiment in Subsection 2.3.3 takes some weather data without RFI and adds simulated RFI, allowing both the effects of RFI and the alleviating influence of the algorithms on meteorological parameters to be quantified; and finally, the experiment in Subsection 2.3.4 uses a method of simulating weather data to examine the performance of the algorithms in a wide variety of meteorological conditions, and also compares the performance of these algorithms to that of some more complex machine learning methods.

2.2 RFI Detection Algorithms

2.2.1 Vaisala Algorithms

The Vaisala algorithms (sometimes called the Sigmet algorithms) were developed to eliminate strong intermittent RFI originating from man-made emitters. The detection phase works by examining the variation in pulse power between the cell under test (CUT) and power of the cells from one and two pulses prior to the CUT, as shown in Figure 2.2. There are three variations of the algorithm, and all are very similar in

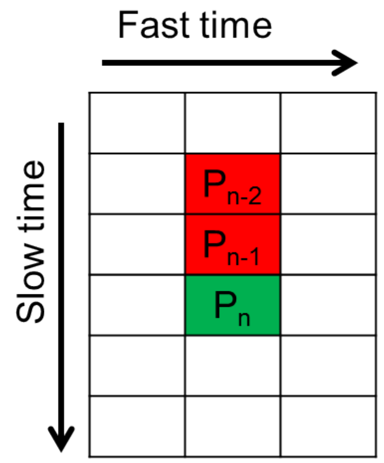


Figure 2.2: Cells used in the Vaisala algorithms, with the CUT in green.

form:

Vaisala 1

$$\begin{cases} |P_{n-1} - P_{n-2}| & < C_1 \quad \text{and} \\ |P_n - P_{n-1}| & > C_2 \end{cases} \tag{2.2}$$

Vaisala 2

$$\begin{cases} |P_{n-1} - P_{n-2}| & < C_1 \quad \text{and} \\ P_n - P_{n-1} & > C_2 \end{cases} \tag{2.3}$$

Vaisala 3

$$\begin{cases} |P_{n-1} - P_{n-2}| & < C_1 \quad \text{and} \\ P_n - \frac{P_{n-1}+P_{n-2}}{2} & > C_2 \end{cases} \tag{2.4}$$

where $P_n, P_{n-1},$ and P_{n-2} are the powers (in dB) at the current pulse, at one pulse before the current pulse, and at two pulses before the current pulse respectively; C_1 and C_2 are user defined constants, typically between 5 and 20 dB and often equal to each other.

The data recovery phase of the Vaisala algorithms is simple replacement: the interference-flagged data IQ data $\{I_n, Q_n\}$ at the CUT is replaced with the IQ data from the previous pulse $\{I_{n-1}, Q_{n-1}\}$ [15].

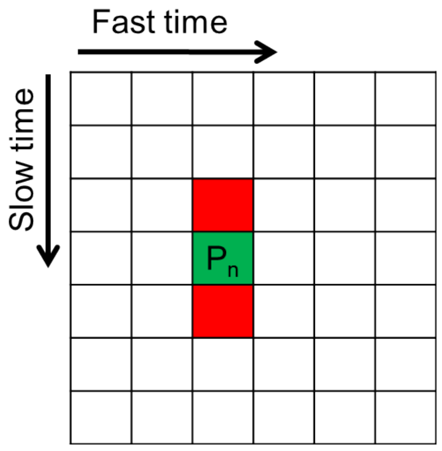
2.2.2 Electromagnetic Interference Filter

The Electromagnetic Interference Filter (EMI) is an algorithm developed internally by Chris Curtis of CIMMS. The detection stage of the EMI filter does an initial statistical analysis of the data and flags as RFI any point at which the power is significantly larger than the median power. The user is able to specify the desired P_{FA} of the detection stage.

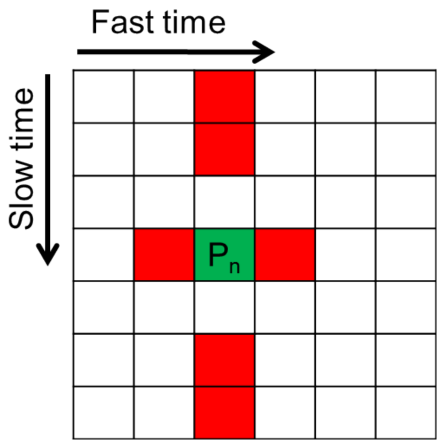
The data recovery stage of the EMI filter is linear interpolation of the IQ data in fast time using data from the range gate before and the range gate after the CUT [13, 17].

2.2.3 Interference Spike Detection Algorithm

The Interference Spike Detection Algorithm (ISDA) was initially developed to help mitigate the intermittent RFI that presented at the NWRT PAR, as shown in Figure 1.4. The detection stage of ISDA is, in essence, a cell-averaging constant false alarm rate algorithm [18]. For each cell-under-test (CUT) in a dwell, a set of “neighbor cells” is defined by the user; by default, and hereafter unless otherwise specified, the neighbors used by ISDA are the cells at the same range gate and from the pulse before and after, as shown in Figure 2.3a. The power in each of these neighbor cells is calculated and subsequently



(a) Default setup for the ISDA Algorithm



(b) Example customization of the ISDA algorithm

Figure 2.3: The neighbor set of ISDA can be defined by the user. The CUT is shown in green, and the neighbor cells are in red. By default (left) the neighbor set is comprised of the data from the cell before and the cell after the CUT.

averaged, giving the mean power of the neighboring cells. A ratio of the power of the CUT to the mean power of the neighbor cells is taken; if this number exceeds a certain threshold, it is flagged as RFI. In equation form:

ISDA

$$\left\{ \frac{P(V_n)}{P\left(V_{neighbors}\right)} > threshold \right. \tag{2.5}$$

where $P(V)$ is the power (not in dB) of whatever term is inside the parentheses; V_n is the IQ data at the CUT; $\overline{P\left(V_{neighbors}\right)}$ is the linear average of the powers (again, not in dB), calculated from the IQ data from all points in the (user-defined) set of neighbors; and *threshold* is a user defined detection threshold[13, 17].

The data recovery stage of ISDA is linear interpolation of the IQ data in slow time using data at the same range gate and from the pulse before and the pulse after the CUT.

2.2.4 Data Recovery Options

Three types of data recovery methods are used: simple replacement, used by the Vaisala algorithms; slow time interpolation, used by default by ISDA; and fast time interpolation, used by the EMI filter. Examples are shown in Figure 2.4

2.3 Temporal Algorithm Results

2.3.1 Testing White Noise

The performance of the algorithms on white noise was explored in the experiments outlined in the next two sub-subsections. First, the performance of ISDA as a function of which cells were included as part of its neighbor set was explored in Section 2.3.1.1. Next, the performance of ISDA, EMI, and the Vaisala algorithms in terms of the probability of false alarm when executed on white noise was examined in Section 2.3.1.2.

2.3.1.1 Exploring ISDA Parameters

The three adjustable parameters of the ISDA are the threshold, the width, and the number of guard cells; the latter two work together to define the the set of neighbors. The impact of adjustments to these parameters was calculated by populating a matrix with uncorrelated, complex, zero-mean, white Gaussian noise. Since the Central Limit Theorem is applicable to meteorological data, any anomalous spikes in this Gaussian distribution would also be present in real meteorological data, meaning that any spike detections found in this noise would be by definition false alarms. The ISDA was then executed on the matrix. The total number of interference spike flags was calculated and divided by the total number of data entries, yielding the probability of false alarm (P_{FA}). This was done for varied values of the threshold, the width, and the number of guard cells, and the results are shown as surface plots in Figures 2.5a and 2.5b.

Varying the number of guard cells had virtually no impact on the P_{FA}. This is expected, as the P_{FA} was calculated using uncorrelated zero-mean complex white Gaussian noise; the lack of correlation means that the neighbor cells' location does not affect the probability of finding a specific number at that location, and thus the average power of the neighbors is independent of location.

Increasing the width lowered the P_{FA} at any specific threshold. This makes sense: as more numbers are considered, the distribution of the samples approaches the distribution of the random process generating the samples, and thus the mean of the samples approaches the mean of the random process. Therefore, with increasing width in the P_{FA} test, the average of the neighbors' power approaches the expected value at the point in question and thus the probability that the power at the point in question normalized by the neighbors' power is more than the threshold is reduced.

Varying the threshold had the most impact on the P_{FA}. Calculating how the P_{FA} varied by threshold for a width of 1 and 0 guard cells using a least-squares fit yielded the relation:

$$P_{FA} = 1.7628 \cdot T^{-1.835} \tag{2.6}$$

where P_{FA} is in linear units and T is the threshold; this relation is shown in Figure 2.6. Integer thresholds for some P_{FA}s of interest are included in the figure, but in summary, the P_{FA} can be manipulated to be anywhere between 0.5 and 10^{-4} using the proper threshold values.

2.3.1.2 Validating Algorithm Parameters

The RFI detection algorithms were next tested on a matrix with injected interference spikes as a diagnostic test to verify settings. The matrix was first populated with uncorrelated, complex, zero-mean, white Gaussian noise, then at known locations additive interference spikes were introduced. The algorithms were then executed on this theoretical data. Because the location of the spikes was known, values for the probabilities of true positives, false positives, and false negatives were able to be determined. Graphs of these probabilities are shown in Figures 2.7 and 2.8. Varying the threshold parameter (for ISDA) and the P_{FA} parameter (for EMI) resulted in relatively constant P_{FA}s as the interference spike power varied, and the P_{FA}s could be coaxed down as low as 10^{-4}. The C_1 and C_2 algorithms were varied over a variety of values and the values that yielded an experimental P_{FA} that most closely matched the desired P_{FA} at each interference spike power level were shown above; despite this effort to glean the best possible performance from the Vaisala algorithms, the chart above shows that they were less steady and unable to reach as low of a P_{FA} as ISDA and EMI. As can be seen, the ISDA detects every interference spike for P_{FA} of 10^{-2} (10^{-3}, 10^{-4}) once the spike powers are 11 dB (14 dB, 16dB), respectively. The EMI algorithm performs even better, reaching near 100% probability of detection once the interferer spike power is about 8 dB no matter the specified P_{FA}. The Vaisala algorithms perform less well, typically maxing out at around 96% detection; this is likely because the Vaisala algorithms intrinsically cannot detect RFI in the first two pulses of a particular dwell. The data at two points are summarized in Table 2.1.

2.3.2 NWRT PAR Data Observed with RFI

The ISDA was tested on a singled dwell from the NWRT PAR data shown in Figure 1.4, using a width of one and no guard cells. The ISDA has varying results based on the threshold chosen. If an aggressive threshold, like $P_{FA} = 10^{-1}$, is used, all of the

Table 2.1: Algorithm Performance Comparison on White Gaussian Noise with RFI Added

Power: 5 dB	ISDA	EMI	Vaisala 1	Vaisala 2	Vaisala 3
P_{FA}	0.0108	0.0084	0.0102	0.0090	0.0110
P_D	0.171	0.544	0.031	0.088	0.164
Power: 5 dB	ISDA	EMI	Vaisala 1	Vaisala 2	Vaisala 3
P_{FA}	0.0010	0.0080	N/A	0.0015	0.0016
P_D	0.022	0.247	N/A	0.014	0.039
Power: 15 dB	ISDA	EMI	Vaisala 1	Vaisala 2	Vaisala 3
P_{FA}	0.0108	0.0084	0.0102	0.0090	0.0077
P_D	1	0.999	0.945	0.959	0.920
Power: 15 dB	ISDA	EMI	Vaisala 1	Vaisala 2	Vaisala 3
P_{FA}	0.0010	0.0080	N/A	0.0015	0.0011
P_D	1	0.999	N/A	0.792	0.965

noticeable spikes are removed, but there are numerous points flagged as interference spikes that to cursory examination do not appear to be RFI; the effects of this can be seen in Figure 2.9b. This alters the noise power and can affect the values of meteorological products which use the noise power in their calculation. If a more conservative threshold, like $P_{FA} = 10^{-3}$, is used, then some smaller spikes are left, though the most noticeable spikes are still removed, and the noise power is less affected; this can be seen in Figure 2.9c.

The ISDA was then tested on the whole of the PPI of the NWRT PAR data. The data before and after ISDA application can be seen in Figures 2.10a and 2.10b. Some RFI, including the RFI in the dwell shown in Figure 2.9, is clearly mitigated; for example, it is easily seen by looking at the change in the datapoint in the green circle between Figures 2.10c and 2.10d. However, other points that are likely afflicted with RFI remain, e.g. the data points to either side of that in the green circle in the figures. Examining the dwell at the location above the green circle in Figure 2.11 gives insight into why the RFI was unmitigated. In this area of the dwell, RFI is detected only at one point: at range gate 767 and pulse 32. ISDA (and the other algorithms) fail to detect RFI at other points on the dwell. This is because the PRI of the interferer (likely a nearby Airport Surveillance Radar) is synced with the PRI of the receiving radar, the NWRT, so the RFI is present in every pulse during the first 25 pulses, after which the PRI swaps to being approximately four times the receiving radar's PRI. The means that, for most of the dwell, the power at the previous pulses (for Vaisala) and neighbor pulses (for ISDA) do not differ significantly from the power at the current pulse. ISDA operates by comparing the neighbors' mean power to the power at the cell in question, an approach that only works if the neighbors are not themselves home to an interference spike. If the spike is persistent between pulses at a single location, ISDA does not see an abnormal jump in power and will not flag this location as interference. These slow-time-continuous interference patterns are difficult to distinguish from point targets; this case has the PRI switch in the middle of a CPI, but this will not always be the case. In Figure 2.11, the sharp drop-off in power in the later pulses is indicative that the return from that location is not from a point target but is instead interference; other nearby locations do not have this sharp dropoff in power at later pulses, and so could theoretically be point targets. Methods to identify RFI that is not intermittent are needed.

2.3.3 NWRT PAR Data Corrupted with Simulated RFI

The experiment covered in Section 2.3.2 did a quantitative examination of algorithm performance on white Gaussian noise with and without simulated RFI and a qualitative examination of algorithm performance on real weather data afflicted with real RFI. To more quantitatively explore algorithm performance on real meteorological data, a region of weather observed by the NWRT and unaffected by RFI (at least to the naked eye) was selected. This data, shown in Figure 2.12, was then additively corrupted with RFI generated according to equation 2.1 for INRs between -30 and 60 dB. To follow the convention of the real RFI observed in Section 2.3.2, if a particular range bin and azimuth was chosen to be afflicted with RFI, then multiple pulses at that range bin of that dwell were corrupted with RFI. Next, each of the detection algorithms was executed on the RFI corrupted weather data; ISDA was executed twice, once with the simple replacement data recovery scheme and once with the linear averaging data recovery scheme. Constants and thresholds were selected such that the P_{FA}s between the Vaisala algorithms, EMI, and ISDA were comparable for this data. Finally, the data was analyzed. Because uncorrupted data was available, the probabilities of detection could be found, and any biases introduced by the RFI or by the RFI mitigation algorithms to the weather data could be measured.

An example execution of the RFI corruption is shown in Figure 2.13. In this example, the RFI is very strong - the interference-to-noise ratio is approximately 60 dB - and quite apparent in the received power. Once afflicted with RFI, the RFI mitigation algorithms were executed on the data and allowed to mitigate the RFI; an example output is shown in the right column of Figure 2.13. To the eye, the RFI seems to be dealt with. However, an answer that is more quantitative than qualitative is desired. Because the "truth" of the data is available, the probability of detection, probability of false alarm, and the bias introduced by the RFI and the leftover bias after RFI mitigation to both reflectivity and radial velocity can be calculated.

Figure 2.14 shows how the probability of detection varied with INR and P_{FA} for ISDA, EMI, and the Vaisala algorithms. As the P_{FA} became more strict, the probability of detection of ISDA at a particular INR dropped. The Vaisala algorithms did not perform as well as ISDA at any of the tested P_{FA} values, but their performance did improve and become comparable to ISDA as the P_{FA} became more strict. By contrast, the performance of EMI did not vary much with the P_{FA} value, remaining relatively constant for a specific INR. This performance was worse than ISDA and some of the Vaisala algorithms at less strict P_{FA} values, but at the strictest P_{FA} values EMI outperformed both ISDA and the Vaisala algorithms.

The bias introduced by the RFI to the reflectivity field is shown in Figure 2.15, and the standard deviation of that bias is shown in Figure 2.16 . The bias introduced by the RFI to the radial velocity field is shown in Figure 2.17, and the standard deviation of that bias is shown in Figure 2.18.   The plots show that the bias in power introduced by the RFI varies with the INR, as expected; the random phase component of equation 2.1 means that the relation is not one-to-one, but the introduced bias varies from 0 dB to about 45 dB for INRs between -30 dB and 60 dB. After the Vaisala algorithms, EMI, and ISDA are executed on the RFI corrupted data, the bias is greatly reduced. Bias after the Vaisala algorithms varied between about 0 dB at low INRs and between 0.7 dB and 2 dB at high INRs; bias after EMI and ISDA was generally between 0 and 0.5 dB and did not vary noticeably with INR. ISDA using the simple replacement mitigation scheme increased the bias by around 0.1 dB, but still performed better than the Vaisala algorithms did, indicating that removal of the RFI is more important than the method of recovery of the corrupted data. The bias in radial velocity was much less dependent on INR. This is expected, since radial velocity is a function of the phase of the signal rather than its power, and the phase of the RFI was uniformly distributed. All the algorithms did a good job at reducing the bias in the radial velocity to near 0 meters per second.

2.3.4 Simulated RFI on Simulated Weather Data

The experiment covered in Section 2.3.3 examined the performance of the RFI mitigation algorithms at detecting RFI and their ability to restore the data to its true value. Evidence indicated there that the detection problem was more important, and that, when RFI was reliably detected, both linear interpolation or simple replacement were able to restore the data; thus, the detection problem was focused on for this experiment. There were two weaknesses in that experiment: first, the weather data was relatively uniform across the domain, and second, getting the desired P_{FA}s required the user to vary algorithm parameters and use their knowledge of the truth data to get the desired P_{FA}, rather than being able to set the algorithm parameters based on the environment.

2.3.4.1 Characterizing Algorithm Parameters

Getting a characterization of algorithm performance in terms of P_{FA} and probability of detection that was valid for a wide variety of meteorological environments would be ideal for an operational algorithm. Meteorological data with a variety of signal powers, signal-to-noise ratios, ambiguous velocities, radial velocities, and spectrum widths was simulated using the formula developed by Zrnić:

$$I(i) + jQ(i) = IFFT\left[-(S_k + N)\ln X_k\right] \tag{2.7}$$

where there are k frequency bins, S_k is the signal power spectrum (typically a Gaussian distribution), N is the noise power, and X_k is a random number between zero and one [19].

The P_{FA} of ISDA and the Vaisala algorithms was calculated for this data and used as a training data set for elastic net regularization, a machine learning regression algorithm.

Elastic net regularization linearly combines the L_1 and L_2 penalties of LASSO and Ridge regression. Ridge regression, also known as Tikhonov regularization or L_2 regularization, is defined as

$$\min_{\beta}\left[||y-X\beta||^2+||\Gamma\beta||_2^2\right]=\min_{\beta_0,\beta}\left[\sum_{i=1}^{N}\left(y-\beta_0-x_i^T\beta\right)^2\right]\text{ subject to }\sum_{j=1}^{p}|\beta_j|^2\le t \tag{2.8}$$

Setting $\Gamma=\lambda\mathbf{I}$, where λ is a penalty parameter, serves to prioritize solutions with smaller coefficients [20].

Least **A**bsolute **S**hrinkage and **S**election **O**perator Regression, also known as L_1 regularization, is defined as

$$\min_{\beta}\left[||y-X\beta||^2+||\Gamma\beta||_1\right]=\min_{\beta_0,\beta}\left[\sum_{i=1}^{N}\left(y_i-\beta_0-x_i^T\beta\right)^2\right]\text{ subject to }\sum_{j=1}^{p}|\beta_j|^2\le t \tag{2.9}$$

Because the penalty term is linear instead of quadratic, it tries to set some coefficients to zero, acting to excise unimportant coefficients [20].

The following relations were found:

- For ISDA: $\log(P_{FA})=-2.04-2.35\cdot10^{-10}\cdot sigPow+4.30\cdot10^{-10}\cdot noisePow+-5.80\cdot10^{-2}\cdot v_r+0\cdot\sigma_v+-1.22\cdot10^{-1}\cdot v_{am}+1.47\cdot10^{-3}\cdot\log(sigPow)+1.37\cdot10^{-6}\cdot SNR+-3.93\cdot10^{-2}\cdot\log(SNR)+-1.33\cdot\log(threshold)+-3.18\cdot10^{-2}\cdot threshold$

- For Vaisala 1: $\log(P_{FA})=-1.76-1.10\cdot10^{-10}\cdot sigPow+1.85\cdot10^{-10}\cdot noisePow+-1.39\cdot10^{-2}\cdot v_r+4.83\cdot10^{-1}\cdot\sigma_v+-1.33\cdot10^{-1}\cdot v_{am}+7.49\cdot10^{-4}\cdot\log(sigPow)+$

$6.09 \cdot 10^{-7} \cdot SNR + -1.83 \cdot 10^{-2} \cdot \log(SNR) + -1.82 \cdot 10^{-1} \cdot C_1 + -6.55 \cdot 10^{-1} \cdot C_2 + 4.20 \cdot 10^{-1} \cdot (C_1 - C_2)$

- For Vaisala 2: $\log(P_{FA}) = -6.12 - 1.29 \cdot 10^{-10} \cdot sigPow + 2.11 \cdot 10^{-10} \cdot noisePow + -1.49 \cdot 10^{-2} \cdot v_r + 3.21 \cdot 10^{-1} \cdot \sigma_v + -8.66 \cdot 10^{-2} \cdot v_{am} + 8.91 \cdot 10^{-4} \cdot \log(sigPow) + 6.49 \cdot 10^{-7} \cdot SNR + -1.93 \cdot 10^{-2} \cdot \log(SNR) + -9.35 \cdot 10^{-2} \cdot C_1 + -7.95 \cdot 10^{-1} \cdot C_2 + 6.50 \cdot 10^{-1} \cdot (C_1 - C_2)$

- For Vaisala 3: $\log(P_{FA}) = -3.13 - 1.28 \cdot 10^{-10} \cdot sigPow + 2.08 \cdot 10^{-10} \cdot noisePow + -1.86 \cdot 10^{-2} \cdot v_r + 3.97 \cdot 10^{-1} \cdot \sigma_v + -1.24 \cdot 10^{-1} \cdot v_{am} + 8.20 \cdot 10^{-4} \cdot \log(sigPow) + 8.67 \cdot 10^{-7} \cdot SNR + -2.70 \cdot 10^{-2} \cdot \log(SNR) + -3.72 \cdot 10^{-1} \cdot C_1 + -1.00 \cdot C_2 + 5.81 \cdot 10^{-1} \cdot (C_1 - C_2)$

where P_{FA} is the calculated probability of false alarm for this data, *sigPow* is the signal power, *noisePow* is the noise power, v_r is the radial velocity, σ_v is the spectrum width, v_{am} is the ambiguous velocity, *SNR* is the signal-to-noise ratio, *threshold* is the ISDA parameter, and C_1 and C_2 are the Vaisala parameters. These relations can all be inverted, meaning that, given some information about the environment, a user can automatically find the algorithm parameter required to attain a specific probability of false alarm. These parameterizations were then tested on another set of simulated data that comprised the test set; the results can be seen in Figure 2.19. The fits are not ideal but are far better and easier than guesswork.

2.3.4.2 Machine Learning Detection Algorithms

The detection of RFI is, at its heart, a classification problem: the algorithms are trying to classify a point as either interference or not interference. Two machine learning algorithms, decision trees and random forest classification trees, are well suited to the classification problem. Additionally, the internal workings of both algorithms are not obfuscated - they are relatively straightforward to interpret and thus wouldn't be overly complicated to implement in an operational setting. While other algorithms, such as neural networks or support vector machines, could likely have performed well, the motivator behind the experiment was to find an algorithm whose inner workings were easily understood and interpretable and which could be easily exported to an FPGA or otherwise simply implemented in hardware.

Decision trees are a machine learning method used primarily for classification, though they can be applied to regression problems. They function by partitioning the learning space of the data into smaller and smaller segments until no information remains. The feature used to partition the data is chosen based on which partition would yield the best information gain. One of the distinct advantages of decision trees is that they are easy to interpret: the decision tree is easily represented as a flowchart. Drawbacks include that a single decision tree can overfit the data quite easily, resulting in trees that perform well for training data but not for new data [21]. This problem can be mitigated using an ensemble of trees, such as a random forest [22].

Random forests are an extension of decision trees. They are an ensemble of individual decision trees functioning in a perturb-and-combine setup. Each tree's training dataset is bootstrapped from the original dataset, meaning that each tree will be trained on a slightly different set of data, ideally making it more robust. Additionally, the features that each tree has available to partition upon are randomly chosen from the set of all features, again working to avoid duplicate trees. Overall, random forest classifiers have a slightly higher bias, but the reduced variance of the random forest output means that the overall performance is better than a single decision tree classifier [22].

One of each was trained on some simulated weather data with simulated RFI. To generate the training data, equation 2.7 was used to simulate weather data, which was injected with RFI using 2.1. The simulated weather radar data was simulated as a set of 5000 range bins, each with 36 pulses, yielding 180000 total samples. The ambiguous velocity of the hypothetical radar was set at 20 ms^{-1}, and the spectrum width was set at 3 ms^{-1}. The signal power was set to 100 W, the signal-to-noise ratio (SNR) was varied between 0 and 60 dB, and, when applicable, the interference-to-noise ratio (INR) was set to between 0 and 30 dB. The radial velocity was uniformly randomly generated between $\pm v_{am}$ for each range bin. The probability that RFI was located at any one range bin and pulse was set to 0.05.

The data passed to the regression algorithm was only data that a radar operator could be reasonably expected to know. The ambiguous velocity is an inherent property of the scanning strategy in use, so it was passed as it was. The other values were measured using the simulated radar data at that particular simulated range bin, using pulse-pair processing when needed:

- signal power (at pulse i) P_i
- radial velocity v_r
- spectrum width σ_v
- average power $\overline{P}$
- $\frac{P_i}{\overline{P}}$
- $P_{i-k}, k \in \{\pm 1, \pm 2 \pm 3, \pm 4\}$
- $\frac{P_i}{P_{i-k}}, k \in \{\pm 1, \pm 2, \pm 3, \pm 4\}$

The total number of features was 37. A separate tree and random forest was trained for every 5 dB of SNR between 0 and 60 dB. Scikit-learn's implementations of both were used [23].

2.3.4.3 Useful Metrics and Algorithm Performance

The receiver operating characteristic (ROC) curve is a measure widely used for model comparison, especially in the field of machine learning. The ROC curve is a plot of true positive rate versus false positive rate; that is, it plots sensitivity versus fallout (where $fallout = 1 - specificity$). It serves to visualize the performance tradeoffs of classification problems by showing how much an increase in true positive rate ”costs” in terms of the requisite increase in false positive rate. By definition, an algorithm that randomly classifies an entry will be right as much as it is wrong, and thus its ROC curve will be a line from $(0,0)$ to $(1,1)$, as seen in Figure 2.20. ROC curves that stay above this diagonal indicate an algorithm that performs better than random assignment [24].

The area under the ROC curve (AUC) is a metric used to quickly quantify the ROC in an easy-to-compare scalar value. It is, as its name suggests, the integrated area under the ROC curve. In terms of probability, the AUC is “equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance” [24]; thus the AUC of a random classifier is 0.5, and the AUC of a perfect classifier that is always correct is 1.

Both metrics are widely used, especially in machine learning applications, and so will be used to evaluate the performance of the decision tree and random classifier and compare their performance to that of the other RFI detection algorithms.

Again, equation 2.7 was used to simulate weather data, which was injected with RFI using 2.1. The SNR and INR were varied and the probability of detection and probability of false alarm were recorded and resulting AUCs of the ROC were recorded. The ROC for when the SNR was 60 dB and the INR was 30 dB is shown in Figure 2.21. The associated AUCs are shown in Table 2.2. Examination of Figure 2.21 and Table 2.2 shows that the industry standard algorithms (Vaisala 1, Vaisala 2, and Vaisala 3) do not perform very well. They do perform better than a random classifier, but their AUC is only around 0.6. By contrast, the experimental ISDA algorithm performs much better,

Algorithm	AUC
ISDA	0.800
Vaisala 1	0.620
Vaisala 2	0.583
Vaisala 3	0.586
Decision Tree	0.784
Random Forest	0.891

Table 2.2: AUCs for the tested RFI classification algorithms.

with an ROC far removed from the random assignment line and an AUC of 0.8. The Scikit-learn implementations of the machine learning algorithms perform quite well; the single decision tree classifier has an ROC only slightly below ISDA's ROC (with a corresponding AUC of 0.784), while the random forest classifier performs much better than ISDA performs, with the best ROC on the chart and an AUC of 0.891.

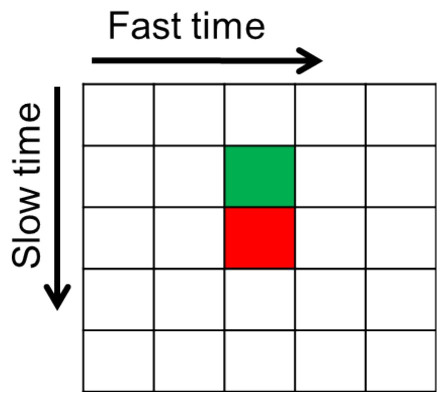
Varying the SNRs and INRs and calculating the AUC at each point can yield more insight into relative algorithm performance, as seen in Figure 2.22. The machine learning algorithms clearly have a higher AUC over a larger set of SNRs and INRs than the other detection algorithms.

Despite their higher performance, there are some drawbacks to the machine learning algorithms. One of the reasons for choosing decision trees was the ease of interpretation, especially when compared to other machine learning classification algorithms such as neural networks. The Scikit-learn implementation of the decision tree has almost 6500 nodes; while still easy to interpret, condensing it into a set of rules to be implemented onto an FPGA is not as simple. If trimming a bunch of nodes from the decision tree or random forests yields a large decrease in performance, it may be worth the slight drop in performance to implement the simpler ISDA over the superior-performing machine learning algorithms, at least in operational settings where speed is a priority. Either way, it's clear that machine learning algorithms can perform better than the currently-used industry algorithms. A single decision tree classifier can perform on par with the experimental algorithm ISDA and far above the operational Vaisala algorithms. Additionally, a random forest classifier can perform better than any other method examined.

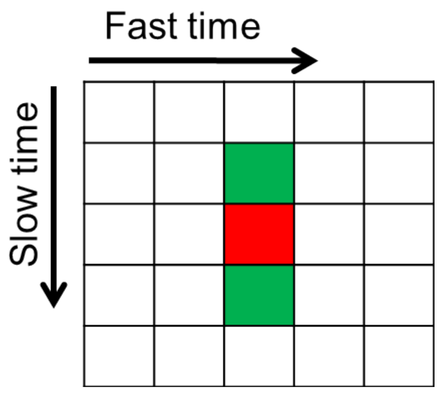
2.4 Conclusions

Radio frequency interference was observed at the National Weather Radar Testbed Phased Array Radar and motivated an inquiry into the efficacy of industry-standard RFI mitigation algorithms. The RFI was intermittent RFI of two types: with a PRI equal to that of the NWRT and with a PRI that was an integer multiple of the NWRT. The industry's Vaisala algorithms were compared to an algorithm developed for this research, a cell-averaging CFAR-type algorithm called ISDA, in a variety of experiments. When tested on white noise afflicted with simulated intermittent integer-multiple RFI, the Vaisala algorithms were less sensitive than ISDA and could not reach P_{FA}s as low as ISDA could. When tested on real weather data afflicted with simulated RFI, the Vaisala algorithms did mitigate the RFI, but the leftover bias in the meteorological variables was still higher than the bias after application of ISDA. When tested on simulated weather data with simulated RFI, the Vaisala algorithms again could not match the performance of ISDA. With the ability to simulate weather data and RFI, training datasets could be formed and machine learning classification algorithms compared to ISDA and the Vaisala algorithms. Decision trees were found to match ISDA's performance, and random forests were able to perform better than any of the other algorithms tested; however, both of these algorithms took a non-trivial amount of time to execute on the radar data. The machine learning algorithms would be best leveraged in offline processing for RFI detection.

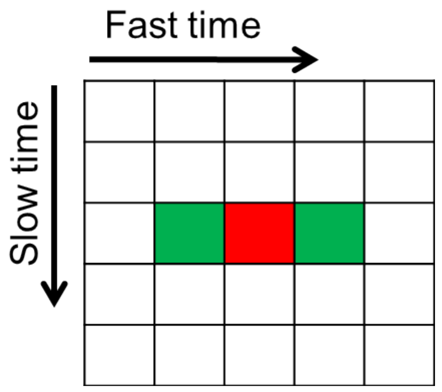
These tests all used simulated RFI that had a PRI that was an integer-multiple of the radar. This is not necessarily realistic; for example, the observed RFI that motivated this research had instances where its PRI was identical to that of the NWRT, making the return near-indistinguishable from that of a point target. This reveals a major weakness of the Vaisala algorithms, ISDA, and the machine learning classification algorithms as they were trained: if the RFI is present in every pulse at a particular range gate, or even just in some multiple contiguous pulses (as was the case with some of the RFI in this observed data), then the meteorological data will be affected in a way that the temporal RFI algorithms cannot mitigate. Fortunately, digital control of phased arrays offers spatial data from the radar which can be leveraged to mitigate RFI. Chapter 3 explores interference mitigation using this spatial data on both weather radar data and communications data.



(a) Simple replacement is used by the Vaisala algorithms.

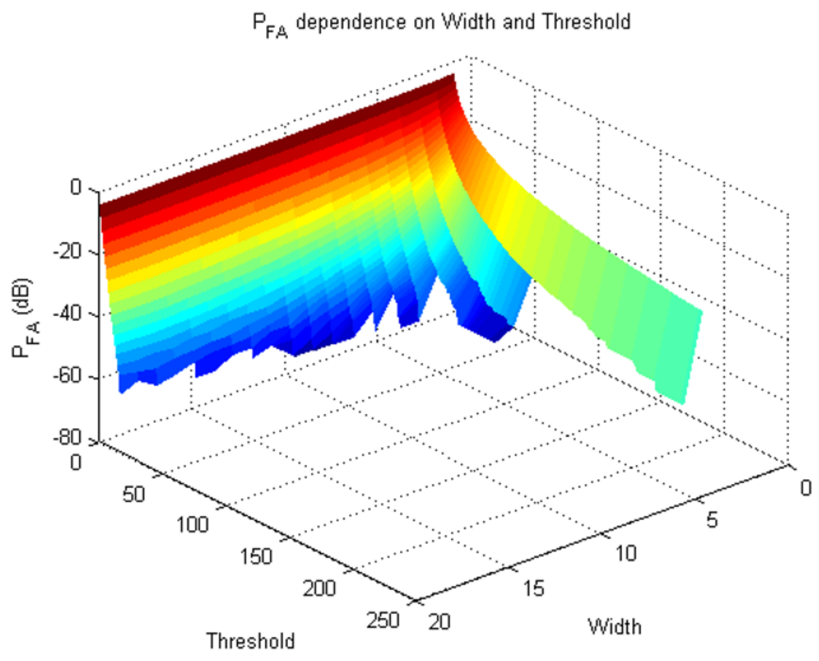


(b) Slow time interpolation is used by default by the ISDA algorithm.

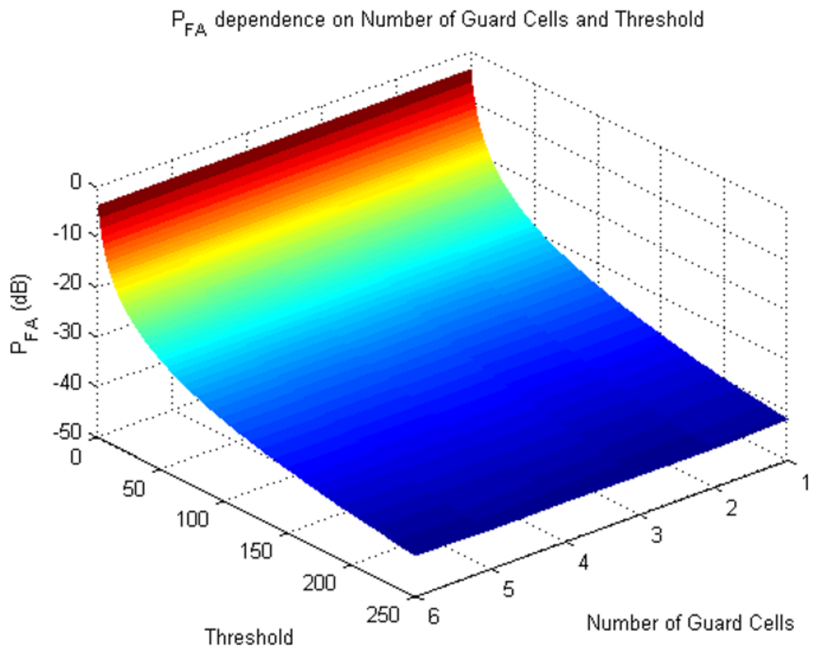


(c) Fast time interpolation is used by default by the EMI algorithm.

Figure 2.4: There are three data recovery schemes tested. Simple replacement is used by the Vaisala algorithms, slow time interpolation is the default for ISDA, and the EMI filter uses fast time interpolation.



(a) Varied Threshold and Width



(b) Varied Threshold and Guard Cells

Figure 2.5: ISDA P_{FA} results from varying the threshold versus the width (top) and number of guard cells (bottom) on white Gaussian noise.

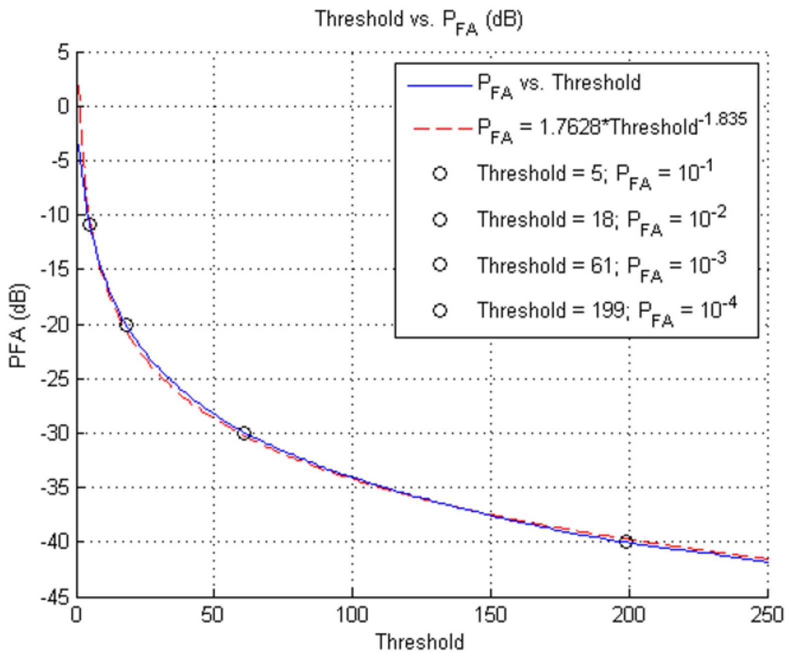


Figure 2.6: Using the default configuration shown in Figure 2.3a, the threshold was varied and the P_{FA}s recorded.

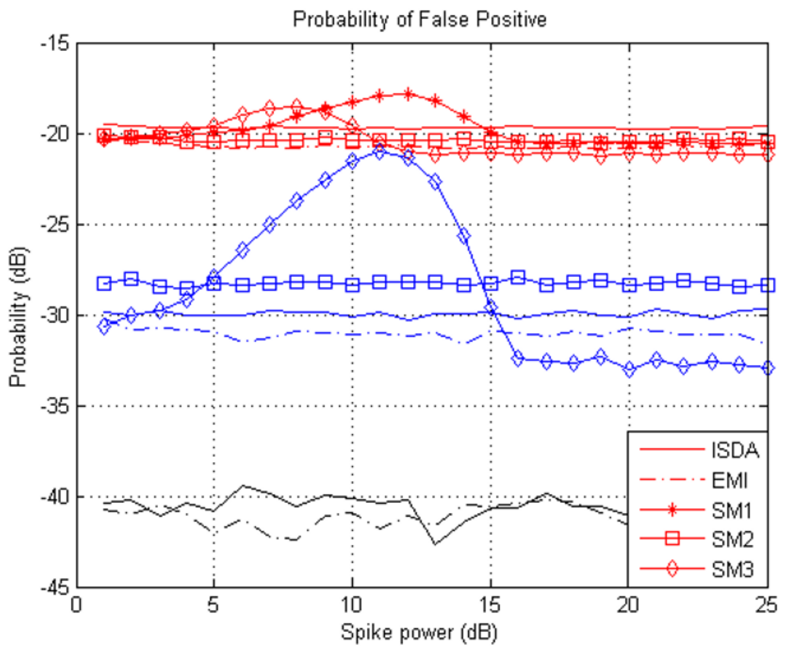


Figure 2.7: The P_{FA} as a function of injected interferer power. The Vaisala algorithms (labeled SM1, SM2, and SM3 here) were unable to reach P_{FA}s as low as the ISDA and EMI algorithms were.

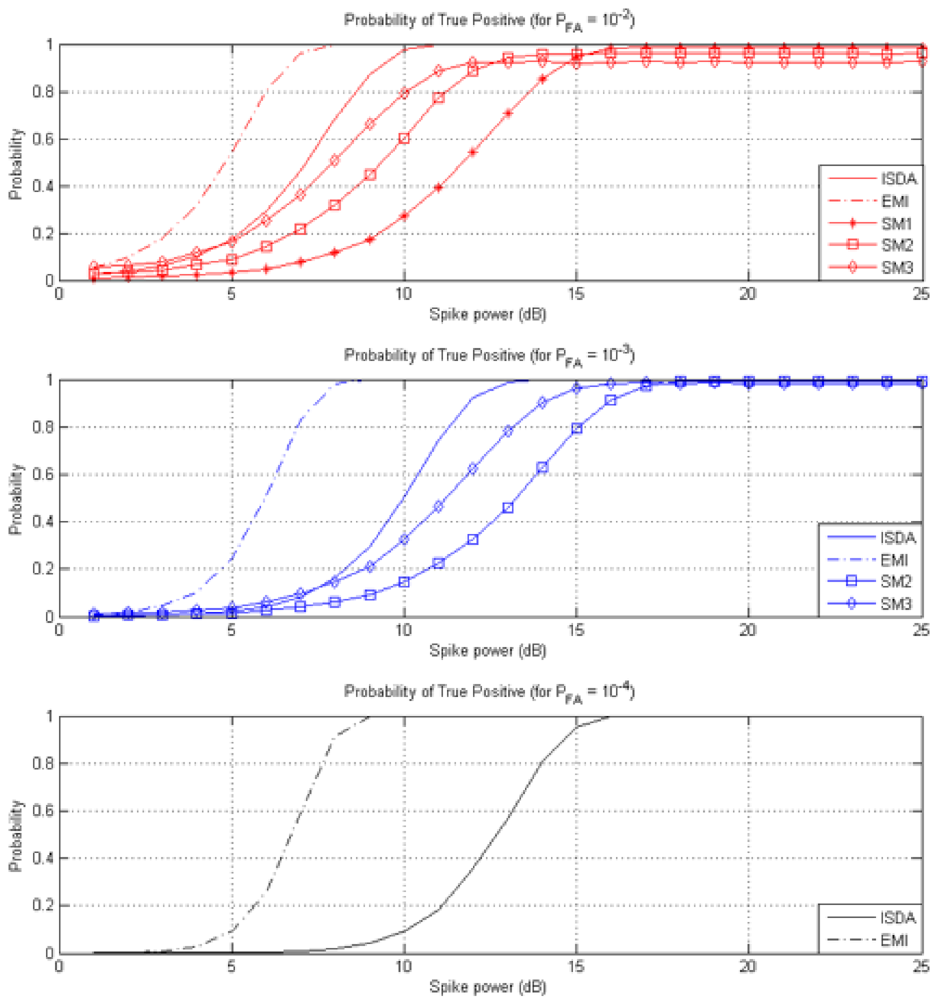
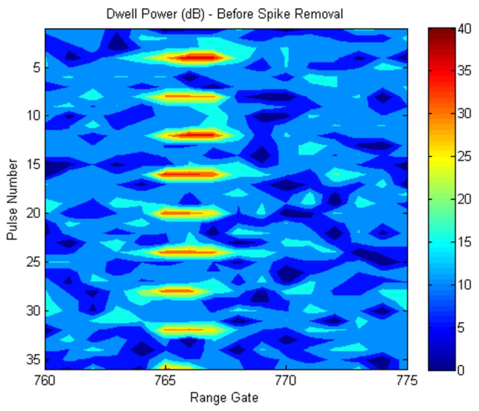
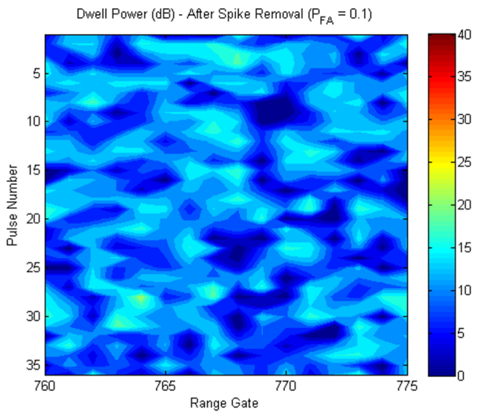


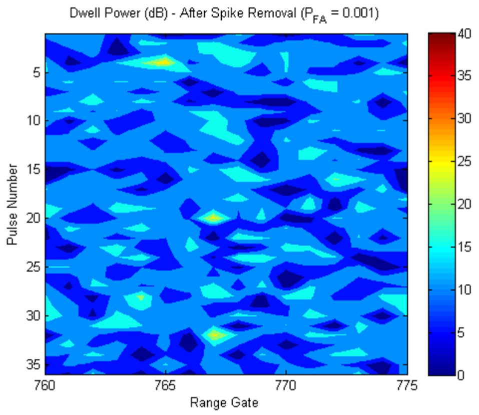
Figure 2.8: The probability of detection as a function of injected interferer power, separated by algorithm P_{FA}. Note that because the Vaisala algorithms (labeled SM1, SM2, and SM3 here) could not reach as low of a P_{FA} as the other two algorithms, their results are not shown in the bottom chart.



(a) Data Before RFI Detection and Mitigation

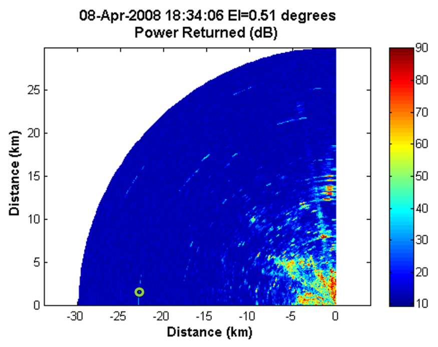


(b) Data After RFI Detection and Mitigation using $P_{FA} = 0.1$

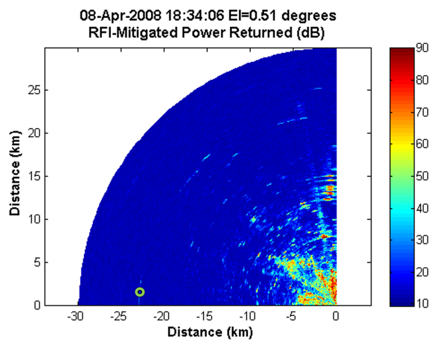


(c) Data After RFI Detection and Mitigation using $P_{FA} = 0.001$

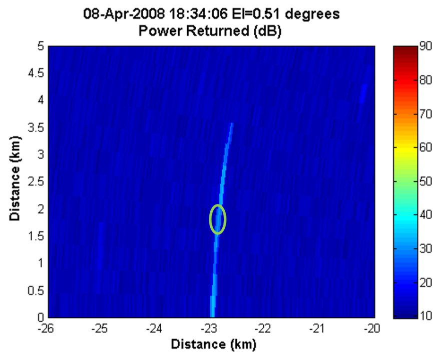
Figure 2.9: Data before (top) and after using ISDA to mitigate RFI with an aggressive P_{FA} of 0.1 (middle) and a more conservative P_{FA} of 0.001 (bottom).



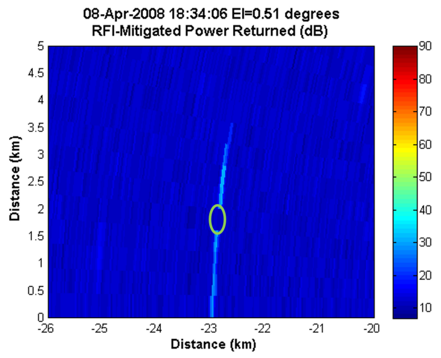
(a) Data before RFI detection and mitigation



(b) Data After RFI Detection and Mitigation using $P_{FA} = 0.001$

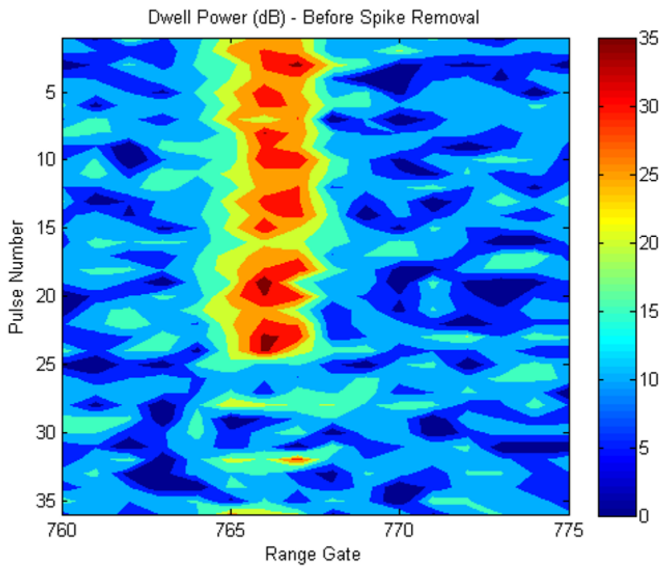


(c) Data before RFI detection and mitigation, zoomed in near the dwell of interest.

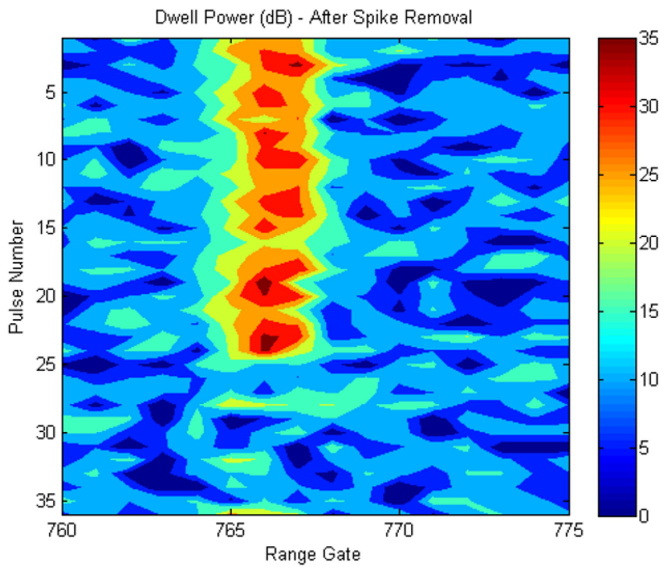


(d) Data after RFI detection and mitigation using $P_{FA} = 0.001$, zoomed in near the dwell of interest

Figure 2.10: Data before (left column) and after (right column) using ISDA to mitigate RFI with a conservative P_{FA} of 0.001. The location of the data used in the dwell shown in Figure 2.9 is noted with the green circle, and the area around that circle is shown in the bottom row.



(a) Data before RFI detection and mitigation



(b) Data after RFI detection and mitigation using $P_{FA} = 0.001$

Figure 2.11: Data before (left column) and after (right column) using ISDA to mitigate RFI with a conservative P_{FA} of 0.001. The only change is at range gate 767 and pulse 32.

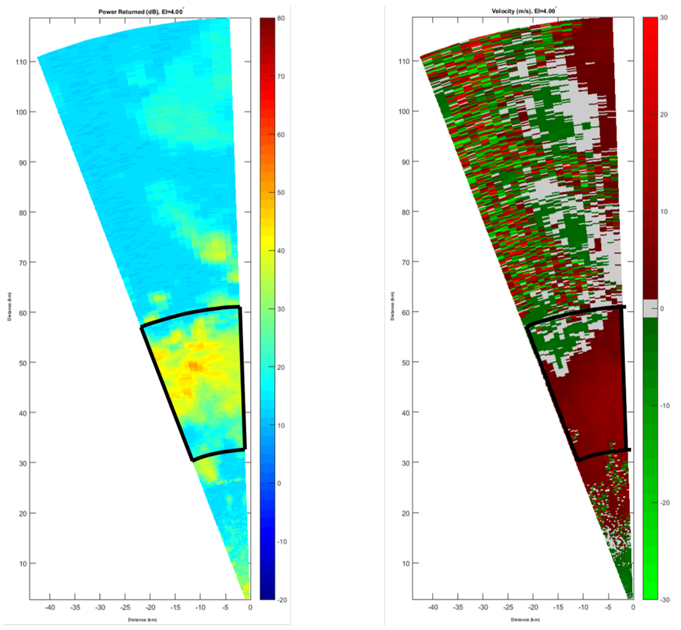
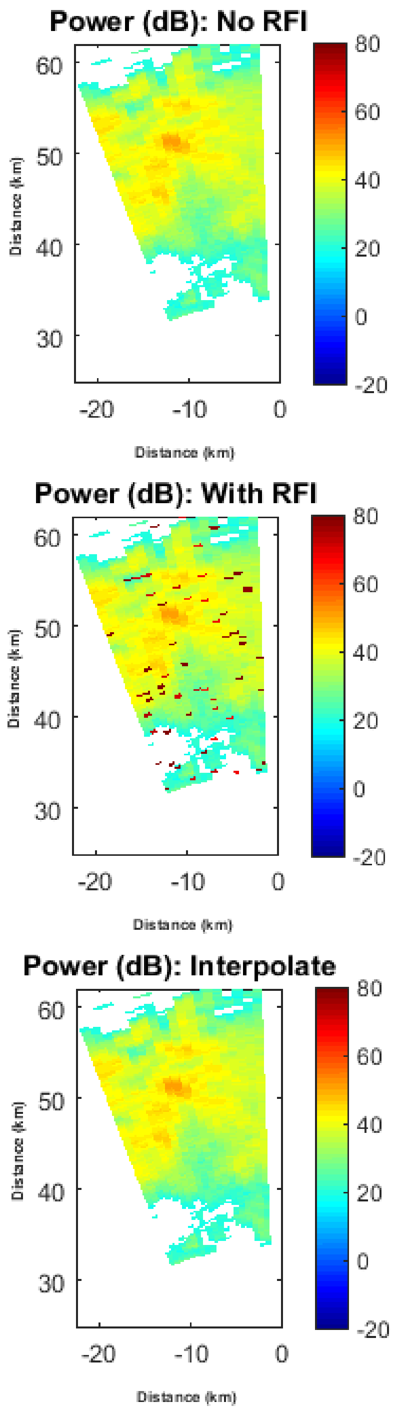
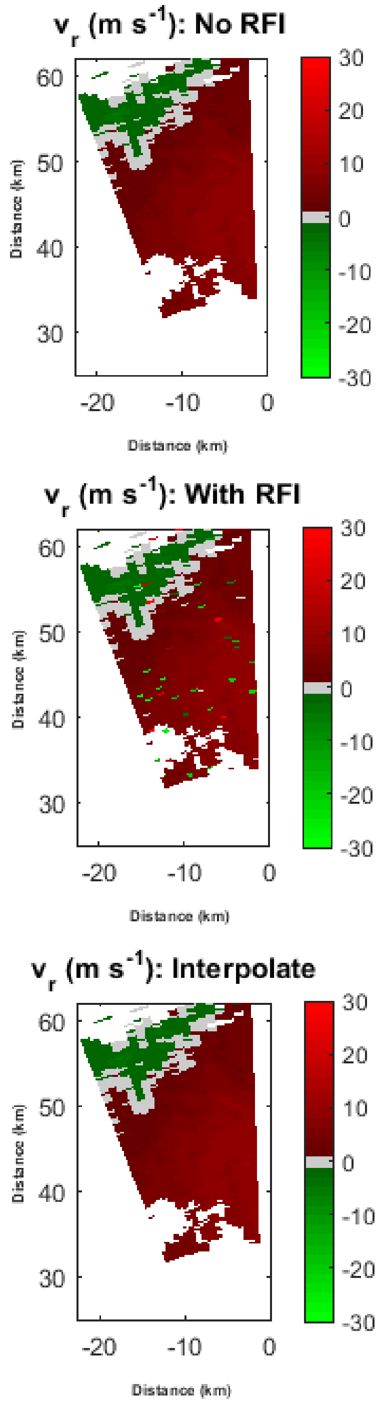


Figure 2.12: This RFI-free data from inside the black box was used as the starting point for this experiment.



(a) Example Execution: Power



(b) Example Execution: Radial Velocity

Figure 2.13: An example execution of this experiment. Power (top) and radial velocity (bottom) without RFI (left), with injected RFI with an INR or 60 dB (center), and after mitigation via linear averaging (right).

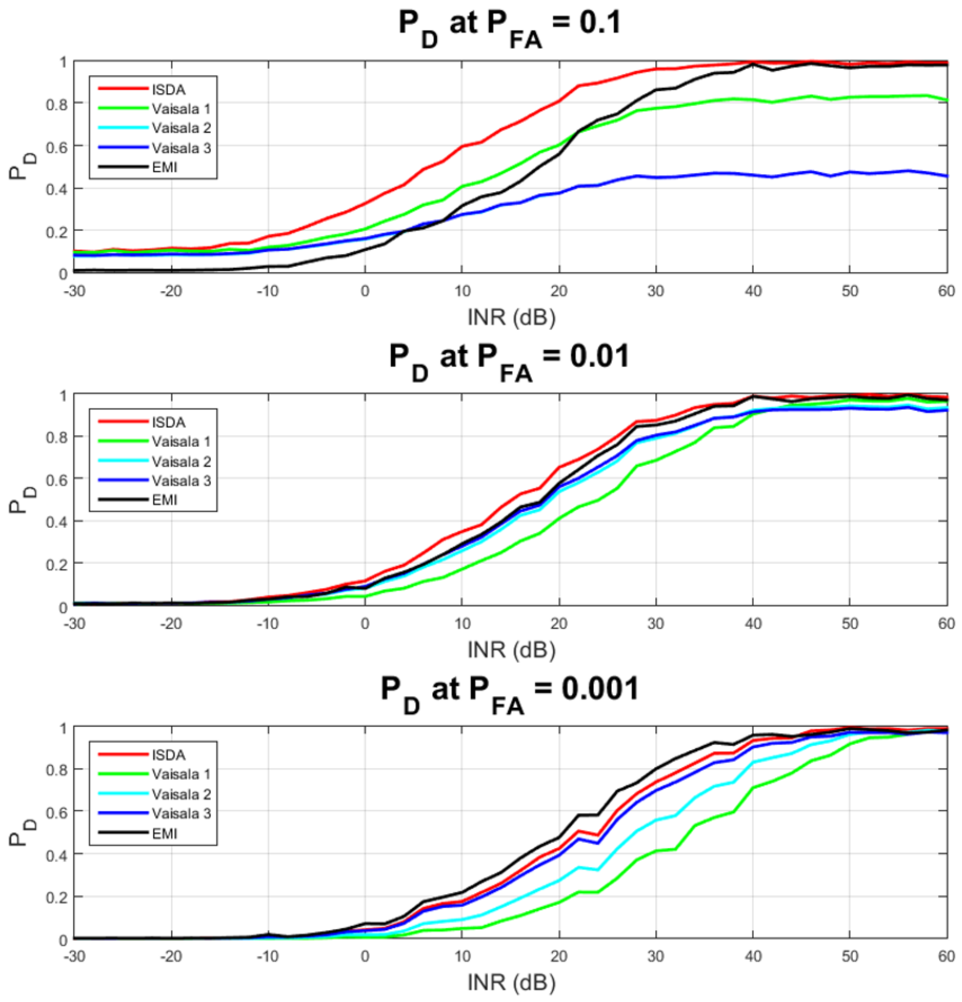


Figure 2.14: Probability of detection for $P_{FA} = 0.1$ (top), 0.01 (middle), and 0.001 (bottom) for ISDA (red), EMI (black), and the Vaisala algorithms (other colors).

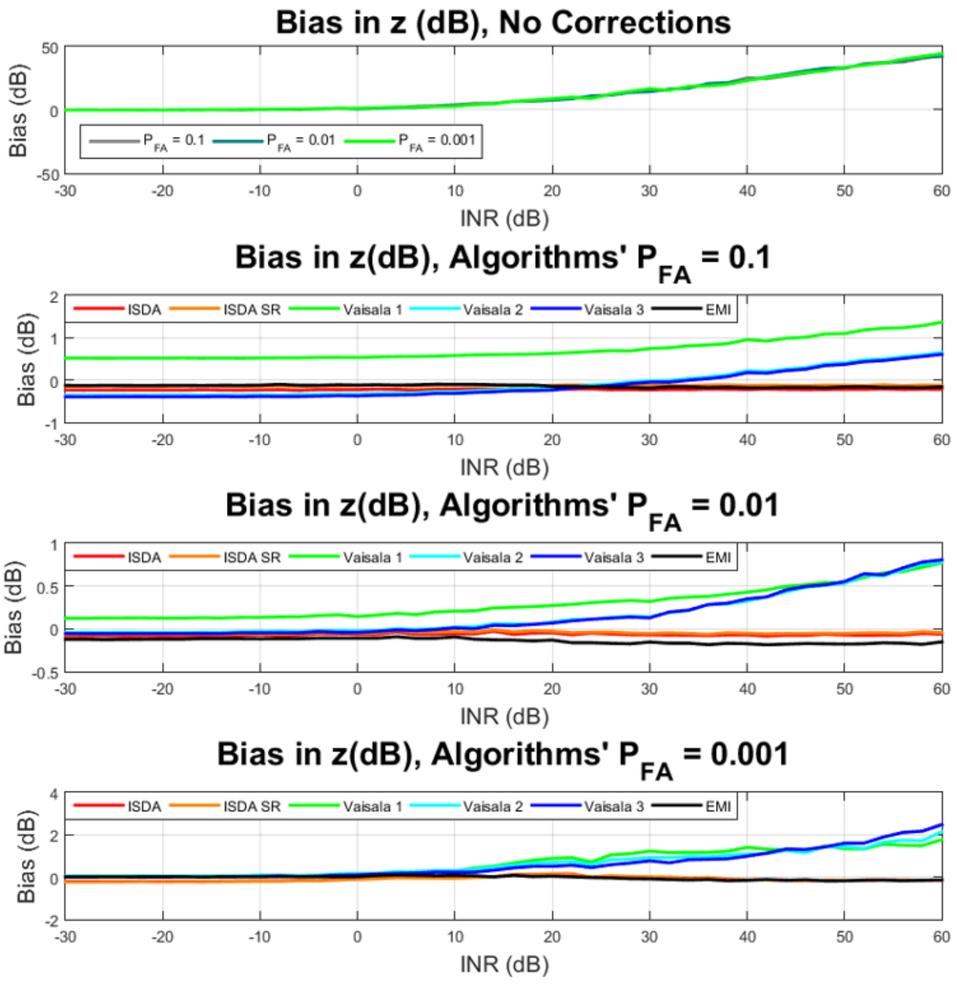


Figure 2.15: Bias (in dB) in reflectivity as a function of INR introduced by RFI (top) and the correction of RFI for $P_{FA} = 0.1$ (middle top), 0.01 (middle bottom), and 0.001 (bottom) for ISDA (red and orange), EMI (black), and the Vaisala algorithms (blues and greens).

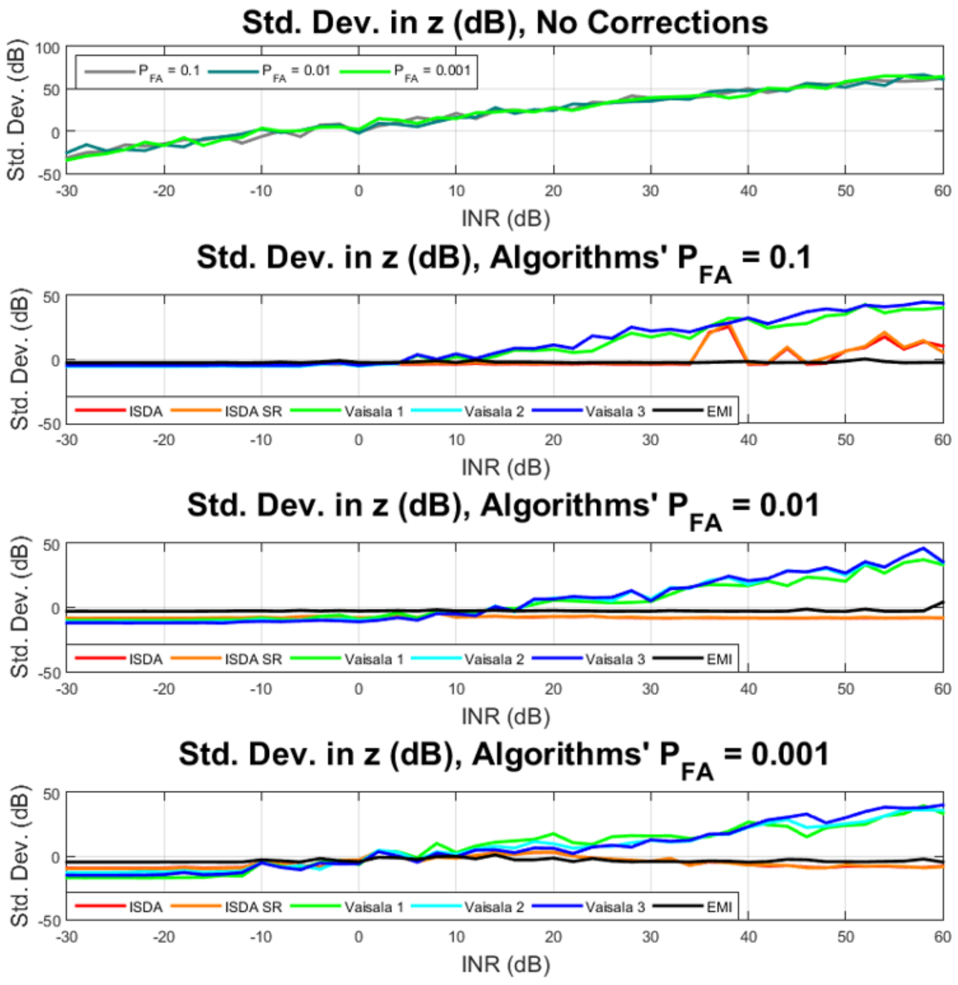


Figure 2.16: Standard deviation of the reflectivity bias as a function of INR introduced by RFI (top) and the correction of RFI for P_{FA} = 0.1 (middle top), 0.01 (middle bottom), and 0.001 (bottom) for ISDA (red and orange), EMI (black), and the Vaisala algorithms (blues and greens).

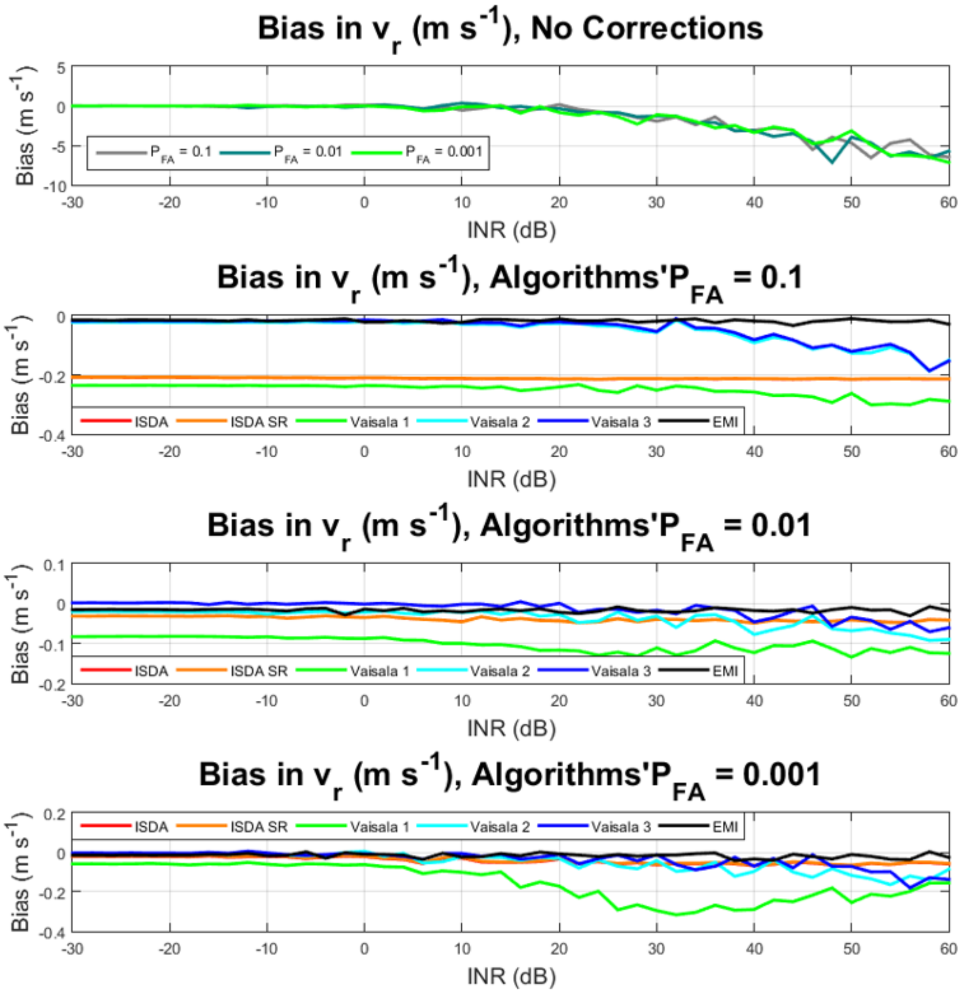


Figure 2.17: Bias (in dB) in radial velocity as a function of INRintroduced by RFI (top) and the correction of RFI for $P_{FA} = 0.1$ (middle top), 0.01 (middle bottom), and 0.001 (bottom) for ISDA (red and orange), EMI (black), and the Vaisala algorithms (blues and greens).

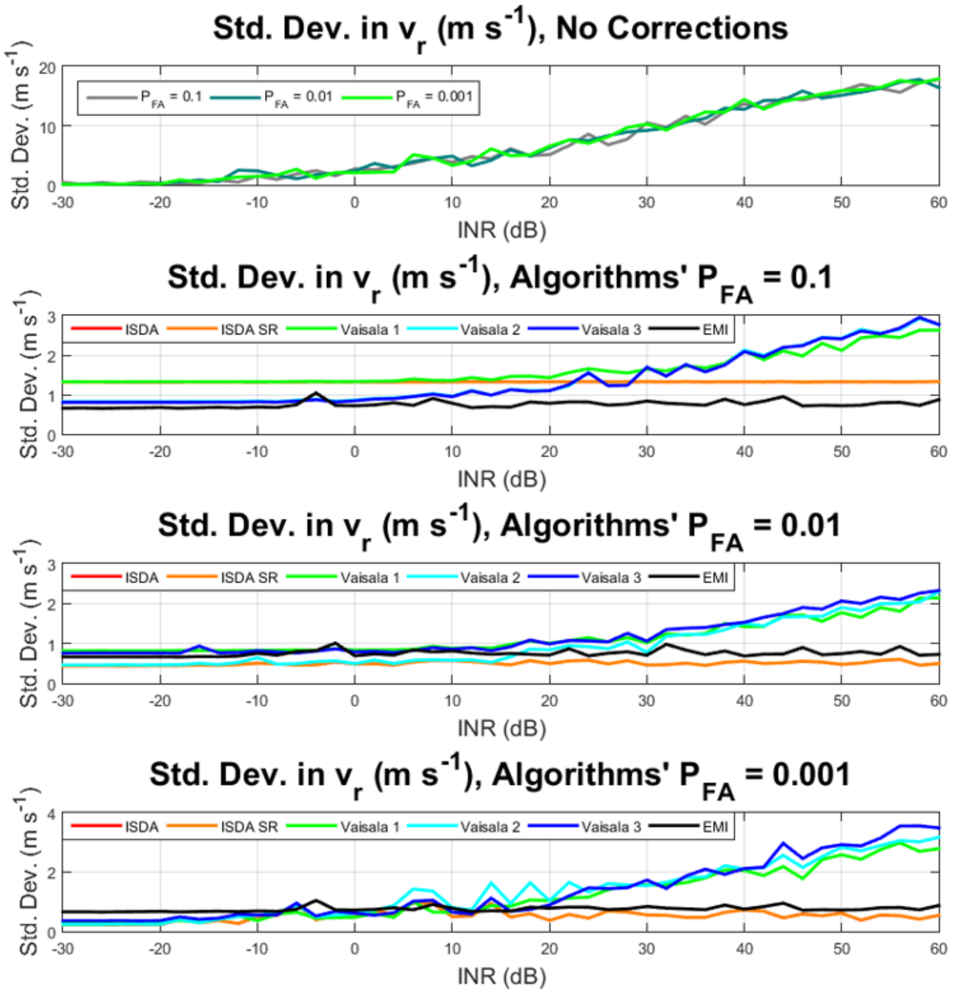
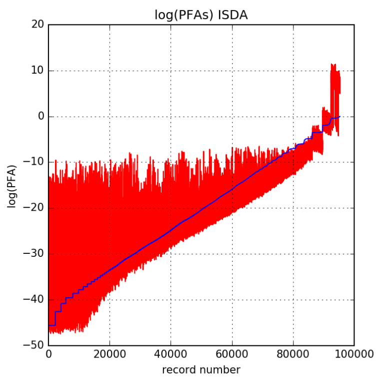
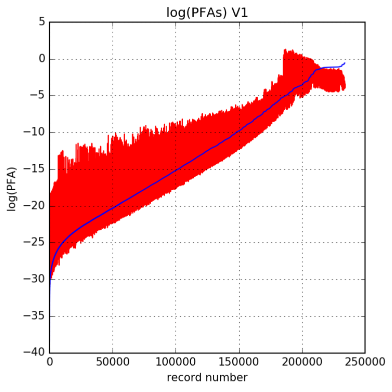


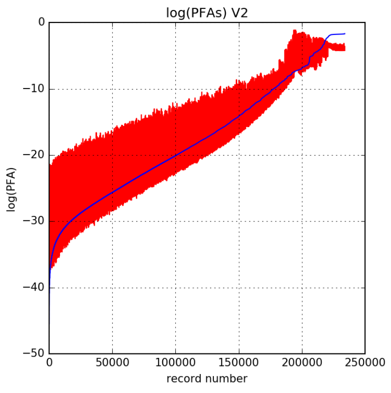
Figure 2.18: Standard deviation of the radial velocity bias as a function of IN-Rintroduced by RFI (top) and the correction of RFI for $P_{FA} = 0.1$ (middle top), 0.01 (middle bottom), and 0.001 (bottom) for ISDA (red and orange), EMI (black), and the Vaisala algorithms (blues and greens).



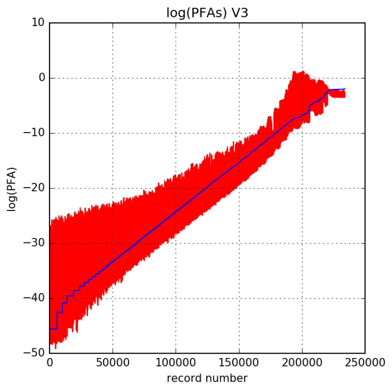
(a) ISDA Parameterization Fit



(b) Vaisala 1 Parameterization Fit



(c) Vaisala 2 Parameterization Fit



(d) Vaisala 3 Parameterization Fit

Figure 2.19: Predicted P_{FA} (blue) versus calculated P_{FA} (red dots) on the test set after elastic net regularization.

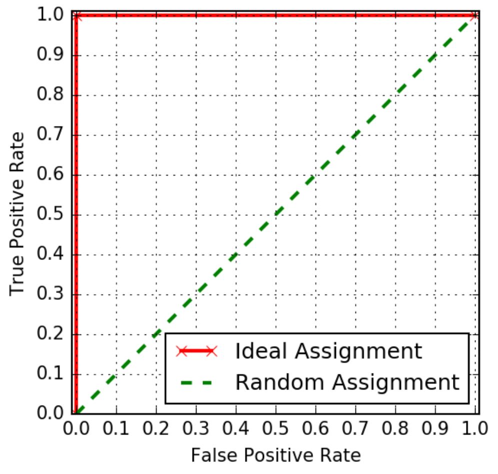


Figure 2.20: The ROC of a random classifier is simply a diagonal line (with an integrated area under the curve of 0.5).

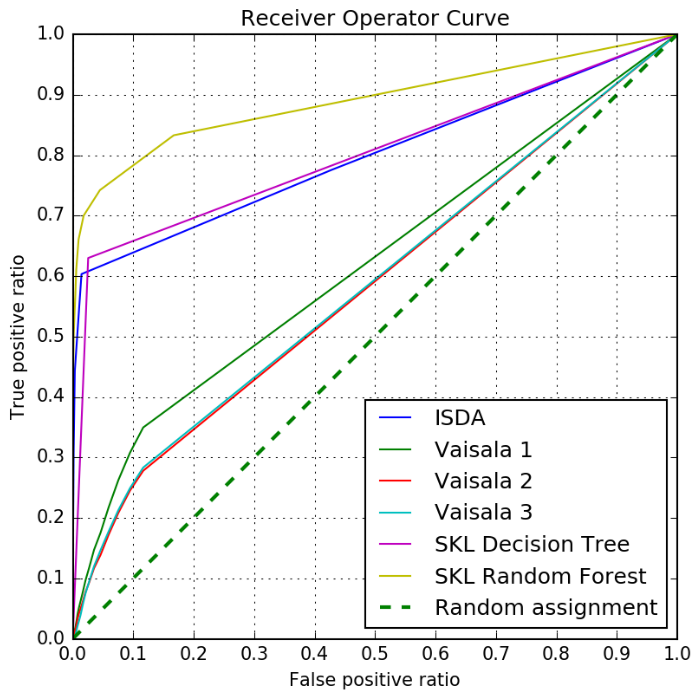


Figure 2.21: The ROC of the various detection algorithms when the SNR was 60 dB and the INR was 30 dB.

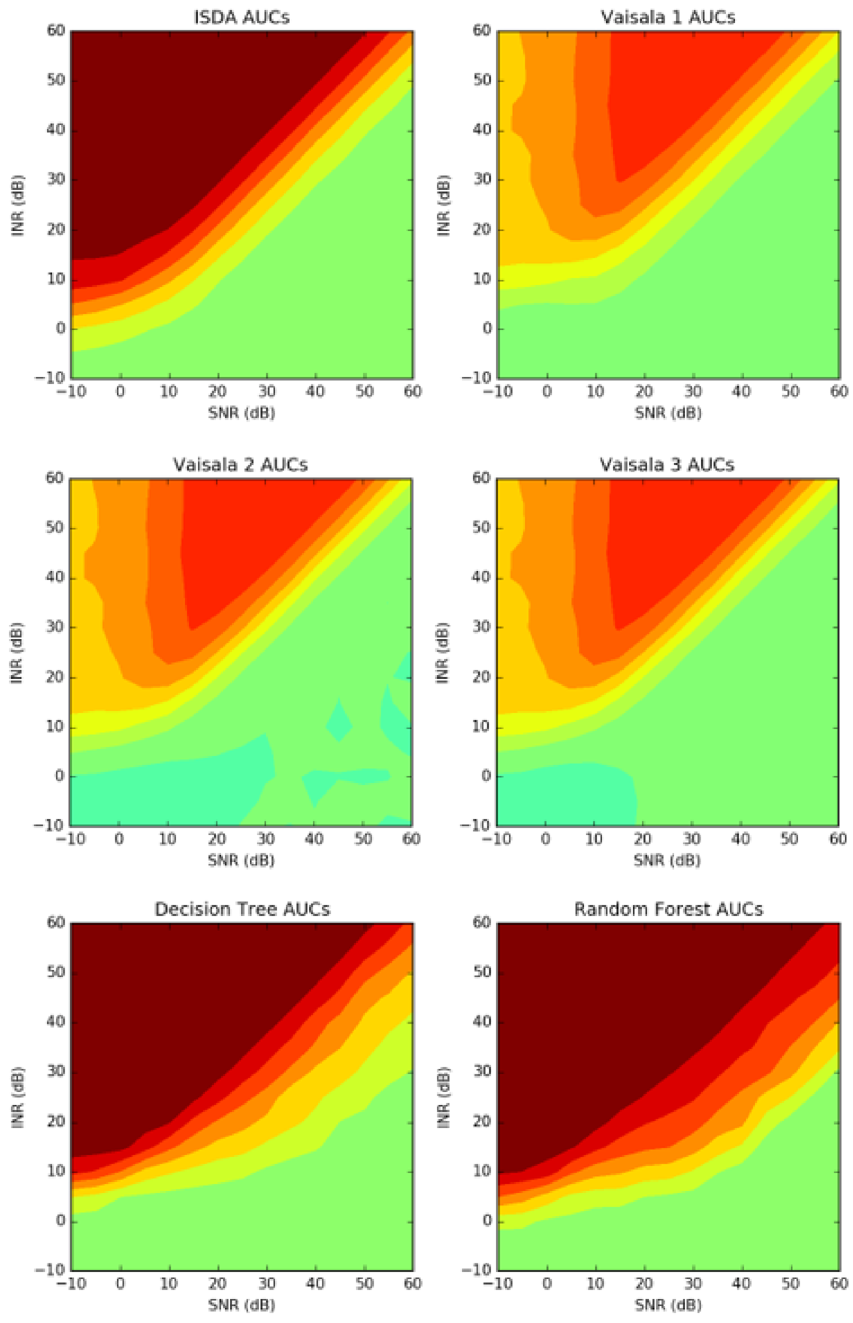


Figure 2.22: The AUCs of the RFI detection algorithms as a function of simulated weather data SNR and RFI INR. ISDA is shown in the top left; the Vaisala algorithms are shown in the top center left, center right, and middle; a decision tree is shown in the bottom far left; and a random forest is shown in the bottom center left. Dark red represents an AUC of 0.9; the color scale is common across the whole figure.

Chapter 3

Spatial Strategies

3.1 Background, Motivation, and Prerequisites

The WSR-88D is a single-dish radar, and so only one stream of complex in-phase and quadrature (IQ) data is converted to a digital signal, and the beampattern of the radar — essentially the “shape” of the electromagnetic wavefront — is immutable. Phased array radars are systems comprised of multiple antennas working in concert, and instead of manually steering a dish to point at a target, they coordinate the phase of the transmitted electromagnetic waves from each antenna so that their phases are aligned in the direction in which the radar wishes to “look”, effectively steering the beam in that direction without moving the antenna. Additionally, the shape of the beampattern can be altered as well by varying the magnitude and phase at each antenna [25].

Improvements in technology have opened the door for multiple streams of IQ data to be received at the subarray and even element level of phased arrays, opening the door for powerful algorithms to be applied to increase data quality. The DARPA Arrays at Commercial Timescales (ACT) program was created to encourage development of a common building block for digital phased arrays that would incorporate 80-90% of an array’s core functionality, reducing development time for radio frequency phased array systems to be used in applications ranging from radar to communications to electronic warfare [26, 27]. Because modern FPGAs are capable of performing tasks traditionally relegated to applications-specific integrated circuits, the processing chain of a radio frequency system can be completely moved to an FPGA. Additionally, modern FPGAs include single-precision floating-point DSPs, allowing more complex algorithms that require higher numerical precision to be implemented on an FPGA with fewer worries over dynamic range and scaling issues. Combined with the potential of modern high-level synthesis tools to generate FPGA layouts given only code in a higher-level language, there is potential for a high performance ACT module contained on an FPGA that performs both RF processing and more complex algorithms to improve data quality [28].

The proliferation of phased array technology is beneficial to weather radar as well — phased array radars are a likely candidate for the next generation of weather and surveillance radar [7, 8, 9, 3]. Though the dual polarization performance of phased arrays remains an issue [29, 30], experiments indicate that the increase in temporal resolution has very beneficial impacts on forecaster performance [31]. Interference is, of course, still an issue for phased array systems. For communications systems, interference can degrade the quality of the connection, potentially corrupting it to the point that data cannot be interpreted from the signal. For weather radar, ground clutter is always an issue, and accurate clutter mitigation algorithms are highly valued by the weather radar community. The simplest methods of mitigating clutter contamination are inflexible; notch filters eliminate undesired ground clutter along with any coincident weather data [32], while static clutter maps cannot adapt to changing environmental conditions' effect on the radar beam's path [33]. Better clutter mitigation can be effected by using the data gathered by the radar. The clutter decision tree proposed by Lee *et al.* detects clutter using a combination of radar moments, statistics, and an adaptive clutter map [34]. Fuzzy logic clutter detectors such as the Radar Echo Classifier proposed by Kessinger *et al.* [35] and the Clutter Mitigation Decision algorithm proposed by Hubbert *et al.* use fuzzy logic on radar moments to classify echoes, detect clutter, and, in conjunction with clutter filtering, can be used to address issues such as anomalous propagation[36]. The more complicated Gaussian Model Adaptive Processing (GMAP) method uses spectral features to recover weather signals after notch filtering has been applied [37]. Spectral features are also used in the CLutter Environment ANalysis using Adaptive Processing (CLEAN-AP) algorithm to detect clutter and recover the weather signal. Simple

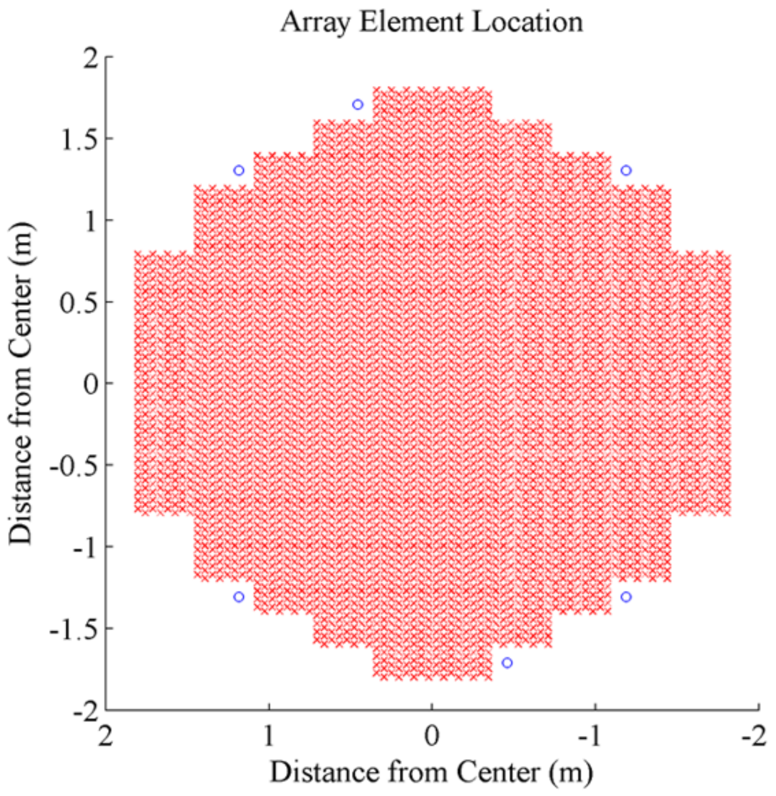


Figure 3.1: Sidelobe canceller channel locations (blue circles) on the NWRT PAR.

Bayesian classifiers (SBCs) are used in the spectrum clutter identification (SCI) algorithm in conjunction with new spectral parameters to detect clutter [38]. Incorporating data from multiple azimuths or polarizations can improve the clutter detection capabilities of these SBCs [39].

All of these sophisticated methods are designed for radars limited to data from a single antenna equipped with a single receiver. Unlike other meteorological radars, the converted Navy SPY-1A phased array radar at the NWRT is equipped with several auxiliary antennas in addition to its main antenna. These antennas are located around the periphery of the radar, as illustrated in Figure 3.1, enabling the gathering of spatial information from the radar return [40, 41]. Adaptive digital beamforming (ADBF) methods such as linearly-constrained minimum variance (LCMV) beamforming have shown their effectiveness at eliminating interference from jammers and other noise sources [42] and have been successfully applied to wind profilers [43], but have not yet been applied to meteorological surveillance radars. The multi-channel receiver at the NWRT provides the spatial data necessary to implement LCMV beamforming, allowing the introduction of nulls in the direction of ground clutter and other undesired targets (nullforming) [44]. Two implementations of the LCMV beamforming algorithm with quadratic constraint were executed on data collected at the NWRT [45].

This chapter is organized as follows: Section 3.2 gives an overview of several ADBF algorithms and their suitability for different applications, and Section 3.3 goes over the performance of these algorithms in several experiments. The experiment in Subsection 3.3.1 goes over the different ADBF algorithms and their simulated performance in a DARPA ACT module; the experiment in Subsection 3.3.2 uses one of these ADBF algorithms to mitigate ground clutter observed by the NWRT SPY-1A radar; and the experiment in Subsection 3.3.3 involves implementing several direction finding techniques for phased array radars on the DARPA ACT module.

3.2 Adaptive Digital Beamforming Algorithms

The spatial information available to phased array systems allows for more complex and better-performing beamforming strategies than the standard Fourier beamforming. Adaptive beamforming algorithms often take the form of statistically optimum beamformers, which use the statistics of the received signal to ”optimize” the signal according to certain criteria; the criteria generally are chosen to minimize the contributions of noise and interferers in the final beamformed signal [46]. Some of the more common statistically optimum beamformers are the multiple sidelobe canceller [47], maximum likelihood method [48], sample-matrix inversion [49], minimum variance distortionless response (MVDR), linearly-constrained minimum variance (LCMV) [50], and Applebaum algorithms [51]; other algorithms, such as the least mean squares (LMS)

[52, 46] and recursive least squares (RLS) [46], are gradient descent algorithms, choosing weights to best match the received signal to some desired signal. Each type and instance of these algorithms is subject to its own limitations in terms of information required and computational cost.

Implementation of effective real time adaptive digital beamforming was a goal of the DARPA ACT project. To that end, as part of the design process a trade study was performed on a subset of these algorithms to analyze the performance and costs of each of these algorithms; the algorithms studied are summarized in Table 3.1 and Section 3.2.1. Unless otherwise specified, the standard notation used in this chapter is: $\mathbf{x}$ is the signal received by the array; $\mathbf{w}$ is the weight vector to be applied during beamforming; y_d is the desired signal, also known as the reference signal; $\mathbf{S}$ is a covariance matrix, so $\mathbf{S_x}$ is the covariance matrix of $\mathbf{x}$; $\mathbf{n}$ is the noise environment, which is what the array would receive if the desired signal wasn't present; $\mathbf{v}_s$ is the steering vector toward the desired signal; $\mathbf{C}$ is the constraint matrix used in LCMV beamforming, and $\mathbf{g}$ is the value of the constraints; $\{\cdot\}^*$ is the conjugation operation, and acts to conjugate the object to which it is applied; $\{\cdot\}^H$ is the Hermitian operator, and acts to conjugate and transpose the object to which it is applied; λ and μ are parameters defined by the algorithm in which they are used; N_{el} is the number of digital channels used in the ADBF calculation, which in this paper is equal to the number of elements in the phased array antenna; N_t is the number of time samples used in the ADBF processing; and N_{con} is the number of constraints given to the LCMV algorithm; the abbreviation "Dir. BF" indicates that only standard Fourier beamforming was used to steer the antenna array [28].

3.2.1 Trade Study

The LMS, RLS, Applebaum, MVDR, and LCMV algorithms were analyzed in terms of their computational cost and ability to reduce the impact of interfering signals and noise on the final beamformed signal. A simulation environment was developed in Matlab to

Table 3.1: Summary of Beamforming Algorithms

Algorithm	Requirements	Equations	Computational Complexity
LMS	Desired signal	$\mathbf{w}(k) = \mathbf{w}(k-1) + \mu\mathbf{x}(k-1)y^*(k-1)$ $y(k) = y_d(k) - \mathbf{w}^H(k)\mathbf{x}(k)$	$2N_{el}N_t$
RLS	Desired signal	$\mathbf{v}(k) = \mathbf{P}(k-1)\mathbf{x}(k)$ $\mathbf{k}(k) = \frac{\lambda^{-1}\mathbf{v}(k)}{1+\lambda^{-1}\mathbf{x}^H(k)\mathbf{v}(k)}$ $\alpha(k) = y_d(k) - \mathbf{w}^H(k)\mathbf{x}(k)$ $\mathbf{w}(k) = \mathbf{w}(k-1) + \mathbf{k}(k)\alpha^*(k)$ $\mathbf{P}(k) = \lambda^{-1}\mathbf{P}(k-1) - \lambda^{-1}\mathbf{k}(k)\mathbf{v}^H(k)$	$4N_{el}^2N_t + 4N_{el}N_t + 2N_t$
Applebaum	Look direction, Knowledge of background environment	$\mathbf{w} = \mu\mathbf{S}_n^{-1}\mathbf{v}_s$	$N_{el}^3 + N_{el}^2N_t + 4N_{el}^2$
MVDR	Look direction	$\mathbf{w} = \mathbf{S}_\mathbf{x}^{-1}\mathbf{v}_s\left[\mathbf{v}_s^H\mathbf{S}_\mathbf{x}^{-1}\mathbf{v}_s\right]^{-1}$	$N_{el}^3 + N_{el}^2N_t + 4N_{el}^2 + 2N_{el}$
LCMV	Look direction, other user constraints (e.g. null directions)	$\mathbf{w} = \mathbf{S}_\mathbf{x}^{-1}\mathbf{C}\left[\mathbf{C}^H\mathbf{S}_\mathbf{x}^{-1}\mathbf{C}\right]^{-1}\mathbf{g}$	$N_{el}^3 + N_{el}^2N_t + 3N_{el}^2 + N_{el}^2N_{con} + N_{el}N_{con}^2 + N_{el}N_{con} + N_{con}^3 + 2N_{con}^2$

perform the trade study. This environment simulated the reception of user-customized signals by a phased array system and subsequently routed signals through the ACT module processing. The ACT module's digital downconversion processing chain includes mixing, decimation, filtration, ADBF, and application of the calculated beamforming weights on the filtered signal.

For the trade study, unless otherwise specified: the phased-array system had a 1×16 uniform linear array (ULA) with 0.5λ spacing at 5 GHz; the digital downconversion chain is as specified in [28]; the target signal was a randomly-generated QPSK signal with a carrier frequency of 5 GHz, a symbol bandwidth of 62.5 MHz, and an angle-of-arrival of 25 degrees from the array broadside; the interfering signal was similarly generated, but with an angle of arrival of 40 degrees from array broadside; the interferer's transmitter strength was 40 dB greater than that of the signal; there was no noise; 128 time samples were used in the ADBF calculations; a null was requested for LCMV in the direction of the interferer [28].

3.2.1.1 Least Mean Squares (LMS)

The least mean square algorithm is aimed at minimizing the error between the weighted received signal and the desired signal. It is computationally simpler than other methods at $O(n)$, but its convergence speed depends on the qualities of the received signal. More specifically, the shape of the error surface, which itself depends on the eigenstructure of the received signal, determines the speed of convergence. More widely spread eigenvalues result in slower convergence, meaning other algorithms with better convergence characteristics may be more suited to the task [46, 52].

3.2.1.2 Recursive Least Squares (RLS)

The recursive least squares algorithm is similar to the LMS algorithm. It is an algorithm that minimizes the error between the signal under the current weights and the desired signal; however, instead of aiming to minimize the current error like LMS does, the RLS algorithm seeks to minimize a weighted sum of the past squared errors. It is $O(n^2)$ and thus is more computationally complex than the LMS algorithm, but offers potentially faster convergence at the cost of being less numerically stable [46, 53].

3.2.1.3 Applebaum

The Applebaum beamformer is an optimum beamformer that maximizes SNR. To do this, it requires two things: a steering vector to the target, and knowledge of the noise environment. Applebaum calculates beamforming weights by inverting the covariance matrix calculated from the noise environment, and is thus of $O(n^3)$ complexity [51].

3.2.1.4 Minimum Variance Distortionless Response (MVDR)

The minimum variance distortionless response beamformer is optimum in that it minimizes the output energy by minimizing variance while maintaining a distortionless response from the desired signal's direction. Minimizing the variance overall serves to reduce the impact of any signals not from the desired direction [50] and any variation of gain in the desired direction is compensated for through appropriate normalization. Interferers are suppressed through the inherent spatial whitening of the beamforming weights [54]. The only information MVDR requires is the steering vector to the target. MVDR calculates beamforming weights by inverting the covariance matrix calculated from the noise environment, and is thus of $O(n^3)$ complexity [50].

3.2.1.5 Linearly Constrained Minimum Variance (LCMV)

The linearly constrained minimum variance beamformer is a more general case of the MVDR beamformer. It is an optimum beamformer since it seeks to minimize output energy, but can have additional user-defined constraints. It requires only the data contained in the user constraints. A typical user constraint is for a distortionless response from the desired direction; if this is the only constraint used, then the LCMV beamformer simplifies to the MVDR beamformer. Other potential constraints include nulls in the directions of known interferers [50, 42], and the algorithm can be extended to include quadratic constraints to preserve mainlobe power if more computational complexity isn't an issue, as was done in [55]. LCMV calculates beamforming weights by inverting the covariance matrix calculated from the noise environment, and is thus of $O(n^3)$ complexity [50, 42].

3.2.2 Increasing ADBF Algorithm Numerical Stability

Both the MVDR and LCMV algorithms are susceptible to signal mismatch and array perturbations. Incorporating a quadratic constraint to minimize $||\mathbf{w}||^2$ can be used to ameliorate this issue. This LCMV QC beamformer minimizes the output noise power subject to the constraint that

$$\mathbf{w}^H\mathbf{w} = T \leq T_\circ \tag{3.1}$$

Using the method of Lagrangian multipliers, the optimum weight vector that satisfies both the linear and quadratic constraints is found to be

$$\mathbf{w} = (\mathbf{S_x} + \beta\mathbf{I})^{-1}\mathbf{C}[\mathbf{C}^H(\mathbf{S_x} + \beta\mathbf{I})^{-1}\mathbf{C}]^{-1}\mathbf{f} \tag{3.2}$$

where β is the amount of diagonal loading added to the signal covariance matrix $\mathbf{S_x}$. There is no closed form solution for β; instead, the β that satisfies the quadratic constraint must be solved for numerically. When $\beta = 0$, the standard LCMV solution in Table 3.1 is obtained; as $\beta \to \infty$, the beamformer approaches the quiescent beamformer [42].

The classic method discussed above requires that the inverse of a matrix be calculated for each β until a β is found that satisfies the quadratic constraint. While the end results of this technique are effective, the computational cost of so many matrix inversions can quickly become prohibitive. A new method of implementing quadratic constraints proposed in [56] reduces the use of the expensive matrix inversion operation using recursive least squares (RLS) updating to the quadratic constraint. First, the weight vector $\mathbf{w}$ is split into two components, one in the constraint subspace ($\mathbf{w}_q$) and one in a subspace orthogonal to it ($\mathbf{w}_a$). The weight vector can then be represented as

$$\mathbf{w} = \mathbf{w}_q - \mathbf{B}\mathbf{w}_a \tag{3.3}$$

where B is a blocking matrix orthogonal to the constraint subspace C. This means that the the quadratic constraint expressed in equation (3.1) can be rewritten as

$$\mathbf{w}_a^H \mathbf{w}_a \leq T_\circ - \mathbf{w}_q^H \mathbf{w}_q \triangleq \alpha^2. \tag{3.4}$$

The quiescent weight vector, $\mathbf{w}_q$, is fixed as

$$\mathbf{w}_q = \mathbf{C}(\mathbf{C}^H \mathbf{C})^{-1}\mathbf{f} \tag{3.5}$$

Using this new notation, the standard LCMV solution in the equation in Table 3.1 can be represented as

$$\mathbf{w}_a = (\mathbf{B}^H \mathbf{S}_\mathbf{x} \mathbf{B})^{-1} \mathbf{B}^H \mathbf{S}_\mathbf{x} \mathbf{w}_q \tag{3.6}$$

By setting $\mathbf{z} = \mathbf{B}^H \mathbf{X}$ and $\mathbf{y}_\mathbf{c} = \mathbf{w}_\mathbf{q}^H \mathbf{X}$, the covariance matrix of $\mathbf{z}$ becomes $\mathbf{S}_\mathbf{z} = \mathbf{B}^H \mathbf{S}_\mathbf{x} \mathbf{B}$ and the cross-correlation vector of $\mathbf{z}$ and $\mathbf{y}_\mathbf{c}$ becomes $\mathbf{p}_\mathbf{z} = \mathbf{B}^H \mathbf{S}_\mathbf{x} \mathbf{w}_q$. The adaptive weights $\mathbf{w}_a$ from the standard LCMV solution then become

$$\mathbf{w}_a = \mathbf{S}_\mathbf{z}^{-1} \mathbf{p}_\mathbf{z} \triangleq \tilde{\mathbf{w}}_a \tag{3.7}$$

When the quadratic constraint is added, the adaptive weight vector is changed only slightly, becoming

$$\mathbf{w}_a = (\mathbf{S}_\mathbf{z} + \lambda \mathbf{I})^{-1} \mathbf{p}_\mathbf{z} \tag{3.8}$$

where λ is the diagonal loading component.

Rewriting the quadratically constrained adaptive weight vector $\mathbf{w}_a$ in terms of the linearly constrained adaptive weight vector $\tilde{\mathbf{w}}_a$ yields:

$$\mathbf{w}_a = (\mathbf{I} + \lambda \mathbf{S}_\mathbf{z}^{-1})^{-1} \mathbf{S}_\mathbf{z}^{-1} \mathbf{p}_\mathbf{z} = (\mathbf{I} + \lambda \mathbf{S}_\mathbf{z}^{-1})^{-1} \tilde{\mathbf{w}}_a. \tag{3.9}$$

Using only the first two terms of the Taylor series expansion transforms $\mathbf{w}_a$:

$$\mathbf{w}_a \approx (\mathbf{I} - \lambda \mathbf{S}_{\mathbf{z}}^{-1})\tilde{\mathbf{w}}_a = \tilde{\mathbf{w}}_a - \lambda \mathbf{v}_a \tag{3.10}$$

where $\mathbf{v}_a = \mathbf{S}_{\mathbf{z}}^{-1}\tilde{\mathbf{w}}_a$. Substituting equation 3.10 into equation 3.4 provides a quadratic equation in λ:

$$\mathbf{w}_a^H \mathbf{w}_a - \alpha^2 = a\lambda^2 + b\lambda + c = 0 \tag{3.11}$$

where $a = ||\mathbf{v}_a||^2$, $b = -2\Re\{\mathbf{v}_a^H \tilde{\mathbf{w}}_a\}$, and $c = ||\tilde{\mathbf{w}}_a||^2 - \alpha^2$. Solving for the smaller root of this equation, i.e.

$$\lambda = \frac{-b - \Re\{\sqrt{b^2 - 4ac}\}}{2a} \tag{3.12}$$

allows for λ, the approximate amount of diagonal loading needed to meet the quadratic constraint, to be directly calculated [56].

3.2.3 Matrix Inversion Lemma (MIL)

The computationally costliest operations in the Applebaum, MVDR, and LCMV algorithms are the inversions of the covariance matrices. For an N channel system, an ADBF calculation using a sample covariance matrix computed from only 5N samples is capable of computing weights that are within 1 dB of optimal [49]. The covariance matrix may be estimated by using the formula

$$\mathbf{S}_{\mathbf{x}} = \tfrac{1}{N} \sum_{k=1}^{n} \mathbf{x}_k \mathbf{x}_k^H \tag{3.13}$$

where N is the total number of samples, k is the iterator in the summation, and $\mathbf{x}$ is the $L \times N$ signal matrix where L is the number of channels. If this sum is expanded, we find that, as new information comes in, the current estimate of the covariance matrix may be updated using the formula

$$\mathbf{S}_{\mathbf{x}}(k) = \tfrac{k-1}{k}\mathbf{S}_{\mathbf{x}}(k-1) + \tfrac{1}{k}\mathbf{x}_k \mathbf{x}_k^H \tag{3.14}$$

We can use this information to keep a running estimate of the inverse of the covariance matrix:

$$[\mathbf{S_x}(k)]^{-1} = \left[\frac{k-1}{k}\mathbf{S_x}(k-1) + \frac{1}{k}\mathbf{x}_k\mathbf{x}_k^H\right]^{-1} = \left[\frac{k}{k-1}\mathbf{S_x}(k-1) + \left(\frac{1}{\sqrt{k}}\mathbf{x}_k\right)\left(\frac{1}{\sqrt{k}}\mathbf{x}_k\right)^H\right]^{-1} \tag{3.15}$$

The matrix inversion lemma, also called the Woodbury matrix identity [57] is relevant:

$$(\mathbf{A}+\mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{C}^{-1}+\mathbf{VA}^{-1}\mathbf{U}\right)^{-1}\mathbf{VA}^{-1} \tag{3.16}$$

By setting $\mathbf{A} = \frac{k-1}{k}\mathbf{S_x}(k-1) \Leftrightarrow \mathbf{A}^{-1} = \frac{k}{k-1}\mathbf{S_x}^{-1}(k-1)$, $\mathbf{U} = \frac{1}{\sqrt{k}}\mathbf{x}_k = V^H$, and $\mathbf{C} = 1$, the matrix inversion lemma can be leveraged to provide an alternate form of the update formula for the inverse of the covariance matrix:

$$\mathbf{S_x}^{-1}(k) = \tfrac{k}{k-1}\mathbf{S_x}^{-1}(k-1) - \frac{\frac{k}{k-1}\mathbf{S_x}^{-1}(k-1)\mathbf{x}_k\mathbf{x}_k^H\mathbf{S_x}^{-1}(k-1)}{(k-1)+\mathbf{x}_k^H\mathbf{S_x}^{-1}(k-1)\mathbf{x}_k} \tag{3.17}$$

Using this formulation has the potential to reduce the computational complexity. If N_c is the number of receiver channels and N_s is the number of samples recorded, then calculating the covariance matrix is an $8N_c^2N_s + 2N_c^2$ operation and inverting that covariance matrix costs an additional N_c^3 operations. Each iteration of the covariance inverse update formula from equation 3.17 costs $45N_c^2 + 5N_c$ operations, so keeping a running estimate of the covariance matrix inverse will cost $45N_c^2N_s + 5N_cN_s$ operations by the end of the CPI. When the number of channels is small, the difference may not be particularly noticeable, but as the number of channels increases, an $O\left(N_c^3\right)$ operation will become prohibitively costly; using the MIL formulation could work to cut some of these costs [58].

3.3 Results

3.3.1 Interference Mitigation for the DARPA ACT Module

Algorithm performance was also assessed as part of the trade study. The default scene (desired signal at 25 degrees, interferer at 40 degrees, SIR of -40 dB, and no noise) was used, with LCMV requesting a null from the direction of the interferer instead of in an arbitrary direction like above. Two statistics in particular were examined: the signal-to-interference ratio (SIR) and the bit error rate (BER). The SIR is the desired signal's power divided by the summed power of the interferers; a higher SIR indicates better performance. The bit error rate was the number of erroneous bits in the received signal after ADBF processing and QPSK demodulation divided by the total number of bits received; a lower bit error rate indicates better performance. The simulation environment was run 1024 times and the statistics averaged as appropriate.

Figure 3.2 shows the average signal-to-interference-plus-noise ratio (SINR) as a function of the number of samples used in ADBF processing; once again, higher SINR indicates better performance. The SINR is remarkably stratified based on the algorithm. On average, LCMV has the highest SINR, followed by Applebaum and MVDR, both of which perform slightly better with a larger number of samples used. The LMS and RLS algorithms do perform better than the case without any ADBF processing, but their performance is less than that of the covariance-matrix-based algorithms. Figure 3.3 shows the average BER as a function of the number of samples used in ADBF processing. There are two clear groups: LMS and RLS beamforming perform only slightly better than the case without any ADBF processing, while MVDR, Applebaum, and LCMV perform noticeably better in this scenario. The performance of MVDR, LCMV, and Applebaum beamforming increases slightly with the number of samples used in ADBF processing.

The LMS and RLS algorithms were not chosen for implementation on the ACT common module; both required knowledge of the desired signal, which may be reasonable for communications applications but is not possible for radar applications. The other three algorithms were considered in more detail. All three of the remaining algorithms required an inversion of a covariance matrix, meaning that algorithm complexity would be by default proportional to the number of elements cubed ($O(N_{el}^3)$), making these

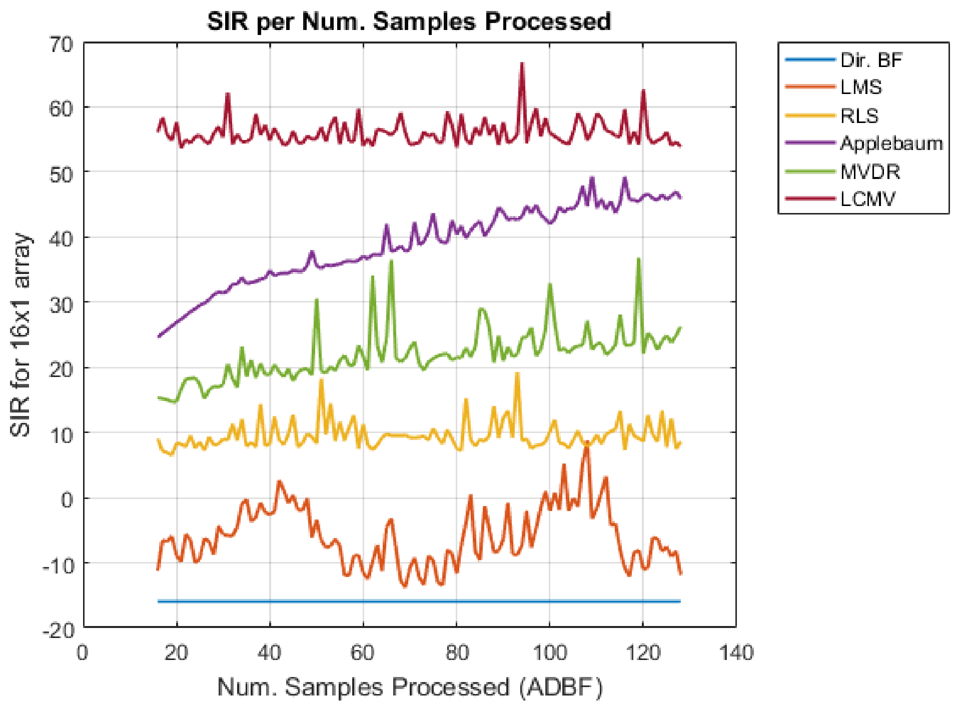


Figure 3.2: The ADBF algorithms' performance in terms of SIR is actually not noticeably correlated with the number of samples used. There are, however, noticeable differences in SIR performance based on the ADBF algorithm used. Note that in this simulation, a null was requested in the direction of the interferer, which allows LCMV to perform better than MVDR.

beamforming algorithms very computationally costly. Applebaum additionally required knowledge of the environment without the desired signal. This is potentially feasible for defense radar — if the desired signal transmits only intermittently and the radar knows when the desired signal will be transmitting, then a characterization of the noise environment may be done outside of that time; this does require the interferers to cooperate and transmit during the noise environment characterization. While it is possible to make Applebaum work, relying on the environment to cooperate was not an ideal setup. MVDR and LCMV required no information about the environment, only information about where to look and, in the case of LCMV, where to place nulls or apply other constraints.

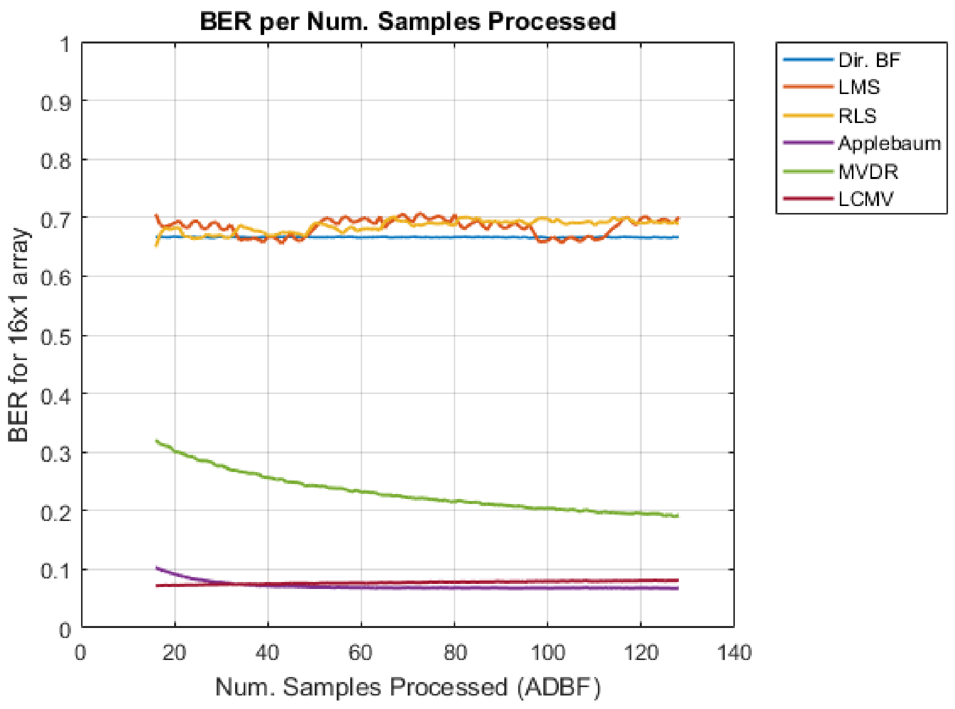


Figure 3.3: The optimum beamforming methods perform noticeably better in terms of bit error rate than do the gradient descent beamforming algorithms. Additionally, the optimum methods perform slightly better as more samples are used for ADBF processing.

Applebaum was not selected for the ACT module despite its smaller computational complexity relative to the other covariance-matrix-inversion algorithms. Because its performance is predicated on its ability to get an accurate observation of the environment sans the desired signal, if a situation arose where this was impossible (e.g. no control over the desired signal, or smart jammers that only activate when the desired signal transmits), its performance could suffer greatly. Similarly LCMV, despite its useful ability to introduce nulls in requested directions, was also not chosen for the ACT module — because space on the FPGA was a valuable and very limited resource, the additional and potentially variable computational cost of LCMV over MVDR was not deemed worth the additional flexibility provided by user constraints.

The ADBF algorithm ultimately chosen for implementation in the DARPA ACT Phase 2 module was MVDR beamforming. Its predictable computational complexity, and thus predictable FPGA usage was an important factor, as was that its only requirement was knowledge of the look direction [28]. For other applications, where computational complexity and space on the electronics was less of a limiting factor, LCMV would be a good choice, especially when the location of interferers is known. The interference present in the data from Figure 1.4 is almost certainly from a nearby airport surveillance radar; the location of this radar is known, and always asking LCMV for a null in that direction could have acted to mitigate all RFI from that radar without the user having to worry about it.

3.3.2 Spatial Filtering Applied to Ground Clutter

The trade study discussed in Section 3.3.1 above was specifically for the DARPA ACT module, and concerned the choice and performance of ADBF algorithms on simulated data with a desired signal and an interferer. There are applications in domains outside that of active interferers; for example, ground clutter can be viewed as a type of interference: it's a non-weather signal that can significantly affect meteorological variable values [16]. In this section, data with ground clutter contamination was collected from the NWRT PAR using the 6-channel receiver comprised of the sum channel (containing signal from the main array) and five sidelobe canceller channels. The two quadratically constrained LCMV methods discussed in Section 3.2.2 were applied to this data to filter out the ground clutter. Because the multi-channel receiver is not phase calibrated, no directional constraints could be applied; instead the LCMV constraint was chosen to reject data common to the sum channel (the first channel) and the sidelobe canceller channels, effectively forming nulls in the direction of any strong signal from the sidelobes:

$$\mathbf{C} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, T_\circ = 2$$

Data collected on October 12, 2012 was inspected to find suitable locations for initial tests of the LCMV beamforming performance. The data can be seen pre-processing in Figure 3.4. From this data, range gates where varying amounts of clutter from the sidelobes contaminated the weather signal were chosen for further study. The velocity spectrum at each of the range gates (range gates 136, 138, 139, and 143 at 355° azimuth) was computed. In all four range gates, the center of the weather signal spectrum was located around 10 m/s. Range gates 136, 139, and 143 additionally show the characteristic ground clutter return at 0 m/s. Figure 3.5 shows the spectra both before and after LCMV beamforming using both the numerical search for the quadratic constraint [50] and the closed form solution for the quadratic constraint [56] discussed in Section 3.2.2. Velocity estimates calculated using pulse pair processing are included in the Figure 3.5, and demonstrate the improvement LCMV beamforming has upon the data. Velocity estimates that were previously not near the center of the weather spectrum, as in range gates 136, 139, and 143, are restored to the center of the weather spectrum once LCMV beamforming removes or mitigates the ground clutter. Velocity estimates of data that was not impacted by ground clutter, such as range gate 138, are not impacted.

Next, the effects of using LCMV beamforming at every range gate along an azimuth were examined. Figure 3.6 shows the results of the application of both types of LCMV QC beamforming. While the final results do slightly differ between the two methods, both result in make major alterations at similar locations (generally closer to the radar, where ground clutter is more prevalent).

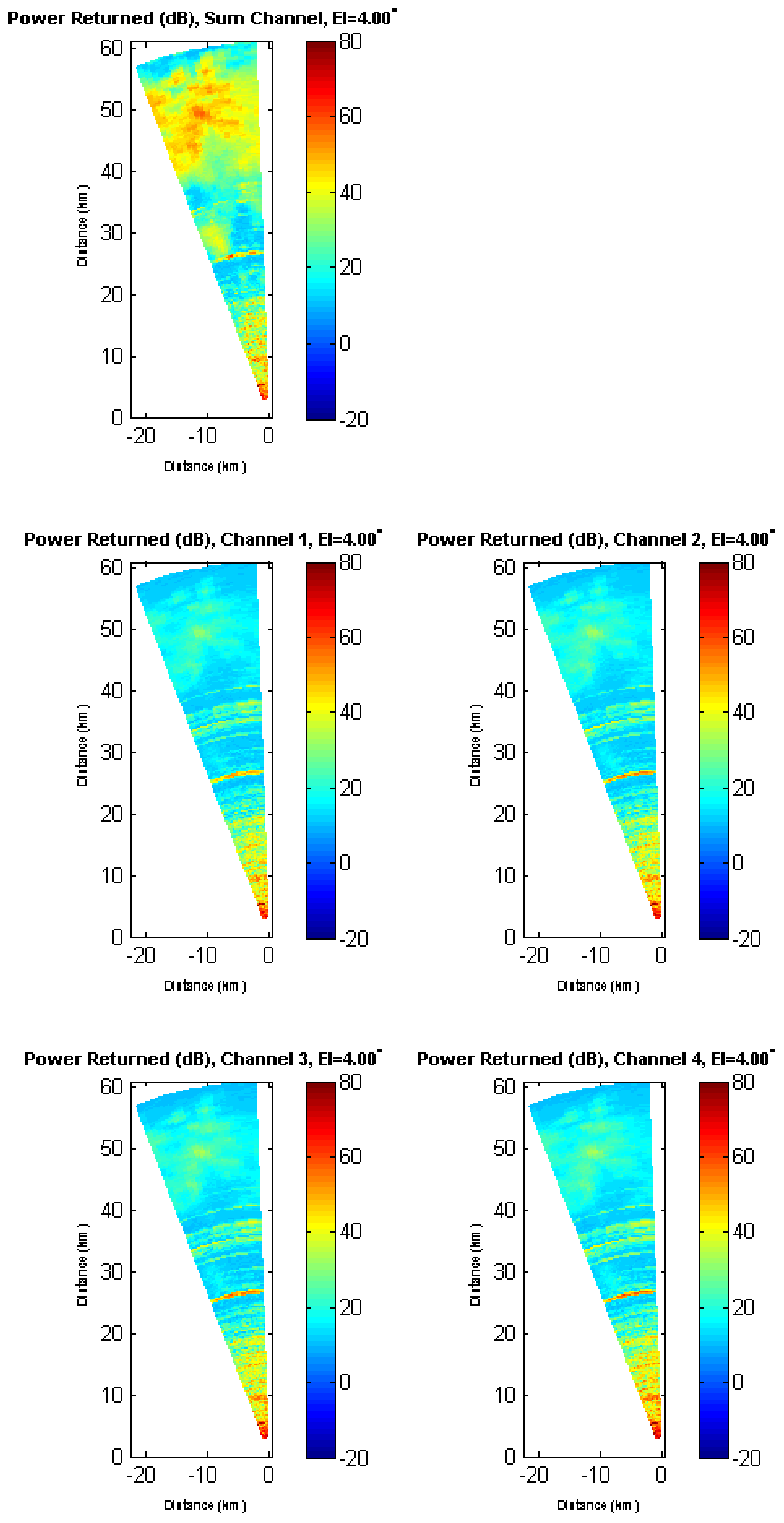
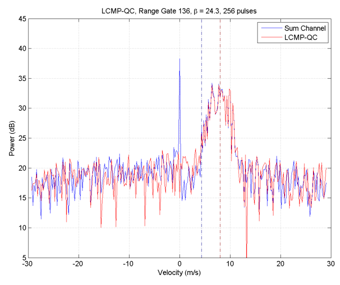
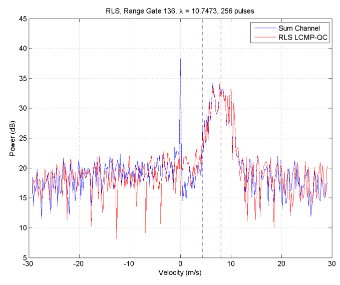


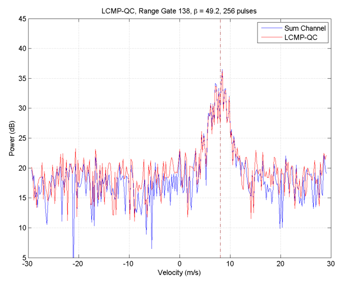
Figure 3.4: PPIs of received echoes of each channel of the multi-channel receiver. The sum channel is in the top left.



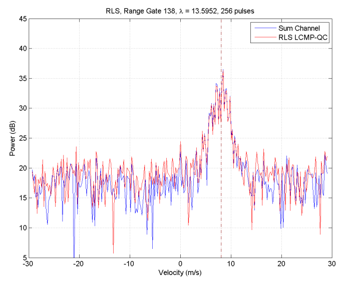
(a) Range Gate 136: LCMP QC



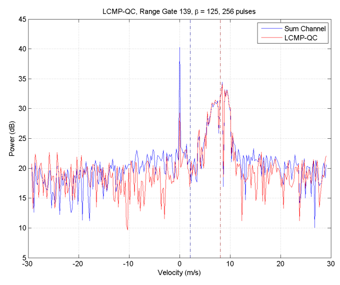
(b) Range Gate 136: LCMP QC (RLS)



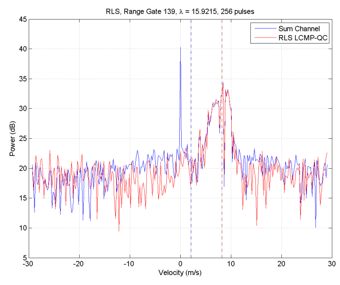
(c) Range Gate 138: LCMP QC



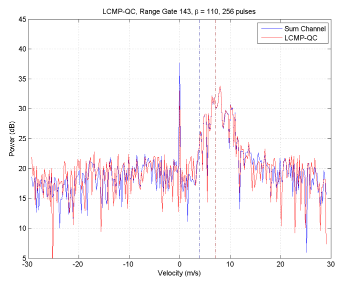
(d) Range Gate 138: LCMP QC (RLS)



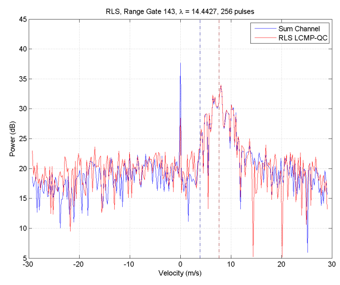
(e) Range Gate 139: LCMP QC



(f) Range Gate 139: LCMP QC (RLS)



(g) Range Gate 143: LCMP QC



(h) Range Gate 143: LCMP QC (RLS)

Figure 3.5: Velocity spectra and estimates of assorted range gates before (in blue) and after (in red) each of the LCMV beamforming methods. The velocity estimates are shown as vertical dashed lines from before (in blue) and after (in red) LCMV beamforming.

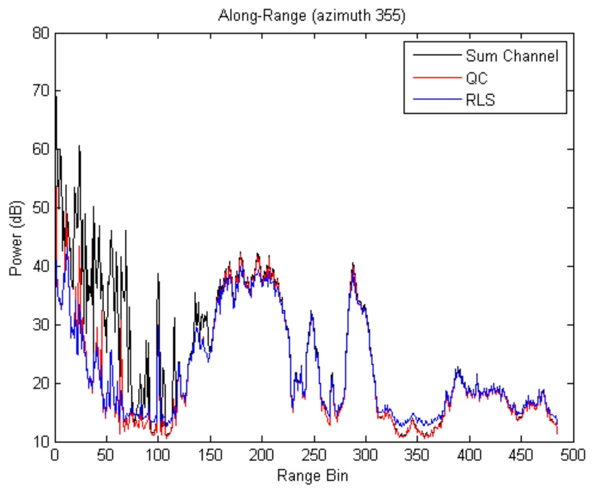
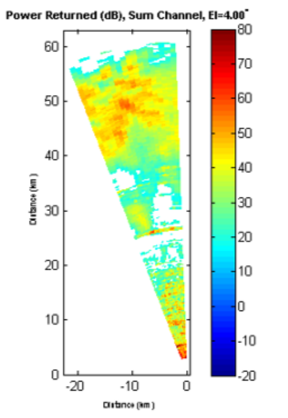


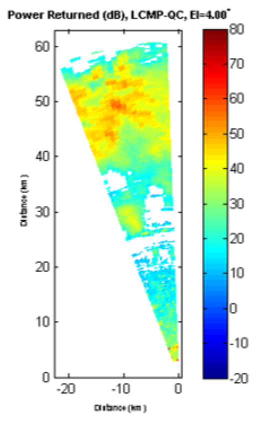
Figure 3.6: Along-azimuth results of application of LCMV beamformer.

After examining the results at individual range gates and along whole azimuths, the LCMV beamforming strategy was applied to a whole sector of a plan position indicator (PPI). The PPIs of the input to the multi-channel receiver were shown in Figure 3.4. The results of LCMV beamforming are shown in Figure 3.7 Figure 3.7a is the power and Figure 3.7d is the velocity of the raw input data — no modifications are made, and contamination due to clutter is readily apparent. Figures 3.7b and 3.7e show the result of LCMV beamforming using the numerical search for the quadratic constraint — much of the clutter contamination is removed, especially close to the radar. This is easily visible in the radial velocity, and some weather signals become less affected by ground clutter as well. Figures 3.7c and 3.7f have much the same results — much of the clutter is attenuated or removed; however, the weather signal is also more attenuated, as can be seen by contrasting Figures 3.7g and 3.7h.

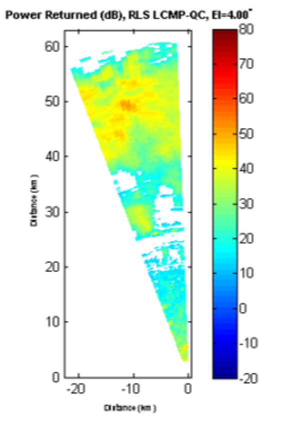
Efficiency concerns arise when examining the workings of the LCMV QC beamformer with no closed solution from Section 3.2.2 [42]. A numerical search must be performed at each gate, and the minimum amount of diagonal loading is desired, meaning that the search for the constraint must start from zero. Ways to increase the efficiency of this search are desired, so the behavior of the constraints was examined in Figure 3.8.



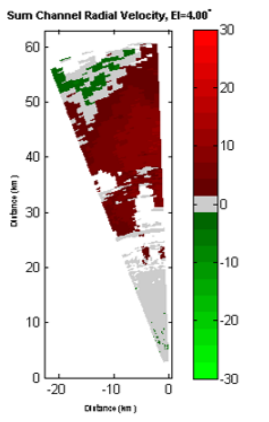
(a) Sum Channel Power (no modifications)



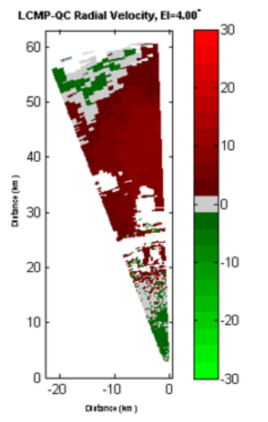
(b) Power After LCMV QC Processing



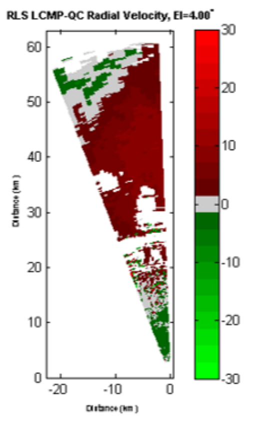
(c) Power After RLS LCMV QC Processing



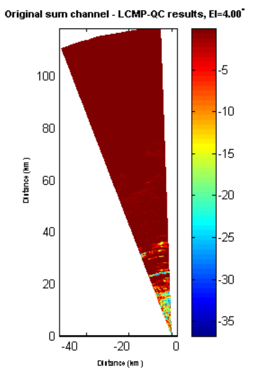
(d) Velocity Sum Channel (no modifications)



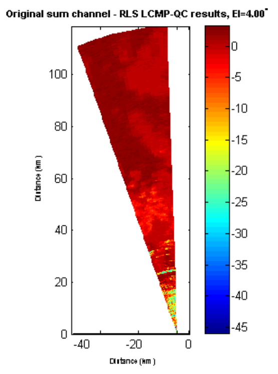
(e) Velocity After LCMV QC Processing



(f) Velocity After RLS LCMV QC Processing

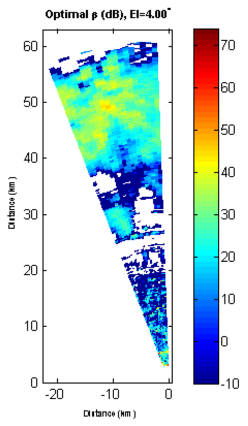


(g) Magnitude of Power Change After LCMV QC Processing

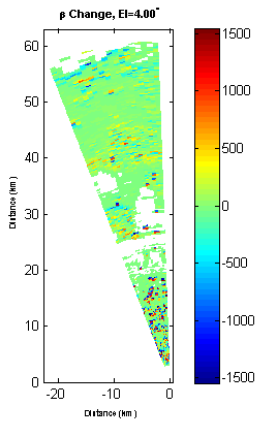


(h) Magnitude of Power Change After RLS LCMV QC Processing

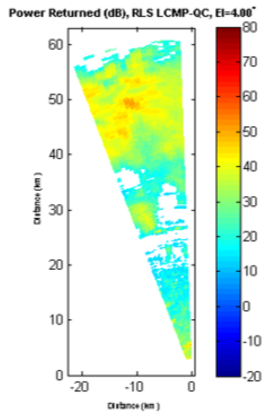
Figure 3.7: PPIs of the power (top), radial velocity (middle), and power change from before (left) and after (middle, right) LCMV processing.



(a) Optimal β (dB) for LCMV QC



(b) Gate-to-Gate Change in β for LCMV QC



(c) Optimal λ for RLS LCMV QC

Figure 3.8: PPIs of quadratic constraints. The β parameter (left and middle) must be found via numerical search; the computational cost could quickly accrue. Because of the high dynamic range of the β and λ parameters, they are shown on a log scale in the left and right plots so that features may be more easily seen.

Figure 3.8c shows the behavior of the solution to the closed-form RLS LCMV beamforming method; since the closed-form solution is known, there is no advantage to be gained by examining its spatial behavior. Figure 3.8a, on the other hand, shows that while there are some sharp jumps in β, for the most part the amount of loading does not vary much from gate to gate. This smoothness is quantified in Figure 3.8b, which shows the gate-to-gate change in β. Since this change in β is so small over much of the PPI, using the previous gate's β as a starting point for the search for a new β can potentially save significant time.

The beamforming techniques were applied for the first time to meteorological data collected at the NWRT phased array radar to filter ground clutter. These techniques have the potential to simplify or even eliminate the need for clutter filtering that is currently mandatory at NEXRAD sites. Initial results show that, while some ground clutter was still present after the adaptive beamforming, a large part of it was removed or at least attenuated. With the closed form solution provided by [56], this adaptive filtering could be done without the need for numerical searches, resulting in relatively quick and efficient clutter filtering [55].

3.3.3 Direction Finding using the DARPA ACT Module

The spatial information observed by the antenna can be used in other applications than adaptive digital beamforming. For example, direction finding algorithms use the spatial data received by a digital phased array to determine the direction from which a signal is coming. This has myriad potential applications; for example, if direction finding is used to find the direction of a signal interfering with a radar, LCMV beamforming can be employed with a constraint added to introduce a null in that direction.

A simple experiment was performed using the Collins Aerospace version of the DARPA ACT module to do direction finding. The ACT module was connected with an 8x8 S-band array. This incarnation of the ACT module only supports 16 digital channels, so only one row of eight elements connected to the ACT module was used in the direction finding. Two algorithms were tested: a beamscan algorithm called the Bartlett beamformer, and a subspace algorithm called MUSIC.

The Bartlett beamformer for determining a spatial spectrum is defined as:

$$\hat{P}_B = \mathbf{v}^H \mathbf{S_x} \mathbf{v} \tag{3.18}$$

where $\mathbf{v}$ is the set of steering vectors that define the search space, and $\mathbf{S_x}$ is the signal covariance matrix. This algorithm is simple in concept — essentially a beam is digitally steered in the directions defined by $\mathbf{v}$ and the power from that direction is recorded. Peaks in the spatial spectrum are noted as the locations of possible sources. It should be noted that the antenna's intrinsic beampattern comes into play, so sources located more closely in angle than the antenna's resolution won't be easily distinguished. Additionally, there will be a number of local maxima equal to the number of antenna elements, as can be seen in Figure 3.9; that does not mean that there are that many sources detected [42].

The MUltiple SIgnal Classification (MUSIC) algorithm for determining a spatial spectrum is defined as:

$$\hat{P}_{MU} = \left[\mathbf{v}^H \hat{\mathbf{U}}_N \hat{\mathbf{U}}_N^H \mathbf{v}\right]^{-1} \tag{3.19}$$

where $\mathbf{v}$ is again the set of steering vectors that define the search space, and $\hat{\mathbf{U}}_N$ is the matrix comprised of the estimated eigenvectors of the noise subspace. The noise subspace matrix is calculated by performing an eigendecomposition on the signal covariance matrix $\mathbf{S_x}$ and removing the eigenvectors associated with the largest D eigenvalues, where D is the number of signals in the environment. If the user is incorrect in specifying D, then algorithm performance can degrade rapidly. This algorithm essentially looks for the directions in the search space that are most orthogonal to the noise subspace; the directions with peaks are those most orthogonal to the noise subspace and are marked as source directions [42].

To verify the capabilities of the ACT module at direction finding, the ACT module was set up in a tapered anechoic chamber. A horn transmitting a tone at S-band was set up at one end of the chamber, and the phased array antenna was set up on a post at the other. The receiving antenna was rotated; this has the effect of changing the source direction relative to the plane of the antenna, and was a suitable setup for testing the direction finding algorithms.

Some snapshots of the spatial spectrums using Bartlett and MUSIC with the antenna at different angles are shown in Figure 3.10. The algorithms do a passable job at direction finding. The results are not as exact as would be desired — for example, two signals are detected by the MUSIC algorithm instead of just one — but a lot of this is attributable to some drawbacks in experimental design and the resources available. First, due to time and equipment constraints, the antenna with the module was rotated manually, and thus the angles of the source relative to the antenna are only approximate.

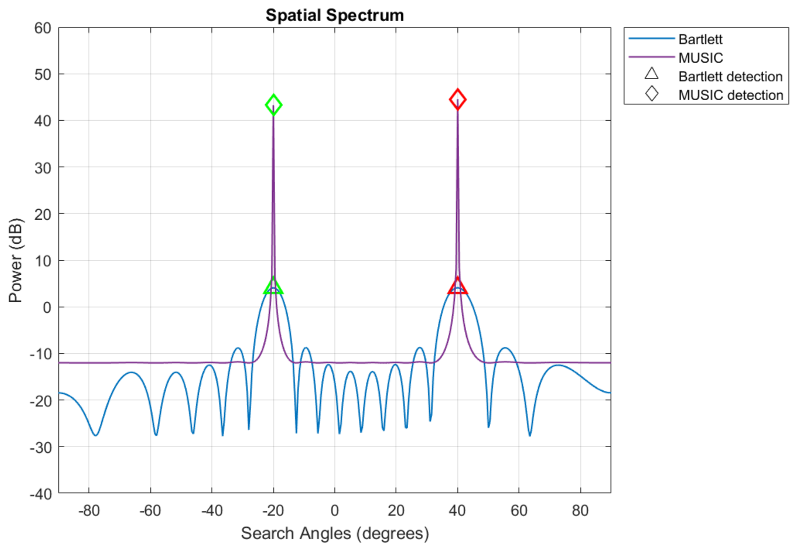
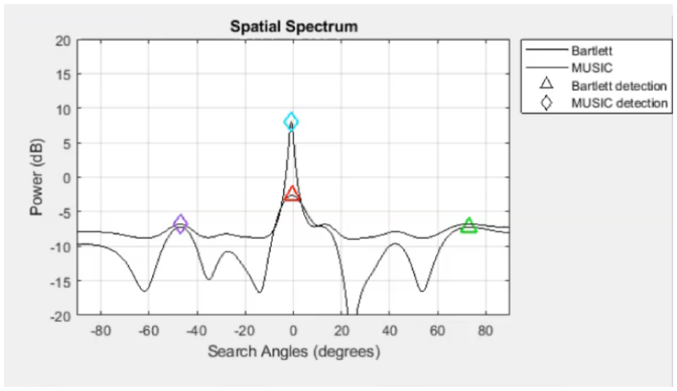
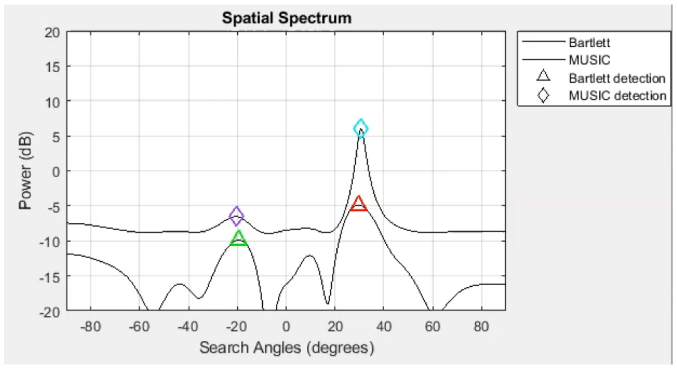


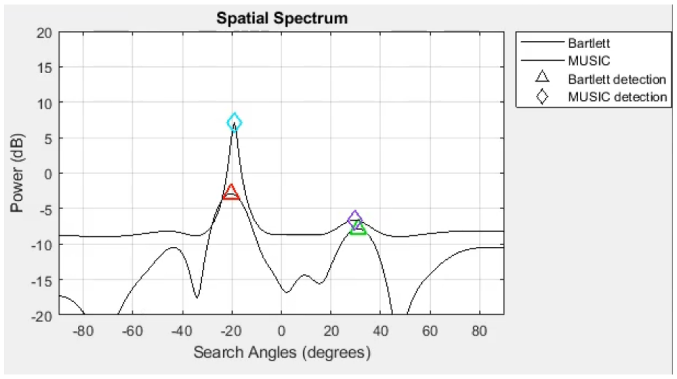
Figure 3.9: An example of direction finding results. An ACT module output comprised of two strong and equal-power signals at different angular locations was simulated and then sent into Matlab post-processing that did direction finding analysis. Shapes indicate detections by different algorithms — triangles represent Bartlett and diamonds represent MUSIC — and each unique shape color indicates a detection at a new angle. Note the many peaks in the Bartlett results compared to MUSIC's two, a result of the antenna beampattern manifesting in the Bartlett processing.



(a) Spatial spectrum, transmitter at antenna broadside.



(b) Spatial spectrum, transmitter at 30 degrees off antenna broadside.



(c) Spatial spectrum, transmitter at -20 degrees off antenna broadside.

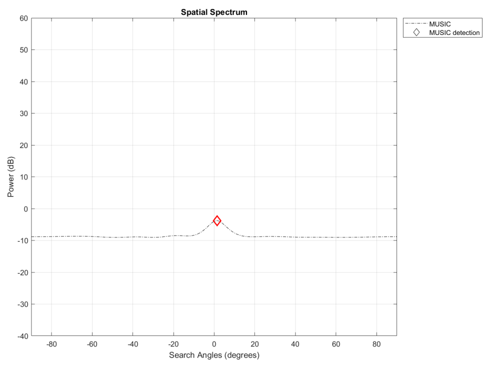
Figure 3.10: Direction finding analysis of the ACT module experimental setup with the antenna oriented at different angles. Note that the environment is set up to find two signals; MUSIC performance was severely degraded when using $D = 1$.

Second, and again due to time and equipment constraints, there was no opportunity for proper calibration of the antenna connected with the module. Instead, very simple digital equalization was performed to improve channel matching and thus direction finding performance; this stopgap measure performed relatively well, but, given that the Bartlett spatial spectrum with does not look much like the simulated beampattern, it is reasonable to assume that there is room for improvement in channel matching. Finally, the doors of the chamber were open by necessity; it is possible that the "second" signal that MUSIC needed to search for in order to perform well is the manifestation of some multipath reception. Regardless, in order to function properly, MUSIC needed to remove at least two eigenvectors associated with the largest eigenvalues rather than one, so two distinct directions must have been associated with the signal subspace. The performance contrast can be seen in Figure 3.11. When only one transmitter is sought, as in Figure 3.11a, there is no direction particularly orthogonal to the noise subspace; when two or more transmitters are sought, as in Figures 3.11b and 3.11c, there is a clear peak in the spatial spectrum.

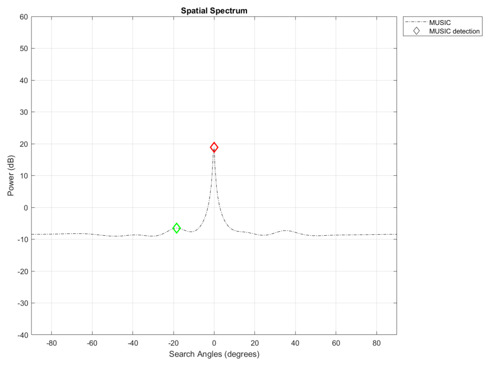
This simple experiment verified the potential for direction finding using an all-digital phased array. While MUSIC gives more distinct and noticeable peaks associated with the spatial spectrum, it must know how many signals are present or else performance is degraded; in contrast, Bartlett does not rely on knowledge of the signal environment, but performs poorly when two sources are located closely in angle. Direction finding can be employed in a variety of applications, including radar and electronic warfare, to improve performance and otherwise accomplish mission goals.

3.4 Conclusions

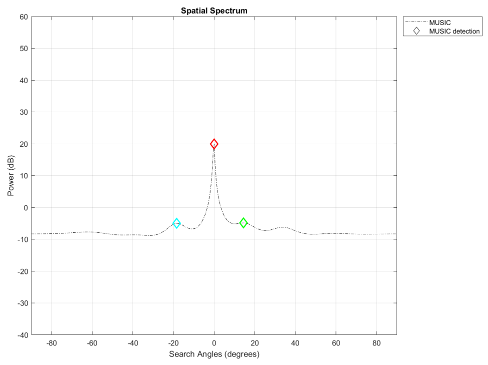
The advent of digital control at the subarray and element level allows spatial processing techniques to be used in a wide variety of applications. The DARPA Arrays at Commercial Timescales project seeks to enable such processing. A variety of adaptive digital



(a) Spatial spectrum, MUSIC looking for 1 source.



(b) Spatial spectrum, MUSIC looking for 2 sources.



(c) Spatial spectrum, MUSIC looking for 3 sources.

Figure 3.11: MUSIC direction finding analysis of the ACT module. Despite there only being 1 transmitter, two signals show up in the noise subspace.

beamforming algorithms were studied and compared in terms of efficacy and computational complexity for radar and communications applications. Ultimately, MVDR was selected for implementation on the DARPA ACT project due to its high performance and predictable computational complexity. Two techniques were explored to enhance the performance of the MVDR algorithm: first, the numerical stability of the MVDR algorithm was enhanced with diagonal loading of the signal covariance matrix; and second, the matrix inversion lemma was found to be a functional method of computing the inverse of the signal covariance matrix in an online fashion.

Unfortunately, no spatial data was available from the NWRT that had the type of RFI observed in Chapter 2. However, for meteorologists ground clutter is, essentially, a type of interference present in every pulse at a particular range bin — this is precisely the situation that the temporal algorithms had issues handling. Since spatial weather radar data with ground clutter contamination was available, ADBF algorithms were employed on this data and were found to be quite effective at mitigating or completely removing the ground clutter signature while leaving the weather signal unaltered. These adaptive digital beamforming methods have promise in mitigating the impact of interference in a wide variety of applications. It should be noted that these algorithms require well calibrated arrays with good channel matching, or else the desired signal may be perceived as an interferer and consequently attenuated by the ADBF algorithm. Chapter 4 uses data from a prototype receiver for a radar in development at the OU Advanced Radar Research Center and explores both the impact of poor channel matching on ADBF performance and also how digital equalization can be used to improve channel matching.

Chapter 4

Digital Equalization

4.1 Background

Digital equalization, a process by which the behavior of a group of antennas can be brought into parity, was precipitated by efforts to improve the performance of adaptive arrays. While early adaptive algorithms could compensate for poorly matched channels, they were slow to converge, especially if eigenvalues were widely spread [52, 59, 46]. Newer adaptive beamforming methods, such as those discussed in Section 3.2, are able to find optimum weights much more quickly [49]; however, these algorithms require channels that are closely matched in terms of amplitude and phase, or else the attainable depth of the nulls produced by the algorithms can be significantly negatively impacted [60, 61, 62, 63, 64]. Digital equalization is a means to effect the requisite channel matching [60].

Digital equalization is typically implemented as a finite input response (FIR) filter through which the signal is fed and which alters the input signal's frequency response to match the frequency response of some reference signal. The most common approach to determine the coefficients for the equalization filter is to collect a calibration signal that has passed through the channel to be equalized (the auxiliary channel) and minimize the error between the auxiliary signal and the reference signal via a least squares approach. However, this approach can be relatively computationally costly — it requires a matrix inversion, which is an $O\left(n^3\right)$ operation. The calibration of the equalizer is typically done in a controlled environment using a known "reference signal" serving as the input to the receiver. There are two popular methods to perform the calibration of the equalizer: the first examines the receiver's response to a series of continuous wave signals that, put together, span the equalization bandwidth, while the second involves examining the receiver's response to broadband noise [65]. The equalizer calibration can be performed with other signals as well; the experiments here use an linear frequency modulated (LFM) signal as the calibration signal, which is a slight deviation from the first method, while signals of opportunity can and have been used as a calibration signal by other phased array radars [66, 67, 68, 69]. If the equalization filter is able to be calibrated in the field, then troublesome issues such as temperature-dependent frequency behavior degrading the performance of an electronically scanned array (ESA) could be ameliorated by an in-the-field calibration of a digital equalizer.

This chapter is organized as follows: Subsection 4.1.1 briefly explains how to calibrate an FIR filter suitable for equalization; Section 4.2 uses this technique on data collected from a prototype receiver at OU and examines the performance, both at increasing channel matching and in a simulated ADBF scenario; and Section 4.3 performs the same experiments on simulated data with simulated mismatch.

4.1.1 Calibrating the Equalization Filter

For an equalization filter with N coefficients, two $M \times N$ signal matrices $\mathbf{X}$ and $\mathbf{Y}$ are formed from the auxiliary and main signals respectively. Each row of these signal matrices is comprised of contiguous series of samples from the output of that channel. It has been calculated that nearly optimum equalizer performance can be attained if $M = 5N$ statistically independent rows are used in constructing these signal matrices [65]. To enable the equalizer to function across a range of frequencies, often a long LFM signal is used, with each individual row coming from a different location in that signal. Other calibration signals may be used, though; another common method is to compare the auxiliary and reference channels' responses to broadband Gaussian noise [65], and signals of opportunity have been used in determining an equalization filter for HF over-the-horizon radars [67].

Once the signal matrices are formed, an error matrix $\mathbf{E}$ is constructed by setting $\mathbf{E} = \mathbf{Y} - \mathbf{XW}$. In this setup, $\mathbf{W}$ is a set of column vectors $\mathbf{w_i}$, where each $\mathbf{w_i}$ is a set of reversed FIR coefficients such that $w_{i,n} = h_{N-n} \forall n \in [1,N]$; each column vector $\mathbf{w_i}$ represents a different potential time delay built into the implementation of the equalizing filter. The different columns of $\mathbf{Y}$ represent time delays in the reference channel, as it is possible to attain better performance with such a delay. The error matrix $\mathbf{E}$ is minimized in a least squares sense:

$$min||\mathbf{E}||^2 = min||\mathbf{Y} - \mathbf{XW}||^2 \tag{4.1}$$

To simply derive the solution, an extended signal matrix $\mathbf{Z} = [\mathbf{XY}]$ is constructed. This extended matrix $\mathbf{Z}$ undergoes QR decomposition; recall that after QR decomposition of a complex matrix, $\mathbf{Q}$ is a unitary matrix (so $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$ where $\{\cdot\}^H$ signifies the conjugate transpose operation) and $\mathbf{R}$ is an upper triangular matrix. Therefore, $\mathbf{Z}^H\mathbf{Z} = \mathbf{R}^H\mathbf{Q}^H\mathbf{QR} = \mathbf{R}^H\mathbf{R}$. The matrix $\mathbf{R}$ can then be partitioned such that

$$\mathbf{R} = \begin{bmatrix} \mathbf{U} & \mathbf{V} \\ \mathbf{0} & \mathbf{T} \end{bmatrix} \tag{4.2}$$

where $\mathbf{U}$ and $\mathbf{T}$ are upper triangular matrices. Since

$$\mathbf{Z}^H\mathbf{Z} = \begin{bmatrix} \mathbf{X}^H\mathbf{X} & \mathbf{X}^H\mathbf{Y} \\ \mathbf{Y}^H\mathbf{X} & \mathbf{Y}^H\mathbf{Y} \end{bmatrix} \tag{4.3}$$

it can be seen that $\mathbf{W} = \mathbf{U}^{-1}\mathbf{V}$. Additionally, $\mathbf{T}$ is the Cholesky triangle of the residual covariance; that is, $\mathbf{E}^H\mathbf{E} = \mathbf{T}^H\mathbf{T}$, meaning that the channel tracking error magnitudes can be computed from

$$|E_{nn}|^2 = \sum_{m=1}^{n} |T_{mn}|^2 \tag{4.4}$$

and the channel tracking error for a particular filter configuration associated with a particular time delay can be computed before solving the $\mathbf{UW} = \mathbf{V}$ equation, helping choose the equalization delay associated with the least error [65].

Figure 4.1: Conceptual drawing of the Horus radar [4].

To determine the effectiveness of the equalization, the channel pair cancellation ratio (CPCR) measure can be used [70]. For this paper, the following definition of CPCR is used:

$$CPCR = \frac{P_{out}}{P_r} \tag{4.5}$$

where P_{out} is the output power of the reference channel and P_r is the output power of the difference of the output signals at the reference and auxiliary channel. Under this definition, higher CPCR is better, and infinite CPCR indicates perfect equalization [65]. In the absence of nonlinearities and with the auxiliary and reference channel gains nearly equal to each other, the theoretical maximum CPCR is about half of the output SNR [70].

4.2 Horus Data

Horus is an all-digital phased array radar system under development at the University of Oklahoma's Advanced Radar Research Center. Seen in Figure 4.1, this highly flexible system will serve as a testbed for advanced digital radar designs and algorithms. Data from a prototype receiver from this system was recorded using the test setup shown in

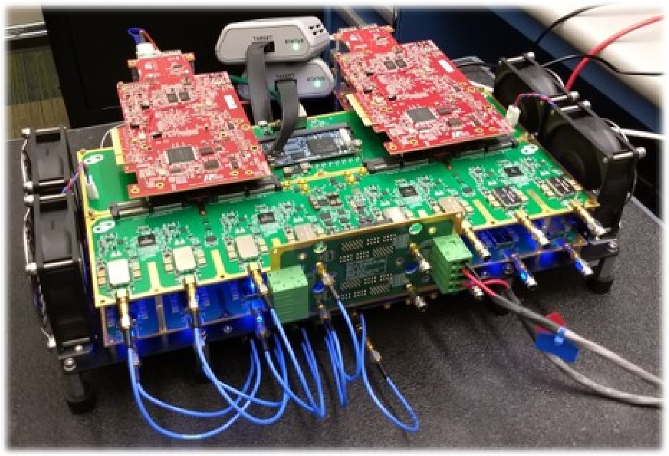


Figure 4.2: Test bench setup for the data collected from this experiment. The signal is passed through four channels of one of these “Octoblade” receivers.

Figure 4.2 [71, 72, 4]. The data suffers from channel mismatch, and was used to explore the effects of equalization on this data. A 500 microsecond chirp spanning 124 MHz was passed through the prototype Horus receiver and the signals at the outputs of the four channels were collected, after being mixed down to baseband, at a sampling frequency of 125 MHz. Three sets of data were collected: the results of a single chirp passed through the receiver, the averaged results of 64 chirps passed through the receiver, and a digital loopback (DLB) of the receiver input signal. There was noticeable channel mismatch throughout the bandwidth; the Fourier transforms of the signals at the receiver input and at each receiver channel’s output are shown in Figure 4.3. Digital equalization was performed using the method outlined in Section 4.1.1 using an equalizer set up in a feedforward configuration, as in Figure 4.4 [73]. The auxiliary signal used was the averaged IQ data from the receiver output. Because the input signal to the receiver was available and had good behavior across the spectrum, it was used as the reference signal for the equalization calculation; because that information — the signal at the input to the receiver — is likely not to always be available, the calculations were repeated using the channel 1 receiver output as the reference signal.

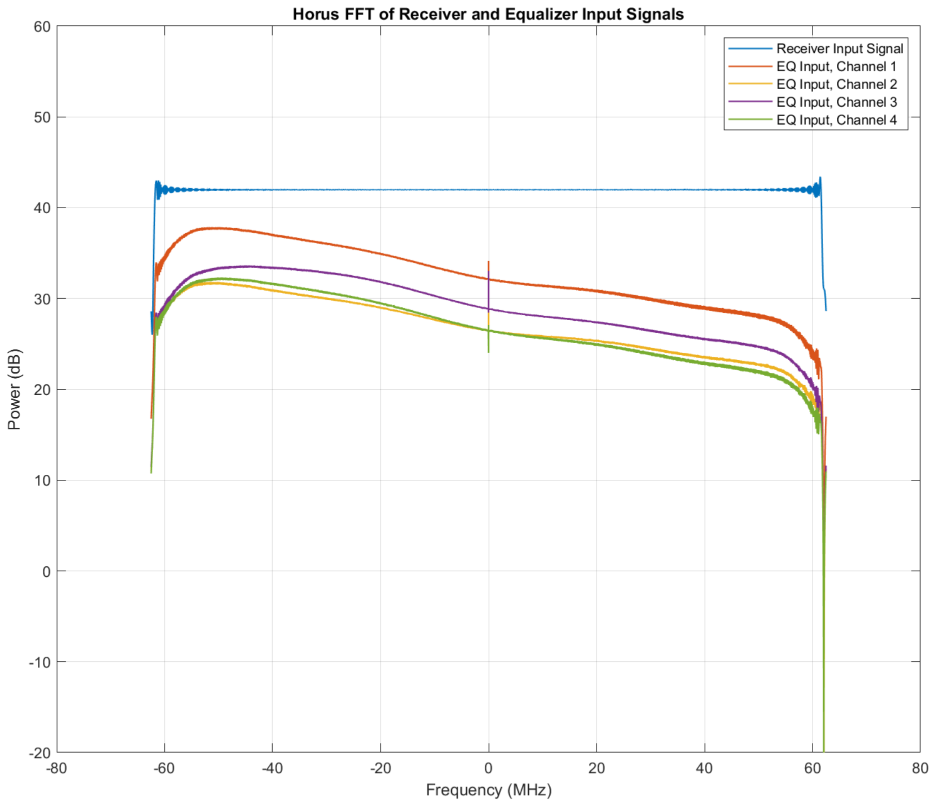


Figure 4.3: Fourier transforms of the input reference signal and of the signals collected at the output of each channel of the prototype Horus receiver.

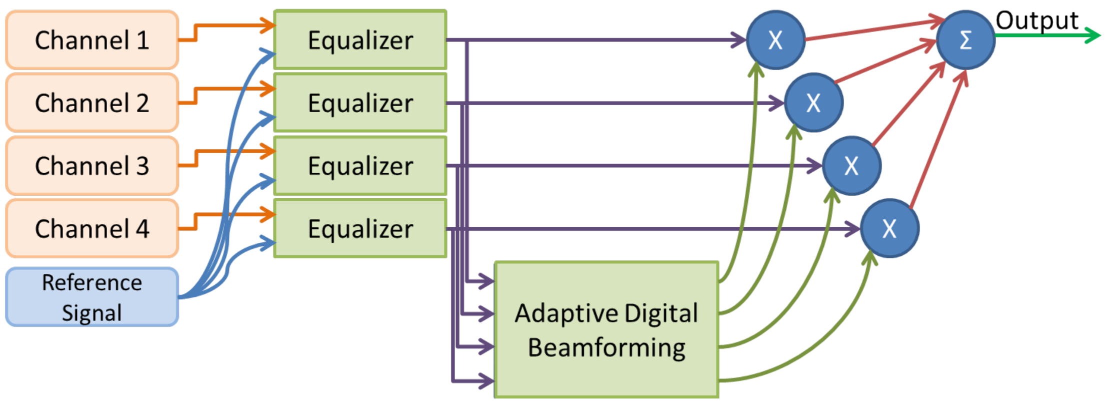
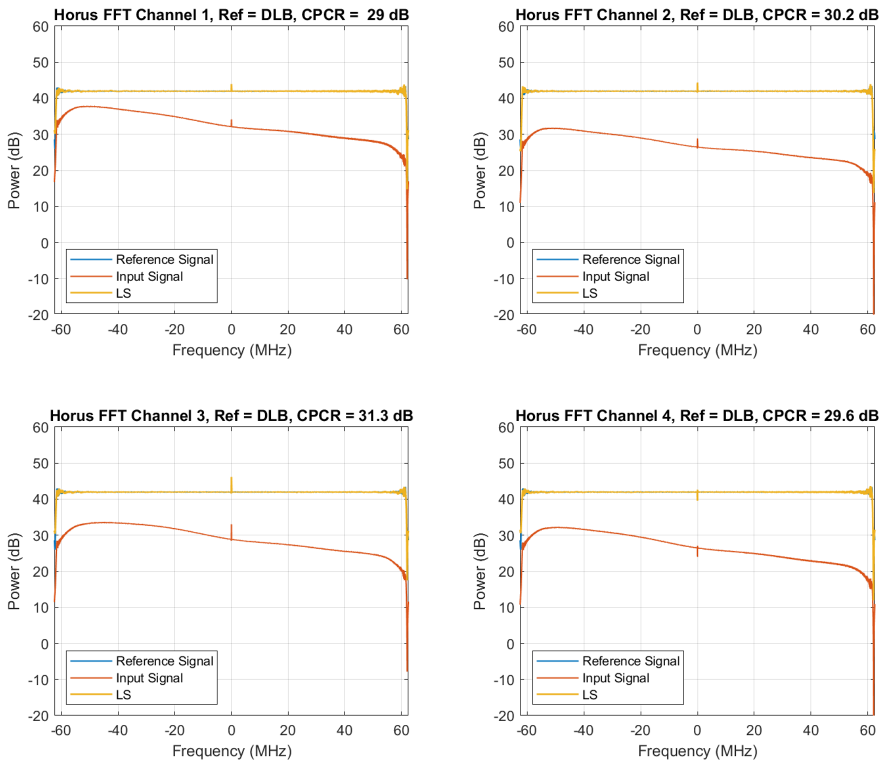


Figure 4.4: Block diagram of a receiver with an equalization stage before the ADBF stage. Because each channel is mismatched in its own particular way, each channel gets its own equalizer.

Table 4.1: Pre- and Post-equalization CPCR

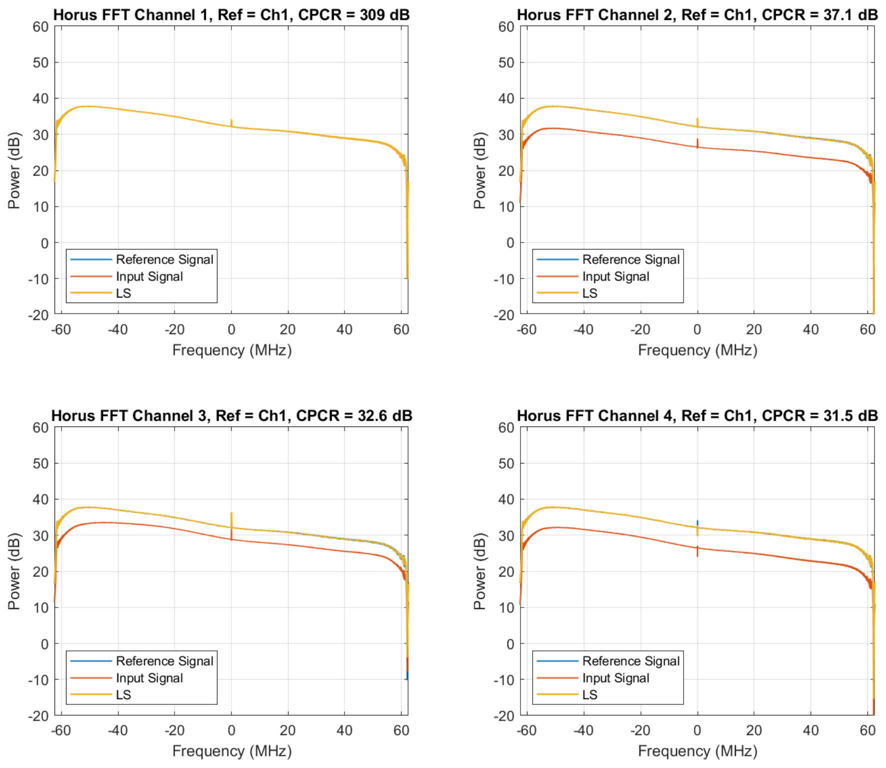
Channel	DLB pre-EQ CPCR (dB)	DLB Post-EQ CPCR (dB)
1	0.678	29
2	0.773	30.2
3	0.883	31.3
4	1.14	29.6
Channel	Ch.1 pre-EQ CPCR (dB)	Ch.1 Post-EQ CPCR (dB)
1	∞	∞
2	2.95	37.1
3	0.415	32.6
4	0.43	31.5

The Fourier transforms of the signals before and after equalization using a 128 tap equalization filter can be seen in Figure 4.5. With equalization using 128 filter coefficients, the signals matched the reference signal very well, attaining the CPCRs shown in Table 4.1. To explore the relation between CPCR and the number of equalization filter coefficients used, that number was varied between 1 and 128. The results are shown in Figure 4.6. For the data used here, the rate of improvement varied on the reference signal used. When the reference signal was the digitally-looped-back signal, there was always room for improvement in the CPCR, though the rate of CPCR improvement slowed as more taps were added. When the channel 1 output was used as the reference signal, the CPCR quickly (after between 10 and 20 taps) reached a value past which there was little variation; though the improvement to CPCR was faster and the ultimate CPCR higher, the amplitude of the output signal as a function of frequency was not as flat as if the DLB signal were used. Note that the CPCR numbers shown here are certainly a function of



(a) Ref. Signal: DLB

Figure 4.5: Fourier transforms of the reference signal (blue), the signals at each channel's output (red-orange), and those signals after equalization (yellow, essentially superimposed on blue) using the digitally-looped-back signal as the reference signal. Figure 4.5b is on the next page.



(b) Ref. Signal: Ch. 1

Figure 4.5: Fourier transforms of the reference signal (blue), the signals at each channel's output (red-orange), and those signals after equalization (yellow, essentially superimposed on blue) using the channel 1 output as the reference signal. Figure 4.5a is on the previous page.

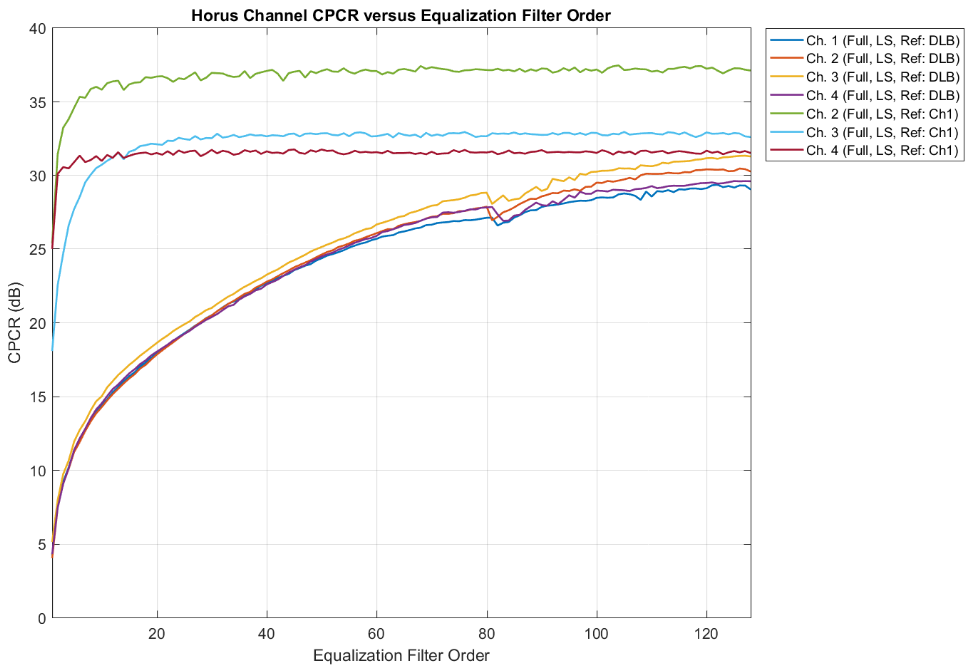
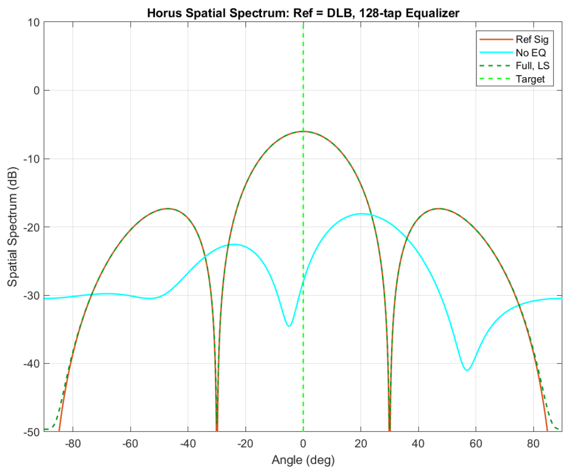


Figure 4.6: The CPCR varies with respect to the number of equalization filter coefficients used.

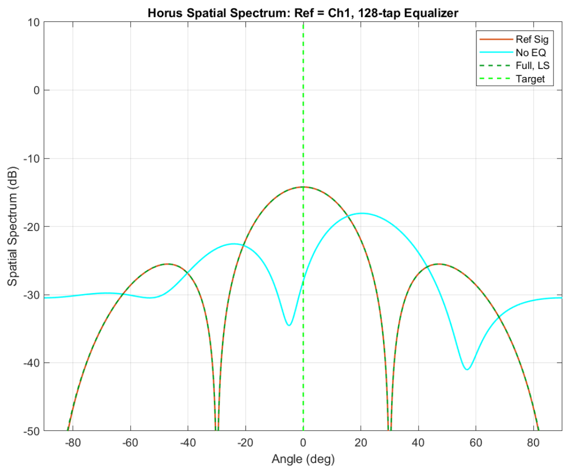
this particular data; data that starts off with worse channel matching will require more equalization filter coefficients to attain its maximum post-equalization CPCRs.

4.2.1 Spatial Spectrum

The effects of equalization in improving the channel matching can be readily seen in the apparent direction of the signal source. Recall that the Bartlett method is a simple direction finding algorithm which essentially performs a sweep across a search space, yielding the incoming power as a function of angle, as covered in Section 3.3.3. The Bartlett method of direction finding was applied to the input data using a search space of -90 to 90 degrees in azimuth [62]. The results are shown in Figure 4.7. The input chirp signal was fed into all of the receiver channels simultaneously, so the signal should appear to come from broadside, and this is the case for the reference signal in red-orange. When the digitally-looped-back signal was the reference signal, the spatial spectrum calculation was straightforward, since there were already four channels of the digitally-looped-back signal available; when the channel 1 output of the receiver was the reference signal, a four channel version of the reference signal was synthesized by duplicating the channel 1 output, yielding a signal that still seems to come in from broadside, but at a lower power. In contrast to the reference signal, the signal at each of the receiver channel outputs before equalization, shown in cyan, appears to come from approximately 20 degrees off of broadside, a significant deviation from what is expected. Equalization alleviates this issue, and the signal after equalization, shown in the green dashed line, appears to come from broadside as expected [74, 73]. For ADBF algorithms very susceptible to beam and array mismatch, such as minimum-variance distortionless response (MVDR) beamforming, an error of even a few degrees could have a significant effect on the beampattern; an error of 20 degrees could easily result in the algorithm actually nulling the signal a user is intending to receive [42].



(a) Spatial spectrum of the original signal before equalization (cyan) and after equalization (green) using the DLB as a reference.



(b) Spatial spectrum of the original signal before equalization (cyan) and after equalization (green) using channel 1 as a reference.

Figure 4.7: The apparent direction of the reference signal (red), the input signal after passing through the receiver but before equalization (cyan), and the signal after equalization (green dashed, superimposed on red). The direction the signal should appear to be coming from is shown in the vertical green dashed line.

4.2.2 ADBF Impacts

The effects of equalization on ADBF techniques were explored in more detail. For this experiment, an artificial input to the receiver was created by duplicating a base auxiliary signal (in this case, the non-averaged IQ data from the receiver output), applying a unique phase shift to each instance of this duplicated signal (to represent a unique source), and adding the two together to yield a composite signal. In this case, the “desired source” or “target source” was phase shifted so it appeared to come from -20 degrees azimuth and scaled up in amplitude by 20 dB. The other source, the “interfering source,” was phase shifted so it appeared to come from 30 degrees azimuth, and had its amplitude scaled up by 40 dB (so the transmitter signal-to-interference ratio was -20 dB). Additionally, to prevent the desired and interfering source from being too similar, the interfering source was comprised of the middle half of the original signal duplicated twice. The spectrogram of this input signal is shown in Figure 4.8. With this method of creating the composite signal, the inputs to each channel had the same channel matching characteristics as they did during the equalization calibration step, mimicking a real-world application of equalization where the coefficients for the equalization filter are determined and then used on a new and unknown signal in an environment where ADBF could enhance performance [74, 73]. For this experiment, MVDR beamforming was the ADBF algorithm used. The formula for MVDR beamforming is

$$\mathbf{w} = \mathbf{S}_{\mathbf{x}}^{-1}\mathbf{v}_s \left[\mathbf{v}_s^H \mathbf{S}_{\mathbf{x}}^{-1}\mathbf{v}_s\right]^{-1} \tag{4.6}$$

where $\mathbf{v}_s$ is the steering vector toward the direction the user wishes to look; $\mathbf{x}$ is the received data; $\mathbf{S}_{\mathbf{x}} = \mathbf{x}\mathbf{x}^H + \lambda\mathbf{I}$ is the signal covariance matrix, where the H superscript denotes the conjugate transpose operation and $\lambda = \mathrm{trace}\left(|\mathbf{S}_{\mathbf{x}}|\right)/N_{channel}$ denotes the amount of diagonal loading applied to the covariance matrix to enhance numerical stability before inversion; and $\mathbf{w}$ are the adaptive beamforming weights. The MVDR calculation’s complexity is $O\left(N_{channel}\right)^3$ [62].

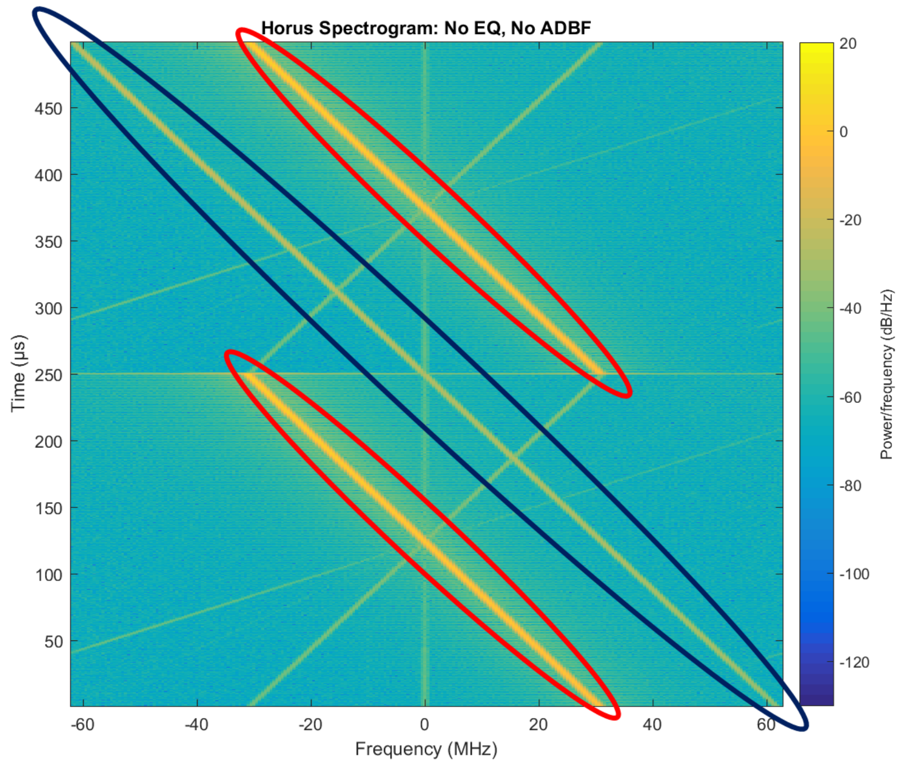
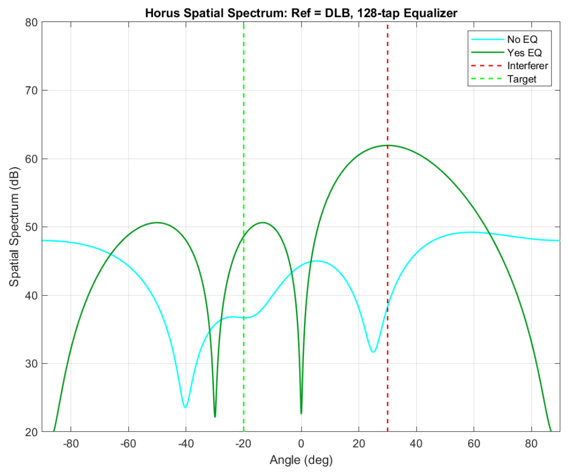


Figure 4.8: A spectrogram of the input signal to the equalizer in the ADBF scenario. The interfering signal is the disjoint signal, circled in red, and is clearly much stronger than the desired signal (which occupies the diagonal and is circled in blue).

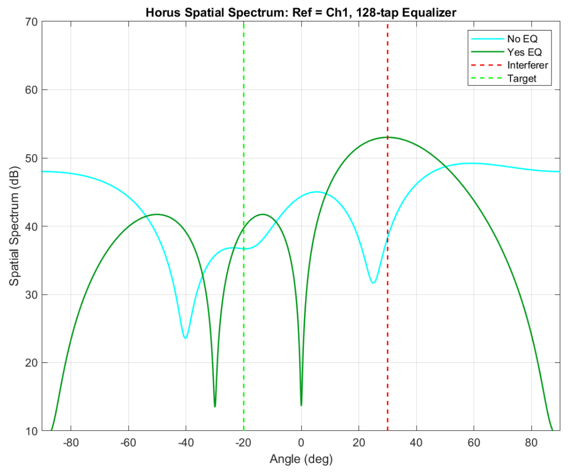
Table 4.2: Post-Processing SIR

Ref. Sig.	Equalization	ADBF	SIR (dB)	Color
N/A	No	No	-36.8	Cyan
N/A	No	Yes	-17.4	Purple (dashed)
DLB	Yes	Yes	9.4	Green (Figure 4.10a)
Ch.1	Yes	Yes	6.4	Green (Figure 4.10b)

The results are shown in Figure 4.10; note that the differences between using the digitally-looped-back signal and the channel 1 output as the reference signal are minimal, manifesting as a higher final SNR but ultimately similar results. Figures 4.9a and 4.9b show the spatial spectrum of the composite signal at the output of the receivers without equalization (cyan) and with equalization (green). The true locations of the sources are shown in the red and green vertical dashed lines. It's clear that, without equalization, the apparent direction of the sources does not resemble the true setup of the environment. Figures 4.10a and 4.10b show the beampatterns without any ADBF (cyan), with ADBF but without equalization (purple dashed), and with ADBF and with equalization (green). The beampattern without equalization does not have any notably deep nulls, meaning that the interferer's contribution to the received signal is not mitigated by much. When equalization is included as a step before ADBF, then the signal-to-interference ratio (SIR) is greatly improved; for the scenario here, the SIRs are shown in Table 4.2. Using equalization before ADBF improves the SIR in this particular setup by around 25 dB. The spectrograms of the post-ADBF signal without an equalization stage and with an equalization stage are shown in Figure 4.11. The color axis is shared between Figure 4.11 and Figure 4.8. As such, it is clear that, without equalization, performing adaptive digital beamforming pushes down the power of the interfering signal only minimally. However, when equalization is performed before the ADBF stage, the

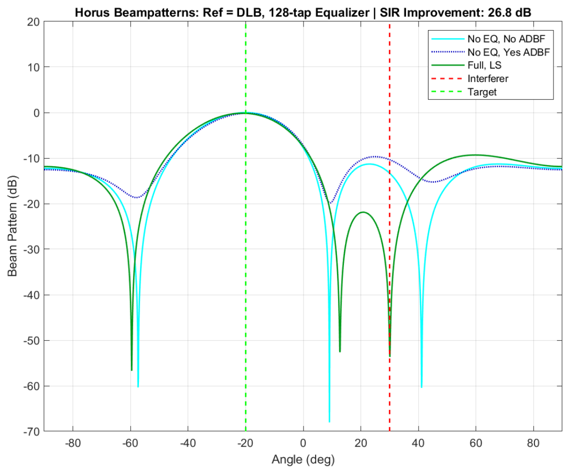


(a) The digitally looped back signal was used as the reference signal.

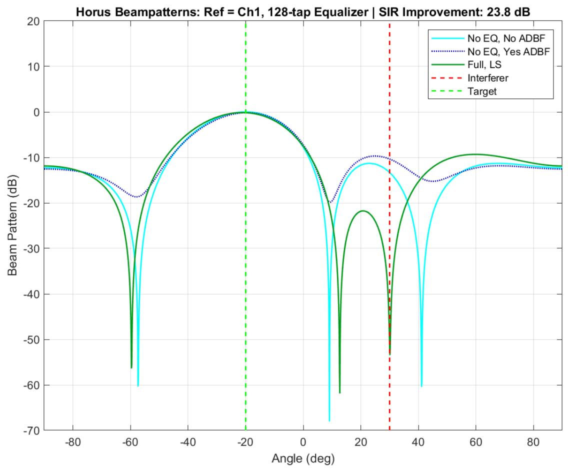


(b) The output of channel 1 was used as the reference signal.

Figure 4.9: The spatial spectrum without equalization (cyan) and with equalization (green) using the DLB signal (top) and the channel 1 signal (bottom) as the reference signal. In both figures, the locations of the desired signal (vertical green dashed) and the interfering signal (vertical red dashed) are also shown.

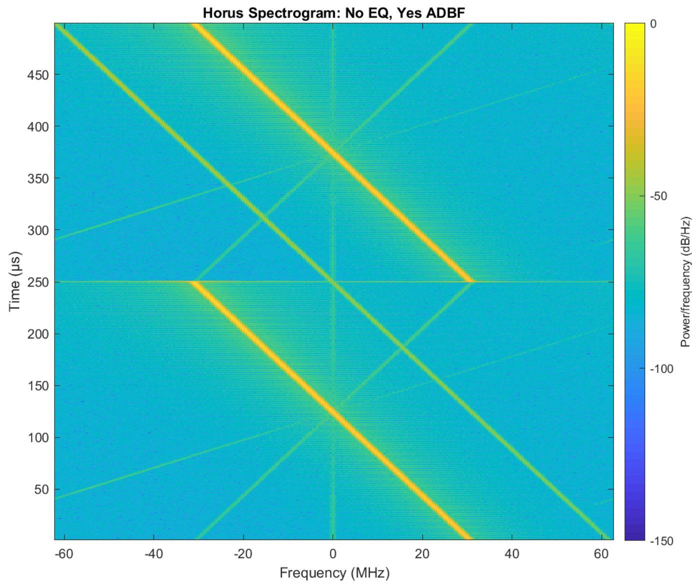


(a) The digitally looped back signal was used as the reference signal.

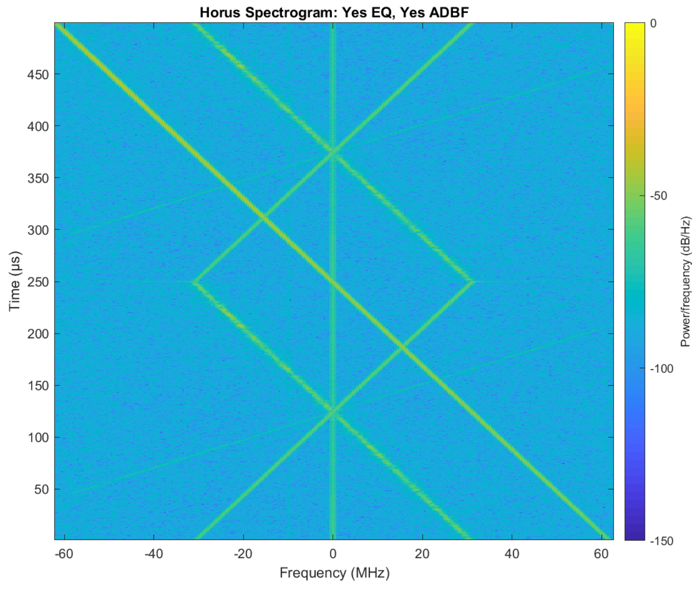


(b) The output of channel 1 was used as the reference signal.

Figure 4.10: The ADBF beampattern with and without equalization using the DLB signal (top) and the channel 1 signal (bottom) as the reference signal. The beampattern without equalization and without ADBF is in cyan, the beampattern without equalization and with ADBF is in purple, and the beampattern with equalization and with ADBF is in green. In both figures, the locations of the desired signal (vertical green dashed) and the interfering signal (vertical red dashed) are also shown.



(a) Post-ADBF, no equalization.



(b) Post-ADBF, with equalization

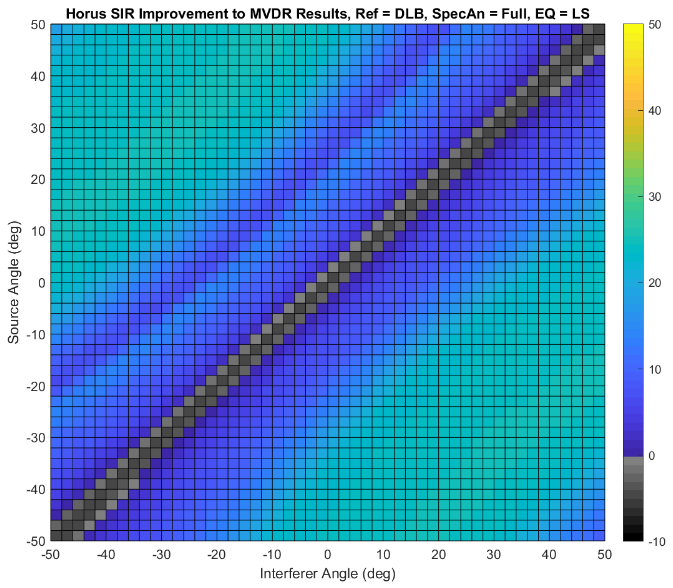
Figure 4.11: Spectrograms of the signal after ADBF without (top) and with (bottom) equalization first. The higher power of the interfering signal is clearly not as mitigated if no equalization is used.

interferer signal power is pushed down to below that of the desired signal, causing the notable signal-to-interference ratio improvement.

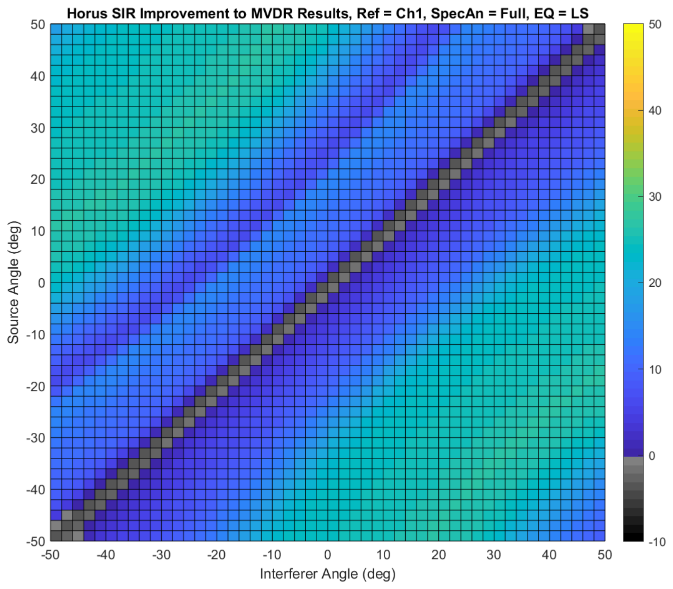
The effects of inserting an equalization stage before performing ADBF were explored for a variety of equalization filter orders and for a variety of desired and interfering signal powers and locations. Results of this inquiry are shown in Figures 4.12 and 4.13. Both sets of figures show the improvement to the SIR caused by including an equalization step before the ADBF step. Figure 4.12 shows the improvement to SIR after equalization as a function of the interferer angle and the source angle. The ADBF algorithm used here is MVDR, which performs poorly in situations where the desired signal and interfering signal are located angularly close to each other; when the sources are sufficiently far apart, equalization always improves the SIR after ADBF. Figure 4.13 shows the improvement to SIR after equalization as a function of the number of filter coefficients used in the equalization step and as a function of the relative powers of the desired and interfering sources. Increasing the filter order does improve performance, but the more noticeable factor is the power of the interferer relative to the power of the source. Note that using equalization before ADBF virtually always improves the SIR [74, 73].

4.2.3 Comparison to Machine Learning Algorithm Performance

The channel matching problem is well addressed by the FIR equalization filter; notably, SIR performance in ADBF scenarios is always improved after employing this method. For comparison, a variety of machine learning regression algorithms were tested on the equalization problem and their effects on CPCR and SINR improvement were compared to the least-squares FIR filter design method. Eight methods were tested: linear regression, linear regression with the ridge penalty (hereafter called “ridge regression”), linear regression with the lasso penalty (hereafter called “lasso regression”), linear regression with the elastic net penalty (hereafter called “elastic net regression”), decision

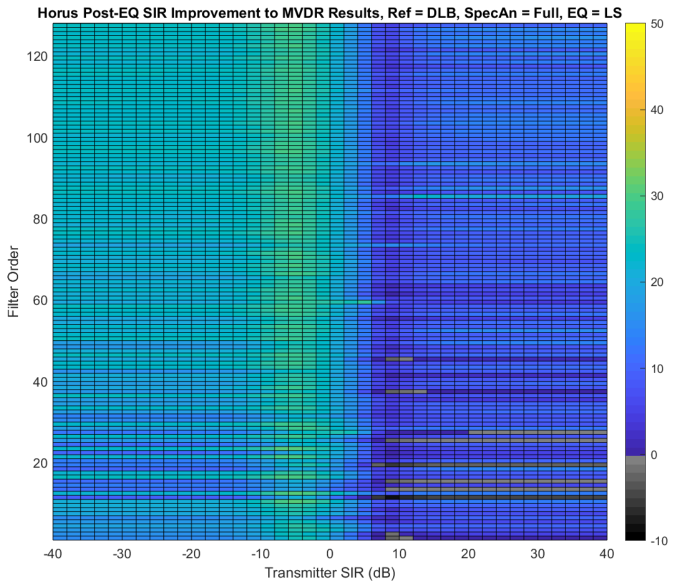


(a) SIR improvements using the DLB signal as a reference.

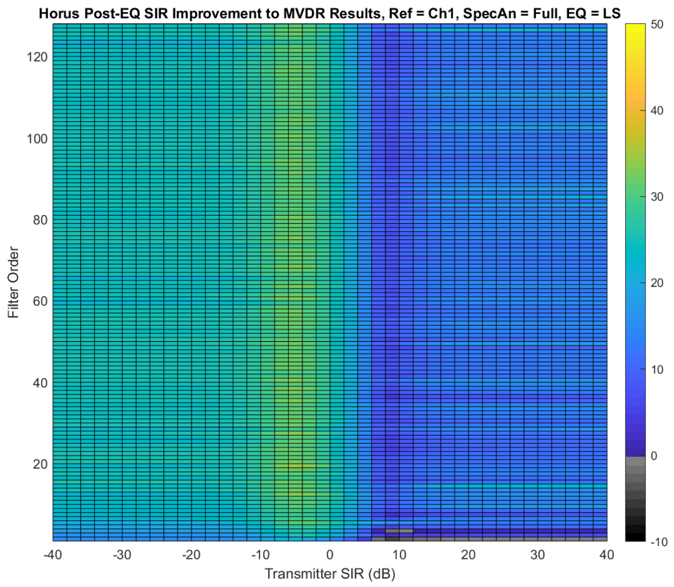


(b) SIR improvements using the channel 1 signal as a reference.

Figure 4.12: When equalization is used before ADBF, the SIR is always improved except when the interferer and desired source are essentially collocated in angle. This is a known weakness of MVDR. The results hold true whether the reference signal is the digitally looped back signal or the channel 1 output signal.



(a) SIR improvements using the DLB signal as a reference.



(b) SIR improvements using the channel 1 signal as a reference.

Figure 4.13: If the interferer transmitter power is of similar or greater power than the desired source transmitter power, using equalization before ADBF always results in an improvement to SIR.

tree regression, random forest regression, gradient boosted tree regression, and multi-layer perceptron regression; short summaries of each of the methods can be found in Appendix B. Scikit-learn's implementations of each of the algorithms were used [23], Numpy [75] and Scipy [76] were used for the processing, and Matplotlib was used to generate the plots [77].

The original least-squares FIR design method was calibrated using the averaged IQ data from the Horus receiver as the auxiliary signal and the channel 1 output as the reference signal; the FIR filter was of order 64. To train the machine learning algorithms, training, validation, and truth data sets were constructed from the same signals. Separate estimators were trained for the real and imaginary components of the signal for each channel. The features extracted for these data sets were:

- $\mathbb{R}\{X_{n-k}\}\forall k \in [0:64)$
- $\mathbb{I}\{X_{n-k}\}\forall k \in [0:64)$

where X_n is the complex signal at the current time, (and so X_{n-1} is the IQ data from the time sample before the current time sample); $\mathbb{R}$ and $\mathbb{I}$ are the real and imaginary components of the following; and 64 was chosen as the number of time delays because it was the same as the order of the FIR filter. These features were normalized so that they fit into a normal distribution with a mean of zero and variance of one. Cross-validation was performed to choose the best-performing parameters, which are shown in Table 4.3. Note that $\alpha = 0$ was found to be the optimal parameter for the linear models (ridge, lasso, and elastic net regression). This indicates that, since the error was lowest if all of the input features were used in the prediction, all of the input features have predictive value; mathematically, this means that they all simplify to be equivalent to linear regression. Additionally, two more features were trialed,

- $|X_{n-k}|\forall k \in [0,64)$
- $\angle X_{n-k}\forall k \in [0,64)$

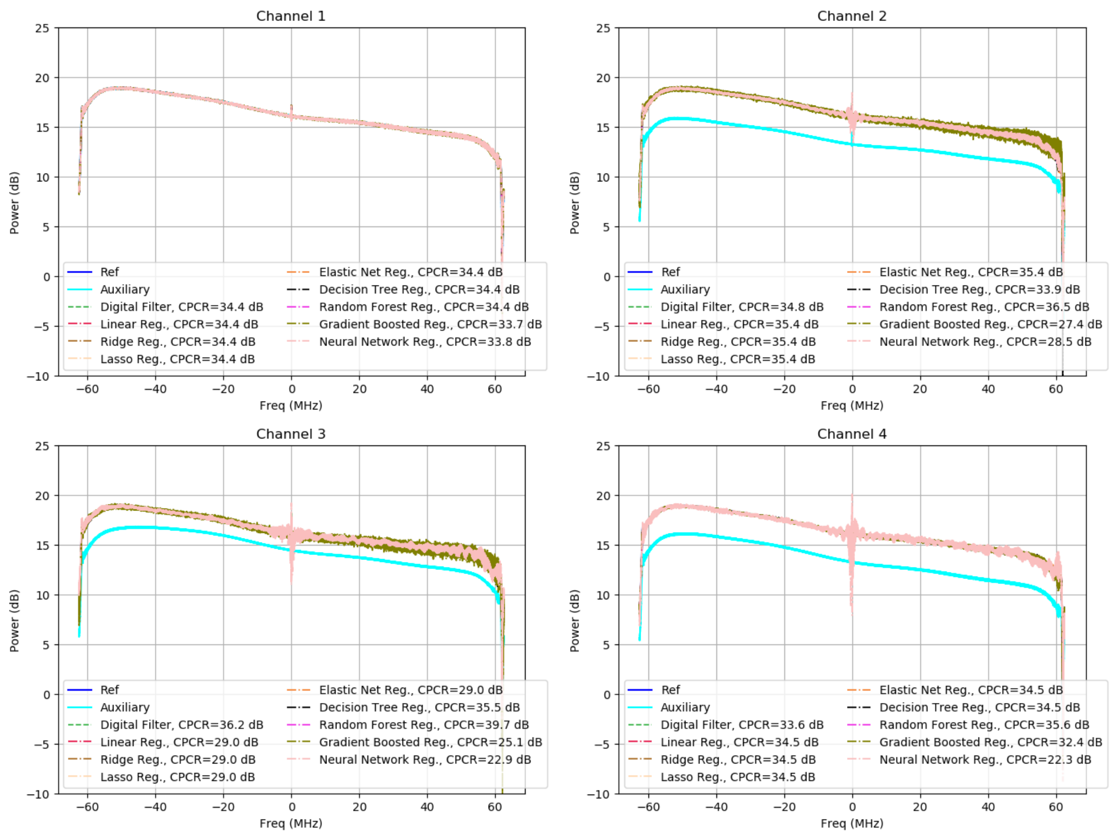


Figure 4.14: The FFTs of the post-equalization signals show consistently improved CPCRs.

where $|\cdot|$ is the magnitude of the enclosed term and $\angle$ is the phase angle of the following term. Performance was worse with these features so they were excluded from further experiments.

The FFTs of the results from testing the calibration on the non-averaged IQ data from the Horus receiver are shown in Figure 4.14, and the resulting CPCRs are summarized in Table 4.4. For the following figures, the reference signal is the solid blue line, the signal before equalization calibration is the solid cyan line, the digital filter equalization method (i.e. the non-machine-learning method) is in the green dashed line, and the algorithms are in a variety of other colors of dot-dashed lines. Most of the algorithms visually do a good job at matching the reference frequency response, though there is

Table 4.3: Optimal Regression Algorithm Parameters

Algorithm	Parameter Name	Parameter Value
Ridge	α	0
Lasso	α	0
Elastic Net	α	0
	L_1 ratio:	0
Decision Tree	Criterion	Mean-squared error (MSE)
Random Forest	Criterion	MSE
	Number of Estimators	50
Gradient Boosted Tree	Criterion	Friedman MSE
	α	0.9
	Learning rate	0.1
	Loss	Least squares
	Maximum depth	3
	Number of Estimators	100
Multilayer Perceptron	Activation Function	Rectified Linear Unit
	α	1
	Hidden Layer Size	100
	Learning Rate	Constant

some noisiness to the fit, especially for the gradient boosted tree; this is reflected in the CPCR numbers.

The spatial spectra of the calibration data are shown in Figure 4.15. Again, the reference signal is in blue, the signal before calibration is in cyan, the non-machine-learning method is in the green dashed line, and the varied machine learning methods are in dot-dashed lines. The beampattern nulls are not as deep as would be desired for some of the algorithms, but all do a passable job at restoring the apparent directionality of the calibration signal.

The same ADBF signal generation technique as in Section 4.2.2 was used to create the same ADBF scene, and the quality of these machine learning equalization solutions were tested. The spatial spectrum of the ADBF scene is shown in Figure 4.16. The signal without equalization is in cyan, the digital filter method is in the green dashed line, and the other methods are in various colors of dotted lines. The locations of the desired and interfering signals are shown in vertical green and red dashed lines respectively. The tree-based methods do get the correct directionality, but do not match the

Table 4.4: CPCRs (in dB) of Equalization Algorithms

Algorithm	Ch. 1	Ch. 2	Ch. 3	Ch. 4
Digital Filter	34.4	34.8	36.2	33.6
Linear Regression	34.4	35.4	29.0	34.5
Ridge	34.4	35.4	29.0	34.5
Lasso	34.4	35.4	29.0	34.5
Elastic Net	34.4	35.4	29.0	34.5
Decision Tree	34.4	33.9	35.5	34.5
Random Forest	34.4	36.5	39.7	35.6
Gradient Boosted	33.7	27.4	25.1	32.4
Neural Network	33.8	38.5	22.9	22.3

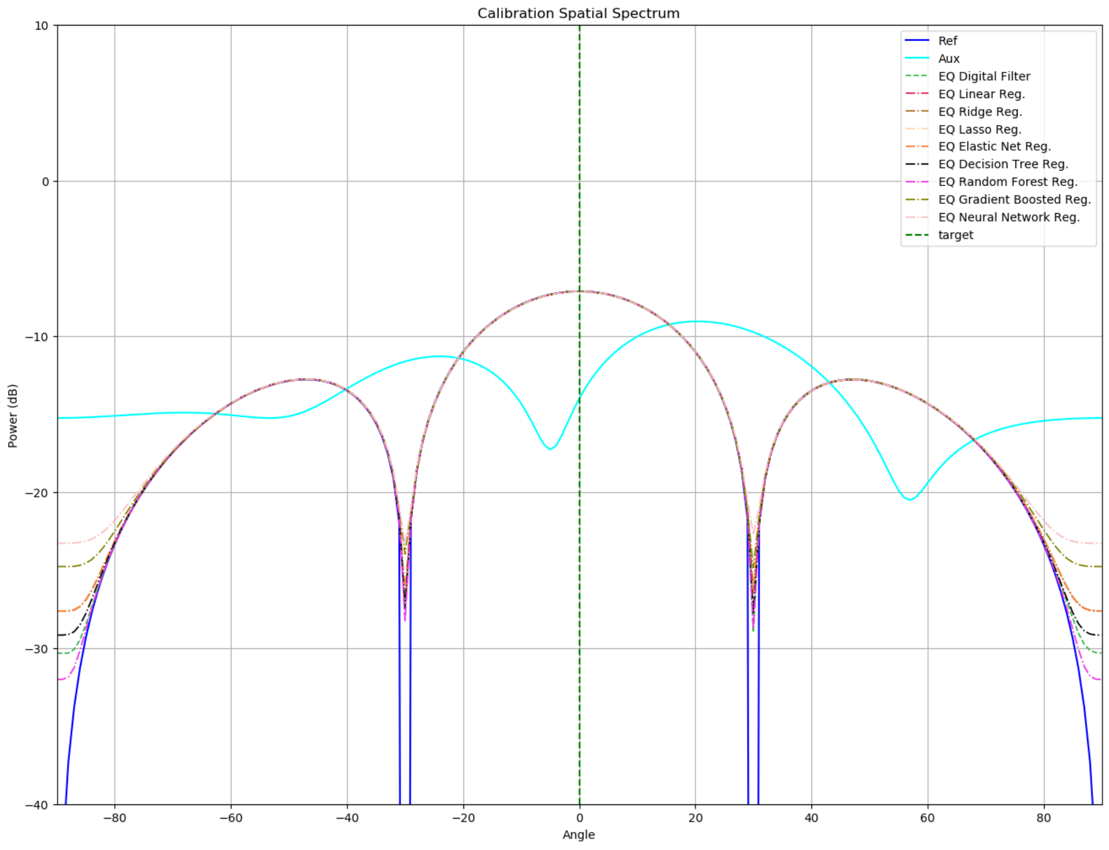


Figure 4.15: The spatial spectra of the post-equalization signals show much improved performance, though some algorithms get better results than others.

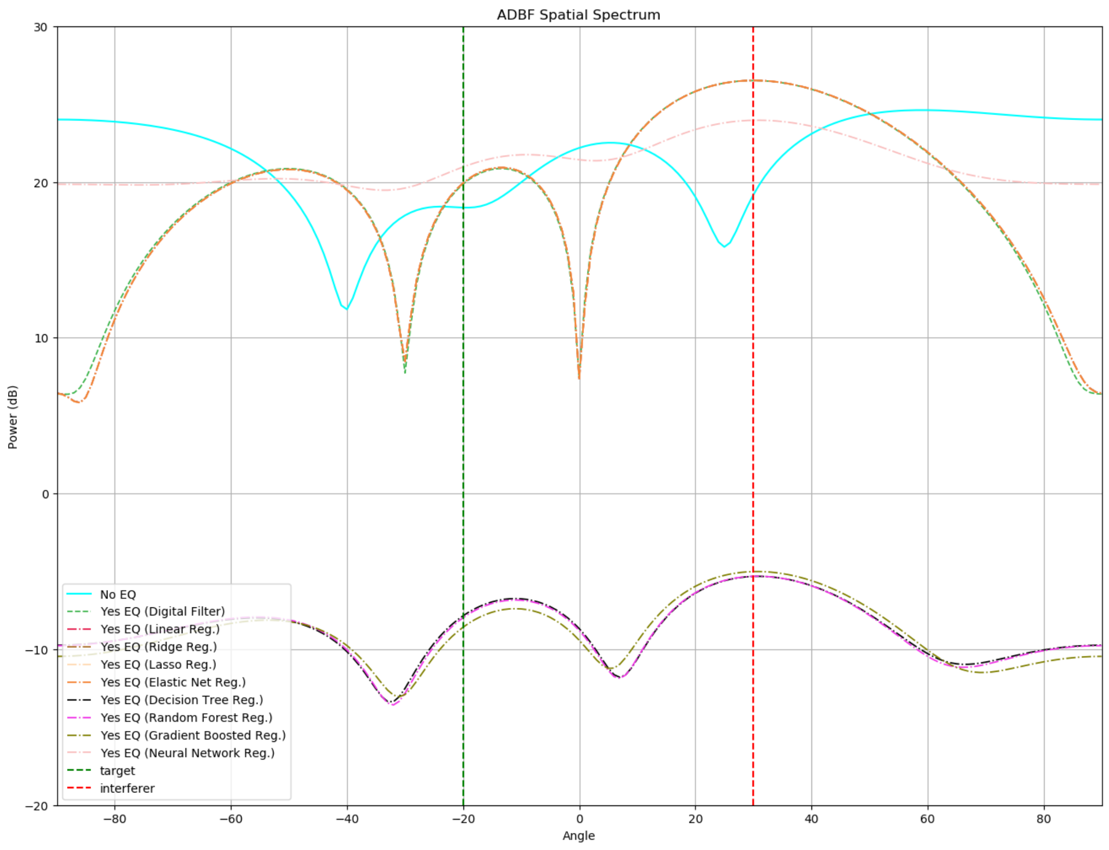
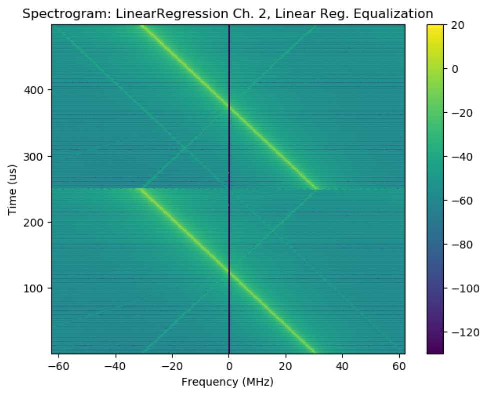


Figure 4.16: The spatial spectra of the post-equalization signals show much improved performance for the linear models, but the nonlinear models (the tree-based methods and neural networks) do not perform well at amplitude matching.

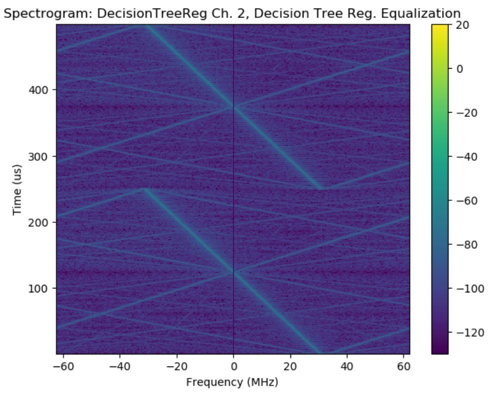
amplitude correctly. Similarly, the neural network implementation does show peaks in its spatial spectrum in the expected directions but has very little amplitude variation. Examining the spectrograms of each type of signal reveals more; Figure 4.17 shows the spectrograms of the equalizer output signal (only channel 2 is shown to save space; all of the channel outputs looked visually similar).

The linear models in Figure 4.17a look as would be expected — the interferer is stronger than the target signal. The tree-based models in Figure 4.17b visually look correct — the correct features are in the correct places — but the amplitude is wrong; these machine learning models have attenuated the signal. The neural network model in Figure 4.17c introduces additional signal components not present in the original signal. The deviations present in both the tree-based models and the neural network are likely because the machine learning models were trained solely on the broadside case and likely overfit to that specific situation. A future experiment would add training data for the neural net that included situations where the calibration signal appeared to come from a variety of directions to determine if neural network performance improved; past work on the use of neural nets to find the direction of arrival for signals indicates that improved performance under this more thorough training scheme is likely [78, 79, 80].

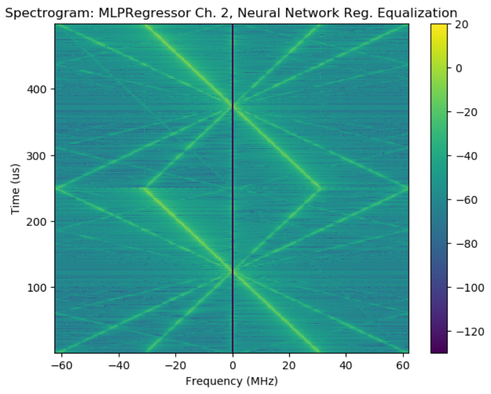
The MVDR ADBF algorithm was applied to the signal after equalization using each of these methods. The results are shown in Figure 4.18 and the SINR improvements from equalization using each method are shown in Table 4.5. The beampattern without ADBF is shown in blue, the beampattern with ADBF but without equalization is shown in the cyan dotted line, the beampattern produced by using the non-machine-learning method is shown in the green dashed line, and the other methods are in various colors of dotted lines. The locations of the desired and interfering signals are shown in the vertical green and red dashed lines respectively. Figure 4.19 has the beampatterns split up by groups of algorithms and is perhaps easier to parse.



(a) Equalization output spectrogram: linear models



(b) Equalization output spectrogram: tree-based models



(c) Equalization output spectrogram: neural network

Figure 4.17: Spectrograms of the equalization output for the ADBF portion of the ML experiment. The linear models (the digital filter design method and linear, ridge, lasso, and elastic net regression) all looked virtually identical, so only linear regression is shown in Figure 4.17a. Similarly, the tree-based methods (decision tree, random forest, and gradient boosted regression) all looked virtually identical, so only decision tree regression is shown in Figure 4.17b.

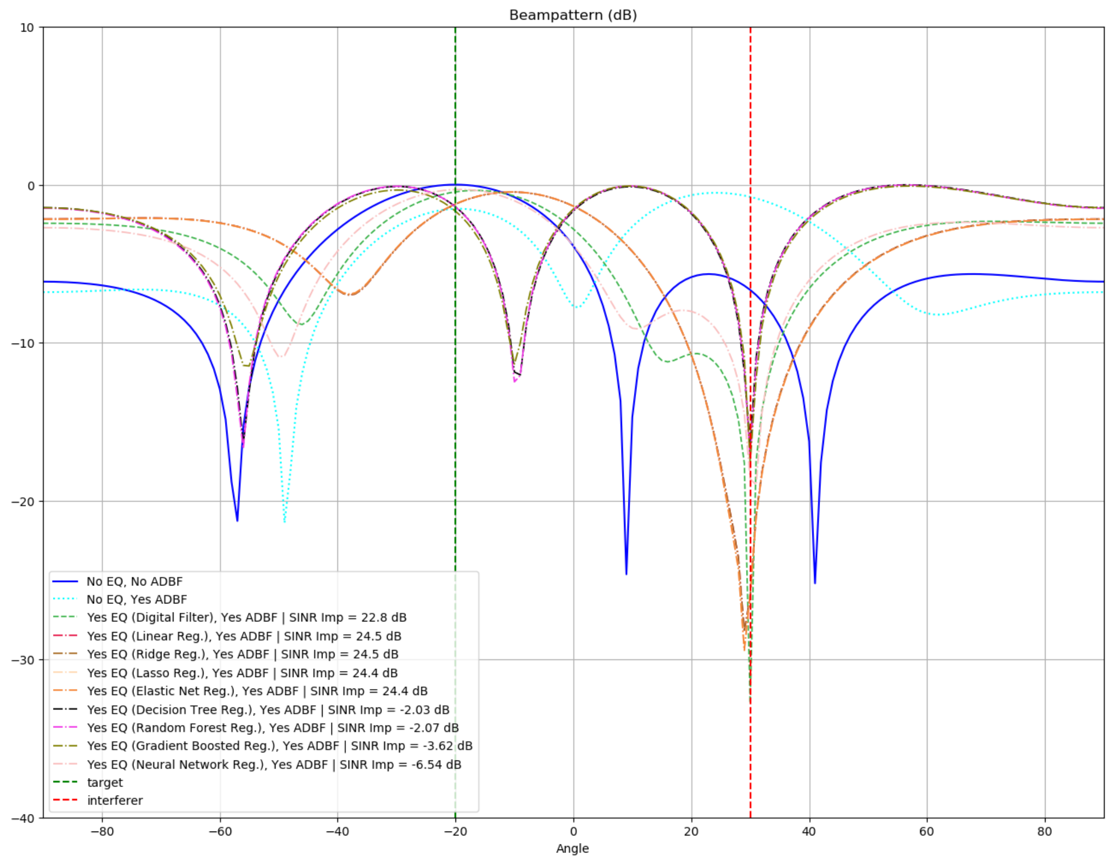
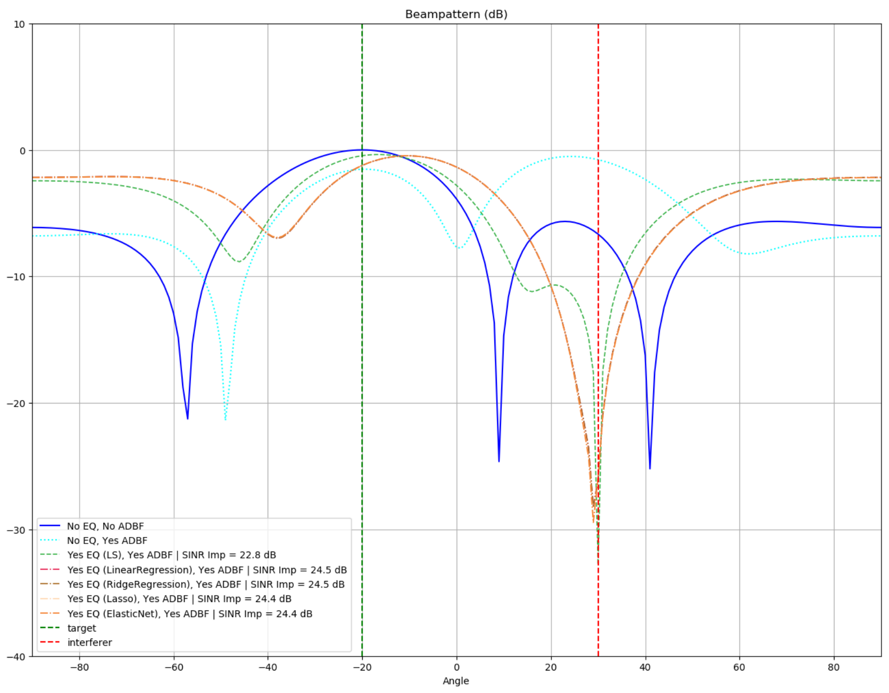
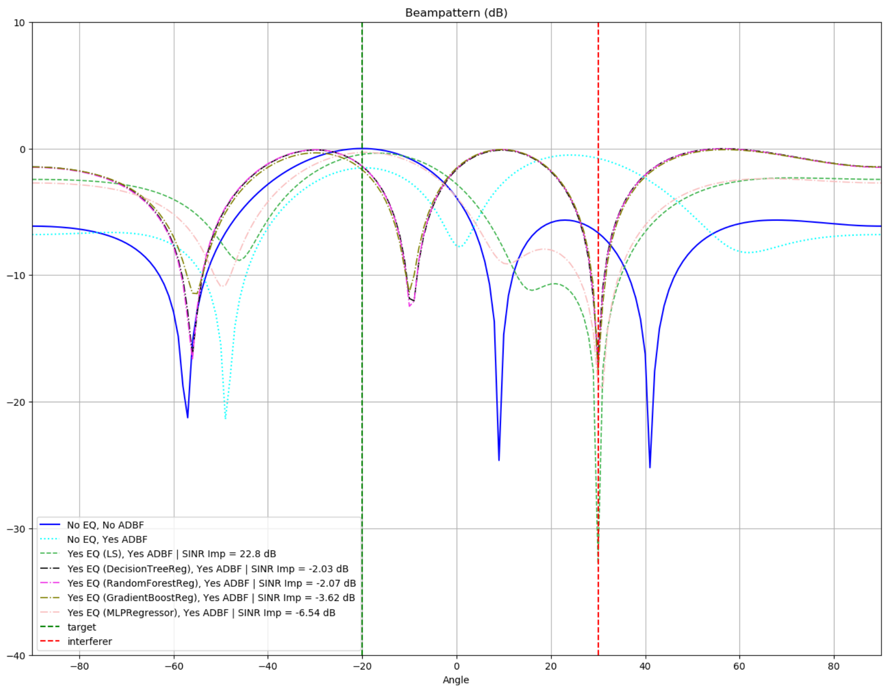


Figure 4.18: The beampatterns determined by MVDR beamforming on an equalized signal.



(a) Beampatterns of linear regression algorithms.



(b) Beampatterns of tree-based and neural network algorithms.

Figure 4.19: The beampatterns determined by MVDR beamforming on an equalized signal, split into different figures for easier viewing.

The linear regression-based algorithms, seen in Figure 4.19a, all improve the SINR after ADBF; linear-model algorithms all do well at introducing a null in the correct direction and improving the SINR. The tree-based algorithms and neural networks are shown in Figure 4.19b. The tree-based algorithms all do a good job at introducing a null in the correct direction, but the associated attenuation caused by the equalizers results in an overall poorer SINR. The neural network's introduced nonlinearities also cause issues, resulting in an overall poorer SINR.

Overall, newer and more complicated machine learning methods, including decision trees, random forests, and neural networks, could probably perform as well as or better than the more classical machine learning methods such as linear least squares regression or ridge regression at digital equalization *if* the training datasets were carefully and comprehensibly crafted; it's very possible, however, that such a training dataset would not be easily obtainable in a real world equalization scenario. However, even if such training datasets were able to be obtained, the expensive computational cost of training

Table 4.5: Improvement to SINR (in dB) after ADBF for each Equalization Algorithms

Algorithm	SINR Improvement (dB)
Least Squares	22.8
Linear Regression	24.5
Ridge	24.5
Lasso	24.4
Elastic Net	24.4
Decision Tree	-2.03
Random Forest	-2.07
Gradient Boosted Tree	-3.62
Multilayer Perceptron	-6.54

these more complicated estimators — potentially as high as $O(n^3 \log n)$ for trees and $O(n^4)$ for neural nets [23] — and the more complex implementation of these estimators methods makes the more classical linear-model-based methods, with their comparable performance, $O(n^3)$ computational cost for training, and simple implementation, better suited to equalization applications.

4.3 Simulated Data

The above experiment tested the performance of equalization for the data collected from the Horus prototype receiver. More general conclusions were desired as to the effects of equalization and how it could improve performance. To that end, data was simulated and manually mismatched. In order to simulate the channel mismatch, the data was passed through an FIR filter with randomly generated complex coefficients; the length of the filter was varied as desired.

First, some relation between the amount of mismatch and the number of coefficients needed to counteract that mismatch was examined. The simulated setup was again a four-channel system. The number of coefficients of the corrupting FIR filter was varied from 1 to 32, with each channel receiving its own coefficients. The number of coefficients in the equalization filter was varied from 1 to 256. At each point, the CPCR was calculated and recorded after equalization using the digitally looped back signal as the reference signal. This was repeated 50 times, and the mean of the CPCRs at each point was calculated. The results can be seen in Figure 4.20. It can be seen that, as more corruption coefficients are added, the number of equalization coefficients needed to counter the corruption increases exponentially. Additionally, CPCRs above approximately 30 dB are not attainable if the mismatch is too severe. A slice of this chart can be seen in Figure 4.21, where the number of corruption filter coefficients was held constant at 4, and the experiment was executed 50 times and then averaged.

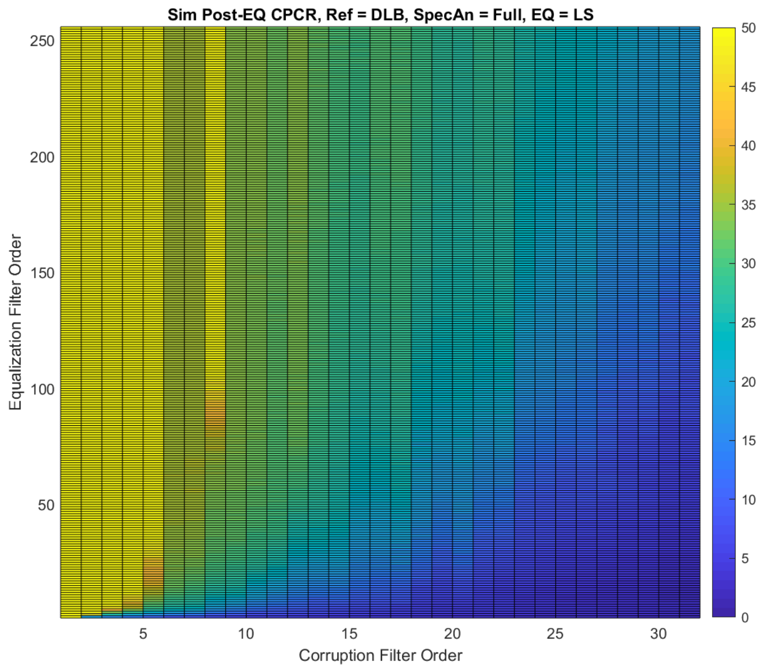
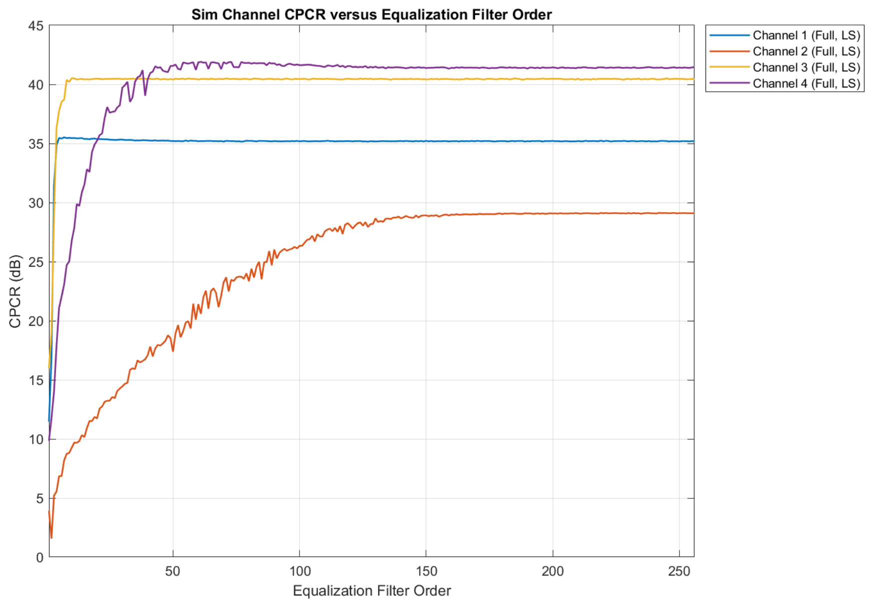
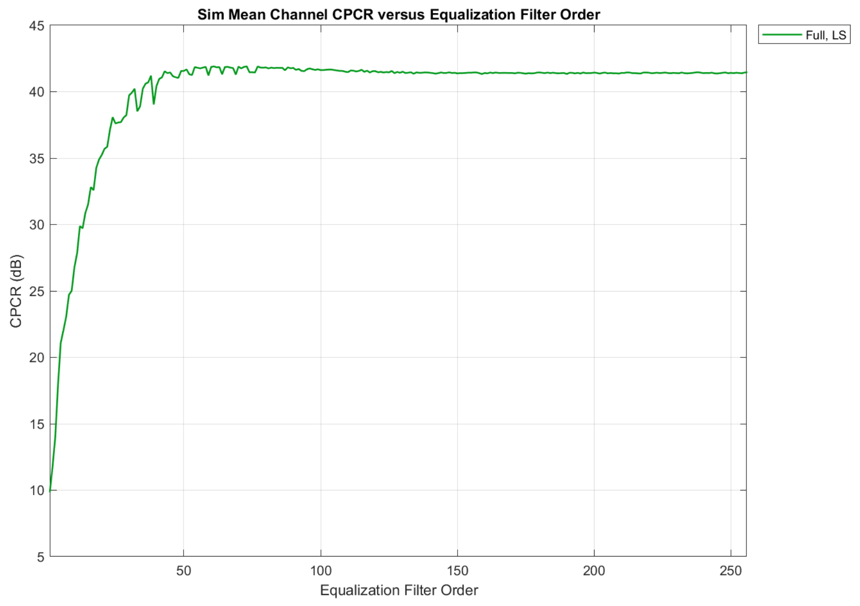


Figure 4.20: The CPCR varies with respect to the number of filter coefficients used in both the corruption and the equalization stages.



(a) CPCR by channel.



(b) Mean CPCR.

Figure 4.21: For simulated data, the CPCR as a function of equalization filter order when the corrupting filter uses 4 taps.

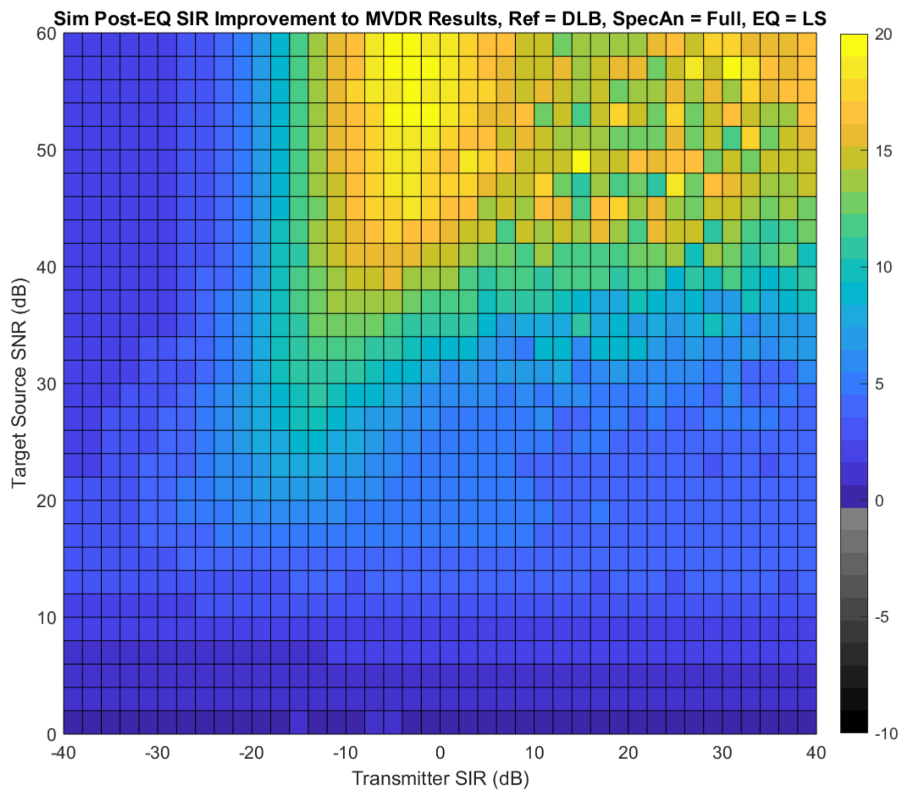


Figure 4.22: Post-ADBF improvement to SIR if equalization is used, as a function of original source SNR and the SIR.

Next, the same adaptive digital beamforming experiments as above were repeated, with one addition — the original source's SNR can be controlled, so it, too, was varied. In all figures, the filter used to induce channel mismatch used 4 coefficients, and the experiment was repeated 50 times and then averaged.

Figure 4.22 shows the improvement to SIR if equalization is used as a function of the source SNR and the transmitter SIR. In general, if the SIR is too low or the source power is not far above the noise power, then improvements are only minimal; as SNR increases, the post-ADBF SIR improvement becomes more noticeable; additionally, having a stronger signal than interferer also helps the post-ADBF SIR.

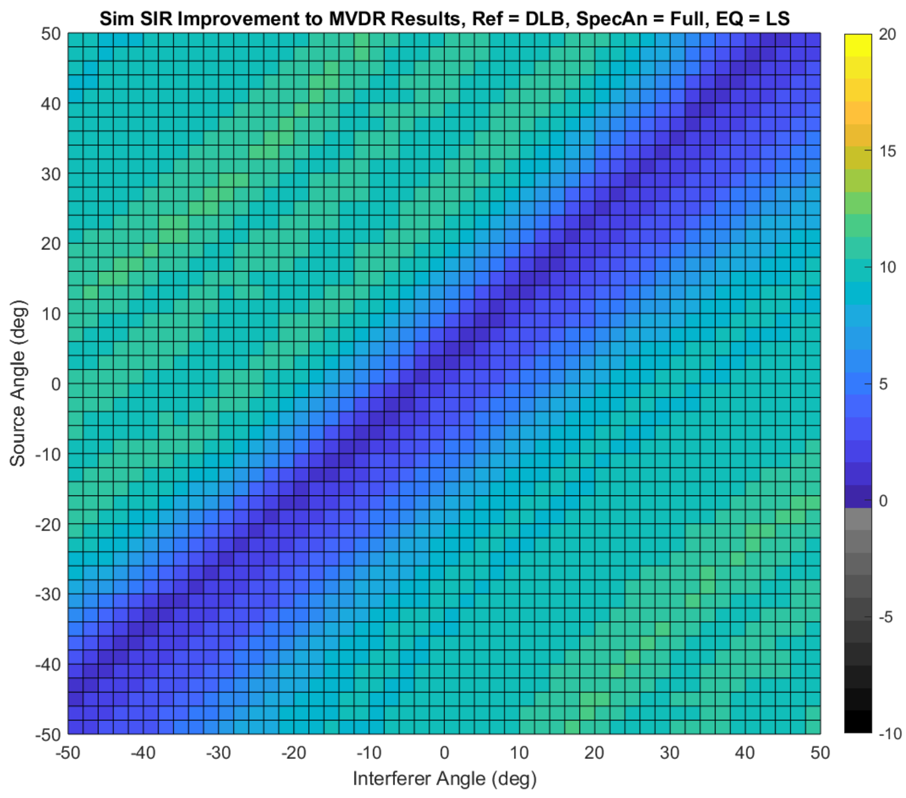


Figure 4.23: Post-ADBF improvement to SIR if equalization is used, as a function of interferer angle and source angle.

Figure 4.23 shows the improvement to SIR if equalization is used as a function of the source and interferer angles. As in the case with the Horus data, the characteristics of MVDR are most evident: having the interferer and source too closely collocated in angle hurts performance. Despite that, employing equalization always improves post-ADBF SIR.

Figure 4.24 shows the improvement to SIR if equalization is used as a function of the transmitter SIR and the order of the equalization filter. The features are similar to the same experiment performed on Horus data: giving taps to the equalization filter improves matching, and the consequent improvement to post-ADBF SIR is more noticeable when the interferer is stronger than the signal.

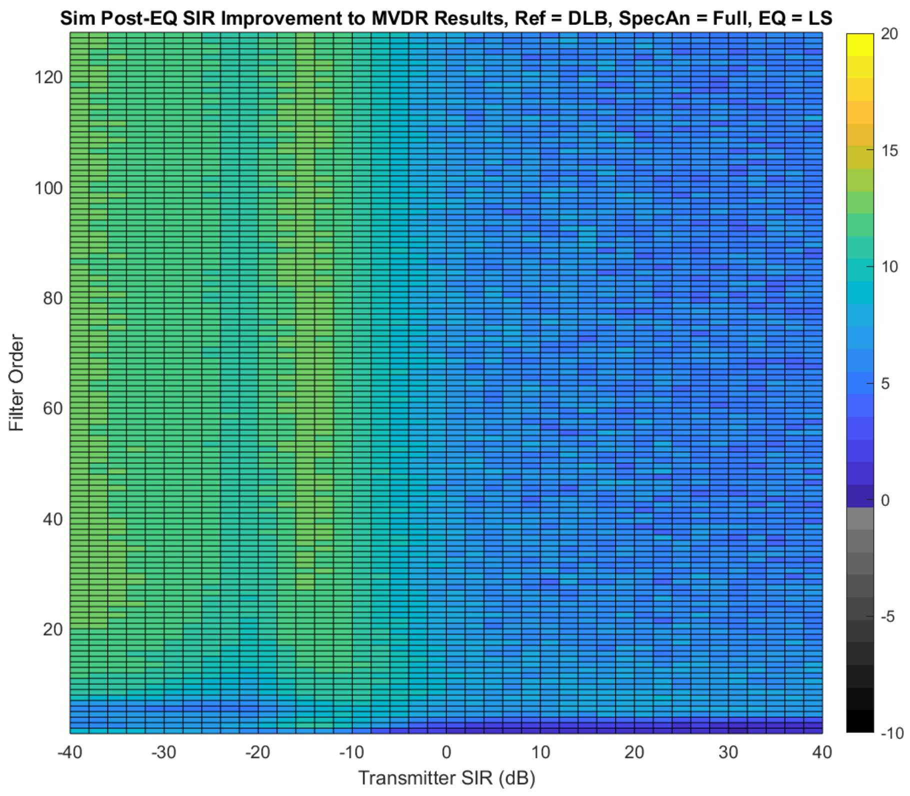


Figure 4.24: Post-ADBF improvement to SIR if equalization is used, as a function of SIR and equalization filter order.

4.4 Conclusions

Data recorded using a prototype receiver and suffering from channel mismatch was used to investigate the efficacy of digital equalization at enhancing channel matching and the consequent effects on ADBF performance. Digital equalization was effected via various methods, including a digital FIR filter design method and a variety of machine learning algorithms. All of the methods tested were good at improving channel matching; all methods tested improved the channel pair cancellation ratio to above 20 dB, and most algorithms tested were able to reach 30 dB. The effects of this improvement on array performance were illustrated by examining the spatial spectra of the input signal to the equalizer and the signals at the outputs of the equalizers. The signal without equalization appeared to come from about 20 degrees off broadside; after equalization, the signal appeared to come from broadside, as expected, a confirmation of the beneficial effects of equalization.

Other data recorded from the same receiver was used to construct an ADBF scenario with a signal and interferer present, with both at different amplitudes and apparent incident angles than that of the training data. This data was used as input to the different methods of equalization that were previously calibrated, and the improvement to SINR compared to ADBF without equalization was calculated. The digital filter method and linear model machine learning methods performed well on this data, improving SINR by almost 25 dB for the specific scene constructed. The nonlinear model machine learning methods did not adapt well to the changes made to the data; the tree-based models greatly attenuated the signal, and the neural network method introduced nonlinearities not present in the input signal. If care was taken to construct a training dataset for these nonlinear models that encompassed a wide range of apparent incident angles and amplitudes, these models could likely perform just as well as the digital filter or the linear models; however, this would likely not be worth the effort. The digital filter and linear models already perform very well and can be trained in $O(n^3)$ time; training trees takes

$O(n^3 \log n)$ time, and training neural nets can take $O(n^4)$ time. If the training dataset is expanded to include the wide variety of conditions needed for these models to perform well in a wide variety of conditions, training time would be prohibitive. The right tool for the job should be used; here, the right tool is one of the linear model machine learning methods or the digital FIR filter design method.

Finally, instead of using data from the prototype receiver, new data with different degrees of simulated mismatch was generated. This allowed investigation into the how well the results above could be generalized. The results above were confirmed: when the right tool is used (here, the digital FIR filter design method of equalization), equalization will *always* improve ADBF performance except when the target signal and the interferer are too closely located in angle, in which case the ADBF algorithm itself is the issue. Equalization, when done properly, is clearly a powerful tool for enhancing digital phased array performance.

Chapter 5

Conclusion

This dissertation explored a variety of methods of improving radar data quality. Already-existing temporal methods, the Vaisala algorithms, performed passably well at mitigating radio frequency interference as long as it was intermittent. A new algorithm, the Interference Spike Detection Algorithm, was developed; it outperforms the Vaisala algorithms in all measures. The Vaisala algorithms and ISDA both require the user to define parameters, but machine learning regression algorithms were applied to the problem to find a relation between the signal characteristics and algorithm parameters as a function of probability of false alarm; while the regression might be improved by using a nonlinear model, the relation found still performs passably well. Additionally, machine learning classification algorithms were applied for the first time to the RFI detection problem, and it was found that they have the potential to outperform ISDA and the other methods; however, these classification algorithms were too complicated to easily implement on an FPGA. Future work should investigate how the machine learning algorithm performance changes if fewer computational resources are available to it. It is very possible that, despite potentially higher performance from the machine learning algorithms, the quick speed and low complexity of ISDA could make them the preferred temporal RFI mitigation method, with the machine learning algorithms applied in post-processing to catch instances of RFI missed during online processing.

Despite this effort, all of the temporal methods fail when trying to address RFI that is continuous wave, noise-like, or intermittent RFI with a PRI perfectly synced with that of the radar; that is, RFI that is present for multiple contiguous pulses at a particular range gate. Fortunately, digital phased arrays and spatial processing algorithms offer another solution. A variety of adaptive digital beamforming algorithms were assessed for their performance in mitigating interferers of precisely the type the temporal algorithms were unable to handle. The MVDR algorithm stood outwith its high efficacy, little prerequisite knowledge required, and predictable computational complexity, and was chosen for implementation on the DARPA ACT program. In the weather radar domain, adaptive digital beamforming was used on weather data collected from spatially distributed channels on the NWRT. While there was no RFI present, the analogy can be made that ground clutter essentially is interference to weather radars; thus, the algorithm was used to filter out ground clutter from this data. The performance was very promising — the adaptive digital beamforming algorithm ameliorated the influence of the ground clutter on the signal statistics while preserving the weather data, and thus offers a viable alternative to state-of-the-art spectral reconstruction ground clutter filtering techniques such as GMAP. The linearly constrained minimum variance algorithm used here is particularly powerful, allowing users to ask it to satisfy constraints; if the location of a problematic emitter or wind turbine is known, the user can ask for nulls in the beampatterns in that direction, mitigating the influence of that interferer or clutter while not altering the signal statistics of the data received from the look direction. Future phased array weather radar systems will likely wish to perform at least some ground clutter filtering via adaptive digital beamforming.

The increased computational performance has additionally opened the door to improve phased array performance even outside the context of interference. In order to have accurate beamsteering and beamforming, the digital channels of a receiver must be well matched — an identical signal fed into separate hardware channels must yield the same signal out (or at least very close to the same signal out). Digital equalization can improve channel matching to high levels, and moreover it has been used to that effect when calibrated using signals of opportunity, meaning that the calibration of the equalizer need not necessarily take place in an anechoic chamber (though matching quality will very likely be higher if it does). With more accurate channel matching comes more accurate steering and consequently more accurate adaptive digital beamforming and thus interference suppression. In an experiment using data taken from a prototype Horus receiver, applying equalization before adaptive digital beamforming improved the final signal-to-interference ratio in virtually all cases. Machine learning regression methods show some promise as an alternative to the older digital FIR filter design techniques, though the right tool for the job must be used; techniques such as neural network regression are not nearly as well suited to the equalization problem as more classical machine learning methods such as linear regression. When implemented properly, digital equalization has great potential to improve phased array radar performance.

Appendix A

Useful Derivations

A.1 Adaptive Digital Beamforming

A.1.1 Minimum Variance Distortionless Response

Van Trees [42] makes the distinction between the minimum variance distortionless response (MVDR) beamformer, which seeks to minimize the power of the noise environment, and the minimum power distortionless response (MPDR) beamformer, which seeks to minimize the total power. The derivation is the same, the only difference is that MVDR uses the noise spectral matrix $\mathbf{S_n}$ instead of the signal spectral matrix $\mathbf{S_x}$. The same applies to linearly constrained minimum power (LCMP) beamforming versus linearly constrained minimum variance (LCMV) beamforming. Because most literature refers to the MPDR beamformer as defined by [42] *as* the MVDR beamformer, that is the convention followed in the text of the dissertation.

The problem of MVDR beamforming is to minimize the total signal power without altering the signal coming in from the steering direction. In math form:

$$\min_{\mathbf{w}} \{\mathbf{w}\mathbf{S_x}\mathbf{w}\} \text{ subject to } \mathbf{w}^H\mathbf{v} = 1 \tag{A.1}$$

where $\mathbf{w}$ are the weights being calculated; $\mathbf{S_x}$ is the received signal spectral matrix; $\mathbf{w}\mathbf{S_x}\mathbf{w}$ is the mean square of the output noise; and $\mathbf{v}$ is a steering vector pointed in the steering direction. Lagrangian multipliers can be used to make rephrase the problem:

$$f(\mathbf{w}) = \mathbf{w}^H\mathbf{S_x}\mathbf{w} + \lambda\left[\mathbf{w}^H\mathbf{v} - 1\right] + \lambda^*\left[\mathbf{v}^H\mathbf{w} - 1\right] \tag{A.2}$$

where λ are the Lagrangian multipliers (and are scalars) and $f(\mathbf{w})$ is the function to be minimized. We take the gradient with respect to $\mathbf{w}^H$ and set it equal to zero:

$$\begin{aligned} \nabla_{\mathbf{w}^H} f(\mathbf{w}) &= 0 \\ \Leftrightarrow 2\mathbf{w}^H \mathbf{S_x} + \mathbf{v}^H \lambda + \mathbf{v}^H \lambda &= 0 \\ \Leftrightarrow \mathbf{w}^H \mathbf{S_x} + \lambda \mathbf{v}^H &= 0 \\ \Leftrightarrow \mathbf{w}^H \mathbf{S_x} &= -\lambda \mathbf{v}^H \\ \Leftrightarrow \mathbf{w}^H &= -\lambda \mathbf{v}^H \mathbf{S_x}^{-1} \end{aligned} \tag{A.3}$$

Note that the signal covariance matrix $\mathbf{S_x}$ is Hermitian; that is, $\mathbf{S_x}^H = \mathbf{S_x}$; additionally, the inverse of a Hermitian matrix is itself a Hermitian matrix. We plug this into the constraint equation and solve for λ:

$$\begin{aligned} \mathbf{w}^H \mathbf{v} &= 1 \\ \Leftrightarrow -\lambda \mathbf{v}^H \mathbf{S_x}^{-1} \mathbf{v} &= 1 \\ \Leftrightarrow \lambda &= -\left[\mathbf{v}^H \mathbf{S_x}^{-1} \mathbf{v}\right]^{-1} \end{aligned} \tag{A.4}$$

We can plug this back into the equation above, canceling out the negatives and getting a formula of the optimum beamforming weights under the MVDR constraint:

$$\begin{aligned} \mathbf{w}^H &= \frac{\mathbf{v}^H \mathbf{S_x}^{-1}}{\mathbf{v}^H \mathbf{S_x}^{-1} \mathbf{v}} \\ \Leftrightarrow \mathbf{w} &= \frac{\mathbf{S_x}^{-1} \mathbf{v}}{\mathbf{v}^H \mathbf{S_x}^{-1} \mathbf{v}} \end{aligned} \tag{A.5}$$

Using these weights will minimize the total power while retaining the signal from the steering direction [42].

A.1.2 Linearly Constrained Minimum Variance

The LCMV beamformer is a more generalized form of the MVDR beamformer. The basic goal — to minimize the total signal power — remains, but with LCMV the constraint on this minimization is more than just distortionless response from the steering direction. The constraints themselves are up the user — some useful applications include steering nulls in particular directions to mitigate interferers, or asking for a strong beampattern to the side of the direction the target is expected to be transmitting from to accommodate for array mismatch — but the derivation is very similar. LCMV seeks to:

$$\min_{\mathbf{w}} \{\mathbf{w}\mathbf{S_x}\mathbf{w}\} \text{ subject to } \mathbf{w}^H\mathbf{C} = \mathbf{g}^H \tag{A.6}$$

where $\mathbf{w}$ are the weights being calculated; $\mathbf{S_x}$ is the received signal spectral matrix; $\mathbf{w}\mathbf{S_x}\mathbf{w}$ is the mean square of the output noise; $\mathbf{C}$ are the constraints parameters, and $\mathbf{g}$ are the desired values of the constraints. When $\mathbf{C} = \mathbf{v}$ (where $\mathbf{v}$ is the steering direction) and $\mathbf{g} = 1$, the LCMV beamformer simplifies into the MVDR beamformer.

To calculate the optimum weights for LCMV beamforming, we define a function $f(\mathbf{w})$ using Lagrangian multipliers λ to capture the constraints:

$$f(\mathbf{w}) = \mathbf{w}^H\mathbf{S_x}\mathbf{w} + \left[\mathbf{w}^H\mathbf{C} - \mathbf{g}^H\right]\lambda + \lambda^H\left[\mathbf{C}^H\mathbf{w} - \mathbf{g}\right] \tag{A.7}$$

We then take the gradient with respect to $\mathbf{w}^H$, set equal to zero, and solve for $\mathbf{w}^H$:

$$\begin{aligned} \nabla_{\mathbf{w}^H} f(\mathbf{w}) &= 0 \\ \Leftrightarrow 2\mathbf{w}^H\mathbf{S_x} + \lambda^H\mathbf{C}^H + \lambda^H\mathbf{C}^H &= 0 \\ \Leftrightarrow \mathbf{w}^H\mathbf{S_x} + \lambda^H\mathbf{C}^H &= 0 \\ \Leftrightarrow \mathbf{w}^H\mathbf{S_x} &= -\lambda^H\mathbf{C}^H \\ \Leftrightarrow \mathbf{w}^H &= -\lambda^H\mathbf{C}^H\mathbf{S_x}^{-1} \end{aligned} \tag{A.8}$$

Note that the signal covariance matrix $\mathbf{S_x}$ is Hermitian; that is, $\mathbf{S_x}^H = \mathbf{S_x}$; additionally, the inverse of a Hermitian matrix is itself a Hermitian matrix. We take this equation and plug it into the original constraint equation, solving for λ^H:

$$\begin{aligned} \mathbf{w}^H\mathbf{C} &= \mathbf{g}^H \\ \Leftrightarrow -\lambda^H\mathbf{C}^H\mathbf{S_x}^{-1}\mathbf{C} &= \mathbf{g}^H \\ \Leftrightarrow \lambda^H &= -\mathbf{g}^H\left[\mathbf{C}^H\mathbf{S_x}^{-1}\mathbf{C}\right]^{-1} \end{aligned} \tag{A.9}$$

We reincorporate to the equation above, canceling out the negatives and getting a formula for the optimum beamforming weights under the LCMV constraints:

$$\begin{aligned} \mathbf{w}^H &= \mathbf{g}^H \left[\mathbf{C}^H \mathbf{S}_\mathbf{x}^{-1} \mathbf{C}\right]^{-1} \mathbf{C}^H \mathbf{S}_\mathbf{x}^{-1} \\ \Leftrightarrow \mathbf{w} &= \mathbf{S}_\mathbf{x}^{-1} \mathbf{C} \left[\mathbf{C}^H \mathbf{S}_\mathbf{x}^{-1} \mathbf{C}\right]^{-1} \mathbf{g} \end{aligned} \tag{A.10}$$

Using these weights will minimize the total power while satisfying the constraints contained in **C** and **g** [42].

A.2 Direction Finding

A.2.1 Bartlett Method

The Bartlett beamformer is just a simple beamscan algorithm. In essence, a digital beam is scanned over the search space and the power of the return is plotted. In math form:

$$\begin{aligned} \hat{P}_B &= \frac{1}{K} \sum_{k=1}^{K} \left|\mathbf{v}^H \mathbf{X}_k\right|^2 \\ &= \mathbf{v}^H \left\{ \frac{1}{K} \sum_{k=1}^{K} \mathbf{X}_k \mathbf{X}_k^H \right\} \mathbf{v} \\ &= \mathbf{v}^H \mathbf{S}_\mathbf{x} \mathbf{v} \end{aligned} \tag{A.11}$$

where there are K samples; $\mathbf{v}$ are the steering vectors that span the search space; $\mathbf{X}_k$ is the signal at sample k; and $\mathbf{S}_\mathbf{x} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{X}_k \mathbf{X}_k^H$ is the signal covariance matrix. Because it's a digital beamscan, its resolution is limited to the resolution of the antenna array. In linear algebra terms, this algorithm is essentially finding the length of $\mathbf{S}_\mathbf{x}$ in the direction of $\mathbf{v}$ [81, 42].

A.2.2 MUSIC

Multiple **Si**gnal **C**lassification, commonly called MUSIC, is a direction finding algorithm based on subspace analysis. Because MUSIC does not rely on the radar geometry at all, its resolution is not limited as the Bartlett method's was. If **X** is a signal comprised of D incoming signals and noise and is received by an M-element radar, then it can be modeled as

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{pmatrix} = \begin{pmatrix} V(\theta_1) & V(\theta_2) & \cdots & V(\theta_D) \\ & & & \\ & & & \\ & & & \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \\ \vdots \\ F_D \end{pmatrix} + \begin{pmatrix} B_1 \\ B_2 \\ \vdots \\ B_M \end{pmatrix} \tag{A.12}$$

which can be reduced to

$$\mathbf{X} = \mathbf{VF} + \mathbf{B} \tag{A.13}$$

where $\mathbf{V}$ are the steering vectors associated with the incident directions of the D signals in $\mathbf{F}$, and B_i are the noise at each of the M elements. The covariance matrix of $\mathbf{X}$ is then

$$\mathbf{S_x} = \mathbf{XX}^H = \mathbf{VFF}^H\mathbf{V}^H + \mathbf{BB}^H \tag{A.14}$$

which, if the incident signals and the noise are uncorrelated, can be simplified into

$$\mathbf{S_x} = \mathbf{VPV}^H + \lambda S_0 \tag{A.15}$$

Unless the signals F_i are completely uncorrelated, the matrix $\mathbf{P}$ will be positive definite, and when the number of signals D is less than the number of array elements M, then the matrix $\mathbf{VPV}^H$ will not be full rank, meaning that its determinant will equal zero. The equation can be rearranged

$$\left|\mathbf{VPV}^H\right| = |\mathbf{S_X} - \lambda S_0| = 0 \tag{A.16}$$

This can only be true if $\mathbf{S} = \lambda \mathbf{S}_0$, meaning that λ must equal one of the eigenvalues of $\mathbf{S}$. Since $\mathbf{V}$ is full rank and $\mathbf{P}$ is positive definite, $\mathbf{VPV}^H$ must be nonnegative definite, requiring that $\lambda = \lambda_{min}$ where λ_{min} is the minimum eigenvalue of $\mathbf{S}$ in the metric of S_0. This can be used to estimate D, the number of incident signals, by counting the number of eigenvalues of $\mathbf{S}$ approximately equal to λ_{min} and subtracting that from the number of elements. That is, $\hat{D} = M - \hat{N}$ where $\hat{N}$ is the number of eigenvalues approximately equal to λ_{min}. When examining matrix eigenvalues and eigenvectors, by definition

$$\mathbf{S}\mathbf{e}_i = \lambda_i S_0 \mathbf{e}_i \forall i \in [1, M] \tag{A.17}$$

which, using equation A.15, can be rewritten as

$$\begin{aligned} \mathbf{V}\mathbf{P}\mathbf{V}^H\mathbf{e}_i + \lambda_{min} S_0 \mathbf{e}_i &= \lambda_i S_0 \mathbf{e}_i \\ \Leftrightarrow \mathbf{V}\mathbf{P}\mathbf{V}^H\mathbf{e}_i &= (\lambda_i - \lambda_{min}) S_0 \mathbf{e}_i \end{aligned} \tag{A.18}$$

which, whenever $\lambda_i = \lambda_{min}$ (meaning that the current eigenvalue is not associated with a signal) will evaluate to zero. In linear algebra terms, when $\mathbf{V}\mathbf{P}\mathbf{V}^H\mathbf{e}_i = 0$, then the current eigenvector $\mathbf{e}_i$ associated with the current eigenvalue $\lambda_i \approx \lambda_{min}$ is orthogonal to $\mathbf{V}\mathbf{P}\mathbf{V}^H$, which is the space spanned by the signal vectors. Using this, the eigenvectors can thus be sorted into two groups: $\mathbf{E}_D$, a set of D eigenvectors that span the signal subspace; and $\mathbf{E}_N$, a set of N eigenvectors that span the noise subspace. Additionally, because the bases for these subspaces are eigenvectors of a Hermitian matrix, they are orthogonal, meaning that the signal and noise subspaces are orthogonal.

To perform MUSIC, then, we simply define a space over which we wish to search, $\mathbf{v}$, and look at the lengths of each v_i in the noise subspace:

$$\begin{aligned} \hat{\mathbf{Q}}_{MUSIC} &= \left|\mathbf{v}^H\mathbf{E}_N\right|^2 \\ \Leftrightarrow &= \mathbf{v}^H\mathbf{E}_N\mathbf{E}_N^H\mathbf{v} \\ \Leftrightarrow &= \mathbf{v}^H\left(\mathbf{I} - \mathbf{E}_D\mathbf{E}_D^H\right)\mathbf{v} \end{aligned} \tag{A.19}$$

Typically this in inverted, since the lowest lengths in the noise subspace are associated with the directions most associated with the signal subspace; that is, $\hat{\mathbf{P}}_{MUSIC} = \frac{1}{\hat{\mathbf{Q}}_{MUSIC}}$ is typically evaluated [82, 42].

Appendix B

Machine Learning Background

B.1 Linear Models

B.1.1 Linear Regression

The following conventions are used: $\mathbf{y}$ is a series of observations; $\mathbf{X}$ is the series of system states that, using coefficients in β, can be used to generate a predicted system state $\hat{\mathbf{y}}$, where $\hat{\mathbf{y}} = \mathbf{X}\beta$. Often a bias term, β_0, is folded into the β and $\mathbf{X}$ matrices.

An example application is the simple case of projectile motion: a ball traveling in one direction at a constant velocity. We want to find the linear model $\hat{\mathbf{y}} = \mathbf{X}\beta$ that will best describe the system states in $\mathbf{y}$. In this case, $\mathbf{y}$ is the set of observations of the ball's position; $\mathbf{X}$ is the times at which the observations are taken, and β are the unknown quantities that govern the ball's motion (in this case, the starting position y_0 and the velocity v). If there are n observations, the series of equations comprising our linear model,

$$\begin{aligned} \hat{y}_1 &= \beta_0 + \beta_1 t_1 \\ \hat{y}_2 &= \beta_0 + \beta_1 t_2 \\ &\vdots \\ \hat{y}_n &= \beta_0 + \beta_1 t_n \end{aligned} \tag{B.1}$$

can, by setting

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & t_1 \\ 1 & t_2 \\ \vdots & \\ 1 & t_n \end{pmatrix}, \text{and } \beta = \begin{pmatrix} y_0 \\ v \end{pmatrix}$$

be packaged up neatly:

$$\mathbf{y} = \mathbf{X}\beta = \hat{\mathbf{y}} \tag{B.2}$$

Note that the column of ones at the left of $\mathbf{X}$ is the byproduct of incorporating the bias term $\beta_0 = y_0$ into $\mathbf{X}\beta$.

There are many methods to solve for β, but the most common is the method of least squares, wherein the β is chosen such that the sum of squares of a residual vector

$$\mathbf{r} = \mathbf{y} - \hat{\mathbf{y}} \tag{B.3}$$

is minimized. When using the Euclidean norm, $||\mathbf{r}||_2 = \left(r_1^2 + r_2^2 + \cdots + r_n^2\right)^{1/2}$,we find the minimum of

$$\begin{aligned} f(\beta) &= ||\mathbf{r}||_2^2 = ||\mathbf{y} - \hat{\mathbf{y}}||_2^2 \\ &= ||\mathbf{y} - \mathbf{X}\beta||_2^2 \\ &= (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) \end{aligned} \tag{B.4}$$

which will be at a location when both of these conditions are true:

$$\begin{cases} \nabla f(\beta) = 0 \\ \nabla^2 f(\beta) \text{ is positive definite} \end{cases} \tag{B.5}$$

where

$$\nabla f(\beta) = \left(\frac{\partial f}{\partial \beta_0}, \frac{\partial f}{\partial \beta_1}\right)^T \tag{B.6}$$

and

$$\nabla^2 f(\beta) = \begin{pmatrix} \frac{\partial^2 f}{\partial \beta_0^2} & \frac{\partial^2 f}{\partial \beta_0 \partial \beta_1} \\ \frac{\partial^2 f}{\partial \beta_0 \partial \beta_1} & \frac{\partial^2 f}{\partial \beta_1^2} \end{pmatrix} \tag{B.7}$$

Expanding the $f(\beta)$ from above, we obtain

$$\begin{aligned} f(\beta) &= (\mathbf{y} - \mathbf{X}\beta)^H (\mathbf{y} - \mathbf{X}\beta) \\ &= \left(\mathbf{y}^T - \beta^T \mathbf{X}^T\right)(\mathbf{y} - \mathbf{X}\beta) \\ &= \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta \\ &= \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\beta + \beta^T\left(\mathbf{X}^H\mathbf{X}\right)\beta \end{aligned} \tag{B.8}$$

From this, we can find that

$$\nabla f(\beta) = -2\mathbf{X}^T\beta + 2\left(\mathbf{X}^T\mathbf{X}\right)\beta \tag{B.9}$$

and

$$\nabla^2 f(\beta) = 2\left(\mathbf{X}^T\mathbf{X}\right) \tag{B.10}$$

The first condition from B.5 yields that

$$\nabla f(\beta) = 0 = -2\mathbf{X}^T\mathbf{y} + 2\left(\mathbf{X}^T\mathbf{X}\right)\beta \Leftrightarrow \left(\mathbf{X}^T\mathbf{X}\right)\beta = \mathbf{X}^T\mathbf{y} \tag{B.11}$$

which, if $\mathbf{X}^T\mathbf{X}$ is non-singular, yields the solution

$$\beta = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y} \tag{B.12}$$

Using the second condition from B.5 we find that $\nabla^2 f(\beta)$ is positive definite if $\mathbf{X}^T\mathbf{X}$ is positive definite.

The requirement that the matrix $\mathbf{X}^T\mathbf{X}$ be nonsingular can become a problem if the features in the different columns of $\mathbf{X}$ are not independent. If the terms are correlated, the $\mathbf{X}^T\mathbf{X}$ matrix can approach singularity, resulting in random errors causing large variance in the system output [20, 83, 23]. This can be dealt with using shrinkage methods such as ridge regression and lasso.

B.1.2 Tikhonov Regularization (Ridge Regression)

Tikhonov regularization, also known as ridge regression or L_2 regularization, changes the minimum residual problem solved by least squares regression into a problem that balances minimizing the residual and minimizing the norm of the coefficients in β. The problem becomes minimizing $f(\beta)$, where:

$$f(\beta) = ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \alpha\,||\beta||_2^2 \tag{B.13}$$

where $\alpha \geq 0$ is a real constant. Minimizing it involves the same conditions. Setting the gradient to zero yields the solution

$$\begin{aligned} \nabla f(\beta) = \left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)\beta - \mathbf{X}^T\mathbf{y} = 0 \\ \Leftrightarrow \beta = \left(\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I}\right)^{-1}\mathbf{X}^T\mathbf{y} \end{aligned} \tag{B.14}$$

Large α will increase the shrinkage that occurs, making the coefficients more robust to issues arising from correlated features. Overall, ridge regression acts to smooth the solutions, and acts to transform ill-posed problems into well-posed problems [20, 83, 23].

B.1.3 Lasso

Lasso, an acronym for "least absolute shrinkage and selection operator", is sometimes known as L_1 regularization. It is another shrinkage method that can be used to improve regression performance over simple least squares regression by adding a different constraint — minimizing the absolute value of the coefficients — to the least squares problem. The problem becomes minimizing $f(\beta)$:

$$f(\beta) = ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \alpha\,||\beta||_1 \tag{B.15}$$

where $||\beta||_1 = |\beta_0| + |\beta_1| + \cdots + |\beta_n|$ is the Manhattan norm. There is no closed form solution for β as there was with ridge regression, but, as α increases, the penalty term tends to prefer solutions with fewer parameter values, resulting in a solution that depends on fewer variables [20, 23].

B.1.4 Elastic Net

Elastic net regularization strives to strike a balance between the L_1 penalty imposed by lasso and the L_2 penalty imposed by ridge regression. The penalty has the form

$$\sum_{i=1}^{n}\left(\alpha\,|\beta_i| + (1-\alpha)\beta_i^2\right) \tag{B.16}$$

making the minimization problem that of finding the minimum of $f(\beta)$ where

$$f(\beta) = ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda\alpha\, ||\beta||_2^2 + (1-\lambda)\,\alpha\, ||\beta||_1 \tag{B.17}$$

where α performs the same functions as in ridge regression and lasso, and λ allows the user to choose the balance between the L_1 and L_2 penalties. Each penalty performs the same functions as above: the L_1 term performs variable selection, excluding features that are deemed unimportant, while the L_2 increases numeric stability [20, 23].

B.2 Trees

The most common application of trees are in classification problems, as was done with RFI detection in Section 2.3.4. Classification trees are simply binary trees that split using the feature that will give the most information. A simple example, the problem of "should I eat at Chipotle?", can be simply represented using the tree in Figure B.1. To construct a tree, the training dataset is split using a criterion, typically to find the feature that provides the most information gain or maximizes the "purity" of each class in its child nodes. This can be done using a few different metrics; for the experiment in Section 2.3.4, the Gini index was used as the splitting criterion. Using the definition

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i} I(y_i = k) \tag{B.18}$$

to represent the proportion of members in a node m that belong to class k, these criteria can be defined. The Gini index is defined by

$$\sum_{k \neq k'} \hat{p}_m k \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{B.19}$$

Similarly, the entropy is defined by

$$-\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk} \tag{B.20}$$

Information gain looks at the difference in one of these values before and after making a split. A positive value indicates a useful split, and more positive values indicate better splits.

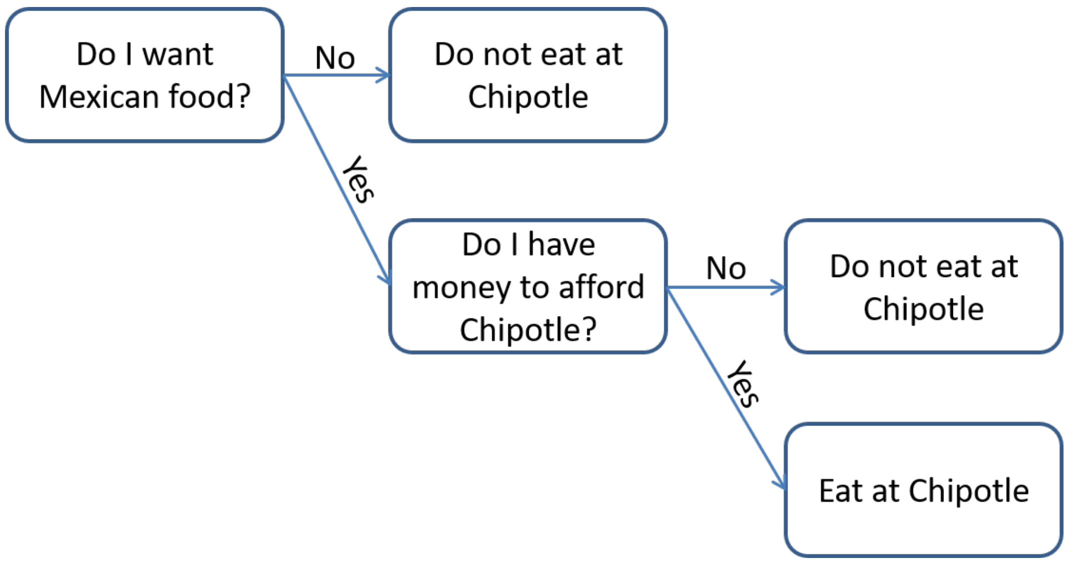


Figure B.1: Simple classification problem governing whether someone should eat at Chipotle, with the root node at the left. An accurate tree would, of course, just be a single node with "Yes".

Regression trees operate similarly, with two primary differences. First, instead of the tree solving a classification problem with a limited number of discrete classes, it solves a regression problem whose outputs are continuous and real. Second, rather than splitting based on a measure of information gain, the split is chosen by finding a constant that minimizes the residual between that constant chosen and the output truth values. This minimization can be in terms of mean squared error:

$$\min_{c} \sum_{xi} (y_i - c)^2 \tag{B.21}$$

or mean absolute error:

$$\min c \sum_{xi} |y_i - c| \tag{B.22}$$

where y_i is the truth data for each corresponding data point x_i and c is the constant being searched for.

Rather than exhaustively test the all potential partitions, the Classification And Regression Trees (CART) algorithm (which is the algorithm used by Scikit-Learn) uses a greedy approach: starting with the current full dataset, it looks at all available variables and all potential values upon which to split that variable; it then chooses the constant that minimizes the error on each side of that split; it chooses to make that node of the tree out of the split that results in the lowest error in the new partitions. More formally: for each feature variable j in the training data X, it splits the data into two regions based on whether the value of that data's feature j is greater than or less than a splitting value s:

$$R_1(j,s) = \{X|X_j \leq s\} \text{ and } R_2(j,s) = \{X|X_j > s\} \tag{B.23}$$

It then finds the constant c_1 and c_2 in each region that minimizes the error between the truth value and the constant. Altogether:

$$\min_{j,s}\left[\min_{c_1}\sum_{x_i\in R_1(j,s)}(y_i-c_1)^2+\min_{c_2}\sum_{x_i\in R_2(j,s)}(y_i-c_2)^2\right] \tag{B.24}$$

when the mean squared error measure is used. When it finds the best split, it chooses that and repeats the process for each branch of the tree. This repeats until some criteria is fit; typically, when the number of data points in each region reaches some threshold; for Scikit-Learn, this threshold is 1 data point, meaning that potentially each piece of data in the training set will have its own path down the tree. While this results in good performance on the training data, if the tree becomes too big then it can become "brittle" and unable to perform well on input data that were not included in the training data.

Random forests are a tree-based machine learning method that can deal with such issues. Random forests are built from a group of individual decision trees that are built using the input data. Rather than sending all the data to all the trees, however, the training dataset is bootstrapped (i.e. a subset is randomly chosen from all the available data) and each tree is trained on its own bootstrapped data; additionally, during training each tree will choose the best feature upon which to split from a random subset of all available features. Both factors combine to produce individual trees that will have their own unique structures. When used to perform a prediction, the outputs from each tree in the forest are averaged. This results in a model that has overall reduced variance in its output.

Gradient boosting trees use a different approach. Whereas random forests average the output of an ensemble of trees, gradient boosting sums the outputs of a group of trees that have been trained to work together in concert. The starting tree is trained in the same way as the original, to predict the constant that minimizes the error between the truth values and the predicted value; this residual error is then used as the truth data for the next tree, which chooses constants to minimize that error. This is repeated with subsequent trees. In mathematical terms, these trees are being trained to predict the negative gradient of the residual error (typically calculated using least squares), hence the name "‘gradient boosting" [20].

B.3 Neural Networks

Neural nets are constructed out of many individual "neurons", so named because they were originally developed to model the neurons in the human brain, working together in concert to perform a machine learning task. Like trees, neural nets are ultimately nonlinear. The neural nets are organized in several layers — an input layer, one or more hidden layers, and an output layer — and each neuron is connected to all the neurons in the next layer. An example is shown in Figure B.2. Each of the interconnections is weighted, and each of these weighted inputs are used to calculated the neuron output value, which can be one of several functions. The one used in this paper was the rectified linear unit

$$f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases} \tag{B.25}$$

To train a neural network, the initial weights of the interconnections are randomized and the input data fed into the network. The output is observed and the error calculated. The error is propagated back through the network and the gradients of the error are used to adjust the weights. The process is repeated until the error has reached an acceptable value; training is stopped before a minimum is reached to avoid overfitting,

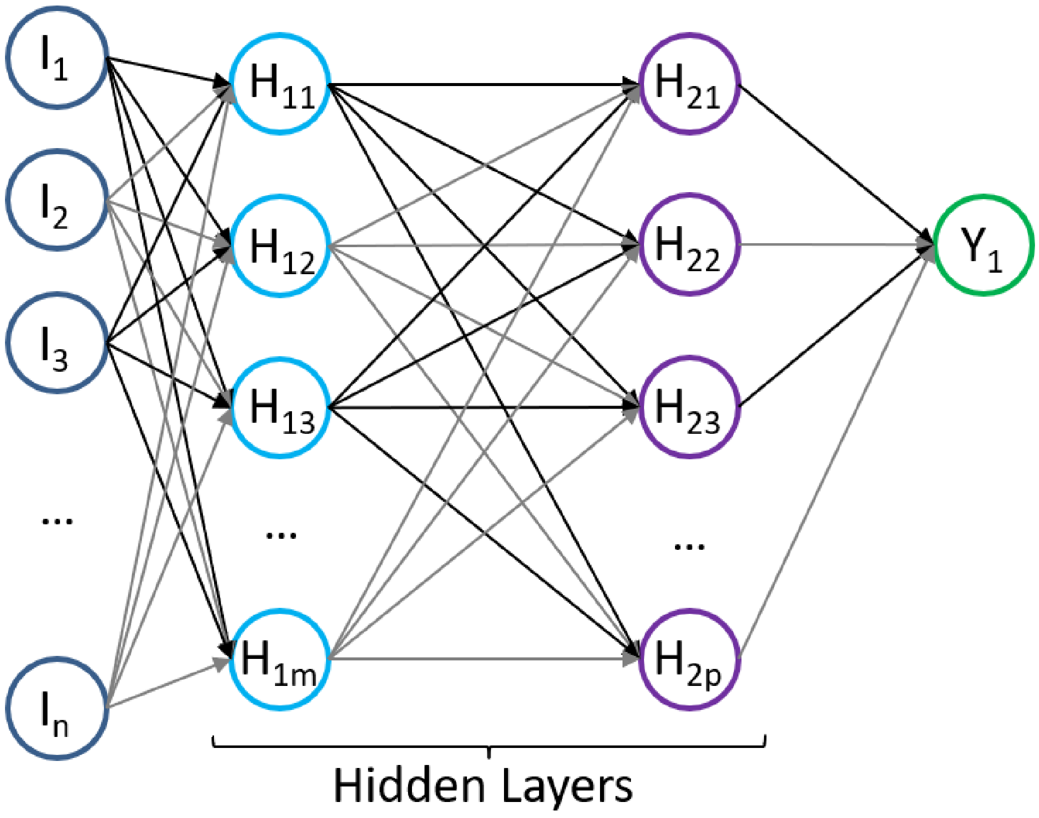


Figure B.2: An example neural network architecture with n nodes in the input layer, two hidden layers with m and p nodes, and an output layer comprised of one node. Each arrow has a weight associated with it.

which would mean a loss of model generalizability [20] This training process takes $O(n^4)$ time, making training from scratch a very expensive process; fortuitously, neural nets are amenable to online training, ameliorating this problem somewhat [23]. Neural networks are sensitive to input data values, and so input data should be normalized to fit in a normal distribution with mean of zero and variance of one. Additionally, neural networks are prone to overfitting, and can lose their ability to perform in circumstances that don't match their training [20].

Appendix C

Useful Matlab Code

The code below is the implementation of ISDA used in the experiments in this dissertation. This code should be useful for performing ISDA in for offline processing, and will hopefully provide a useful starting point for anyone wishing to implement ISDA.

```
function [flaggedAsRFI] = ISDA(dataIQ,neighborSet,threshold)
%ISDA Interference Spike Detection Algorithm, detects RFI.
%   A cell-averaging CFAR method useful for the detection of
%     radio frequency interference.
%   This performs ISDA after the whole dwell has been recorded;
%     if it is desired to perform ISDA as data streams in,
%     modifications will have to be made.
%
%   Arguments:
%     dataIQ:       numPulses x numRangeBins
%                   The IQ data that comprises a single dwell
%                     pointed in a single azimuth direction.
%     neighborSet:  numNeighbors x 2
%                   The location of the neighbors of each cell
%                     relative to the current cell. [-1,0] will
%                     make a neighbor of the cell above, [0,1]
%                     will make a neighbor of the range bin
%                     after, and so on.
%                   The function MakeNeighborSet() may prove
%                     useful for procedurally generating neighbor
%                     sets.
%     threshold:    Scalar value.
%                   The value to which the ratio of the power at
%                     the current cell and the mean of the powers
%                     of the neighboring cells is compared. If it
%                     exceeds this value, then the current cell
%                     is flagged as RFI.
%                   When tested on white Gaussian noise, this
%                     approximate parameterization proved useful:
%                         threshold = (PFA / 1.7628) ^ (-1/1.835)
%                     where PFA is in linear units.
%                   Alternatively:
```

```
%                       a threshold of 5  gives a PFA near 0.1,
%                       a threshold of 18 gives a PFA near 0.01,
%                       a threshold of 61 gives a PFA near 0.001.
%
%   Returns:
%     flaggedAsRFI: numPulses x numRangeBins
%                   A Boolean array (ones and zeros) of where the
%                     RFI was detected.
%                   Follow up with a function like RestoreData()
%                     to deal with the RFI detections, though be
%                     warned: using a PFA that is too high can
%                     result in very aggressive RFI detections
%                     and thus a loss of signal detail.
%

% Figure out how many neighbors are in the neighborSet.
numNeighbors = size(neighborSet,1);
% Pre-allocate for efficiency.
neighbors = zeros([numNeighbors,size(dataIQ)]);
% Put all the neighbors in one big array.
for idxNeighbor = 1:length(numNeighbors)
  neighbors(idxNeighbor,:,:) = ...
    circshift(dataIQ,neighbors(idxNeighbor,:));
end
% Find the power of the neighbors.
neighborPower = neighbors.^2;
% Take the mean.
meanNeighborPower = squeeze(mean(neighborPower,1));

% Find the power of the cell under test.
currDataPower = dataIQ.^2;

% Take the ratio of the two.
powerRatio = currDataPower ./ meanNeighborPower;

% If the ratio exceeds the threshold, flag as RFI.
flaggedAsRFI = (powerRatio > threshold);


end
```

```
function [neighborLocs] = MakeNeighborSet(params)
%MAKENEIGHBORSET Make a set of neighbor locations for ISDA.
%   Simple function used to generate the locations of the
%     neighbors that ISDA uses as part of its check for RFI.
%   This function doesn't need to be used for ISDA, but proved
%     useful for testing the performance of ISDA where the
%     location of neighbors varied.
%
%   Arguments:
%     params:           A structure containing any or all of the
%                         fields below.
%
%     params.widthPulse:      Default: 1
%                             The number of points to grab on
%                               each slow-time side of the cell
%                               under test. Thus, a width of 1
%                               will yield 2 locations; a width
%                               of 3 will yield 6 locations, and
%                               so on.
%     params.widthRange:      Default: 0
%                             The number of points to grab on
%                               each fast-time side of the cell
%                               under test. Again, a width of 1
%                               will yield 2 locations, and so
%                               on.
%     params.guardNumPulse:   Default: 0
%                             The number of cells to skip around
%                             the cell under test when grabbing
%                             neighbors from the slow-time
%                             dimension. A guardNumPulse of 0
%                             will result in the use of data
%                             points immediately adjacent (in
%                             slow-time) to the cell under test.
%     params.guardNumRange:   Default: 0
%                             The number of cells to skip around
%                               the cell under test when grabbing
%                               neighbors from the fast-time
%                               dimension. A guardNumPulse of 0
%                               will result in the use of data
%                               points immediately adjacent (in
%                               fast-time) to the  cell under
%                               test.
%
%   Returns:
%     neighborLocs:           (widthPulse+widthRange)*2 x 2
%                             Set of locations from which to grab
%                               the neighbors for ISDA. Useful as
%                               arguments into circshift().
%                             If using default arguments, will
%                               use data from the pulses
%                               immediately before and after the
%                               cell under test, yielding:
%                                 [ 1,0 ;
%                                  -1,0 ]
%

% Set default values if not otherwise specified.
if isfield(params,'widthPulse')
  widthPulse    = params.widthPulse;
else
  widthPulse = 1;

end
if isfield(params,'widthRange')
  widthRange    = params.widthRange;
else
  widthRange = 0;
end

if isfield(params,'guardNumPulse')
  guardNumPulse = params.guardNumPulse;
else
  guardNumPulse = 0;
end
if isfield(params,'guardNumRange')
  guardNumRange = params.guardNumRange;
else
  guardNumRange = 0;
end

% Pre-allocate for efficiency's sake.
neighborLocs = zeros(widthPulse+widthRange,2);

% Generate the neighbor locations.
for idxWidthP = 1:widthPulse
```

```
  currRow = idxWidthP;
  neighborLocs(currRow,:) = [idxWidthP+guardNumPulse, 0];
end
for idxWidthR = 1:widthRange
  currRow = widthPulse+idxWidthR;
  neighborLocs(currRow,:) = [0,idxWidthR+guardNumRange];
end

% The above generates the neighbors below and to the right of the
%   point of interest. This line below makes it symmetric about
%   the cell under test.
neighborLocs = [neighborLocs ; -1*neighborLocs];


end
```

```
function [restoredDataIQ] = RestoreData(dataIQ,flagPlot,method)
%RESTOREDATA Performs data restoration of RFI-afflicted data.
%   Restores RFI-afflicted data using the specified method.
%
%   Arguments:
%     dataIQ:         numPulses x numRangeBins
%                     IQ data that comprises a dwell in a single
%                       azimuth direction.
%     flagPlot:       numPulses x numRangeBins
%                     Boolean array (ones and zeros) indicating
%                     where RFI is present. Use an algorithm such
%                     as ISDA to generate such a plot.
%     method:         string
%                     What method of data restoration to use. By
%                     default, will perform slow time
%                     interpolation  (i.e. it will interpolate
%                     interpolate across the pulse before and
%                     after the pulse where RFI is present.
%
%   Returns:
%     restoredDataIQ: numPulses x numRangeBins
%                     IQ data after data restoration has been
%                        performed.
%

if nargin < 3
  method = 'slowtime';
end

% Convert string to lower case.
method = lower(method);

% [numPulses,numRangeBins] = size(dataIQ);
restoredDataIQ = dataIQ;

switch method
  case 'slowtime'
    % Interpolate across slow time (pulses).
    neighbors = zeros([2,size(dataIQ)]);
    neighbors(1,:,:) = circshift(dataIQ,[-1,0]);
    neighbors(2,:,:) = circshift(dataIQ,[ 1,0]);
    meanNeighbors = mean(neighbors,1);

    restoredDataIQ(flagPlot==1) = meanNeighbors(flagPlot==1);

  case 'fasttime'
    % Interpolate across fast time (range).
    neighbors = zeros([2,size(dataIQ)]);
    neighbors(1,:,:) = circshift(dataIQ,[0,-1]);
    neighbors(2,:,:) = circshift(dataIQ,[0, 1]);
    meanNeighbors = mean(neighbors,1);

    % I don't want to interpolate the first and last range gates
    %   across fast time, because that doesn't make any since -
    %   they're not related at all. In those cases, just do
    %   simple replacement with data from the neighboring range
    %   gate.
    meanNeighbors(1,:) = dataIQ(2,:);
    meanNeighbors(end,:) = dataIQ(end-1,:);

    restoredDataIQ(flagPlot==1) = meanNeighbors(flagPlot==1);

  case 'simplereplacement'
    % Just copy data from the pulse before.
    replacementData = circshift(dataIQ,[0,-1]);
    restoredDataIQ(flagPlot==1) = replacementData(flagPlot==1);

  otherwise
    fprintf(1,'Not supported, returning original data.\n');
end


end
```

Bibliography

[1] National Weather Service, "Using and understanding Doppler radar," https://www.weather.gov/mkx/using-radar, accessed: 2016-12-11.

[2] Wikipedia contributors, "File:atmospheric electromagnetic opacity — Wikipedia, the free encyclopedia," 2019, [Online; accessed 5 March 2019]. [Online]. Available: https://en.wikipedia.org/wiki/File:Atmospheric_electromagnetic_opacity.svg

[3] J. Y. Cho, "OEP terminal and CONUS weather radar coverage gap identification analysis for nextgen," 2010.

[4] R. Palmer, C. Fulton, J. Salazar, H. Sigmarsson, and M. Yeary, "The Horus radar - an all-digital polarimetric phased array radar for multi-mission surveillance," in *Env. Inf. Proc. Tech.* Phoenix, AZ: Amer. Met. Soc., January 2019.

[5] R. J. Doviak and D. S. Zrnić, *Doppler Radar and Weather Observations*. Dover Publications, 2006.

[6] S. Cocks, J. Boettcher, and P. Schlatter, "An operational assessment of pre-deployment dual polarization WSR-88D radar data," in *35th Conference on Radar Meteorology*. Amer. Met. Soc., 2011.

[7] D. E. Forsyth, J. F. Kimpel, D. S. Zrnic, R. Ferek, J. F. Heimmer, T. McNellis, J. E. Crain, A. M. Shapiro, R. J. Vogt, and W. Benner, "The National Weather Radar Testbed (phased-array)," in *32nd Conference on Radar Meteorology*, 2005, pp. 24–29.

[8] D. Zrnic, J. Kimpel, D. Forsyth, A. Shapiro, G. Crain, R. Ferek, J. Heimmer, W. Benner, F. T. McNellis, and R. Vogt, "Agile-beam phased array radar for weather observations," *Bulletin of the American Meteorological Society*, vol. 88, no. 11, pp. 1753–1766, 2007.

[9] M. E. Weber, J. Y. Cho, J. S. Herd, J. M. Flavin, W. E. Benner, and G. S. Torok, "The next-generation multimission US surveillance radar network," *Bulletin of the American Meteorological Society*, vol. 88, no. 11, pp. 1739–1752, 2007.

[10] B. Isom and C. Curtis, "Can NEXRAD and industry share the S-band? Exploring the impact of RF interference on the WSR-88D estimators," in *8th European Conf. on Radar in Met. and Hydro.* Amer. Met. Soc., 2014.

[11] H. Griffiths, S. Blunt, L. Cohen, and L. Savy, "Challenge problems in spectrum engineering and waveform diversity," in *2013 IEEE Radar Conf.* Ottawa, ON: IEEE, 2013, pp. 1–5.

[12] President's Council of Advisors on Science and Technology (PCAST), "Realizing the full potential of government-held spectrum to spur economic growth," President's Council of Advisors on Science and Technology (PCAST), Washington, D.C., Tech. Rep., 2012.

[13] J. L. Lake, M. Yeary, and C. D. Curtis, "Adaptive radio frequency interference mitigation techniques at the National Weather Radar Testbed: First results," in *2014 IEEE Radar Conference*, May 2014, pp. 0840–0845.

[14] M. E. Weber, "FAA surveillance radar data as a complement to the WSR-88D network," in *Preprints, 9th Conference on Aviation Range and Aerospace Meteorology*, vol. 150, 2000.

[15] Vaisala Oyj, *Vaisala User's Manual: Digital IF Receiver/Doppler Signal Processor, RVP8*. Helsinki, Finland: Vaisala Oyj, 2013.

[16] D. Franc, "The effects of interference on NEXRAD: A requirements-based approach," National Oceanic and Atmospheric Administration and Office of Oceanic and Atmospheric Research, Tech. Rep., 2014.

[17] J. L. Lake, M. Yeary, and C. D. Curtis, "Effects of radio frequency interference mitigation strategies on meteorological data," in *2016 IEEE Radar Conference (RadarConf)*, May 2016, pp. 1–5.

[18] H. Finn, "Adaptive detection mode with threshold control as a function of spatially sampled clutter-level estimates," *RCA Rev.*, vol. 29, pp. 414–465, 1968.

[19] D. S. Zrnić, "Simulation of weatherlike Doppler spectra and signals," *Journal of Applied Meteorology*, vol. 14, no. 4, pp. 619–620, 1975.

[20] T. Hastie, R. Tibshirani, and J. Friedman, "The elements of statistical learning: Data mining, inference, and prediction," 2009.

[21] T. Mitchell, *Machine Learning*. New York, NY: McGraw-Hill Science/Engineering/Math, 1997.

[22] A. Liaw and M. Wiener, "Classification and regression by random forest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[24] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Machine learning*, vol. 31, no. 1, pp. 1–38, 2004.

[25] C. A. Balanis, *Antenna Theory: Analysis and Design*. John Wiley & Sons, 2016.

[26] B. Epstein, R. H. Olsson, and K. Bunch, "Arrays at commercial timescales: Addressing development and upgrade costs of phased arrays," in *2018 IEEE Radar Conference (RadarConf18)*, April 2018, pp. 0327–0332.

[27] T. Rondeau, "Arrays at commercial timescales (ACT)," https://www.darpa.mil/program/arrays-at-commercial-timescales, accessed: 2019-03-06.

[28] A. Saunders, J. Lake, C. Fulton, M. Yeary, F. Robey, T. Hoffman, T. Karrels, and D. Jensen, "Fully adaptive digital beamforming on an FPGA for the DARPA ACT program," in *2018 GOMACTech Conf.*, Miami, FL, 2018.

[29] G. Zhang, R. J. Doviak, D. S. Zrnic, J. Crain, D. Staiman, and Y. Al-Rashid, "Phased array radar polarimetry for weather sensing: A theoretical formulation for bias corrections," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 11, pp. 3679–3689, Nov 2009.

[30] J. E. Stailey and K. D. Hondl, "Multifunction phased array radar for aircraft and weather surveillance," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 649–659, March 2016.

[31] K. A. Wilson, P. L. Heinselman, C. M. Kuster, D. M. Kingfield, and Z. Kang, "Forecaster performance and workload: Does radar update time matter?" *Weather and Forecasting*, vol. 32, no. 1, pp. 253–274, 2017. [Online]. Available: https://doi.org/10.1175/WAF-D-16-0157.1

[32] H. Groginsky and K. Glover, "Weather radar canceller design," in *19th Conf. Radar Meteorology*. Miami Beach, FL: Amer. Met. Soc., April 15-18 1980, pp. 192–198.

[33] M. Meischner, *Weather Radar: Principles and Advanced Applications*. Berlin, Germany: Springer-Verlag, 2002.

[34] R. Lee, G. Deruna, and J. Joss, "Intensity of ground clutter and of echoes of anomalous propagation and its elimination," in *27th Conf. Radar Meteorol.* Vail, CO: Amer. Met. Soc., 1995, pp. 651–652.

[35] C. Kessinger, S. Ellis, and J. Andel, "The radar echo classifier: A fuzzy logic algorithm for the WSR-88D," in *3rd Conf. on Artificial Intelligence Applications to the Environmental Science*. Long Beach, CA: Amer. Met. Soc., 2003, pp. 1–11.

[36] J. Hubbert, M. Dixon, and S. Ellis, "Weather radar ground clutter. part ii: Real-time identification and filtering," *J. Atmos. Oceanic Technol.*, vol. 26, pp. 1181–1197, July 2009.

[37] A. Siggia and R. Passarelli, "Gaussian model adaptive processing (GMAP) for improved ground clutter cancellation and moment calculations," in *3rd European Conf. on Radar in Met. and Hydro.* Atlanta, GA: Amer. Met. Soc., 2004, pp. 67–73.

[38] Y. Li, G. Zhang, R. Doviak, L. Lei, and Q. Cao, “A new approach to detect ground clutter mixed with weather signals,” *IEEE Trans. Geosci. Remote Sens.*, vol. 51, no. 4, pp. 2373–2387, April 2013.

[39] Y. Li, G. Zhang, and R. Doviak, “Dual-scan and dual-polarization radar measurements to detect ground clutter mixed with weather signals,” in *Env. Inf. Proc. Tech.* Austin, TX: Amer. Met. Soc., January 9 2013.

[40] M. Yeary, J. Crain, A. Zahrai, C. Curtis, J. Meier, R. Kelley, I. Ivić, R. Palmer, R. Doviak, G. Zhang, and T.-Y. Yu, “Multi-channel receiver design, instrumentation, and first results at the National Weather Radar Testbed,” *IEEE Trans. Inst. Meas.*, vol. 61, no. 7, pp. 2022–2033, July 2012.

[41] M. Yeary, J. Crain, A. Zahrai, R. Kelley, J. Meier, Y. Zhang, I. Ivić, C. Curtis, R. Palmer, T.-Y. Yu, and R. Doviak, “An update on the multi-channel phased array weather radar at the National Weather Radar Testbed,” in *2011 IEEE Radar Conference*, Kansas City, MO, May 2011, pp. 971–973.

[42] H. Van Trees, *Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory*. New York, NY: John Wiley and Sons, Inc., 2002.

[43] D. Hélal, M. Crochet, H. Luce, and E. Spano, “Radar imaging and high-resolution array processing applied to a classical VHF-ST profiler,” *J. Atmos. and Solar-Terrestrial Physics*, vol. 63, no. 2, pp. 263–274, 2001.

[44] C. Curtis, M. Yeary, and R. Palmer, “Using the multi-channel receiver to study sidelobe cancellation on the National Weather Radar Testbed,” in *28th Conf. on Interactive Information and Processing Systems (IIPS) for Met., Oceano., and Hydro.* New Orleans, LA: Amer. Met. Soc., January 2012.

[45] J. L. Lake, M. Yeary, and C. D. Curtis, “Multichannel nullforming at the National Weather Radar Testbed,” in *2015 IEEE Radar Conference*, May 2015, pp. 1072–1077.

[46] B. D. Van Veen and K. M. Buckley, “Beamforming: A versatile approach to spatial filtering,” *IEEE assp magazine*, vol. 5, no. 2, pp. 4–24, 1988.

[47] S. P. Applebaum and D. J. Chapman, “Adaptive arrays with main beam constraints,” *IEEE Trans. Antennas and Propagation*, vol. 24, no. 5, pp. 650–662, Sept. 1976.

[48] J. Capon, “High-resolution frequency-wavenumber spectrum analysis,” *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.

[49] I. S. Reed, J. D. Mallett, and L. E. Brennan, “Rapid convergence rate in adaptive arrays,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-10, no. 6, pp. 853–863, Nov 1974.

[50] O. Frost, "An algorithm for linearly constrained adaptive array processing," *Proc. of the IEEE*, vol. 60, no. 8, pp. 926–935, August 1972.

[51] R. Monzingo and T. Miller, *Introduction to Adaptive Arrays*, 1980.

[52] B. Widrow, P. Mantey, L. Griffiths, and B. Goode, "Adaptive antenna systems," *Proceedings of the IEEE*, vol. 55, no. 12, pp. 2143–2159, 1967.

[53] A. P. Liavas and P. A. Regalia, "On the numerical stability and accuracy of the conventional recursive least squares algorithm," *IEEE Transactions on Signal Processing*, vol. 47, no. 1, pp. 88–96, 1999.

[54] M. A. Richards, *Fundamentals of Radar Signal Processing*. Tata McGraw-Hill Education, 2005.

[55] C. D. Curtis, M. Yeary, and J. L. Lake, "Adaptive nullforming to mitigate ground clutter on the National Weather Radar Testbed phased array radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 3, pp. 1282–1291, March 2016.

[56] Z. Tian, K. L. Bell, and H. V. Trees, "A recursive least squares implementation for LCMP beamforming under quadratic constraint," *IEEE Trans. Signal Proc.*, vol. 49, no. 6, pp. 1138–1145, June 2001.

[57] M. A. Woodbury, "Inverting modified matrices," *Memorandum report*, vol. 42, no. 106, p. 336, 1950.

[58] C. Fulton, M. Yeary, D. Thompson, J. Lake, and A. Mitchell, "Digital phased arrays: Challenges and opportunities," *Proceedings of the IEEE*, vol. 104, no. 3, pp. 487–503, March 2016.

[59] L. E. Brennan and L. S. Reed, "Theory of adaptive radar," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-9, no. 2, pp. 237–252, March 1973.

[60] L. L. Horowitz, H. Blatt, W. G. Brodsky, and K. D. Senne, "Controlling adaptive antenna arrays with the sample matrix inversion algorithm," *IEEE Transactions on Aerospace and Electronic Systems*, vol. AES-15, no. 6, pp. 840–848, Nov 1979.

[61] A. Farina, "Digital equalisation in adaptive spatial filtering for radar systems: a survey," *Signal Processing*, vol. 83, no. 1, pp. 11–29, 2003.

[62] H. L. Van Trees, *Optimum array processing: Part IV of detection, estimation, and modulation theory*. John Wiley & Sons, 2004.

[63] Z. L. Yu, W. Ser, M. H. Er, Z. Gu, and Y. Li, "Robust adaptive beamformers based on worst-case optimization and constraints on magnitude response," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2615–2628, July 2009.

[64] A. Mandal and R. Mishra, "Digital equalization for cancellation of noise-like interferences in adaptive spatial filtering," *Circuits, Systems, and Signal Processing*, vol. 36, no. 2, pp. 675–702, 2017.

[65] J. R. Johnson, A. J. Fenn, H. M. Aumann, and F. G. Willwerth, "An experimental adaptive nulling receiver utilizing the sample matrix inversion algorithm with channel equalization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 39, no. 5, pp. 798–808, May 1991.

[66] Y. Rockah and P. Schultheiss, "Array shape calibration using sources in unknown locations–part i: Far-field sources," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 3, pp. 286–299, March 1987.

[67] G. A. Fabrizio, D. A. Gray, and M. D. Turley, "Using sources of opportunity to compensate for receiver mismatch in HF arrays," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 1, pp. 310–316, Jan 2001.

[68] A. G. Stove, "Calibration of active arrays using signals of opportunity," in *IEEE Seminar on Calibration of Active Phased Array Antennas*, March 2005, pp. 1–2.

[69] A. Kintz and I. J. Gupta, "Airborne antenna array calibration with signals of opportunity," in *2013 IEEE Antennas and Propagation Society International Symposium (APSURSI)*, July 2013, pp. 1264–1265.

[70] K. Lauritzen, H. Krichene, and S. Talisa, "Hardware limitations of receiver channel-pair cancellation ratio," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 48, no. 1, pp. 290–303, Jan 2012.

[71] M. Yeary, C. Fulton, R. Palmer, J. Salazar, and H. Sigmarsson, "Update on the all-digital phased array radar Horus program at the Advanced Radar Research Center at OU," in *Government Microcircuit Applications & Critical Technology Conference*. Miami, FL: GOMAC, March 2018.

[72] J. Salazar, J. Diaz, J. Ortiz, C. Fulton, H. Sigmarsson, M. Yeary, and R. Palmer, "Update on an ultra low-cross-polarization S-band active array antenna for a fully digital polarimetric phased radar system," in *IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*. Boston, MA: IEEE, July 2018.

[73] J. Lake, M. Yeary, and R. Palmer, "Real-time digital equalization to enhance element-level digital beamforming," in *2019 IEEE Radar Conf.*, Boston, MA, 2019.

[74] J. Lake and M. Yeary, "Real-time digital equalization: A first step in the calibration process of element level digital beamforming," in *Env. Inf. Proc. Tech.* Phoenix, AZ: Amer. Met. Soc., January 2019.

[75] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[76] E. Jones, T. Oliphant, P. Peterson *et al.*, "SciPy: Open source scientific tools for Python," 2001–, [Online; accessed May 6, 2019]. [Online]. Available: http://www.scipy.org/

[77] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[78] H. L. Southall, J. A. Simmers, and T. H. O'Donnell, "Direction finding in phased arrays with a neural network beamformer," *IEEE Transactions on Antennas and Propagation*, vol. 43, no. 12, pp. 1369–1374, Dec 1995.

[79] K. Toh and C. Lee, "Efficient network training for DOA estimation," in *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, vol. 5, Oct 1999, pp. 383–388 vol.5.

[80] M. Agatonovic, Z. Stankovic, I. Milovanovic, N. Doncov, L. Sit, T. Zwick, and B. Milovanovic, "Efficient neural network approach for 2D DOA estimation based on antenna array measurements," *Progress In Electromagnetics Research*, vol. 137, pp. 741–759, 2013.

[81] Q. T. Zhang, "A statistical resolution theory of the beamformer-based spatial spectrum for determining the directions of signals in white noise," *IEEE Transactions on Signal Processing*, vol. 43, no. 8, pp. 1867–1873, Aug 1995.

[82] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation*, vol. 34, no. 3, pp. 276–280, March 1986.

[83] J. M. Lewis, S. Lakshmivarahan, and S. Dhall, *Dynamic data assimilation: a least squares approach*. Cambridge university press, 2006, vol. 13.