Passive Direction Finding

A Phase Interferometry Direction Finding System for an Airborne Platform

A Major Qualifying Project
submitted to the faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree of Bachelor Science

Submitted by:

Daniel Guerin

Shane Jackson

Jonathan Kelly

Submitted to:

Project Advisors:

Professor Edward A. Clancy

Professor George T. Heineman

Professor Germano Iannacchione

Project Supervisors:

Lisa Basile, MIT Lincoln Laboratory

Kelly McPhail, MIT Lincoln Laboratory

Christopher Strus, MIT Lincoln Laboratory

October 10, 2012

This work is sponsored by the Department of the Air Force under Air Force Contract #FA8721-10-C-0007. Opinions, interpretations, conclusions, and recommendations are those of the authors and not necessarily endorsed by the United States Government.

Abstract

This paper describes the development of a phase interferometry direction finding system for an airborne platform developed for MIT Lincoln Laboratory. A phase interferometer uses the phase difference to determine the Angle of Arrival (AoA) of a received signal, but is unable to distinguish phase differences of more than one period, giving rise to phase ambiguities. The team utilized three antennas to resolve phase ambiguities and was able to determine the azimuthal AoA for a received electromagnetic signal in the X band to within $\pm 0.1^\circ$ in simulations including realistic noise models for a 170° field of view. A prototype was implemented using an FPGA-based board for data acquisition connected via USB to a PC for analysis, which connected to another PC via a TCP connection for tracking and display. The hardware was only able to utilize two channels. This limitation resulted in ambiguous solutions in AoA calculations. The team developed a graphical user interface for the system to display results to a system operator.

Acknowledgments

The authors would like to profess their gratitude towards the following people:

Our WPI Advisors:
Professor Ted Clancy, Professor George Heineman and Professor Germano Iannacchione.

Our MIT Lincoln Laboratory Supervisors:
Lisa Basile, Kelly McPhail and Christopher Strus.

As well as Emily Anesta, Sarah Curry, Chris Massa, Dennis Roux and all of the other MIT Lincoln Laboratory staff that made this project possible.

Statement of Authorship

This project has been completed in partial fulfillment of the requirements for the degree of Bachelors of Science in the fields of Electrical and Computer Engineering, Computer Science, and Physics.

Daniel Guerin

- Primary Author: Background, Prototype Methods, Prototype Results
- Testing: Testing C algorithm, compare prototype with model, certainty testing in model
- Added signal pulsing, signal detection to MATLAB model
- Developed high-level flow of MATLAB model
- Wrote drivers to communicate with hardware in C
- Identifying and correcting dissimilarities between model and algorithm

Shane Jackson

- Primary Author: Executive Summary, Background, MATLAB Methods, MATLAB Results, Discussion
- Testing: MATLAB model
- Developed signal generation and Error Correction in MATLAB
- Identified and resolved errors in the MATLAB model

Jonathan Kelly

- Primary Author: Executive Summary, Introduction, Prototype Methods, Discussion
- Testing: GUI verification
- Added ambiguity removal to MATLAB model
- Primary developer for C algorithm
- Designed and developed GUI
- Setup TCP Communication between algorithm and GUI

Executive Summary

The United States Air Force (USAF) is constantly working to understand and address new technology that could threaten U.S. aircraft. Developing countermeasures to threats such as new weapons systems, electronic countermeasures, and air surveillance systems allows the USAF to keep the nation safe. The Tactical Defense Systems Group, Group 108, at MIT Lincoln Laboratory aids the USAF by performing flight, field, and laboratory testing and by developing and prototyping new systems and instruments. The purpose of our project was to model and test a passive direction finding algorithm and to create a prototype for laboratory evaluation and future integration with a larger airborne system.

Radar, short for Radio Detection and Ranging, uses electromagnetic waves to detect and locate distant objects. By measuring the time between the transmission of a wave and the return of its reflection, the radar system determines the distance to the target. Passive DF systems receive and analyze signals from external emitters. By not transmitting, passive systems allow the user to obtain information about emitters without revealing themselves. The information can be used to warn the operator, map radar locations, or direct countermeasures. In order to fulfill the needs of Lincoln Laboratory, our DF system had the following requirements: $\pm 2.5°$ of accuracy, $90°$ field of view on the azimuth plane in front of the aircraft ($\pm 45°$), 40 dB dynamic range, and total processing time of under one second. The system processes X band radar pulses (8-12 GHz) on a 100 MHz intermediate frequency (IF) band. The system monitors the 4 GHz band 100 MHz at a time. In order to ensure that our system could meet these requirements, our group first constructed a MATLAB model of the system. This model allowed us to verify and test our phase interferometry DF implementation before converting it to a C algorithm and performing tests with live data.

The three main techniques used in passive direction finding (DF) are time difference of arrival (TDOA), amplitude comparison, and phase interferometry. Each method was simulated by a previous WPI student group, The Beacon Locator Project, in 2011 (Silva et al., 2011). All three methods of DF measure differences in the signals received at two or more separated antennas. The TDOA method measures the difference in arrival time of one signal at multiple antennas to calculate the Angle of Arrival (AoA) and range of the emitter. The degree of accuracy of the TDOA method depends on the distance between the antennas. The system's accuracy requirements would require antenna spacing on the order of kilometers using existing technology. The airborne platform that the system is intended for can support separations of at most 10 to 20 meters making the TDOA method impractical for our application. The amplitude comparison method uses two directional antennas pointed in different directions that the ratio of the gains for the two antennas will be unique for each angle within the field of view. By comparing the amplitude ratio of the signals received by the two antennas to known gain patterns,

the system calculates the AoA. The Beacon Locator Project found that the amplitude comparison method could not determine the AoA within ±2.5° across a ±45° field of view with a 40 dB range, the limits specified by Lincoln Laboratory. Phase interferometry is the third method for determining AoA and was the method used in this project. This method, like TDOA, relies on the time delay of the arrival of the signal between two or more antennas. Instead of measuring the time of arrival, phase interferometry measures the difference in signal phase between antennas. Unlike the time difference, the phase difference can be measured accurately over short distances. A 12 GHz signal undergoes a 2π phase change in less than five centimeters. The ability to accurately measure the phase difference makes the phase interferometry method more accurate than the TDOA method on airborne platforms. A two antenna phase interferometer is shown in Figure 1.

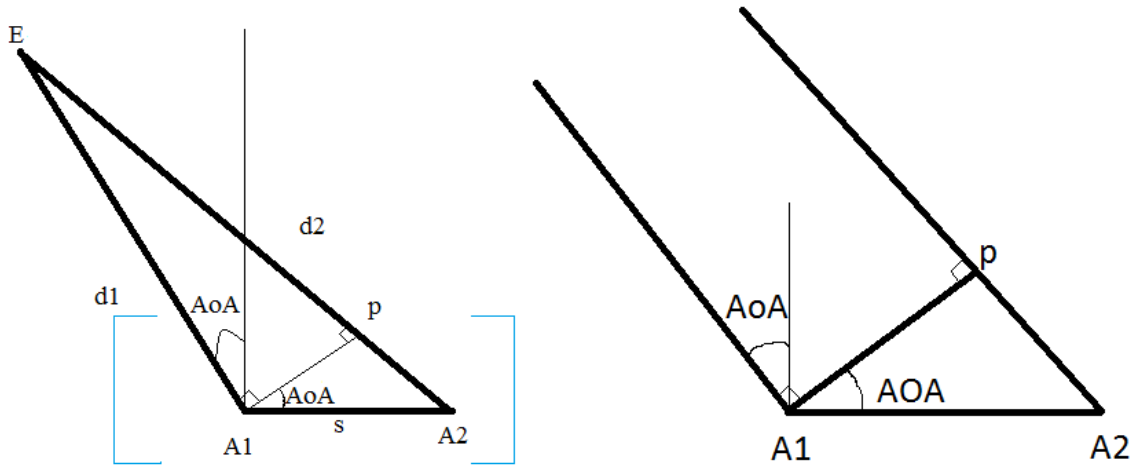


Figure 1: An interferometer system. (Left) An emitter distance d1 and d2 from two antennas separated by a distance s. Here, s is much less than d1 and d2. (Right) The two antenna system close to the antennas where d1 and d2 are effectively parallel.

The extra distance that the wave has to travel to reach antenna A2, line segment $\overline{pA2}$ in Figure 1, is a function of the phase difference ($\Delta\phi$), as shown in Equation 1:

$$\overline{pA2} = \lambda(\frac{\Delta\phi}{2\pi} + I), \tag{1}$$

where λ is the wavelength of the signal and I can be any integer value greater than or equal to 0. I is the number of full wavelengths the signal goes through along $\overline{pA2}$. When d1 and d2 are much larger than s, d1 and d2 can be assumed parallel in the vicinity of s and the points A1, A2, and p form a right triangle. Using these assumptions, the system calculates the angle of arrival as shown in Equation 2:

$$sin(AoA) = \frac{\lambda(\frac{\Delta\phi}{2\pi} \pm I)}{s}. \tag{2}$$

The accuracy of the interferometer system increases with antenna separation; however, increasing antenna separation leads to multiple results for the AoA calculation. The interferometer only measures phase between $-\pi$ and π so the phase measurement for different values of I are indistinguishable. Ambiguities begin occurring when the antenna separation is more than one half the wavelength of the signal.

The interferometer created in this project consists of three horn antennas placed in a horizontal line perpendicular to the direction of flight. The configuration is shown in Figure 2. The antenna spacing between antennas 2 and 3 is larger than that between

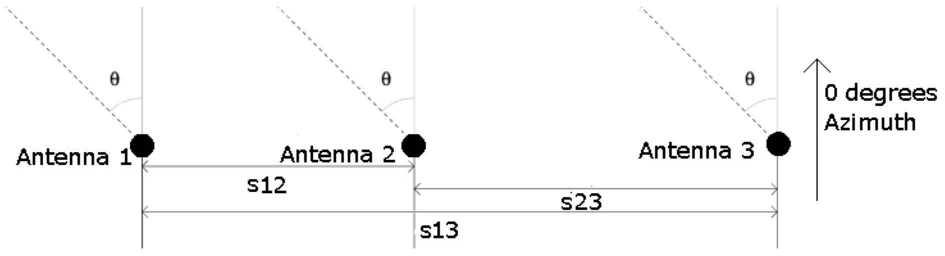
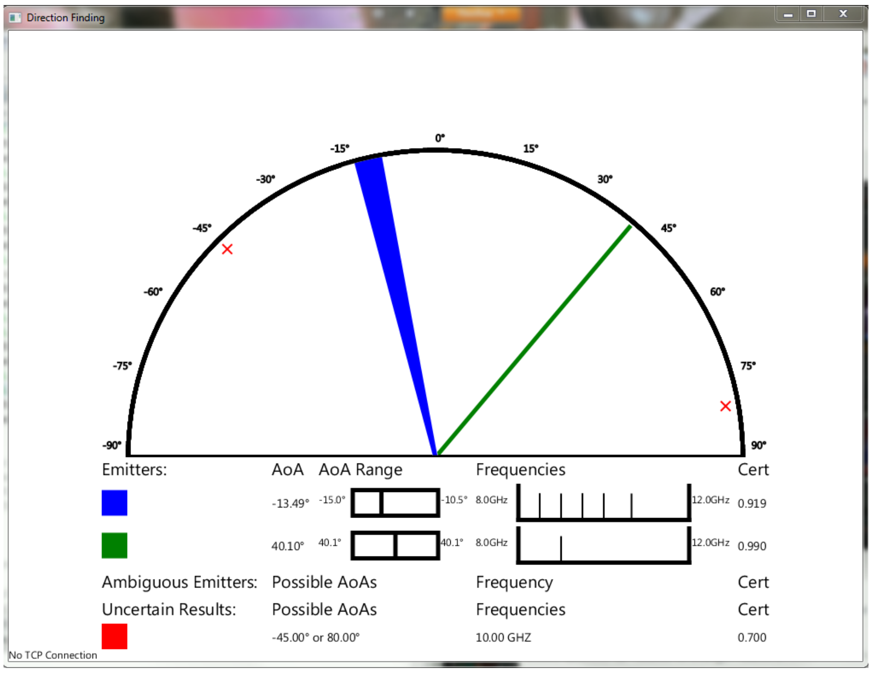


Figure 2: Setup of a three antenna interferometer.

antennas 1 and 2. The phase difference is calculated between antennas 1 and 2 and antennas 1 and 3. The two calculated phase differences are compared to determine an unambiguous AoA. The final calculation is performed using the phase difference between antennas 1 and 3 as the larger separation produces a more accurate result. Each of the antennas connects to a down converter. The down converter reduces the frequency of the signals from the radio frequency (RF) of 8-12 GHz to the IF of 15-115 MHz. The IF signals are easier to analyze and can be sampled by the analog to digital converters (ADC). These hardware components were provided by our Lincoln Laboratory supervisors. The sampled signals are then sent over a USB connection to our C signal processing algorithm running on a Windows 7 PC. The algorithm first detects if a signal is present then determines phase difference, frequency, and angle of arrival. The result of the calculation is sent over a TCP connection to another PC running the graphical user interface (GUI). The GUI, written in Java, performs error checking based on the history of angle results and then displays the results as shown in Figure 3.

The system was tested in both MATLAB and C to ensure that it would meet our requirements. In the presence of noise the system may not be able to disambiguate the angle. There are two types of error in our system: error due to noise and ambiguity error due to incorrectly resolved angles. To reflect the possibility of ambiguity error the system reports both of the two most likely ambiguous angles and a certainty value between 0.5 and 1.0 demonstrating how certain the system is of the most likely angle. A graph showing the AoAs calculated by the MATLAB model across the range of angles $-90°$ to $90°$ with worst case scenario parameters is shown in Figure 4. The worst case system parameters are: signal to noise ratio of 20 dB, carrier frequency of 12 GHz, and



Emitters:	AoA	AoA Range			Frequencies			Cert
🟦	-13.49°	-15.0°		-10.5°	8.0GHz		12.0GHz	0.919
🟩	40.10°	40.1°		40.1°	8.0GHz		12.0GHz	0.990
Ambiguous Emitters:	Possible AoAs				Frequency			Cert
Uncertain Results:	Possible AoAs				Frequencies			Cert
🟥	-45.00° or 80.00°				10.00 GHZ			0.700

Figure 3: Sample GUI display. The blue and green lines represent different emitters. The narrow green line shows one result, while the thick blue line shows a range of results close together. The red Xs show the possible angles for an uncertain result. An uncertain result is a signal for which the system was unable to confidently resolve the ambiguity. Each portion of the top display has a color coded description in the table.

emitter distance of 1 km. Under worst case conditions, the first angle reported by the system is correct 99% of the time. Figure 4 contains 181 samples, two of which, when the true AoA was 5° and 50°, the system resolved incorrectly and provided the incorrect angles as the most likely AoA. The certainties corresponding to these cases were 0.63 and 0.64 respectively and the second reported angle was correct, so the GUI had a chance to resolve both cases.

For all angles within ±85°, the system met the accuracy requirement. The system had ambiguity errors and resolved the wrong AoA approximately 1% of the time in worst case conditions. The errors in ambiguity resolution are infrequent enough to consider the system successful in meeting the accuracy requirement across the full field of view and dynamic range.

Analyzing the same signals with the C algorithm and the MATLAB model yields angles that are different by less than 10^{-4} degrees in every case and certainties which differ by less than 10^{-4} likely due to round off errors. The strong agreement between the results provided confidence that the C algorithm performed as well as the verified MATLAB simulation. In addition to the tests with MATLAB data the prototype system was tested with two signal generators producing IF signals. Only two signal generators were used as a third ADC was not available for the third input channel. The system was not able to resolve ambiguities without the third input, but the tests verified that

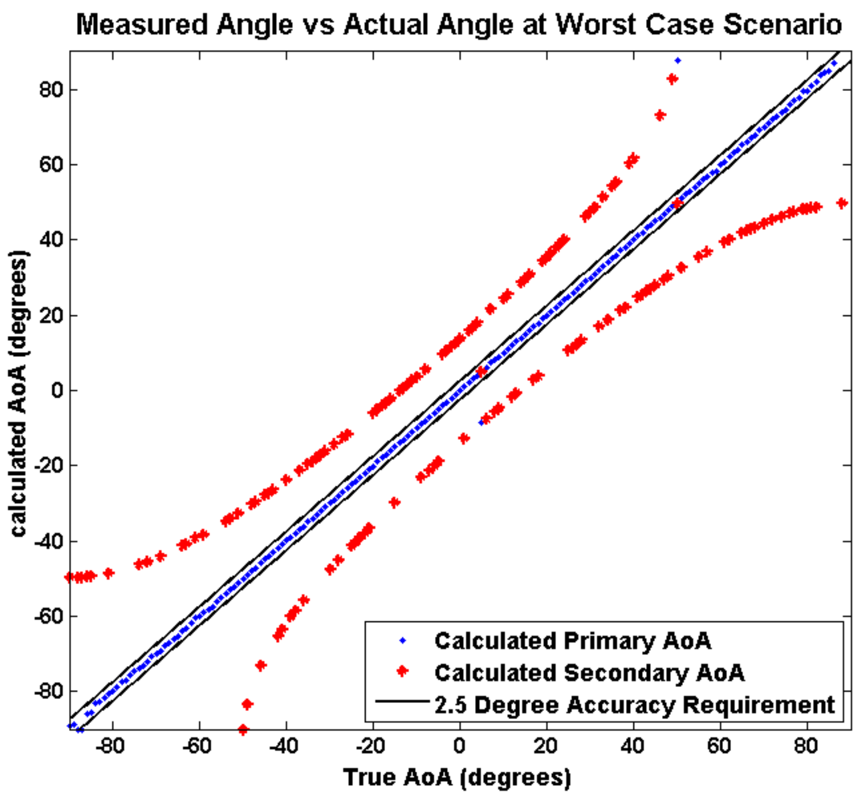


Figure 4: The primary (blue diamonds) and secondary (red stars) reported AoAs plotted against the known true angle with 1° steps in worst case conditions. In every case except for two, where the true AoAs were 5° and 50°, the most likely of the two angles reported by the system was correct. The black lines show the boundaries of the required ±2.5° accuracy.

the system would meet the accuracy requirement on live data if the angle was resolved correctly.

There are several extensions to this project that would improve the functionality of the system. The system should be adjusted by adding the ability to distinguish multiple emitters in a single pulse. Currently, the system only processes the strongest of the signals rather than looking for radar signals present at multiple frequencies. To enhance the speed of the system, the direction finding algorithm should be implemented on a field-programmable gate array (FPGA) or on an embedded system because communication is the primary cause of delay in the system. Implementing the algorithm on an FPGA would likely require changes to operate on fixed point data unlike the current system, which uses double-precision floating point data. Changing the GUI display to allow for user configuration of display parameters would make it more useful and interactive. Testing the third input channel with live data and the entire system with true antenna data would further validate the performance of the system. With or without these improvements, the prototype system created by this project is a valuable addition to the larger system being developed by Group 108.

Contents

1 Introduction

Since World War II, radar use has become widespread in civilian and military service. Used primarily to determine information about distant objects, such as vehicles and aircraft, a radar system emits electromagnetic waves and analyzes the returning reflections to determine the location and velocity of the object. Militaries around the world responded to the development of the radar with a variety of countermeasures. Examples of countermeasures include disrupting radar by emitting pulses to overwhelm and confuse the radar and dropping chaff so that the signal hits clutter instead of the plane. These countermeasures are aided by passive direction finding (DF) systems which determine the presence and direction of emitting sources (Wiley, 1985).

Although passive direction finding is well known and documented, Lincoln Laboratory and the United States Air Force require their own direction finding system which will be integrated into a larger system on an airborne platform. By developing their own DF system, Lincoln Laboratory can easily expand and alter the system. Purchasing an existing DF product may reduce development cost; however, it would not be optimized for the larger system.

The purpose of this project was to create a prototype airborne direction finding system that determines the Angle of Arrival (AoA) of incoming signals and displays that information on a graphical user interface (GUI). For the sake of simplicity, the system is only concerned with the AoA in the azimuth plane rather than both the horizontal angle (azimuth) and the angle of elevation, and the effects of aircraft tilt are ignored. Passive direction finding is performed using interferometry. Interferometry is the practice of comparing characteristics, such as frequency and phase, of two or more signals in order to gain information about the waveforms. Passive DF systems utilize interferometry on the same signal received in multiple locations to determine information about its source. The resolution of interferometers improves as antenna separation increases; however, the spatial constraints imposed by an airborne platform require our system to function with antennas approximately 5 cm by 7.5 cm, placed only a few meters apart. As a result there are only slight time differences between them. Utilizing these small differences to provide highly accurate AoAs was the primary challenge of this project.

In 2011, another student team from WPI compared and contrasted three methods of direction finding: time difference of arrival (TDOA), amplitude comparison, and phase comparison. Their project, titled the Beacon Locator Project, implemented the amplitude comparison method, but was not able to fully meet the system requirements (Silva et al., 2011). These requirements were determining the AoA within $\pm 2.5°$, a $\pm 45°$ field of view in front of the aircraft, a dynamic range of 40 dB, and to provide at least one update or calculation per second. The system must operate on signals in the X band frequency range of 8 to 12 GHz, utilizing an intermediate frequency (IF) with 100 MHz bandwidth.

Secondary objectives were expanding the field of view to 180° and simultaneously tracking multiple beacons. Our project implemented a system which used phase interferometry to fulfill these requirements.

The project produced three deliverables: a MATLAB model of the DF system, a C algorithm hosted on a PC which communicated with the hardware provided by our Lincoln Laboratory supervisors and calculated, logged, and transmitted the AoA based on data received from the hardware, and a graphical user interface (GUI) that communicated with the C algorithm over an Ethernet connection. The MATLAB model provided us with an opportunity to easily prototype and finalize our DF algorithm before translating the algorithm to C. The simulation covered the full DF system from antennas to processor output. The front-end hardware required by the system to receive a signal and prepare it for processing by the software was comprised of: antennas, down converters, analog-to-digital converters (ADCs), and a field-programmable gate array (FPGA) based digital signal processor. The hardware was provided to the group by our Lincoln Laboratory supervisors. The final prototype of the system consisted of the front end hardware provided by Lincoln Laboratory, a computer running the C algorithm to retrieve and process the incoming data from the hardware, and another computer running the GUI and displaying the information.

2 Background

In order to meet the requirements for the project, the team researched a number of topics to obtain background knowledge of radar systems and direction finding techniques. The following sections contain an overview of radar systems, focused on radar receivers, direction finding systems and techniques, and a discussion of methods to solve ambiguities that result from phase comparison direction finding.

2.1 Radar Overview

Radar systems have a myriad of uses, from radar speed detectors and air traffic control systems to missile tracking and surface mapping. Regardless of the application, all radar systems share the same general concepts and structure. A radar system is composed of several basic components: a transmitter, an antenna or antenna array, a receiver, and a signal processor (Holm and Richards, 2010). The basic functionality of a radar system is shown in Figure 5.

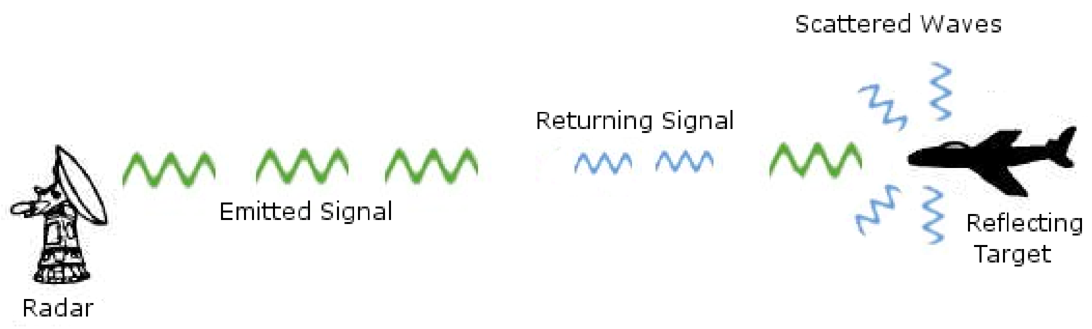


Figure 5: A basic depiction of a radar system detecting an aircraft. The emitted signal (large green waves) hits the aircraft and scatters in every direction (smaller blue waves). Some of the scattered signal returns to the emitter.
Modified from electriciantraining.tpub.com/14190/img/14190_14_1.jpg and www.srh.noaa.gov/jetstream/doppler/how.htm.

The transmitter emits a structured electromagnetic wave depicted by the large green wave in Figure 5. The electromagnetic wave is generated by creating an initial signal, called a baseband signal, with a frequency between 0 Hz and a specified maximum cutoff frequency. The baseband signal is then modulated with a sine wave. The modulation process shifts the signal to the intermediate frequency. In general, modulating the signal in this way doubles its original bandwidth, however single-sideband modulation, which uses a filter to remove the images of the frequency-shifted signal, can be used to prevent this effect. Once the signal is at IF, it is modulated once again to bring the signal up to radio frequency (RF) (Skolnik, 2001). The two step modulation process allows a radar to change its RF, called frequency hopping, without changing the IF used to process signals. Processing at IF allows radars to use simpler, less expensive hardware in both the transmitter and receiver.

The wave generated by the transmitter travels through space until it hits a target. When the wave hits a target, it scatters off the object, sending waves in every direction (the small blue waves in Figure 5). A small portion of the scattered reflections return to the radar system via the receiver antenna. A large portion of the signal's power is lost in transit and from the reflection scattering, so the receiver's main function is to identify and amplify these relatively low energy signals. The first step is to reduce the frequency of the signal to the IF so that it is easier to process. ADCs can sample the IF signals for digital processing while RF signals in the X band would require analog processing. The signal is then sent to a signal processor, either directly to an analog processor or through an ADC to a digital processor. Regardless of the processing type, the signal processor analyzes the signal to determine characteristics of the target. While most radars determine range, some also determine a number of other characteristics such as angle to target, location of target, and radial velocity.

The techniques used to calculate information gathered by radars depend on the type of wave sent by the transmitter. Two main types of electromagnetic waves used by radars are continuous waves (CW) and pulsed waves. Continuous wave systems continuously emit a sinusoidal wave. When the radar system detects a return signal, the signal is compared to the original transmitted wave to determine target characteristics. The change in amplitude, phase, and frequency all give information about the target hit by the wave. Pulsed radars do not transmit continuously. Instead, these radars transmit a series of short sinusoidal signals. By only sending pulses, pulsed radar systems require less power than their CW counterparts and do not have issues of interference between a continuously emitted signal and returning signal. A sample of a pulsed radar signal is shown in Figure 6. The time spent emitting continuously is the pulse width (PW), and the pulse width plus the time between pulses is called the pulse repetition interval (PRI). The PW, PRI, and frequency allow systems to distinguish one radar system from another as opposed to just the frequency in the case of CW. Using the assumption that the signal travels at the speed of light, the radar system calculates the range to a target by using the time delay between the transmission of the pulse and the arrival of the reflected signal. If the PRI is shorter than the time difference between the return signal and emission time, the radar will incorrectly determine that the target is in close proximity (Skolnik, 2001). To avoid the potential ambiguity, the PRI must be greater than the time it takes a pulse to travel to and return from a target at maximum range. Consequently, determining the maximum range in relationship to the PRI is essential for radar design.

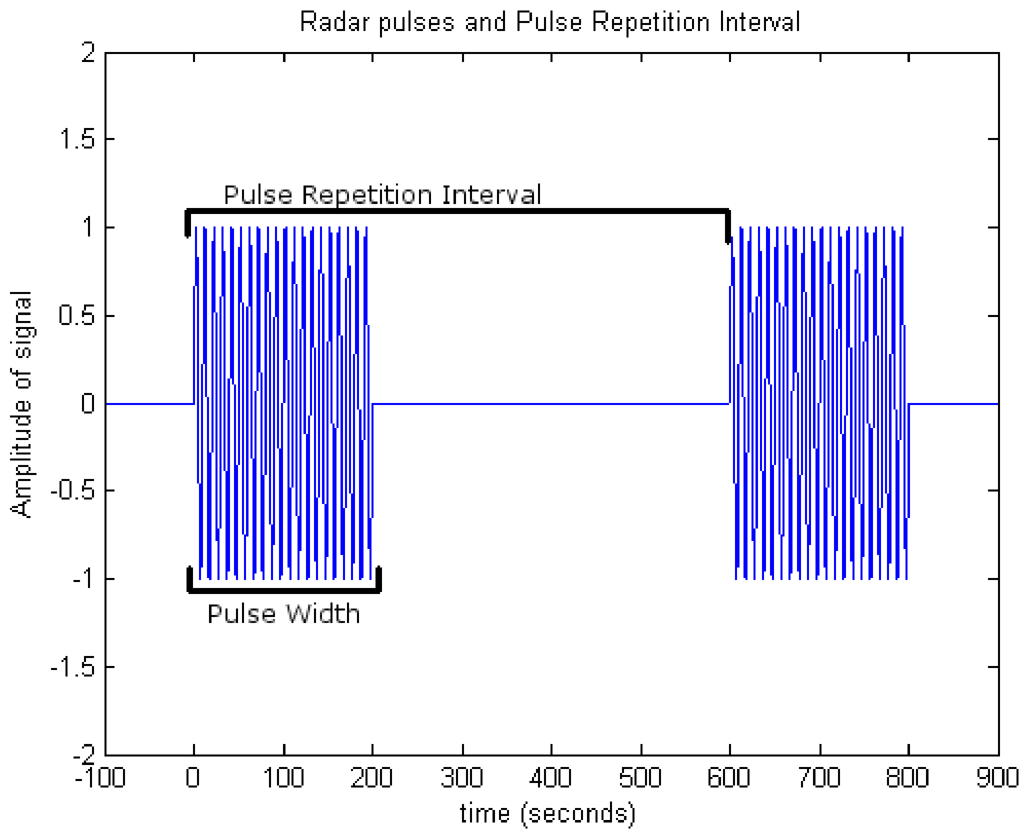


Figure 6: Two radar pulses with the same pulse width of 100 ms and a PRI of 400 ms. The first pulse arrives at t=0 and the second pulse arrives at t=400 ms.

2.1.1 Radar Equation

The effectiveness and range of a radar system (R_{max}) depends on many factors as shown in the radar range equation:

$$R_{max}^4 = \frac{P_t G A_e \sigma}{(4\pi)^2 k T_0 B F_n \frac{S}{N}_{min}} \tag{3}$$

(Holm and Richards, 2010). Equation 3 defines a radar system's maximum range in terms of transmitted power (P_t), transmitting antenna gain (G), receiving antenna effective aperture (A_e), radar cross section of target (σ), minimum signal to noise ratio ($\frac{S}{N}$), Boltzmann's constant (k), the IEEE standard temperature (T_0) in Kelvin, receiver half power bandwidth (B), and receiver noise figure (F_n). The range equation calculates the maximum range for which a signal can travel to a target, scatter off of it, and then return as a detectable signal. If the target is beyond this range, the returned echos will not be strong enough to register at the receiver. There are more complex versions of the radar equation that take into account other factors such as pulse data and weather clutter, but even this simplified view can be of great use when designing a radar system (Holm and Richards, 2010). The (R_{max}^4) in the radar equation is actually a combination of the transmit and return distances, $(R^2_{transmit})$ and (R^2_{return}). The passive DF system relies solely on the transmitted distance of the received wave and does not depend on σ. The range of a passive DF system is given by:

$$R^2_{transmit} = \frac{P_t G A_e}{4\pi k T_0 B F_n \frac{S}{N}_{min}}, \tag{4}$$

where A_e, B, and $\frac{S}{N}_m in$ are determined by the DF system.

2.2 Radar Receivers

This project's direction finding system is a radar receiver. Instead of receiving the echos of signals that it transmits, the DF system receives and process signals sent by other radar systems. The most common type of radar receiver is the superheterodyne receiver. Superheterodyne receivers are composed of five main components: antennas, down converters, ADCs, a signal detector, and a signal processor. A block diagram of a superheterodyne receiver is shown in Figure 7.

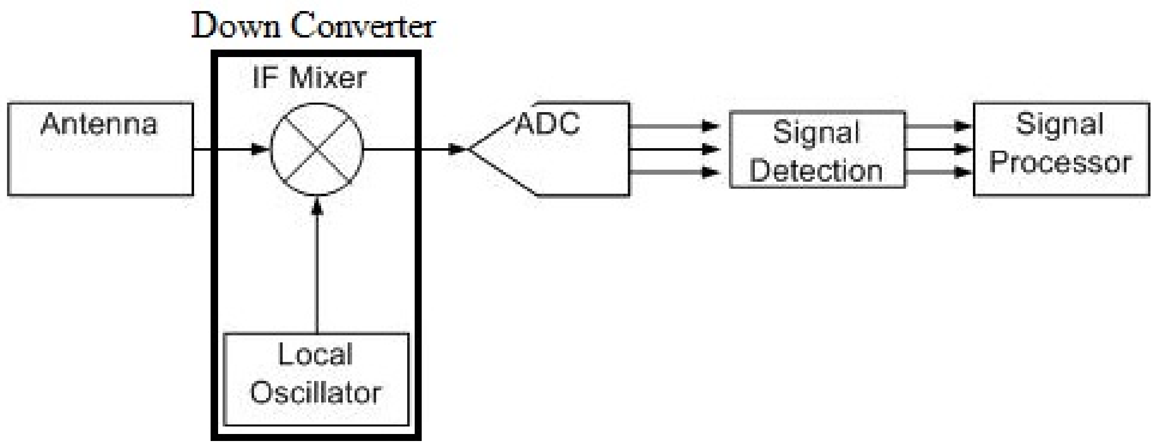


Figure 7: The setup of a superheterodyne receiver system. The antenna and local oscillator feed into the IF mixer. The mixer output is sampled by the ADC. If the signal detector finds a signal present, it is then passed off to the signal processor (Skolnik, 2001).

2.2.1 Antennas

The purpose of antennas in a DF system is to receive radar signals. Essential characteristics of a receiver antenna include the gain, the antenna area, the effective aperture, and the field of view. The gain is a measure of how well the antenna amplifies the power of signals it receives. The effective aperture describes the absorbing section of an antenna. A larger effective aperture increases the gain, thus increasing received signal power. Depending on the desired field of view, the system may use directional antennas or omnidirectional antennas. Directional antennas have different gains depending on the reception angle. The direction of maximum gain is the antenna's boresight (Holm and Richards, 2010). Omnidirectional antennas have equal gains in all directions, but if a smaller view is desired when using omnidirectional antennas, signals outside of the desired range of angles must be physically blocked to prevent noise from other directions (Skolnik, 2001).

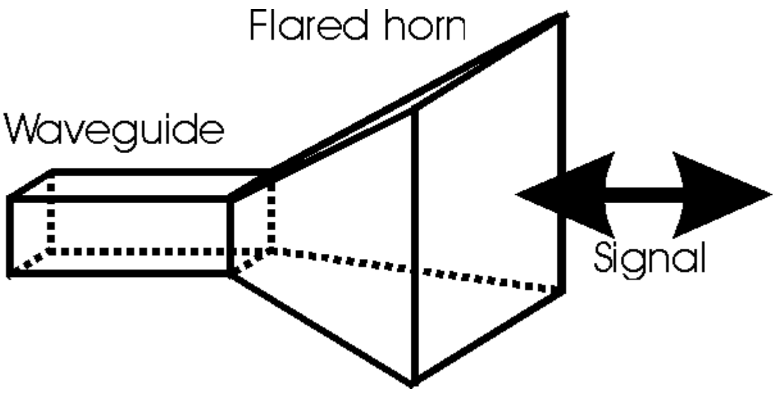


Figure 8: A typical horn antenna.
Source: www.radio-electronics.com/info/antennas/horn_antenna/horn_antenna.gif.

The horn antenna, shown in Figure 8, is a type of directional antenna commonly used to detect signals in the X band (8-12 GHz), the desired operating range for our system (Holm and Richards, 2010). A horn antenna is simply a waveguide which is flared out at one end like a horn. One advantage of the flare is that it provides a large opening to accept signals while at the same time blocking out signals from other directions. The cross-section of the flare can be adjusted as desired. Shrinking the flare increases directivity at the cost of gain (Bakshi, 2009). Horn antennas come in a wide variety of sizes, many of which are small enough to place on an aircraft.

2.2.2 Down Converter

High frequency signals, like those found in the X band, have smaller amplitudes for the same transmitted power and are difficult to sample and analyze for changes in phase. A down converter, which reduces the frequency of the incoming signal and applies gain, is used to allow an ADC to sample the signal accurately. To sample the signal without loss of information, the ADCs must sample at a rate greater than or equal to the Nyquist rate, which is defined as twice the signal's bandwidth. Sampling signals directly in the X band, which has a bandwidth of 4 GHz, would require an 8 GHz sampling rate. Current technology cannot sample this rapidly, so only subsections of the X band can be sampled at any given time. In order to monitor the entire X band, a system must sample and process data from a smaller bandwidth, for example 100 MHz, within X band and then repeat the process until every section of X band has been examined. Because each window is a different frequency, it becomes convenient to use a down converter to frequency shift the RF down to a known intermediate frequency. Having a known IF greatly aids in filter design because only one filter needs to be designed, instead of designing a separate filter for each section of the RF band that the system will examine. The down converter reduces the frequency through use of a mixer and a local oscillator. The local oscillator is configured to produce a sinusoidal signal with a frequency close to the expected carrier frequency of the signal and at least 7 dB greater than the largest signal the system intends to analyze (Holm and Richards, 2010). The mixer takes in two signals in the form of cosine waves: the signal from the antenna, $A_1cos(\omega_1 t)$, and a signal from the local oscillator, $A_2cos(\omega_2 t)$. These two signals are then multiplied together and transformed by combining the trigonometric identities for $cos(A+B)$ and $cos(A-B)$ yielding:

$$A_1A_2cos(\omega_1 t)cos(\omega_2 t) = \frac{A_1A_2}{2}cos((\omega_1 - \omega_2)t) + cos((\omega_1 + \omega_2)t). \tag{5}$$

The mixer outputs a signal with two frequency components, one at $(\omega_1 - \omega_2)$ and one at $(\omega_1 + \omega_2)$. The component at $(\omega_1 + \omega_2)$ is filtered out using a low pass filter leaving only the IF component at $(\omega_1 - \omega_2)$ (Wolff, 1997).

When the original frequency of the signal is unknown, using a local harmonic oscillator with a constant frequency only allows the system to monitor one subsection of the X band. Consequently, tunable local oscillators are used to sweep over the bandwidth. A tunable oscillator for X band begins at 8 GHz and steps up frequency in increments less than or equal to the IF bandwidth until it reaches 12 GHz. Between each step the system receives signals for a set period of time. The steps allow the receiver to process signals across the entirety of X band, but the slow increments give rise to the possibility of missed signals (Holm and Richards, 2010). If a signal arrives outside of the window the system is currently tuned for it will be filtered out in the down conversion process. In order to reconstruct the signal at its original RF, the local oscillator's frequency must be known by the rest of the system. To restore a down converted signal to its original form, the same mixing process is used with a high pass filter in lieu of a low pass filter.

2.2.3 Analog to Digital Converters

While radar systems can be made using only analog processing, most modern radar systems use digital signal processing to determine information about the received signals. The ADC samples the analog wave received by the antennas to form a digital signal that approximates the original signal. The two most important characteristics of an ADC are the sample rate and the resolution. Higher sample rates allow larger IF bands, which in turn reduce the probability of missing a signal. When sampling rates become significantly higher than the Nyquist rate, oversampling occurs. Oversampling results in a better signal to noise ratio (SNR). Oversampling also creates separation between aliases of the signal which allows filters to have more gradual cutoffs than when sampling at exactly the Nyquist rates where the alias begins where the signal ends (Candy and Temes, 1992). The resolution and dynamic range of an ADC increases with the number of bits. The ADC can only represent a number of values equal to 2^b, where b is the number of bits. If a set resolution is required, the range that can be represented is the required resolution times the number of possible values. As analog values do not have exact digital maps, they are rounded or truncated to the nearest match, causing a quantization error.

2.2.4 Signal Detection

In passive DF system, the sampled data from the ADC are passed to the signal detector. When no signal is detected, the signal detector does not pass the sample on to the rest of the system. By avoiding sending data devoid of a signal, the system does not waste computational resources on meaningless data. Ensuring the existence of a signal becomes significantly more difficult in the presence of noise. The simplest method of ignoring noise is setting a threshold voltage in the time domain and ignoring all samples with peak voltages below it. The value for the threshold voltage has to be chosen carefully. At lower threshold values, the false positive rate goes up as the system accidentally passes noise through; however, when the threshold is raised, the missed detection rate goes up because low power signals are mistaken for noise (Skolnik, 2001). Ideally, signal detectors would simultaneously minimize both false positives and missed detections. One way to intelligently set the detection threshold of the signal detector is monitoring returns in the absence of a signal to determine typical noise levels. This typical level can be multiplied by a predetermined constant in order to ensure a minimum false alarm rate. In order to maintain an appropriate value for the threshold noise in changing conditions, the system needs to continuously monitor the noise. This self-adjusting technique is called constant false alarm rate detection (Skolnik, 2001). Over time, conditions and noise levels change requiring the continuous monitoring of noise to maintain appropriate threshold values. If the signal detector mistakes a signal for noise then the noise threshold might become too large, causing repeated missed signals.

2.2.5 Signal Processor

Once a signal has been received and detected it is passed to the signal processor. The signal processor analyzes properties of the signal to determine characteristics of the signal, such as phase, frequency, and amplitude. A wide variety of tools and techniques have been developed to help extract this information from signals. Two common techniques in radar processing are in-phase and quadrature analysis and frequency domain analysis using the Fourier transform.

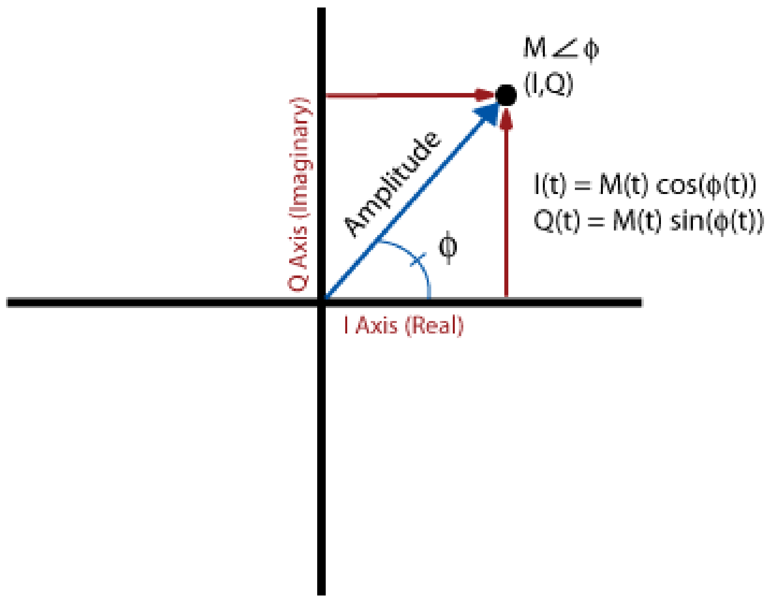


Figure 9: Polar and I-Q coordinates, where M is the magnitude of the wave, t is time, ϕ is the phase, I is the real component of the wave, and Q is the imaginary component.

In-phase and quadrature analysis (I-Q analysis) represents a real valued signal as a combination of real (I) and imaginary (Q) components. Having a signal split into real and imaginary parts reduces the complexity of determining frequency and phase (Smith, 2007). The general function for a real valued sine wave is given by Equation 6:

$$x(t) = Asin(2\pi ft + \delta), \tag{6}$$

where A is the amplitude, f is the frequency (Hz), and δ is the phase shift. In this form, the total instantaneous phase of the wave, $\phi(t) = 2\pi ft + \delta$, depends on δ. In order to calculate ϕ and A directly, the function must be altered. Using trigonometric identities Equation 6 can be rewritten as:

$$x(t) = Asin(\delta)cos(2\pi ft) + Acos(\delta)sin(2\pi ft) = A_1cos(2\pi ft) + A_2sin(2\pi ft), \tag{7}$$

where $A_1 = Asin(\delta)$ and $A_2 = Acos(\delta)$. The two components of the sinusoid are: the sine (in-phase) component, $I = A_2sin(2\pi ft)$, and cosine (quadrature) component, $Q = A_1cos(2\pi ft)$. The I and Q components are a conversion of polar coordinates to the Cartesian plane as shown in Figure 9. The relationship between I and Q allows for the calculation of instantaneous phase by taking the arctangent of the I and Q components as shown in Equation 8. The amplitude is calculated using the Pythagorean Theorem as shown in Equation 9:

$$\phi = tan^{-1}(Q/I) \tag{8}$$

$$A^2 = I^2 + Q^2. \tag{9}$$

Much like I-Q analysis, the Fourier transform represents the real signal as a combination of real and imaginary parts; however, the Fourier transform also changes the signal from the time domain to the frequency domain, showing which parts of the signal are present at each frequency. The Fourier transform is given by the equation:

$$X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt, \tag{10}$$

where $x(t)$ is the signal in the time domain, $X(\omega)$ is the signal in the frequency domain, t is the time, and ω is the frequency. The Fourier transform uses an integral across the entire time domain. The integral spans every point in time; however, in a digital system, data only exist for a finite time range and only at sampling intervals within that range. Therefore, digital systems use the discrete Fourier transform (DFT) which replaces the integral over all time with a sum of samples in a specified range. The DFT can be found by the equation:

$$X(k) = \sum_{n=0}^{N_0-1} x_n e^{-jk(\frac{2\pi}{N_0})n}, \tag{11}$$

where x_n is the nth sample of the signal in the time domain, $X(k)$ is the kth sample of the signal in the frequency domain, and N_0 is the number of samples taken. The transformation expresses the waveform as a series of harmonics, each with a specific amplitude, frequency, and phase. By comparing the magnitude of the different frequency components, the frequency or frequencies of the signal are determined. When the strength of a signal is large compared to the noise present, the frequency components of the signal have a higher magnitude than the frequency components of the noise. The DFT of a weak 33 Hz signal and a strong 80 Hz signal in the presence of noise is shown is shown in Figure 10. The high amplitudes at specific frequencies verify the presence of a signal. The dominant frequency of the signal, once determined, allows the system to calculate the phase associated with the signal at that frequency. Figure 11 displays the graphs of amplitude versus frequency and the phase versus frequency.

For practical applications, the DFT can be inconvenient to use because the number of computations required is O(N^2) where N is the number of samples used in the calculation. The accuracy of the DFT increases with the number of terms used because it becomes a closer approximation of the regular Fourier transform. Due to the potentially large number of calculations required for the DFT, many systems use fast Fourier transform (FFT) algorithms which take advantage of the linearity of the DFT to perform the DFT

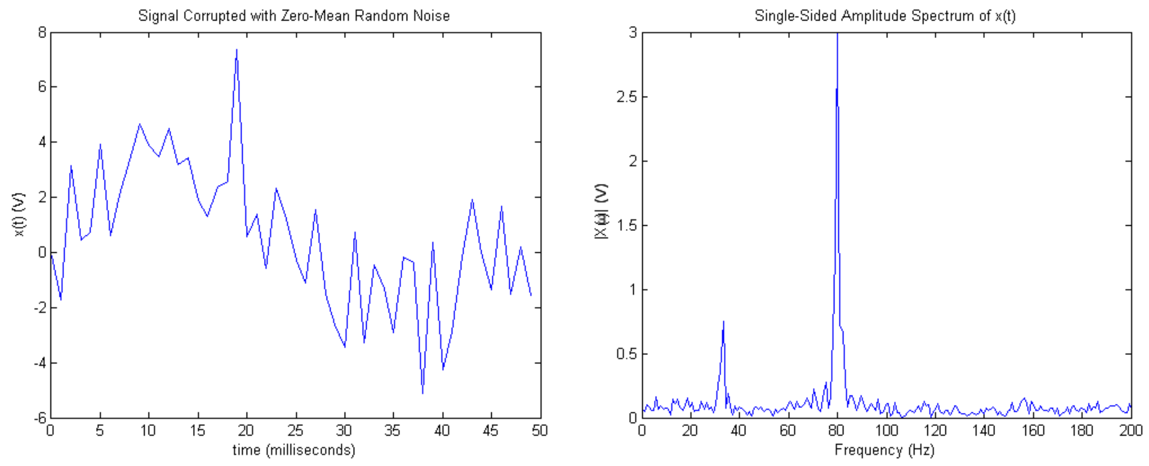


Figure 10: The received time-domain signal (Left) composed of a 33 Hz and 80Hz sine wave with Gaussian noise and its Fourier transform in the frequency domain (Right). This exaggerated case exemplifies the usefulness of the Fourier transform in isolating noise.

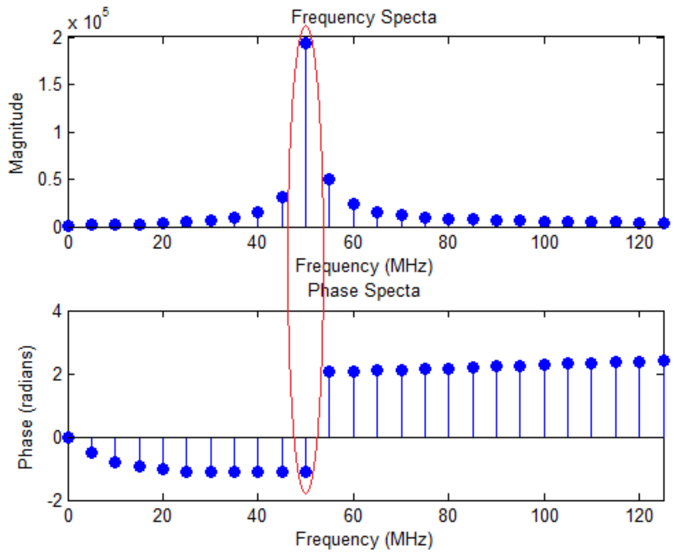


Figure 11: Frequency and phase spectra of a pulsed 50 MHz sine wave with 10 dB SNR. The phase and the magnitude at the signal's frequency are circled in red.

on a number of shorter signals and then combine the results. FFT algorithms reduce the number of computations needed to O(N$\log_2$N) in cases where N is a power of 2 (Lathi, 2005). These signal processing techniques simplify the process of extracting phase and frequency information that direction finding systems can use to calculate the AoA of a signal.

2.3 Passive Direction Finding

Passive radar systems do not transmit waves. Instead, they analyze waves sent by external emitters. The DF system does not have any knowledge of the signal's properties at transmission, so comparisons cannot be made between the transmitted and received signal to determine range or velocity. In spite of these limitations, there are still advantages to using a passive DF system in conjunction with or in lieu of conventional radar systems. Because passive DF systems do not transmit, they require significantly less power and do not reveal their location to other receivers. Even without transmitting, DF systems can aid in target recognition and determine direction to emitters. DF systems can identify the type of emitter by analyzing the received signal (Skolnik, 2001). DF systems are designed to determine characteristics of incoming signals in particular, their AoA. There are three main methods for determining the AoA of an incoming radar signal: time difference of arrival (TDOA), amplitude comparison, and phase interferometry. All three of these methods require a minimum of two antennas and yield better results with more antennas.

2.3.1 Time Difference of Arrival (TDOA)

The TDOA method relies on two or more antennas, positioned with a known geometry, and the difference in arrival time of the signal at each antenna to calculate the angle of arrival. In a two antenna system, combining the knowledge of the geometry of the antennas and the time difference allows the signal processor to calculate a curve of possible values for the position of the emitter (Wiley, 1985). An emitter and a two antenna TDOA system are shown in Figure 12. The distance formula provides the distance between two points, in this case the distance between the emitter and antenna 1:

$$d_1 = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2}, \tag{12}$$

where x_1, y_1 and z_1 are the coordinates of antenna 1 and x, y, z are the coordinates of the emitter. Noting that the speed of light, c, is effectively constant in the atmosphere at $3 * 10^8$ m/s, solving for the time of arrival (t_1, ToA) is straightforward:

$$t_1 = d_1/c. \tag{13}$$

The same relationship holds between the ToA at antenna 2, t_2, and the location of the second antenna, $d_2 = \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2}$. In practice, d_1, d_2, t_1, and t_2 are unknown; however, the measured time difference $(t_2 - t_1)$ allows the system to calculate the difference in radial distance from the emitter to each antenna, $d_{12} = d_1 - d_2$,

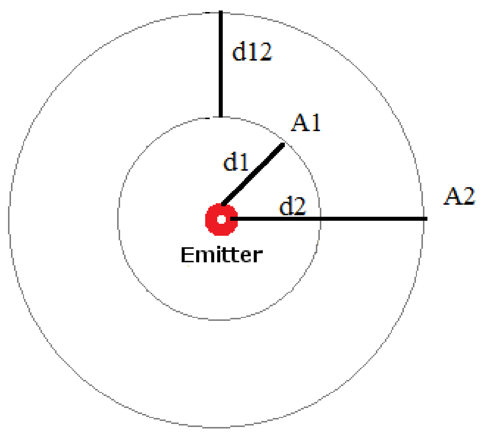


Figure 12: An example of a TDOA system. Antennas A_1 and A_2 are distance d_1 and d_2 away from the emitter. Each antenna sits on one of two concentric circles centered on the emitter. The radial distance between these circles is given by d_{12}.

using (13). Algebraic manipulation and using Equation 12 to replace d_1 and d_2 yields:

$$\sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2} = \sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2} + d_{12}. \tag{14}$$

Squaring both sides and performing more manipulation yields:

$$1 = \frac{(x_1^2 - x_{12}x - x_2^2 + y_1^2 - y_{12}y - y_2^2 + z_1^2 - z_{12}z - z_2^2)}{2d_{12}\sqrt{(x_2 - x)^2 + (y_2 - y)^2 + (z_2 - z)^2}}. \tag{15}$$

Squaring equation 15 gives the formula for a hyperboloid (Fang and Martin, 1990). The location of the emitter lies on some point (x,y,z) on this hyperboloid.

Adding a third antenna to the system allows the generation of another hyperboloid. The location of the emitter must be a point in both hyperboloids. The intersection of the two hyperboloids limits the possible values to a 2D curve. Additional antennas continue to reduce the ambiguity by adding further constraints on possible locations. The TDOA method of direction finding has the potential to be very accurate and yields location in addition to direction; however, its heavy reliance on accurate measurements of ToA from multiple antennas creates problems in small-scale systems. ToA accuracy in the time measurement determines the accuracy of the AoA calculation. Consequently, to calculate the AoA within the accuracy required for this project, the antennas need to be placed a distance on the order of kilometers apart. While the minimum distance may be feasible for ground-based systems, the Beacon Locator Report found that it is impractical for airborne systems (Silva et al., 2011).

2.3.2 Amplitude Comparison

The amplitude comparison method of determining AoA uses two or more directional antennas whose boresights are pointed in different directions (Skolnik, 2001). The number of antennas used depends on the desired field of view and resolution. The boresights are typically offset from each other such that the gain patterns overlap along the 3 dB edge (shown in Figure 13) (Wiley, 1985). When a signal arrives, the ratio of power amplitudes between the two antennas is compared to known antenna gain patterns which is then used to calculate the AoA. Since the power ratios are compared to known values, the AoA can only be resolved to one of the pre-stored values. The resolution of this method is proportional to the amount of memory dedicated to the table of known values; however, increasing the size of the table simultaneously increases the time required to search through that table. Noise increases error in the amplitude method more so than in

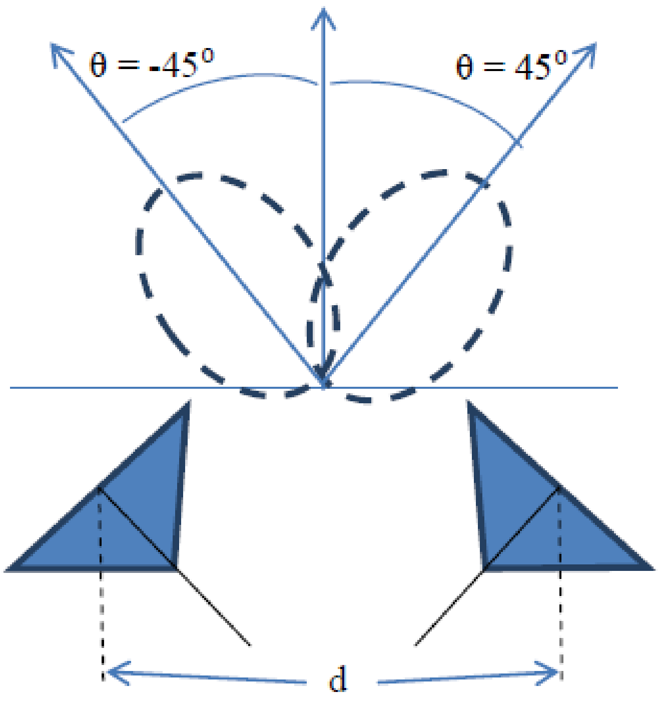


Figure 13: A two antenna amplitude comparison system. Each blue triangle represents a horn antenna. The antenna boresights are offset by 90°.
Source: (Silva et al., 2011).

the other two methods of direction finding. The noise changes the amplitude of the signal, causing the amplitude ratio to differ from the expected value. The change in amplitude can cause the device to resolve to a different angle. At the fringes of the detectable range, the signal to noise ratio is lower than at the boresight. Consequently, the noise has a larger impact on the error in the power ratio calculations at angles at the extremes of the field of view. In the Beacon Locator Project report, the amplitude comparison method was implemented with errors as large as 4° for angles near the extremes of their range

(±45°) (Silva et al., 2011).

2.3.3 Phase Interferometry

The third and final method for determining the AoA of an incoming signal is the phase interferometry method. The setup of a two antenna phase interferometry system is shown in Figure 14. In Figure 14, d_1 is less than d_2 so the point p is placed on the line segment $\overline{EA2}$ such that d_1 is equal in length to the line segment $\overline{Ep}$. The system is oriented such that an AoA of 0° corresponds to a target directly in front of the aircraft. Like the TDOA method, this method relies on the delay of the signal arriving at two different antennas; however, this method compares the change in phase between the two waves received rather than the difference in the time of arrival. An advantage to comparing phase is that when the distance between antennas is on a similar scale to the wavelength of the received signals, the phase difference is significant enough to measure accurately. For the X band, the wavelength of the signals is generally smaller than the

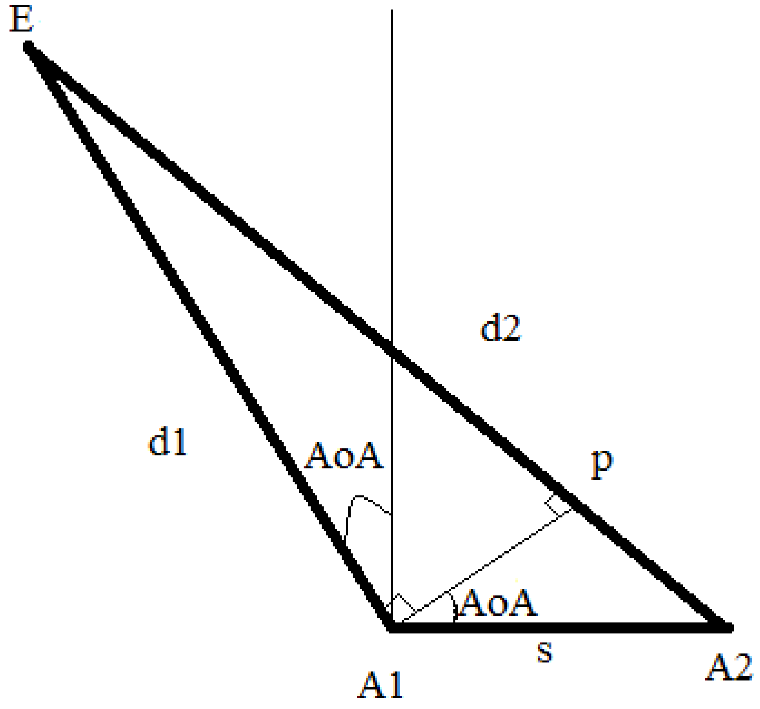


Figure 14: A phase interferometry system. An emitter distance d_1 and d_2 from two antennas separated by a distance s. Point p is placed on the line segment $\overline{EA2}$ such that d_1 is equal in length to the line segment $\overline{Ep}$. Here s is much less than d_1 and d_2.

antenna separation; therefore a phase interferometry system does not have the same small-scale accuracy issues present in a TDOA system.

The setup of the system allows the processor to use trigonometry to find the AoA of the signal. Typically, the distances from the antennas to the emitter, d_1 and d_2, are orders of magnitude larger than the antenna spacing, s, so that near the antennas the lines $\overline{EA1}$ and $\overline{EA2}$ can be considered parallel as shown in Figure 15. When $\overline{EA1}$ and

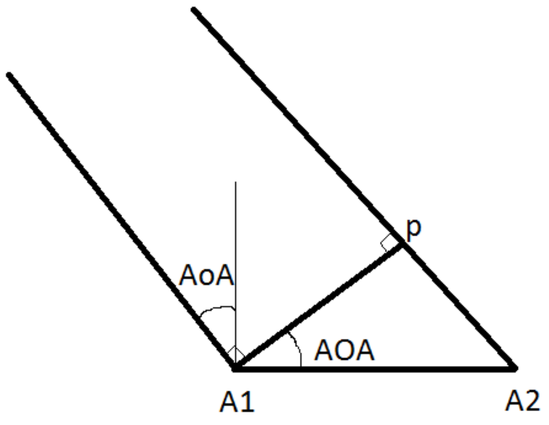


Figure 15: A closer look at the system depicted in Figure 14.

$\overline{EA2}$ are parallel the points A1, A2, and p form a right triangle. The extra distance the wave had to travel to the second antenna, the line segment $\overline{pA2}$, can be found using the phase difference. The length of $\overline{pA2}$ is given by:

$$\overline{pA2} = \lambda(\frac{\Delta\phi}{2\pi} + I), \tag{16}$$

where I is a non-negative integer, λ is the wavelength of the received RF signal, and $\Delta\phi$ is the phase difference in radians. The phase difference, $\Delta\phi$, is measured relative to antenna 1 and is restricted to values between 0 and 2π. The restriction on $\Delta\phi$ restricts the first term of Equation 16 to values between 0 and 1, requiring the addition of I to account for distances greater than λ.

Since the line segments $\overline{A1p}$, $\overline{pA2}$ and $\overline{A2A1}$ form a right triangle and the angle formed by A1A2p is given by $\pi/2 - AoA$ and $cos(A + B) = cos(A)cos(B) - sin(A)sin(B)$, the cosine relationship can be used to calculate the AoA:

$$cos(\pi/2 - AoA) = sin(AoA) = \frac{\lambda(\frac{\Delta\phi}{2\pi} + I)}{s}, \tag{17}$$

or in terms of the AoA

$$AoA = sin^{-1}\left(\frac{\lambda(\frac{\Delta\phi}{2\pi} + I)}{s}\right). \tag{18}$$

When the emitter moves from the left of the interferometer to the right, the calculation changes slightly. Figure 16 shows the setup for an emitter on the right of inteferometer. The point p is again placed so that the length of the line segment $\overline{Ep}$ is equal to d_1. Because d_1 is less than d_2 $\overline{Ep}$ extends past antenna A2. The right triangle formed by A1, A2, and p is once again used to calculate the AoA and Equation 16 is still used to calculate the length of $\overline{pA2}$. In this case, however, the value of I will be negative resulting

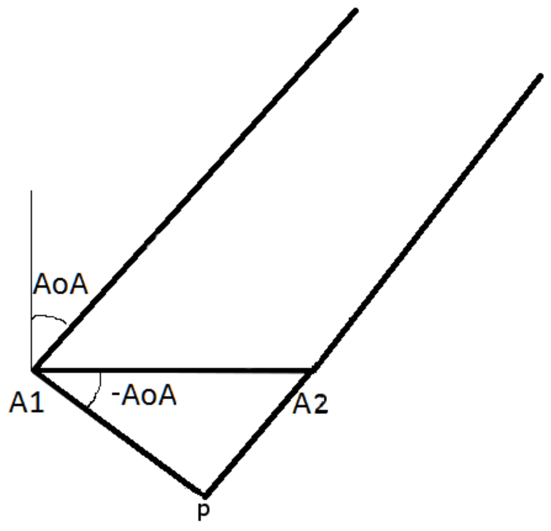


Figure 16: Figure 14 with the emitter moved to the right side of the interferometer. The line segment $\overline{EA2}$ is artificially extended to point p so that antenna 1 can still be used as a reference. Emitters to the right of the interferometer result in negative total phase difference and negative AoAs.

in a negative length for $\overline{pA2}$. The negative length reflects the fact the signal arrives at antenna 2 before antenna 1 and allows antenna 1 to be used as the baseline in both cases. A negative value for I results in a negative value for the AoA from Equation 18. It is more intuitive to have negative AoAs for emitters on the left of the system and positive AoAs for emitters on the right so the final AoA equation used multiplied Equation 18 by -1 resulting in Equation 19:

$$AoA = -1 * sin^{-1}\left(\frac{\lambda(\frac{\Delta\phi}{2\pi} + I)}{s}\right). \tag{19}$$

In Equation 19 $\Delta\phi$ is the measured phase difference between 0 and 2π, s is the antenna separation, and I is an integer corresponding to the number of 2π phase changes.

The phase interferometry method yields accurate AoA calculations; however, when antenna separation is greater than $\frac{\lambda}{2}$, the phase difference can be greater than π. Interferometers can only calculate phase differences between $-\pi$ and π, allowing for multiple answers, or ambiguities, when calculating the AoA. Additional techniques to resolve ambiguities are needed in order to make this technology viable for direction finding.

The AoA calculation is valid so long as d, the distance from the emitter to the midpoint between the antennas, is much larger than s. To discover the limits of this assumption, we considered the triangle formed by antenna 1, the midpoint between the antennas, m, and the position of the emitter. The geometry is shown in Figure 17. The angles of the triangle, α_1 and α_2, can be written in terms of the AoAs, AoA1 and AoA2:

$$\alpha_1 = \frac{\pi}{2} + AoA1 \tag{20}$$

$$\alpha_2 = \frac{\pi}{2} - AoA2. \tag{21}$$

The law of cosines produces:

$$d^2 = (s/2)^2 + d_1^2 - sd_1cos(\alpha_1). \tag{22}$$

Rearranging Equation 22 to solve for α_1 yields:

$$\alpha_1 = cos^{-1}\left(\frac{s}{4d_1} + \frac{d_1}{s} - \frac{d^2}{sd_1}\right). \tag{23}$$

Again, using the law of cosines to solve for d_1 produces:

$$d_1 = \sqrt{d^2 + (s/2)^2 - dscos(\alpha_2)}. \tag{24}$$

Combining equations 23 and 24 yields:

$$\alpha_1 = cos^{-1}\left(\frac{1}{\sqrt{d^2 + (s/2)^2 - dscos(\alpha_2)}}\left[\frac{s}{2} - dcos(\alpha_2)\right]\right). \tag{25}$$

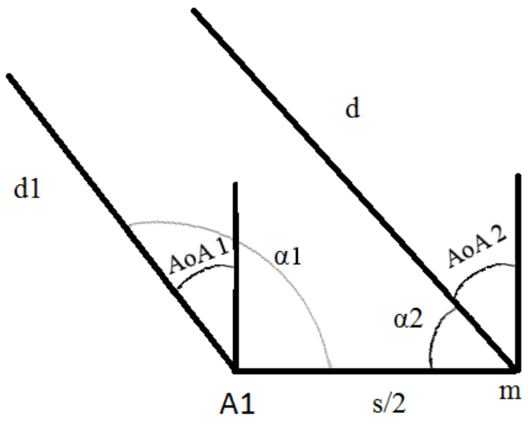


Figure 17: This figure shows the antenna 1, A1, and the midpoint between the two antennas, m. The distance from the emitter, not shown, to the first antenna is d_1 and the distance from the emitter to the midpoint is d. The separation between A1 and m is $s/2$ and the angle of arrival for each point is shown (AoA1 and AoA2 respectively). The angles of the triangle formed by the emitter, m, and A2 are α_1 and α_2.

In the limit as d approaches infinity, Equation 25 becomes:

$$\alpha_1 = cos^{-1}\left(\frac{1}{\sqrt{d^2}}\left[\frac{s}{2} - dcos(\alpha_2)\right]\right) = cos^{-1}\left(\frac{s}{2d} - cos(\alpha_2)\right) = \pi - \alpha_2, \tag{26}$$

which is the result of adding Equations 20 and 21 when θ_1 and θ_2 are equal. From Equation 25, for realistic ratios of d to s on the order of 1000:1, the difference between α_1 and α_2 is on the order of 10^{-4} degrees.

2.3.4 Resolving Phase Ambiguity

The greatest technical difficulty in implementing the phase interferometry method of direction finding is resolving ambiguities. Phase interferometers can only measure phase differences between $-\pi$ and π. Phase differences outside of this range result in ambiguous solutions and, without additional information, the true AoA cannot be determined. The simplest way to remove ambiguity is to place the interferometer's two antennas less than $\frac{\lambda}{2}$ apart; however, this cannot be used for our application. For a 12 GHz signal, the maximum X band frequency, half of the wavelength is 1.25 cm. Placing the antennas this close together would require the antennas to be less than 1.25 cm wide in order to not physically overlap one another. Using antennas this small would result in a small effective aperture and poor signal gain. Additionally, when the separation between the antennas is small, the phase difference becomes more susceptible to noise and the measurement loses accuracy.

Combining amplitude comparison and phase interferometry techniques provides another method of solving ambiguities with two antennas. The amplitude comparison provides a rough, but unambiguous angle of arrival, which can be used to determine which of the several ambiguous angles produced by phase intereferometry is the true AoA. The amplitude method can be accomplished with directional antennas, or by placing an asymmetric gain medium placed between two omnidirectional antennas. The scattering box reduces the amplitude of the received signals by varying amounts dependent upon the incident angle. Asymmetric boxes allow unique determination of every AoA. The scattering box method allows for 360° field of view (Zhou et al., 2011). Due to the time constraints of the project we focused on solving the ambiguities with only the phase interferometry method rather than combining the phase interferometery and amplitude comparison methods.

The problem of ambiguity can also be mitigated by moving from two antennas to three antennas. The antennas are placed in a line with differing spacing as shown in Figure 18. The three antennas are treated as two pairs; the short baseline of antenna 1 and antenna 2 and the long baseline of antennas 1 and 3. Adding the third antenna helps mitigate some of the issues with placing smaller antennas half of a wavelength apart. The system

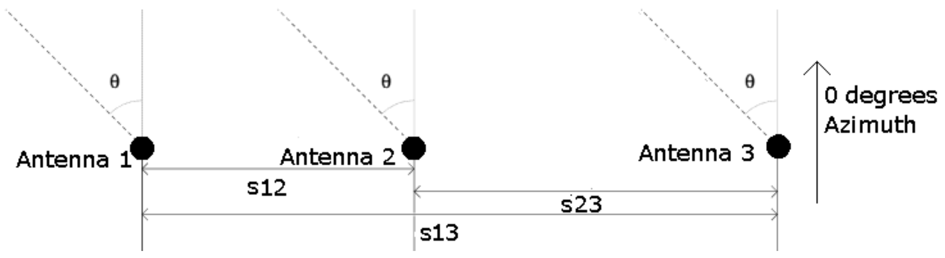


Figure 18: Setup of a three antenna phase interferometer. Each of the three lengths, s12, s23 and s13, has a distinct value.

maintains its overall accuracy by measuring the angle of arrival based on antennas 1 and 3, which are far enough apart to be resistant to noise, after ambiguities are removed by antennas 1 and 2.

In cases where the antennas cannot be placed half a wavelength apart using three antennas still helps remove ambiguities. The phase difference can now be calculated between the first and second and first and third antennas giving two sets of ambiguous results. Comparing these results reduces the ambiguities to only those angles that appear among the possible angles for both pairs of antennas. This method is further improved by taking the geometry of the antennas into account. The maximum number of full phase changes that can occur between a pair of antennas, and therefore the number of ambiguities, n, is shown in Equation 27:

$$n = \frac{s}{\lambda} sin\theta_{max}, \tag{27}$$

where s is the distance between the antennas, λ is the minimum wavelength of the signal and θ_{max} is the maximum angle of arrival for the system (Jacobs and Ralston, 1981). The true phase difference for a measured phase difference of $\Delta\phi_{12}$ is $\Delta\phi_{12} + i * 2\pi$ where i is an integer in the range of $-n \leq i \leq n$. These equations apply to both pairs of antennas (the first and second antenna and the first and third antenna). By substituting these equations into the equation for angle of arrival, equation 18, the following two equations can be derived:

$$\Delta\phi'_{12} + i_{12} = \frac{s}{\lambda} sin\theta_{AoA}$$
$$\Delta\phi'_{13} + i_{13} = \frac{s}{\lambda} sin\theta_{AoA}.$$

In these equations $\Delta\phi'_{12}$ and $\Delta\phi'_{13}$ are $\Delta\phi_{12}$ and $\Delta\phi_{13}$ normalized by 2π so that they are in the range of 0 to 1. Combining the two equations yields:

$$\Delta\phi'_{13} = \frac{s_{13}}{s_{12}} \Delta\phi'_{12} + \frac{s_{13}}{s_{12}} i_{12} - i_{13}. \tag{28}$$

For any valid value of i_{12} and i_{13} Equation 28 yields a line representing a set of possible phase differences. Collectively these lines cover every theoretically possible combination of phase differences. In a no noise environment the measured phase difference will always fall on one of these lines. By finding the line that is closest to the measured phase difference, the true value for the phase difference can be found (Jacobs and Ralston, 1981). An example of this process is shown in Figure 19. In this plot the blue lines are

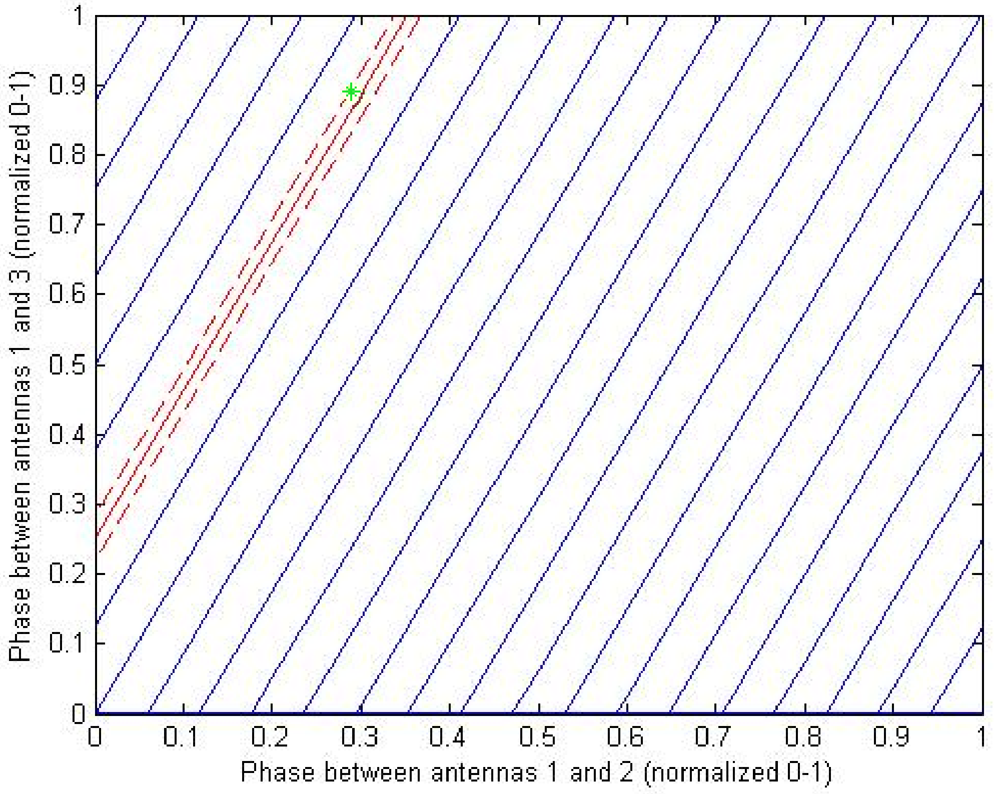


Figure 19: Example plot of the phase resolution technique. The blue lines are phase lines generated by Equation 28, the green star is the location of the normalized measured phase differences ($\hat{\Delta\phi}_{12}$, $\hat{\Delta\phi}_{13}$), and the red line is the chosen result. The red dotted lines show the maximum distance from the chosen line that is treated as a certain result.

determined by specific values of i_{12} and i_{13}, the green dot is the location of the measured phase difference, and the red line is the closest result. The true phase is determined by: $\hat{\Delta\phi}2\pi + I2\pi$, where $\hat{\Delta\phi}$ is the normalized measured phase difference and I is the number of full phases as determined in Figure 19. Noise can cause the measured phase to move away from one of the true phase lines. A range of confidence, shown by the red dotted lines in Figure 19, is defined to reflect the possibility of noise causing an incorrect phase resolution. If the measured result is within the confidence range, between the red dotted lines and the red line, then the system is confident that it has resolved the phase correctly. Increasing the confidence range raises the possibility that the system will confidently report the incorrect angle while reducing it raises the possibility of reporting a correct result as uncertain.

In high noise cases the measured phase can bounce between two or more possible true phases. To help mitigate these cases, a history of calculated angles can be kept to help determine which angles to use. Adjacent lines of true phase will yield drastically different results for AoA because moving over one line represents changing the phase difference by 2π radians, so it can be quite clear when noise causes the calculation to choose the wrong phase line. This risk can be mitigated by choosing the antenna spacing such that there is the greatest possible distance between true phase lines. There are more ambiguities at high frequency so testing a variety of antenna spacing with the 12 GHz provides a spacing that can be used for any frequency in X band. In an ideal, noiseless environment the measured phase will always appear on one of the true phase lines. When the lines are far apart, the noise has to corrupt the phase measurement by more before it will resolve to an incorrect angle. The worst case scenario occurs when two incident true phase lines are generated by Equation 28. In this case, there will always be two completely ambiguous results for a certain set of measured phases. If the needed reliability cannot be achieved through optimal spacing and noise reduction, the field of view can be limited to reduce the total number of lines.

3 MATLAB Methodology

The MATLAB model of the DF system is composed of two distinct components: the signal generator and the direction finding algorithm. The layout of the model and the functions performed by these two components are shown in the block diagram in Figure 20.

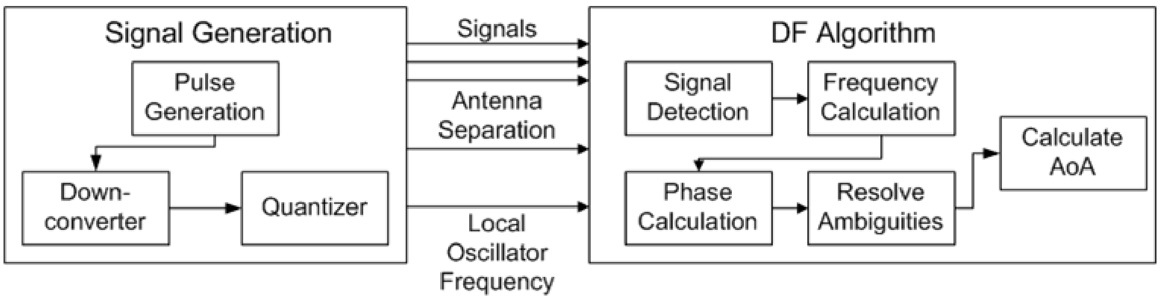


Figure 20: A block diagram showing the structure and components of the MATLAB model. The signal generator generates signals and replicates the processing performed by hardware as signals are received. The DF algorithm analyzes the three signals produced by the generator and determines the AoA

The signal generator models the hardware of our system, a superheterodyne receiver, up to and including the ADCs. The generator simulates a pulsed sine wave at RF, down converts the sine wave to IF, passes the new waveform through a low pass filter, and then quantizes the signal as if it had passed through the ADCs. After quantization, the signals are sent to the DF algorithm which performs the steps required to determine the AoA: detecting signals, determining signal frequency, determining the phase difference, and removing ambiguities. The algorithm outputs the frequency of the detected signal, the two most likely AoA values and the likeliness of the two most probable solutions. Designing the DF algorithm in MATLAB allowed for the use of MATLAB's high level functions, such as the FFT function and vector processing, and built in signal processing tools to rapidly design and test the algorithm. MATLAB's tools and functions permitted us to make fundamental changes to the components without addressing the complexities of memory management and system input and output.

3.1 Project Plan

The model architecture utilized modular design principles in order to further reduce complexity. The MATLAB model was created using a spiral development method, gaining components with each iteration. The schedule for the model's development in shown in the Gantt chart in Figure 21. Each component in the MATLAB simulation was designed and tested separately by a member of the group. Components were added to the simulation individually until the model reached the final and most accurate representation

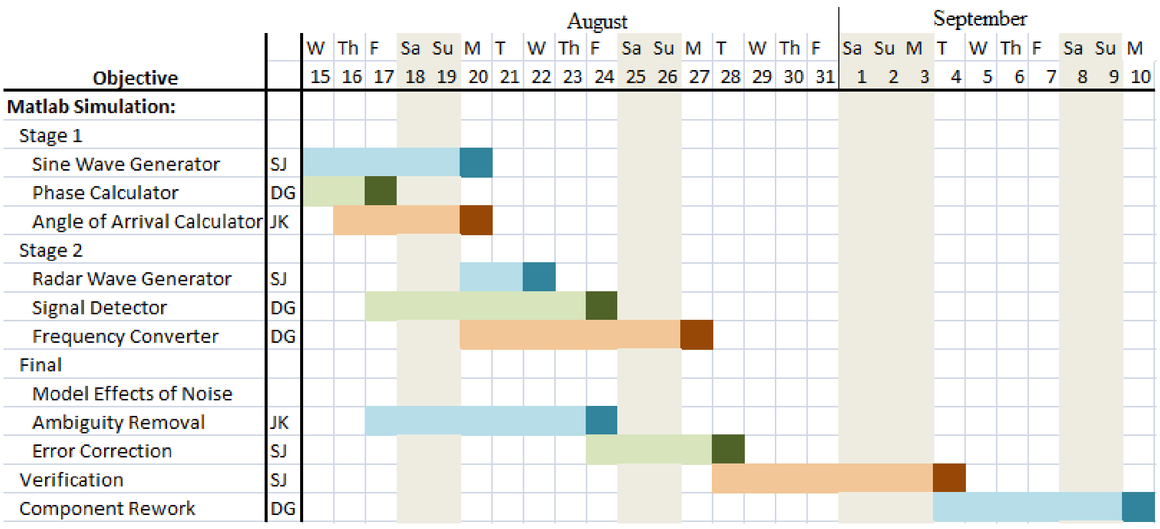


Figure 21: Gantt chart showing the schedule for the MATLAB model. Work began on August 15th, 2012 and continued to September 10th, 2012. Each component was developed by a different individual whose initials are shown in the second column. The components are grouped into iterations representing significant improvement in the system.

of the physical system. Each logical element is discussed below.

3.2 Signal Generation

The signal generator simulates the generation of pulsed X-band radar signals and the conversion of these signals from analog to digital. To generate the appropriate signal, the following parameters must be specified: the AoA at antenna 1 (degrees), the signal carrier frequency (Hz), the distance between the emitter and midpoint between antennas 1 and 2 (m), the separation between antennas 1, 2, and 3 (m), the signal strength (dBfs), the number of pulses, the pulse repetition interval (s), and the pulse width (s). The RF waves are generated with a 50 GHz sampling rate to be as close to continuous as possible while still allowing enough memory to generate multiple pulses. The simulated RF waves are down converted to the IF, sampled at 500 MHz, and quantized by 12 bits.

3.3 Pulse Generation

The signal generator creates three sine waves of the form $Asin(\omega t + \phi)$. Each sine wave corresponds to the signal received at one of the three antennas. To create a series of pulses, the three waves are multiplied by a pulse train specified by the PW and PRI. The start time of the pulse train is randomized to begin at any time during the PRI. The differences in phase offset, ϕ, is the parameter used by the DF algorithm to calculate the AoA. In order to calculate ϕ, the MATLAB model calculates the distance from the emitter to each of the antennas. The distances between each of the antennas and the emitter are calculated using the law of cosines as shown in Equation 29.

$$d_a^2 = (s)^2 - (2ds)cos(\alpha). \tag{29}$$

In Equation 29, d_a is the distance to the emitter from the antenna, s is the distance from the antenna to the midpoint between antenna 1 and 2, d is the distance from that midpoint to the emitter, and α is the angle formed at the midpoint, $90 + AoA$ for antennas 2 and 3 and $90 - AoA$ for antenna 1. After the distance is calculated, the phase of the waves when they reach the antennas is determined using Equation 30:

$$\phi = 2\pi d_a(f/c), \tag{30}$$

where f is the carrier frequency of the signal and c is the speed of light estimated at $3 * 10^8$ m/s. The results of this calculation are used for the phase shift in the generated sine waves.

The amplitude, A, of the sine waves is determined by the signal strength parameter. The signal strength parameter is in dBfs which is a measure of the signal's power relative to the maximum possible value. The maximum value that can be received by a digital system is determined by the ADCs' range. The Texas Instruments ADS5400 ADCs used in our system have 12 bits of resolution so their range is from -2^{11} to $2^{11} - 1$. The pulse generator maps these values to -1 and 1 so the amplitude of a signal with 0 dBfs will be one. Because the maximum value is one, the amplitude of the sine wave at any signal strength (S) is found by converting the dBfs value to a magnitude using Equation 31:

$$A = 20log_{10}(S). \tag{31}$$

The last step in generating the signals is simulating noise. Our model defines a fixed noise level of -60 dBfs based on the performance defined in the ADS5400 ADC vendor data sheet. This noise level results in a signal to noise ratio (SNR) from 60 to 20 dB across the 40 dB range requirement. The simulated noise is white Gaussian noise produced by MATLAB's awgn (add white Guassian noise) function. Once the noise is added, the pulsed signal is passed to the down converter.

3.4 Down Converter

Using the signal's frequency, the width of the IF band, and the sampling rate, the down converter frequency shifts the signal to IF. In a physical system, down converting is performed by tuning a local oscillator (LO) to generate a sine wave at a known frequency, multiplying the LO signal by the incoming signal to create a signal with two components, and using a low-pass filter to isolate the lower frequency component. The frequencies of the two components are the sum and difference of the LO and input signal. In an operational system the local oscillator is shifted in 100 MHz increments across the 4 GHz band. If the local oscillator is at the wrong frequency when a signal arrives, that signal will not be detected (Holm and Richards, 2010). The model automatically chooses a local oscillator frequency that will down convert the signal and then uses a low-pass 20th order Butterworth filter with a cutoff frequency of 10 GHz to remove the high frequency component. The Butterworth filter is designed to minimize the effect on the frequency response of the wave in the passband at the expense of having a gradual cutoff. The Butterworth filter removes half of the original magnitude of the signal. The resulting IF signal therefore ranges from -0.5 to 0.5 rather than -1 to 1. The IF signal is then sent to the quantizer which simulates quantization error of a 12 bit ADC.

3.5 Quantizer

The ADCs are simulated by down-sampling the IF signal from the original "continuous" form with a 50 GHz sampling rate into the 500 MHz sampling rate that we expected to use in the hardware. To simulate a 500 MHz sampling rate, the simulation takes every 100th sample of the waveform and saves that collection of samples as the new wave. Additionally, ADCs do not store the actual voltage values of samples. Instead, the ADC converts these values to integers between some maximum and minimum values specified by the number of bits used by the ADC. For the 12-bit ADCs used in the prototype there are 2^{12} possible values ranging from -2^{11} through 2^{11}-1. On a range of -0.5 to 0.5 volts, the minimum and maximum amplitude of our simulated IF pulses, the quantization results in a step size of 0.244 mV. Perfect simulation of an ADC would be complicated, so for simplification the simulated quantizer rounds the voltages down to the nearest step, so any result in the digital signal may be off by as much as the quantization step value.

3.6 DF Algorithm

The signals generated by the signal generator are used as inputs to the direction finding process. The DF algorithm takes one batch of samples, 4096 values, and checks to see if there is a signal present in all three waveforms. If there is a signal present the algorithm determines its frequency and phase. It then removes the ambiguities from the phase calculation and uses the final result to calculate the AoA. The signals passed to the DF algorithm are at IF, however the AoA calculation requires the full RF of the signal. In order to calculate the RF, the DF algorithm is also provided with the frequency of the local oscillator used to down convert the signal. Once the algorithm has processed the entire waveform, it reports the frequency, the two most likely AoAs, and the certainty calculated for each sample. Each step of the DF algorithm is detailed below.

3.6.1 Signal Detector

The signal detector examines the incoming data and, based on the presence or absence of a signal, decides whether or not to forward the signal for further processing. In order to determine if a signal is present each batch of samples is checked for voltage amplitudes above a threshold value. If at least one sample from each antenna has a value above the threshold, the detector determines that a signal is present and the batch is forwarded. Otherwise the batch is discarded and the next batch of samples is taken. The detector stores the average magnitude of the five most recent samples without a signal and sets the threshold value to four times the average of these five values. Dynamically setting the threshold allows the algorithm to adjust to changes in noise.

3.6.2 Frequency Calculation

In order to determine the AoA, the algorithm must first determine the frequency of the signal. To calculate the frequency, the algorithm takes the discrete Fourier transform of the signal. The system removes the DC bias of the signal before taking the DFT to ensure that the zero-frequency component will never be reported as the largest. The model removes the DC bias by subtracting the average value of the batch of samples from each individual sample. Removing the DC bias ensures that the zero frequency component of the received waveform will not affect the results of the DFT while leaving all other frequency components unaffected. With the DC bias removed, the algorithm examines the magnitude of the signal in the frequency domain. The location of the largest magnitude is the dominant IF of the signal. If the batch contains multiple signals, only the frequency component with the largest magnitude is considered. The intermediate frequency of the dominant signal is added to the frequency of the local oscillator to determine the signal's original RF.

The frequency stage also performs another signal detection check. If the magnitude of the highest frequency component in the batch is not more than four times the average magnitude, the batch is rejected. The cutoff value was chosen because simulations showed that it provided a reliable method of detecting signals while not producing a high false alarm rate. In the absence of noise, the only frequency present is the signal frequency, but in a noisy environment there are many other smaller frequency components. If high noise levels had caused the sample to pass the signal detector the secondary detection algorithm will remove it. To ensure consistency between the three antennas, data is only counted as a signal when a signal is detected in all three batches of samples.

3.6.3 Phase Calculation

The algorithm uses the dominant frequency determined by the frequency calculator to determine the signal of interest. The dominant frequency is the location where the signal is the strongest and, therefore, least susceptible to noise. The angle between the real and imaginary components of the DFT at this point is equal to the phase of the signal. With the presence of noise, the dominant frequency calculated for each of the three antennas will be slightly different. In order to accurately calculate the phase differences the phases of the three signals have to be calculated at the same point. The dominant frequency at antenna one is assumed to be the dominant frequency for the other two antennas to ensure that the same point is used in all three phase calculations.

3.6.4 Ambiguity Removal

The phase difference calculated by the phase calculation component is always between -π and π. The true phase difference between the antennas, however, is not limited to this range. The true phase difference between the antennas is $\Delta\phi + I2\pi$ where $\Delta\phi$ is the measured phase difference and I is an integer corresponding to the number of full 2π phase changes. In order to accurately calculate the phase difference the algorithm needs to determine the appropriate value for I. By calculating the phase differences between antenna 1 and 2 and antenna 1 and 3, the algorithm is able to find the number of full phase changes that occur between the antennas. The maximum number of full phase changes is limited by the wavelength of the signal, the antenna separation, and the field of view. The maximum number of full phase changes between a pair of antenna, i_{max} is given by $s/\lambda * sin(\theta_{max})$ where s is the antenna separation and θ_{max} is the largest angle in the field of view. Every possible pair of phase differences calculated by the system should fall on one of the lines given by Equation 32:

$$\Delta\phi'_{13} = \frac{s_{13}}{s_{12}}\Delta\phi'_{12} + \frac{s_{13}}{s_{12}}i_{12} - i_{13}, \tag{32}$$

where $\Delta\phi'_{12}$ and $\Delta\phi'_{13}$ are the measured phase differences normalized by 2π and i_{12} and i_{13} are any two values of I valid between antennas 1 and 2 and antennas 1 and 3 respectively. Figure 19 of Section 2.3.4 (Resolving Phase Ambiguity) shows the technique for resolving phase ambiguities in more detail. In the presence of noise the measured result will not lie directly on one of these lines. The true phase differences can be estimated by determining the closest line. Since all of the lines are parallel the distance to any line can be found using Equation 33:

$$distance = |\Delta\phi'_{13} - (s_{13}/s_{12})\Delta\phi_{12} + i_{13} - (s_{13}/s_{12})i_{13}|. \tag{33}$$

The algorithm reports the values of i_{12} and i_{13} that minimize the distance as the most likely result. It also reports the certainty of that result compared to the second closest using $1 - (min_1/(min_1 + min_2)$ where min_1 is the distance to the closest line and min_2 is the distance to the second closest line. The maximum certainty value of 1 is reported when the measured phase values lie directly on one of the phase lines. The minimum reported certainty value, 0.5, occurs when the distance from the measured phases to both of the closest lines is equal in which case, the algorithm cannot determine how to resolve the phase ambiguity. The algorithm then calculates the AoA for these two results using the AoA equation:

$$AoA = sin^{-1}\left(\frac{\lambda(\frac{\Delta\phi}{2\pi} + i)}{s}\right).$$

The confidence range chosen for the disambiguation process is 0.25. As long as the distance from the measured result to the closest phase line is less than a quarter of the total separation between the two closest lines, corresponding to a certainty value greater than 0.75, the system is confident. Any results with certainty values less than 0.75 will be checked against previous results to check the angles reported. The confidence range of 0.25 caused few false alarms, wrong answers reported as certain, while still allowing for mostly correct detections at the low end of the dynamic range.

To maximize the chance that the correct ambiguity is chosen, the antenna separation should be chosen so that the distance between the lines generated with Equation 32 is maximized (Jacobs and Ralston, 1981). The MATLAB model included a function that calculated the optimal antenna separation for the first and third antennas given the separation between the first and second antenna and the maximum frequency. The line spacing function calculates the minimum separation between lines for every value of s_{13} between $2s_{12}$ and $2s_{12} + 50$ cm with increments of 0.1 mm. The s_{13} value with the highest minimum line spacing is the optimal value. The results of running this function are shown in Figure 22.

The peaks in Figure 22 correspond to values for s_{13} with high spacing between phase lines. For $s_{12} = 10$ cm and a maximum frequency of 12 GHz the ideal value for s_{13} is 21 cm. Values for s_{13} with minimum phase line spacing of zero, such as 30 and 40 cm, are the worst choice for antenna separation.

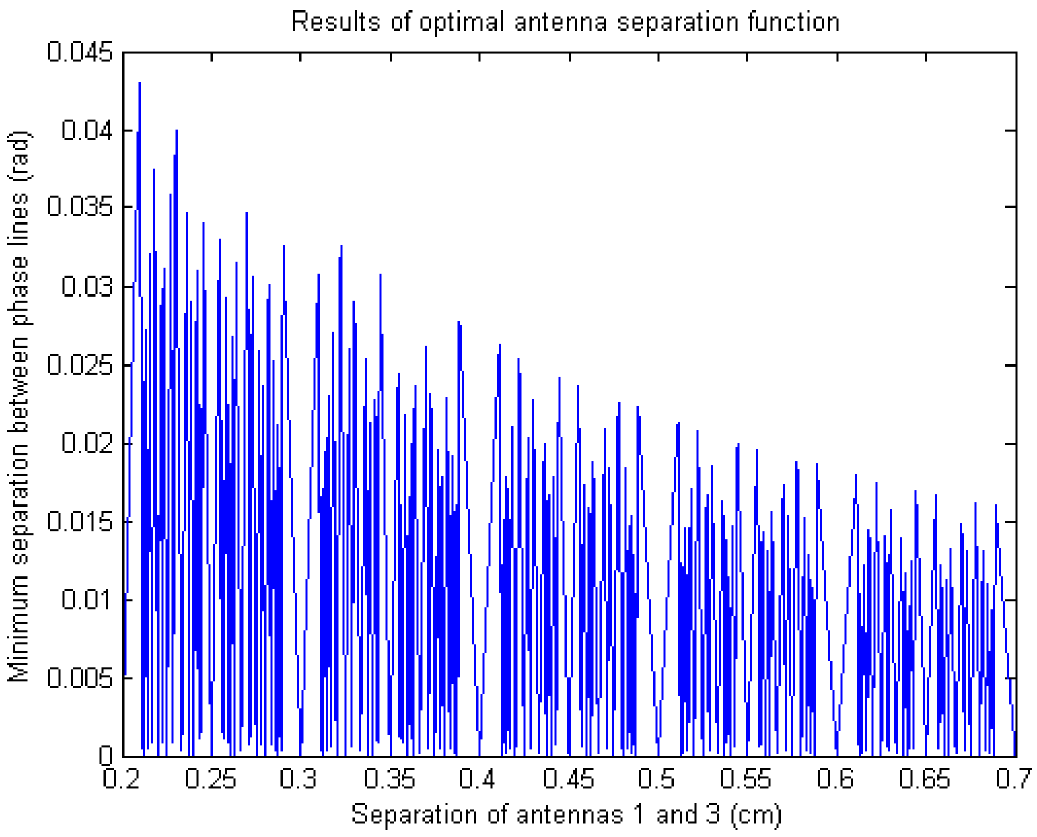


Figure 22: Plot of separation between antennas 1 and 3 versus the minimum separation between phase lines. This plot shows the result of the optimal antenna spacing function. The results shown are for $s_{12} = 10$ cm and a maximum frequency of 12 GHz. The highest peak corresponds to the value for s_{13} that maximizes the phase line spacing.

4 MATLAB Model Results

Before implementing our DF algorithm in C, the algorithm was verified in MATLAB to ensure that it met all the design requirements. Antenna separation was optimized for our worst case conditions, and then held constant for other tests. Once the antenna separation was chosen, the system was tested in worst case conditions over the range of $\pm 90°$ with a step size of $1°$. Once verified, individual parameters were tested to identify their impact on error and ambiguity resolution. Each test was run under worst case conditions because validating in the worst case strongly suggests validation for all other cases. The worst case conditions that the system is required to handle successfully are: an emitter distance of 1 km, signal to noise ratio of 20 dB, RF of 12 GHz, and pulse width of 500 ns.

4.1 Error versus Antenna Separation

The antenna separation is the only parameter we have physical control over and, by extension, have the ability to optimize. As mentioned in the background and methodology, the ideal antenna separation maximizes the spacing of phase ambiguity lines and increases the probability of reporting the correct AoA. The MATLAB model calculates the optimal separation between antennas 1 and 3 based on the maximum expected carrier frequency and the separation between antennas 1 and 2. Higher carrier frequencies result in more ambiguities, so when a 12 GHz waveform can be analyzed with no ambiguities, so can all lower frequencies. The minimum separation between antennas 1 and 2 is 5 cm because the antennas themselves are 2.5 cm wide. To determine the optimal antenna spacing between antennas 1 and 2, we tested all antenna 1-2 spacings between 5 cm and 20 cm with a step size of 0.5 cm. The optimal position of antenna 3 was calculated using our spacing function.

A plot of the average certainty corresponding to the correct AoA versus the separation between antennas 1 and 2 is shown in Figure 23. The certainty was averaged for 100 runs at each antenna separation with a true AoA at 45°. Throughout the MATLAB results, the certainty plots are of the correct AoA and are separated into three regions, green, yellow and red. When the certainty is in the green region, or above 0.75, the ambiguity has been resolved correctly and the system is confident of the answer. In the yellow region, the certainty is between 0.5 and 0.75 and the system reports the correct angle, but is not confident that the angle is correct. The red region, or certainty values below 0.5, corresponds to failures in the DF algorithm to accurately determine the correct AoA. When the certainty of the correct AoA is this low, the correct AoA is the second reported angle. In other words, the system has resolved the ambiguity incorrectly. A certainty value of 0 indicates that the correct AoA was not one of the returned angles.

Increasing the antenna separation improves the accuracy of the AoA calculation when

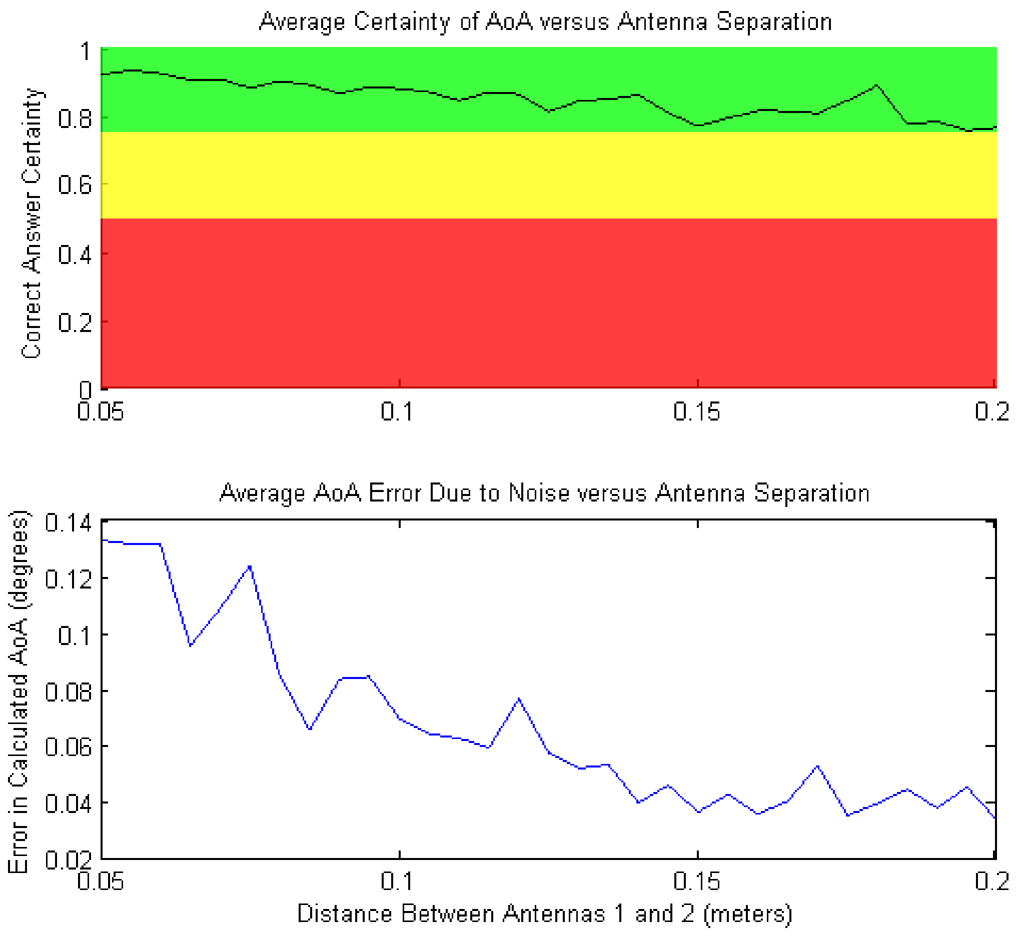


Figure 23: The graph of average certainty of the correct AoA (Top) and average error of the AoA calculation (Bottom) vs. separation between antennas 1 and 2. The separation between antennas 1 and 3 was optimized for every case. One hundred signals with an AoA of 45° and worst case parameters were used to test each of the different antenna configurations. The certainty and error were averaged to produce the plots. As the separation was increased, the average error and the average certainty decreased. The reduction in certainty increases the frequency of incorrect ambiguity resolution; however, when resolved correctly, the calculated AoA is more accurate.

the phase is chosen correctly, but also increases the probability of resolving the ambiguity incorrectly. Both the mean error and the mean certainty decreases with antenna separation. For the system, the 10 cm and 21 cm separation was chosen to provide a balance between mean error due to noise and certainty values.

To validate our method of antenna placement, we tested all angles between -45° and 45° for the calculated optimal distance between antennas 1 and 3 of 21 cm as well as 20 and 22 cm. The certainty plots and error values are shown in Figure 24. The 21 cm separation provided the correct AoA with a certainty value above 0.75 for the majority of runs. The 20 and 22 cm separations have more ambiguity errors reflected by average certainties around 0.5. With the incorrect separations the system looses its ability to

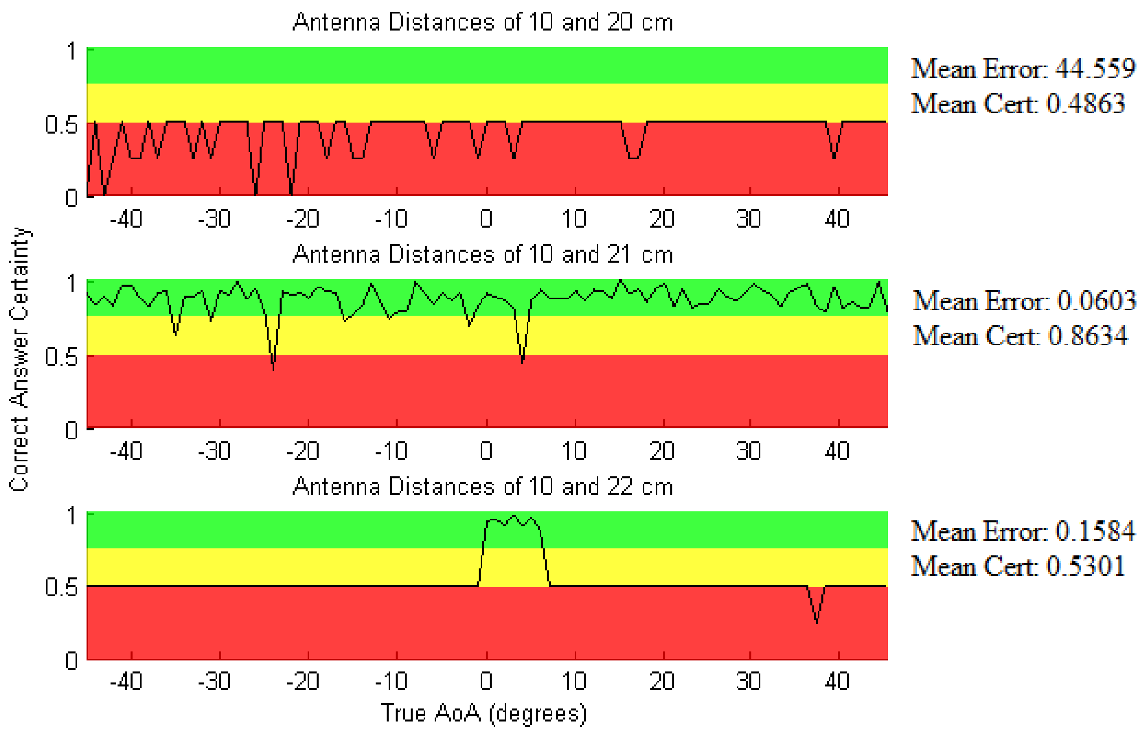


Figure 24: The graph of certainty of the correct AoA vs. the true AoA for all angles between -45 and 45 degreers. Three different antenna (1-3) separations were tested, the calculated separation (21 cm) and the calculated separation ±1 cm. The tests were run in worst case conditions. The average error and certainty are also reported for each of the antenna geometries. At 21 cm the mean error is minimized and the certainty is maximized as expected.

resolve angles.

To ensure that the system would not stop functioning if the antenna separation was slightly different than the optimal calculated separation, the effects of changing the separation by ±1 mm were tested. Figure 25 shows the certainty and mean error for the three separations. By comparing Figures 24 and 25, it is clear that the random error induced by noise has a larger impact on error and certainty than variations in antenna spacing on the order of 1 mm.

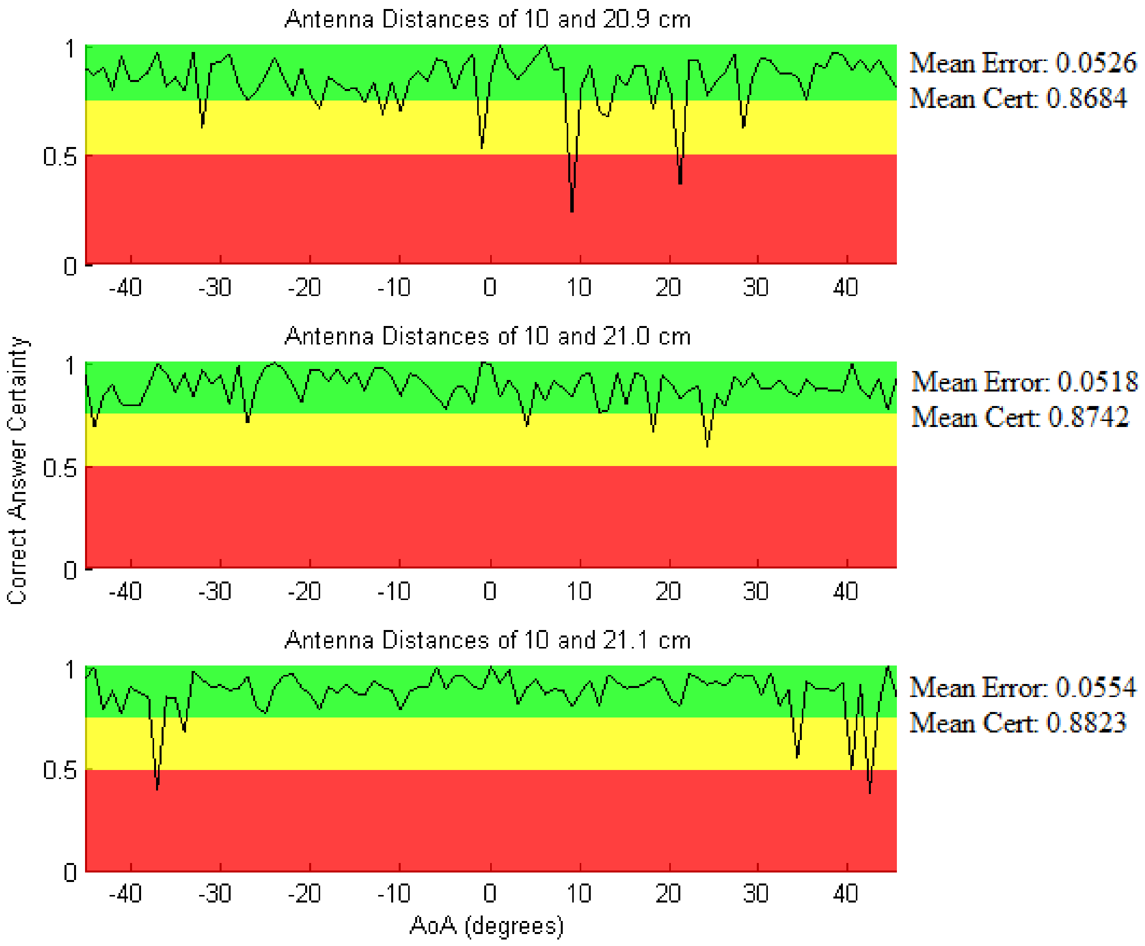


Figure 25: The graph of certainty of the correct AoA calculations vs. true AoA for all angles between -45 and 45 degrees . Three different antenna (1-3) separations were tested, the calculated separation (21 cm) and the calculated separation ± 1 mm.

4.2 Error in MATLAB Model

Once the 10 cm and 21 cm antenna separation configuration was selected, we tested the system across a 180° field of view under worst case conditions. Shown in Figure 26, when the AoA is resolved correctly, the error is within specifications (black, solid lines). The blue diamonds show the calculated angle with the highest certainty value and the larger red stars correspond to the second reported AoA, with a certainty below 0.5. At 5° and 50°, the system resolved the ambiguity incorrectly; however, the correct angle was the second of the two reported angles. Figure 27 shows the primary AoA plotted against the true AoA with the same data. This test verified that the model met the system requirements in cases without ambiguity error. The accuracy was within ±2.5° for all angles between ±85° except for the two angles when the ambiguity was resolved incorrectly. To gain a better understanding of the model, the effects several parameters on error and certainty were tested individually.

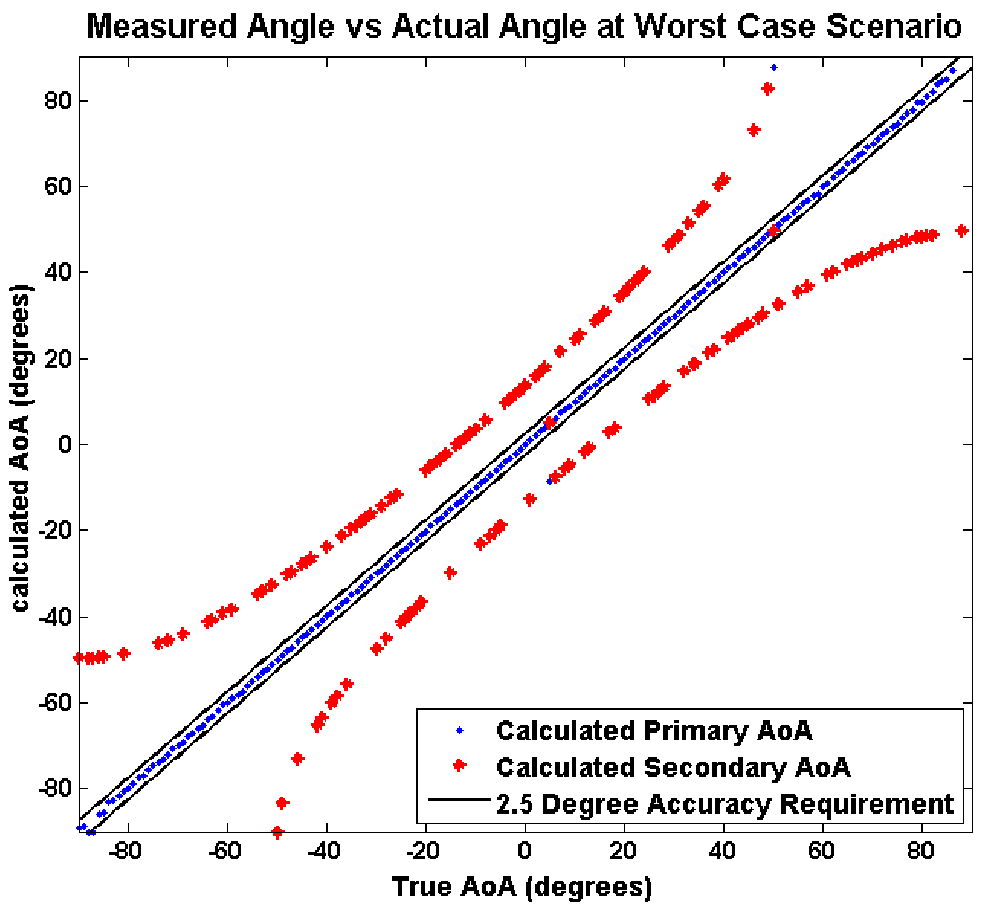


Figure 26: The graph of calculated AoA versus the true AoA. The primary AoA, the angle with the highest associated certainty (blue diamonds), was within our requirements (the solid, black lines) for all but two cases, 5° and 50°. However, this was due to resolving the ambiguity incorrectly, and, in both cases, the second calculated AoA (large red stars) was within 10^{-2} degrees of the true AoA value.

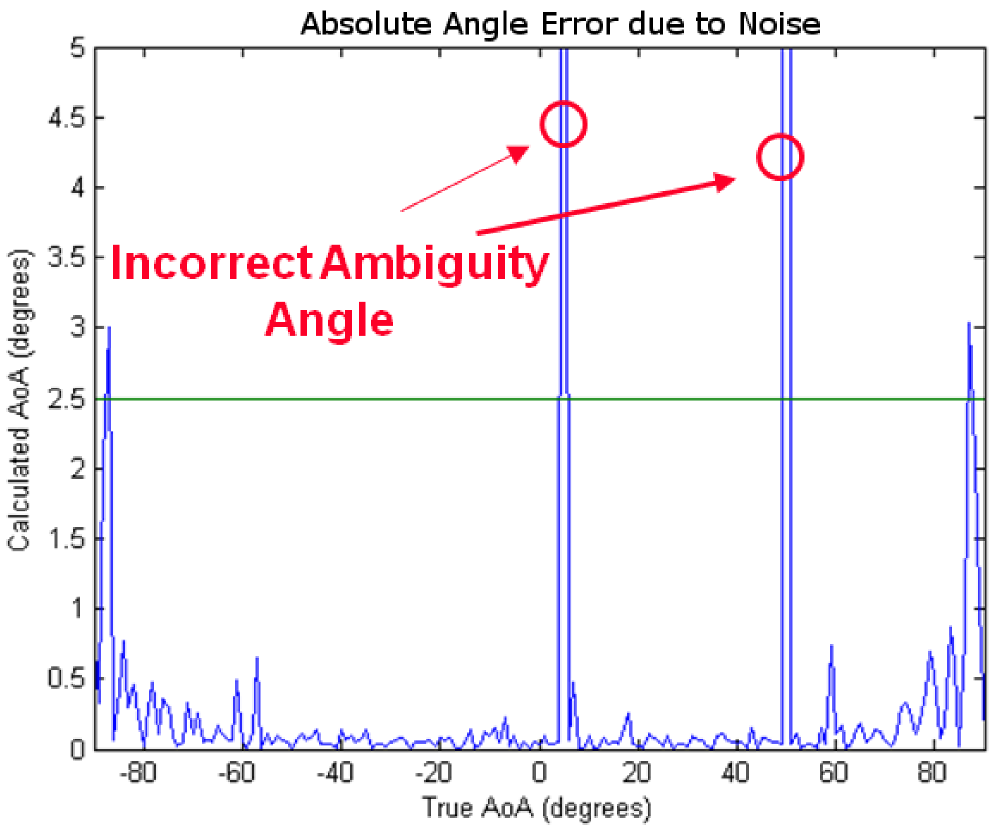


Figure 27: The graph of the primary calculated AoA versus the true AoA. The error is within the requirements of ±2.5° (horizontal green line) for angles between -85° and 85° except for the two cases where the ambiguity was resolved incorrectly (circled in red).

4.3 Error versus Signal Strength

The strength of the signal, and by extension the signal to noise ratio, has the largest effect on error and certainty. Figure 28 shows the certainty for several SNRs: 5 dB, 20 dB, 40 dB and 60 dB. The 20 dB and 60 dB SNRs are the minimum and maximum values of SNR that our system is required to handle. The discontinuities found in the 5 dB plot in Figure 28 are due to the system mistaking the signal for noise and consequently ignoring the data set, thus not performing any AoA calculations. Even when the SNR is at 20 dB, the certainty rarely drops to 0.

Figure 29 shows the error and certainty at 45° for a range of SNRs. The certainty tends to remain in the green region for SNRs above 30 dB. The error appears to decrease exponentially with the SNR.

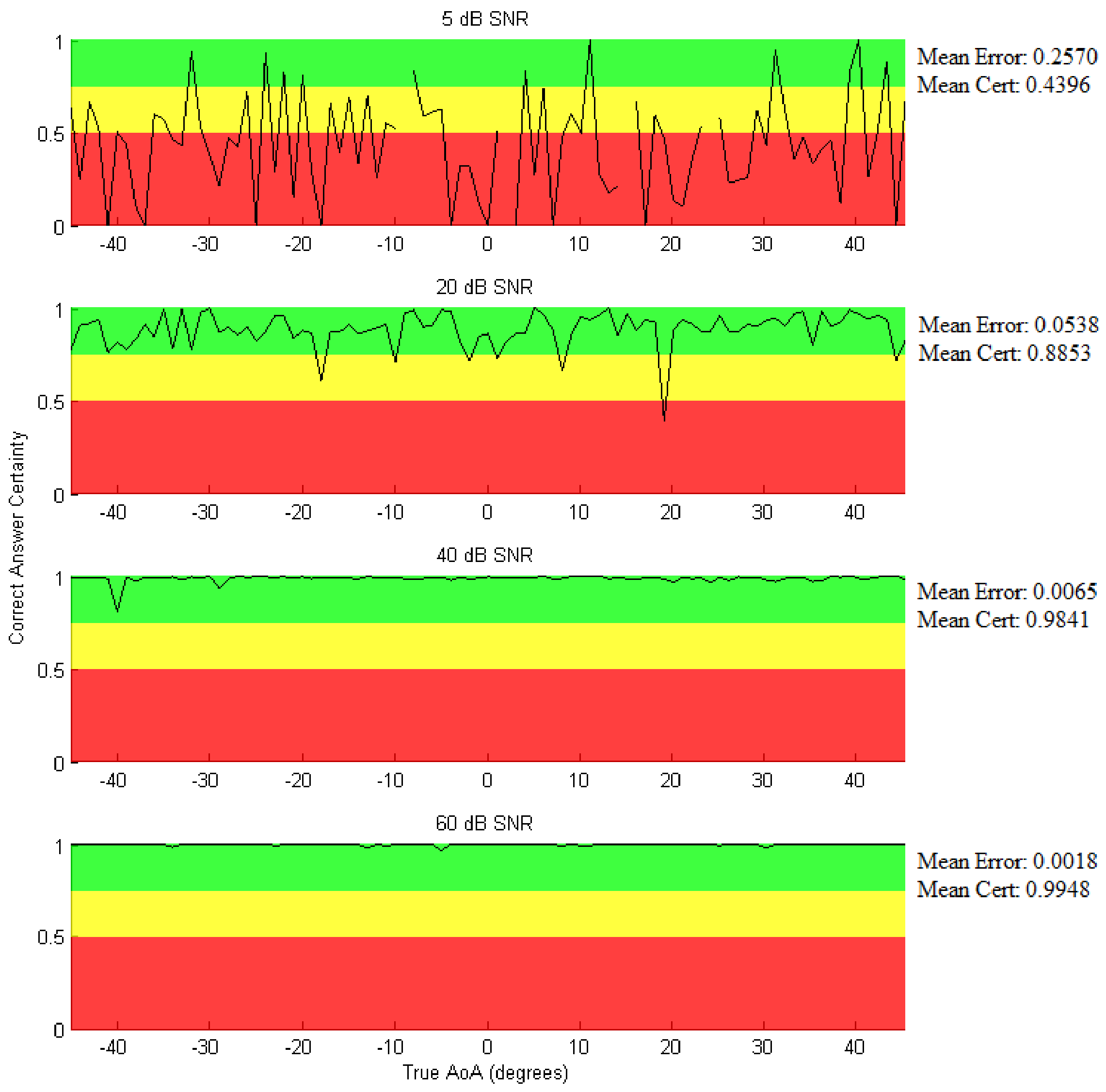
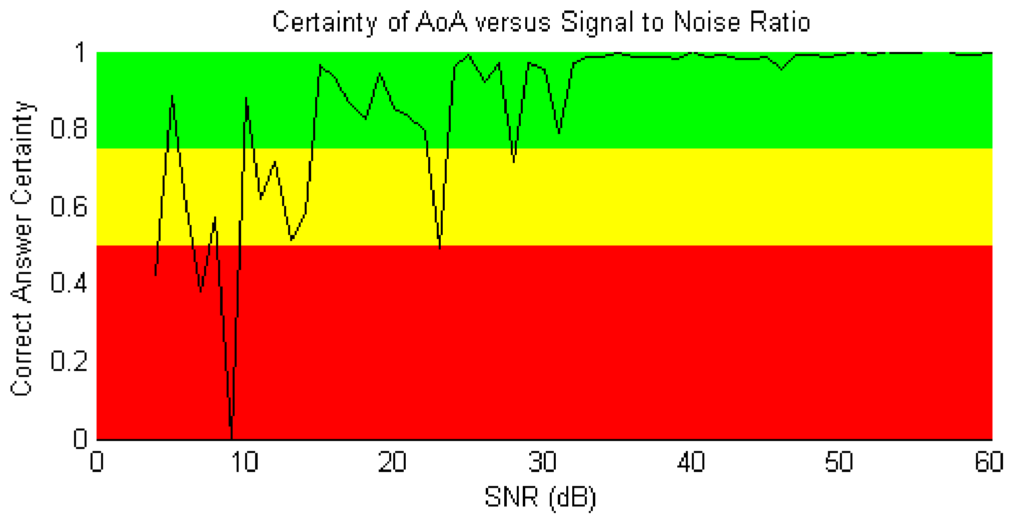


Figure 28: The graph of certainty in AoA calculations vs. true AoA for all angles between -45 and 45 degrees. Four SNRs were tested: 5 dB, 20 dB, 40 dB and 60 dB. When the SNR was set to 5 dB (Top), the signal was not detected in some cases, causing discontinuities in the graph. For SNRs within our 40 dB range, the system rarely dismissed signals as noise and rarely resolved the ambiguity incorrectly.



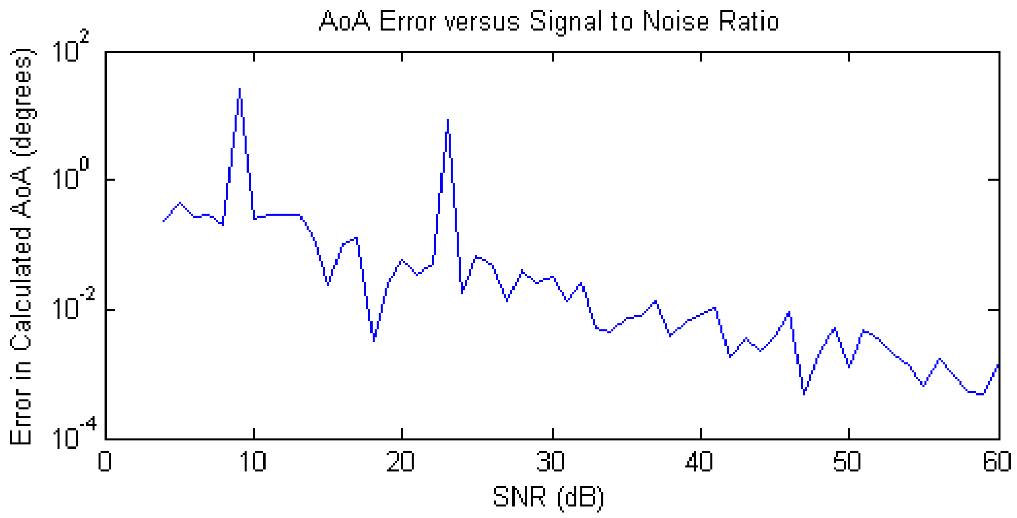


Figure 29: The graphs of certainty and error in AoA calculations vs. the SNR for an AoA of 45° between 5 dB and 60 dB with an SNR step size of 1 dB. The error axis is logarithmic. The certainty rises with SNR and the error decreases in an exponential manner.

4.4 Frequency

Over the X band, carrier frequency has a small but noticeable effect on both certainty and error. Figure 30 shows the average certainty of the correct AoA and error due to noise as a function of carrier frequency with 50 tests per frequency value and a step size of 40 MHz. The average error due to noise and certainty decrease as frequency increases. At 8 GHz, the mean error and certainty were 0.0765° and 0.9203 respectively, while at 12 GHz the mean error and certainty were 0.0558° and 0.8839 respectively.

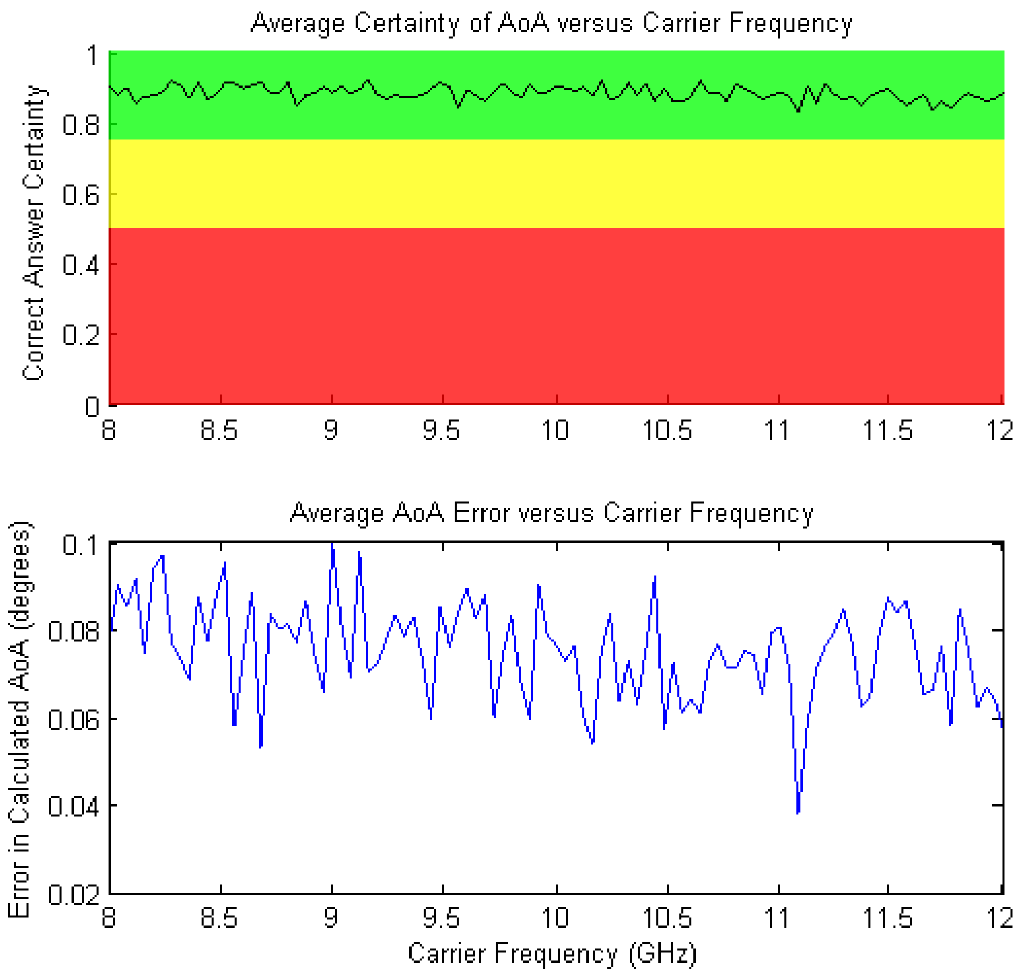


Figure 30: The graph of certainty of the correct AoA calculations vs. the carrier frequency (Top) and error due to noise versus carrier frequency (Bottom). For every carrier frequency between 8 GHz and 12 GHz, with a step size of 40 MHz, the test was run 50 times. The shown certainty and error is the average of the 50 tests. The noise graph ignores error due to phase ambiguity error. As the frequency increases, both the mean error and the mean certainty decrease.

4.5 Emitter Distance

The effects of emitter distance on error and certainty were tested in two ways, by holding the received power constant while allowing the distance to change and by setting the received power as a function of distance, effectively holding the transmitted power constant. Although the latter case is more realistic, the former allowed us to isolate the effects of emitter distance. Figure 31 shows the effects of emitter distance on the certainty and AoA error. At emitter distances above 20 m changing the emitter distance does not have a noticeable effect on the certainty or error. As long as the emitter is more than 20 m away, the lines drawn from the antennas to the emitter can be considered parallel and the assumptions made in the phase calculation are valid. The minimum range the system must operate at is 1 km, so changes in emitter distance do not have a large effect on the system.

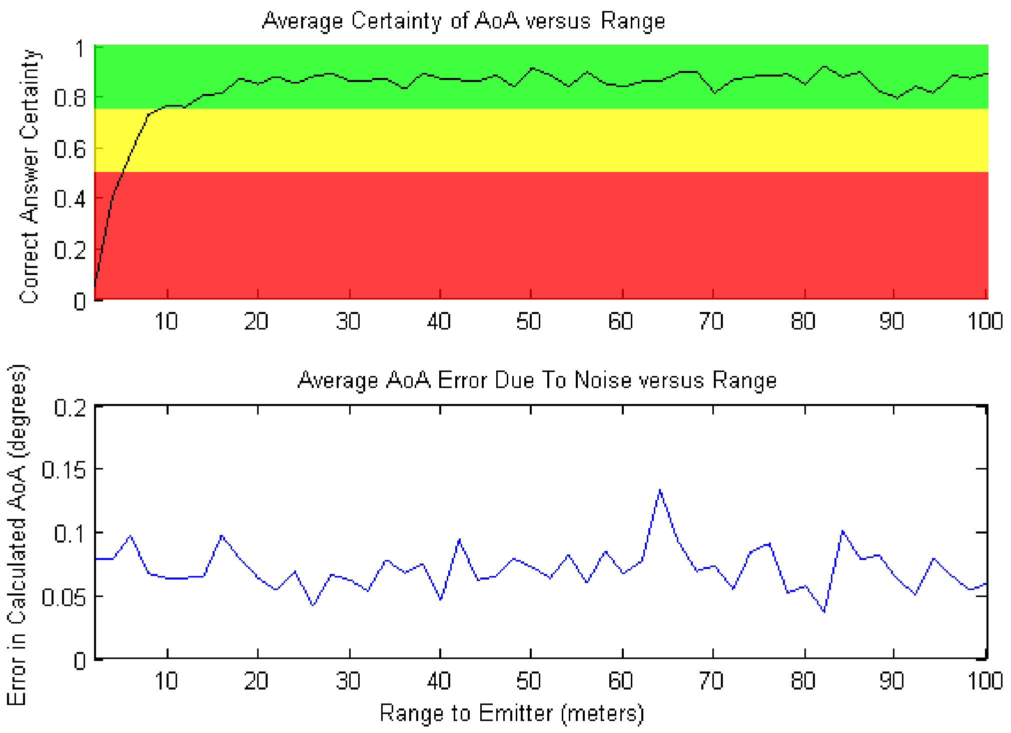


Figure 31: The graph of certainty in AoA calculations (Top) and error in AoA calculation (Bottom) vs. emitter distance from 2 to 100 m. The tests were run with worst case conditions with a true AoA of 45°.

When isolated, operational emitter distance does not have a large effect on the certainty or error in the system. A more accurate test of the system, however, has emitter distance linked to signal strength. Linking these parameters models an emitter with constant transmitted power and varying distance. A variation of the radar range equation (Equation 4 from Section 2.1.1) gives the received power in decibels as shown in Equation 34:

$$P_r = P_t - 10log_{10}\left(\frac{4\pi}{\lambda^2}\right) + G_r - 10log_{10}\left(\frac{1}{4\pi d^2}\right) + G_l, \tag{34}$$

where P_r is the received power, λ is the RF wavelength, d is the distance to the emitter, P_t is the effective radiated power from the emitter, G_r is the gain of the receiving antenna, and G_l is the gain of a local noise amplifier. The effective radiated power was simulated at 10 kW with a 30 dB gain from the transmitting antennas to represent reasonable values for an X band radar system. The gain of the receiver antenna was modeled as 10 dB. A simulated local noise amplifier was used so that the maximum received power from an emitter one kilometer from the system radiating a 12 GHz signal would be equal to the maximum power the ADC can accurately read. With no local amplifier, the received power was found to be -4 dBm in these conditions, compared to a desired maximum power of 4 dBm, so an 8 dB amplifier was included.

Figure 32 shows how the power at the ADCs changes with distance for an emitter with an effective radiated power of 100 dBm and a total gain of about 18 dB on the DF system.

Given the received signal strengths from Figure 32, simulations were run with various emitter distances and their corresponding SNRs. Figures 33 and 34 show how the error increases and certainty decreases with distance. As expected, increasing the emitter distance produces the same results as reducing the SNR. The large change in certainty and error from changing the SNR overwhelms the effects of changing the range. The SNR ratio reached the edge of the system requirements at 100 km. However, even at 500 km, the error due to noise is well within the $\pm 2.5^\circ$ requirement.

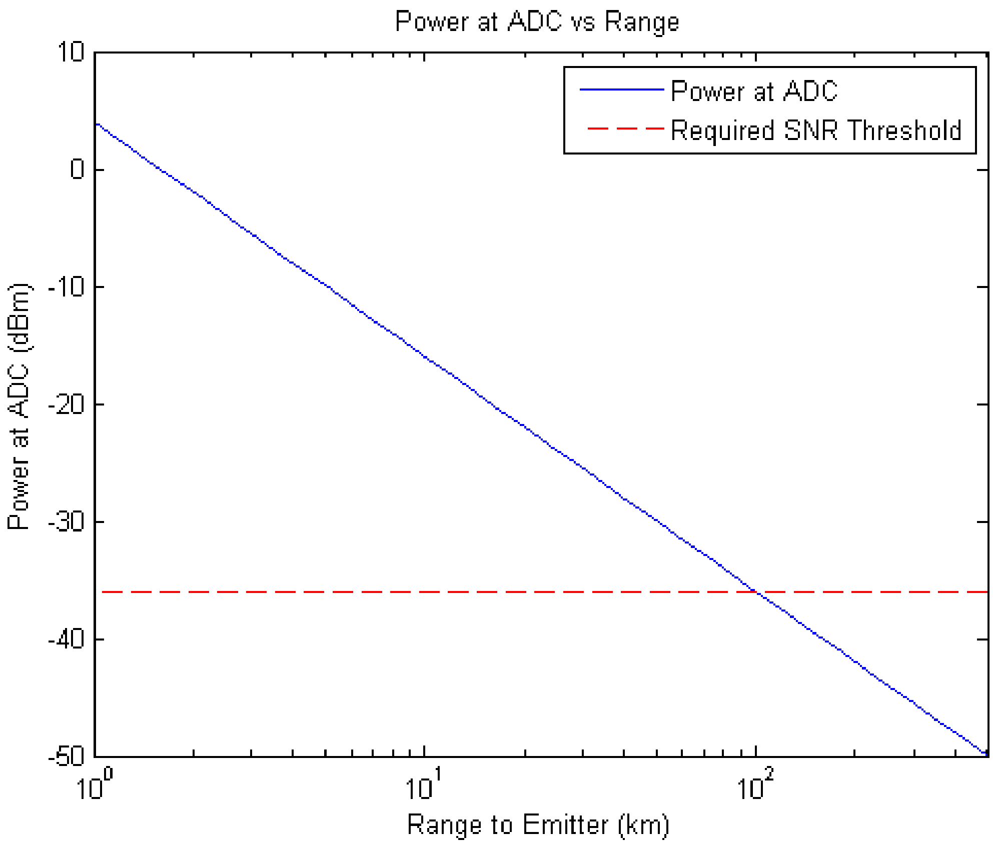


Figure 32: The graph showing how received power changes as distance to emitter increases, as determined by Equation 34. At 1 km from the emitter, the system receives signals at 4 dBm, 0 dBfs. The received power decreases proportionally to the square of the distance. At 100 km the signal strength at the ADCs is -36 dBm, -40 dBfs, the minimum signal strength the system is required to process.

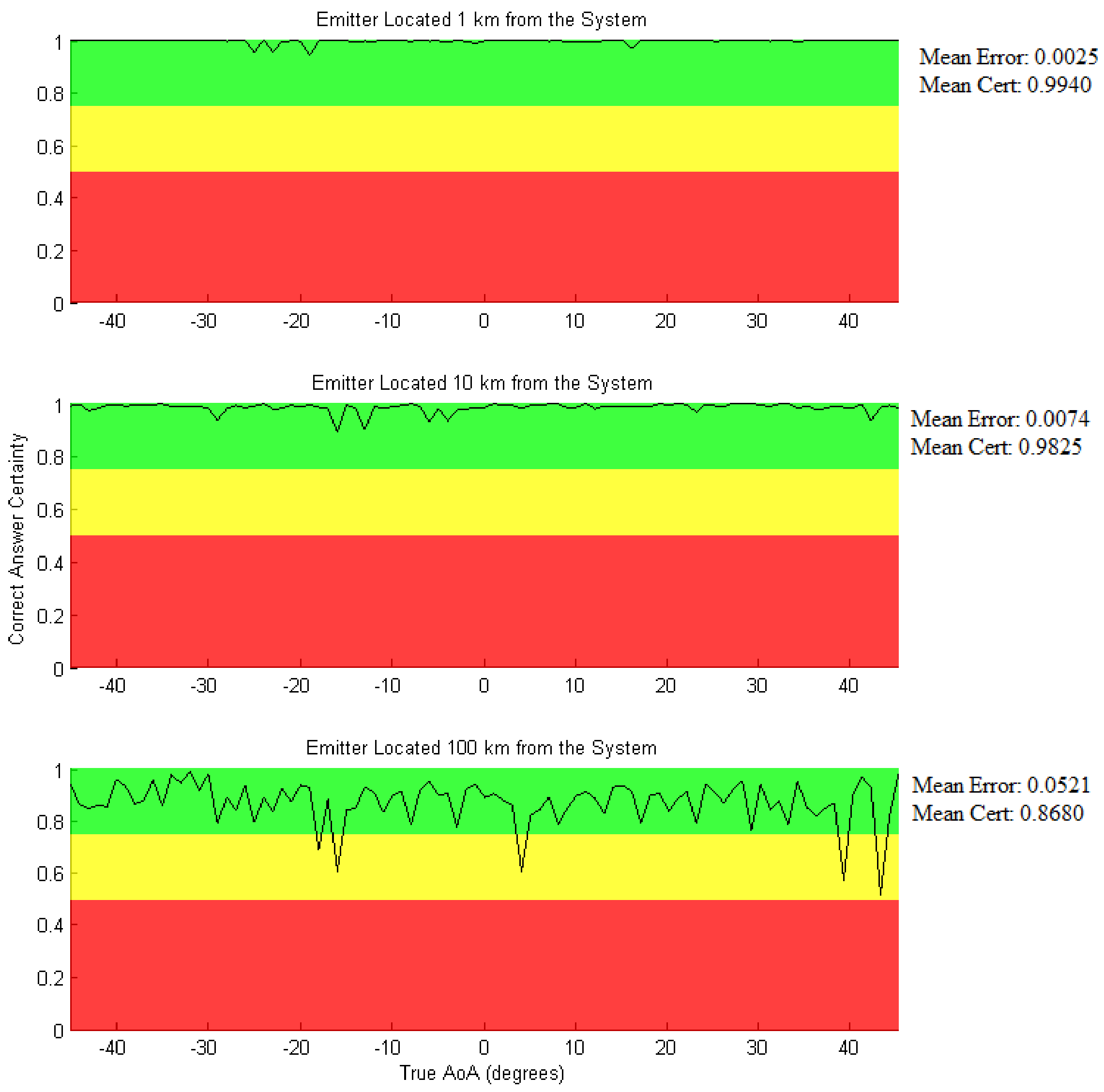


Figure 33: The graph of certainty in AoA calculations vs. emitter distance for all angles between -45° and 45°. The effective radiated power of the emitter was held constant at 100 dBm for each plot. The receiver gain was simulated at 10 dB, with an 8 dB local noise amplifier. The carrier frequency of the signal was 12 GHz with a noise floor of -56 dBm. At 100 km, the SNR reaches the minimum value specified for the system, causing ambiguities and erroneous ambiguity resolutions to appear more frequently.

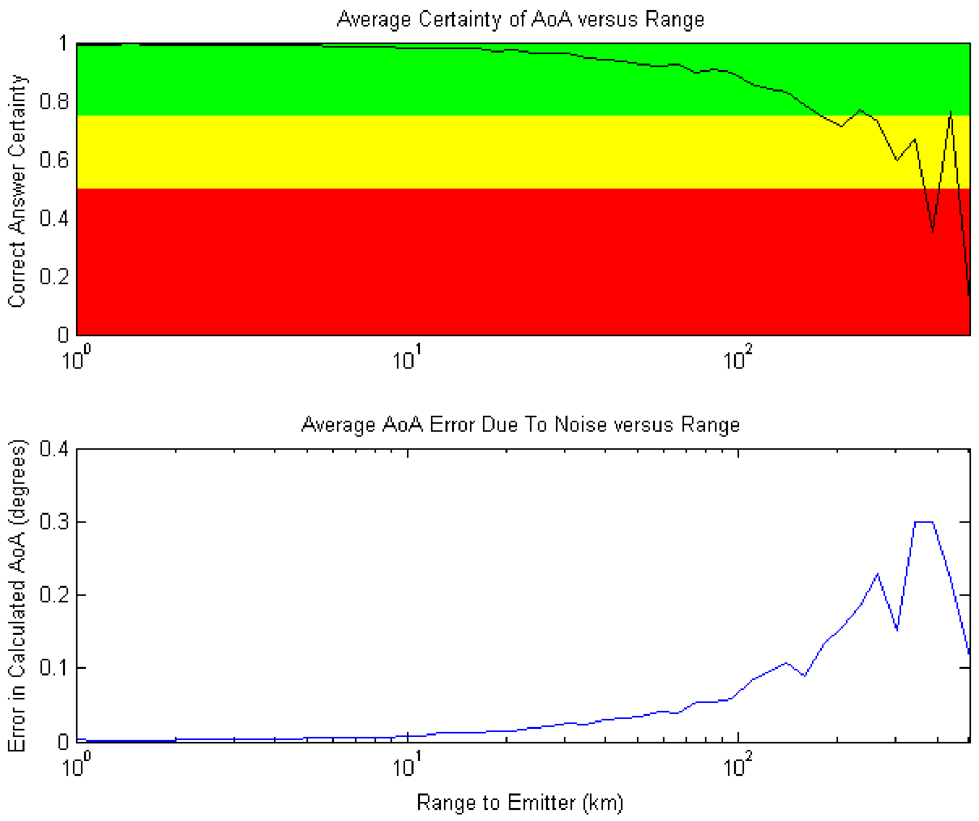


Figure 34: The graph of certainty of the correct AoA, the error in AoA calculations due to noise, and the number of times a signal was detected over 100 pulses vs. Emitter Distance for 50 distances logarithmically spaced between 1 and 500 km. Using an effective radiated power of 100 dBm and a total gain of the DF system of 18 dB, the limit of the system's 40 dB dynamic range requirement was reached when the distance was 100 km. At distances beyond 100 km, the error in AoA calculation due to noise is within specifications, but the certainty drops significantly and the probability of resolving the phase ambiguity incorrectly increases.

5 Prototype Methodology

5.1 System Design

Based on the results of the MATLAB simulation, a prototype DF system was created. The setup of the proposed full hardware system is shown in Figure 35. The RF energy is captured by three horn antennas, placed in a line with separations of 10 cm (between the first and second antenna) and 21 cm (between the first and third antenna). Directional horn antennas were chosen in order to increase the directional gain for the desired field of view. Once received, the signals are down converted to an intermediate frequency between 15 and 115 MHz and then sampled by 500 MHz ADCs.

The front end hardware, shown in Figure 36, uses Texas Intstruments ADS5400 ADCs with 12-bit resolution and a 500 mV full scale range. The ADCs are located on a 4DSP FMC110 daughter card which connects to a Xilinx ML605 development board. The development board has a Xilinx Virtex-6 LX240T field-programmable gate array (FPGA) which captures the digital data from the ADCs. Each FMC110 card contains two ADCs, so two cards are needed to support three antennas. All of the hardware used in this project was provided by Lincoln Laboratory, so the main focus of the project was the software systems and the communication between the algorithm and the hardware.

The ML605 boards communicate with the PC running the DF algorithm serially over a USB connection. The computers used in developing and testing the system were PCs running Windows 7 64-bit with twelve gigabytes of ram and six cores running at 3.5 GHz. The DF algorithm reads two batches per antenna from the FPGA, determines the AoA of the radar signal, and then sends the result to the GUI over TCP. The GUI listens for connections on network port 64185 and then reads in any results sent to it by the DF algorithm. The GUI performs error checking to resolve any uncertain results and then

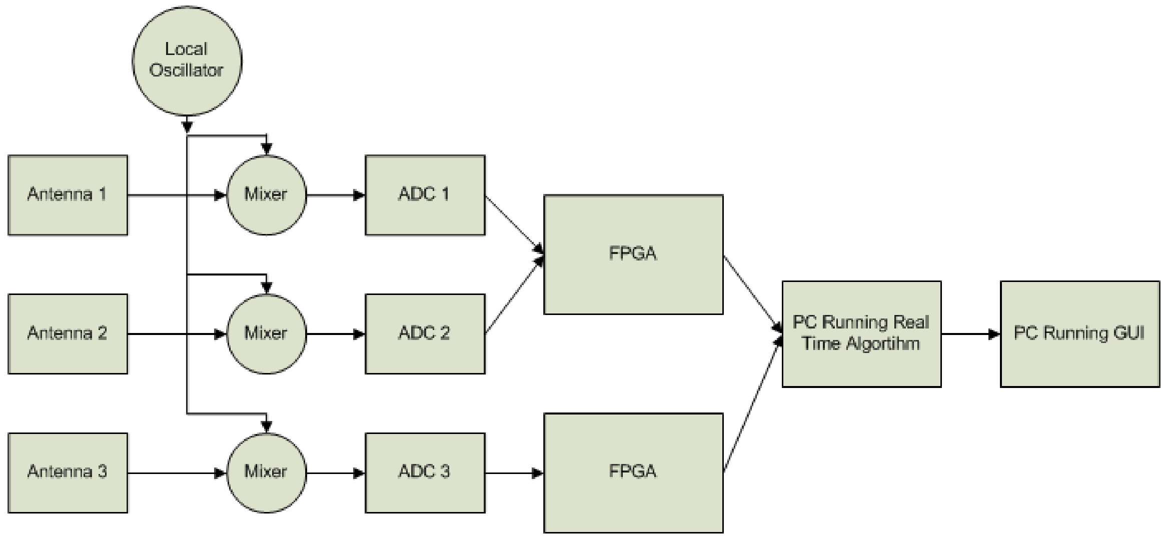


Figure 35: Block diagram showing the setup for the prototype DF system.

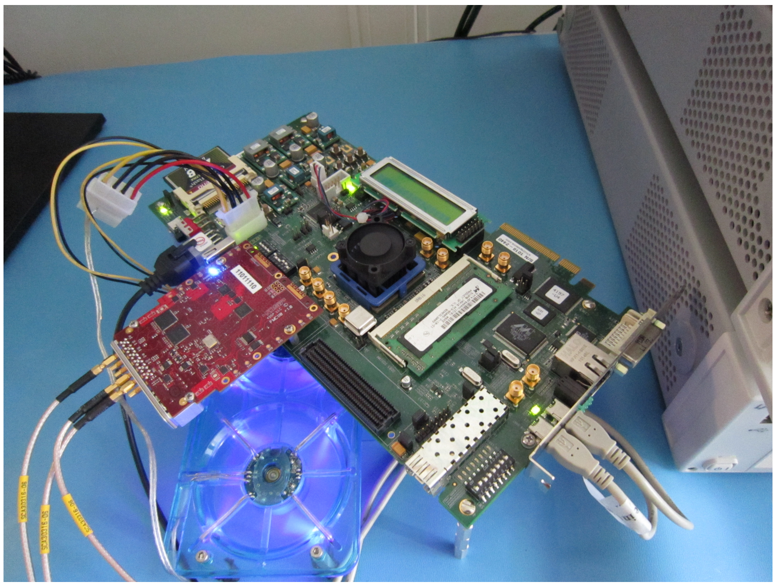


Figure 36: A picture of the hardware supplied by Lincoln Laboratory. The development board is shown with the daughter card (red) and FPGA (under the black fan).

displays the information to the operator.

The system was originally designed to use a three antenna interferometer; however, one of the FMC110 cards was not operational, meaning there were only two ADCs available. The prototype, therefore, was modified to allow for either two or three channel inputs. Antennas were not available for testing, so the two channel hardware implementation was tested using signal generators as shown in Figure 37 and the three channel system was tested using simulated MATLAB data. The processing steps for the two configurations are almost identical. The only difference is that the three channel system resolves the ambiguities while the two channel implementation does not.

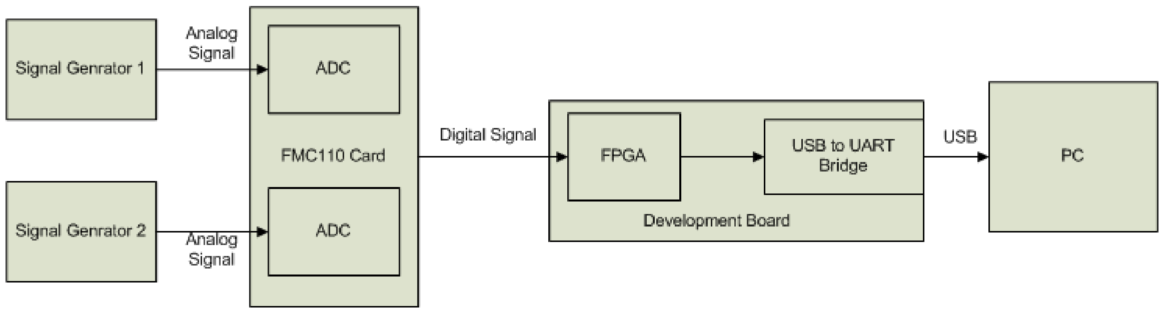


Figure 37: A block diagram showing the test setup of the two channel hardware implementation. The signal generators produce signals at IF. These signals are sampled by the ADCs and passed to a PC running the DF algorithm over a USB connection.

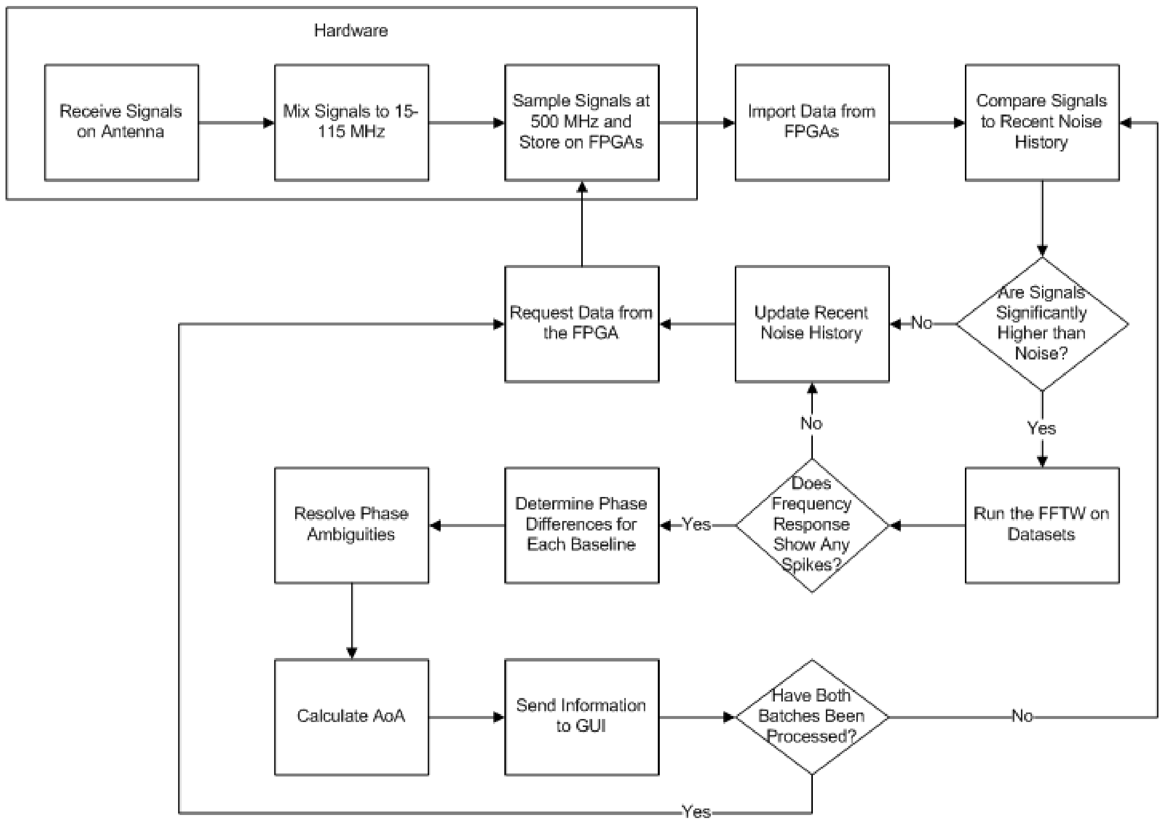

Hardware
Receive Signals on Antenna
Mix Signals to 15-115 MHz
Sample Signals at 500 MHz and Store on FPGAs
Import Data from FPGAs
Compare Signals to Recent Noise History
Are Signals Significantly Higher than Noise?
No
Update Recent Noise History
Request Data from the FPGA
Yes
Run the FFTW on Datasets
Does Frequency Response Show Any Spikes?
No
Yes
Determine Phase Differences for Each Baseline
Resolve Phase Ambiguities
Calculate AoA
Send Information to GUI
Have Both Batches Been Processed?
No
Yes

Figure 38: Control flow diagram of the prototype direction finding system.

5.2 DF Algorithm

In order to improve the performance of the prototype DF system, the final DF algorithm was implemented in C rather than in MATLAB. The flow of control in the C algorithm is shown in Figure 38. The C code is similar to the MATLAB algorithm with many of the functions implemented in the same way. The main difference was that MATLAB provides a number of high level functions that C does not, such as vector operations, signal processing functions, and simplified memory management. Therefore, the C algorithm required some additional components in order to perform its calculations and interface with the other portions of the system. In order to convert the algorithm, the vector operations were turned into iterative processes and the signal processing functions were implemented using lower level operations. MATLAB functions with multiple return values were replicated by specifying meanings for particular return values, such as -1 or NULL. If there were no unique return values or this method was insufficient, passing a pointer argument or defining a structure allowed the function to return more than one variable. The FFT function was implemented in the C algorithm by utilizing an existing C library named the Fastest Fourier Transform in the West (FFTW).

In the MATLAB simulation all of the inputs and outputs were contained within the MATLAB environment. Consequently, specifying arguments and recording output was straightforward. In contrast, the C algorithm does not have immediate access to the input signals and the physical system parameters. In order to access the signals, the C algorithm requires a component that communicates with the FPGA and parses the transmission. To store results, the algorithm contains both a logger and a component to manage the TCP connection with the GUI. The system stores its physical parameters in a configuration file which the algorithm reads on start up. This file specifies the antennas' separations, the local oscillator frequency, the number of antennas used, the configuration of the logging component, and the host name of the PC running the GUI. The configuration file allows the algorithm to adapt to a different system setup without requiring internal modification or recompilation.

The following sections detail the components unique to the C implementation of the algorithm: the FPGA communication, the FFTW library, and the logger.

5.2.1 FPGA Communication

In the MATLAB model, the signals from the signal simulator were passed as parameters to the DF algorithm directly; however, the hardware implementation must communicate with the FPGA to read the samples. FPGA communication was only implemented in the two channel system as a third ADC was not available for the three channel system. Several methods are available to communicate between a host computer and an embedded system. USB was chosen in spite of its slow transfer rate compared to other methods because USB communication was the simplest to implement while still meeting the timing requirement. This project focused on rapidly developing a prototype with the assumption that an operational system would use a faster method of communication.

To gather data from the FPGA via USB, the system initializes communications with the FPGA, reads in the data, and then parses the received data to extract integer outputs. The first step in initializing the communication is verifying that the communication port exists and is not in use. When the port is present and available, the algorithm configures the port for the specific settings used to communicate with the FPGA: 1,228,800 baud, 8 bits per byte, one stop bit, and no parity. The algorithm sets maximum timeouts of 50 ms to prevent missing data while also preventing perpetual stalling if communication is interrupted. Once the port is set up, communication with the FPGA is initialized. The algorithm sends a one-line command to the FPGA which instructs the device to take sample data and calibrate for timing offsets due to perform internal calibration of the ADC devices. When the FPGA notifies the host machine that the process is complete, the FPGA, the algorithm, and the host machine are all ready to sample and transfer data.

The data obtained by the ADCs are captured in the FPGA. The host computer communicates with the FPGA via a Silicon Labs CP2103 USB to UART Bridge chip embedded in the development board. The bridge allows the development board to plug directly into a computer's USB port and send and receive characters through that port one word at a time. The computer communicates over this line at a speed of 1,228,800 baud, the fastest possible speed for successful communication. With three 12-bit ADCs, each working at 500 MHz, we would need a speed of 18,000,000,000 baud to transfer the data directly into the PC while sampling continuously and not missing any samples. This high communication rate, which is 10^4 times larger than the maximum USB communication rate, is why the system samples and processes in batches rather than continuously.

When the algorithm is ready for more input, it asks the FPGA to sample 8192 data points from each ADC. The computer sends a one-line command to begin capturing the data on the FPGA, and another one-line command for the algorithm to read in the data and store them on the host machine. To ensure that the FPGA is done sampling before the algorithm tries to read the data, the algorithm polls the output for an indication of completion. Once the process is complete, the algorithm is able to read in the captured data. The FPGA sends the sampled data over the USB connection as a set of 8192 12-bit samples per antenna, for a total of 98,304 bits of data. At a speed of 1,228,800 baud, the shortest possible amount of time required to read these data is 80 ms. However, the data are not sent simply as a 98,304-bit long stream, but as a series of formatted characters with a marked beginning and an end with separations between the samples. With a two antenna setup, every set of two samples is stored as three 8-bit characters separated by spaces. In effect, 3 bits of data are sent for every 2 bits of sampled signal, plus the 240 bits of header and 64 bits at the end:

$$\frac{4096\ samples}{batch}\frac{2\ batches}{transfer*antenna}(2\ antennas)(\frac{12\ bits}{sample})\left(\frac{3}{2}\right)+240\ \frac{bits}{transfer}$$
$$+64\ \frac{bits}{transfer}=295216\ \frac{bits}{transfer}. \quad (35)$$

Serial communication also requires one stop bit for every byte of data transferred, so the shortest possible time to complete one transfer of a given size is given by:

$$T=\frac{9B}{8S}, \quad (36)$$

where T is time to transfer data, B is bits to transfer and S is the rate of transfer in bits per second. Equation 36 yields 270.3 ms as the minimal possible transfer time for two batches of data. The two batches transferred during this time are processed separately so two independent AoAs can be calculated for each transfer. The delay due to data transfer comprises a large portion of the system's latency, so any further development of the system should investigate faster methods of communication.

Data must be parsed into meaningful numbers before processing can occur. The data are sent and received in an "abc def" format with "a" and the top half of "b" representing one sample from antenna 1, and the bottom half of "b" and all of "c" another sample from antenna 1. The "def" follows the same pattern with antenna 2, with a new line and a carriage return before the next line of data. An example of this format is shown in Figure 39. The use of a consistent format simplifies the parsing process because the position of the data is already known so the process of parsing becomes a series of bit masks, shifting, and adding the separate parts of the samples together. Difficulties arise when a character is lost during communication because this shifts every subsequent character from its expected place. By verifying that the fourth character of every line is a space and declaring non-compliant lines invalid, shifted characters only remove one line of data instead of disrupting the entire file. The FPGA communication was only implemented

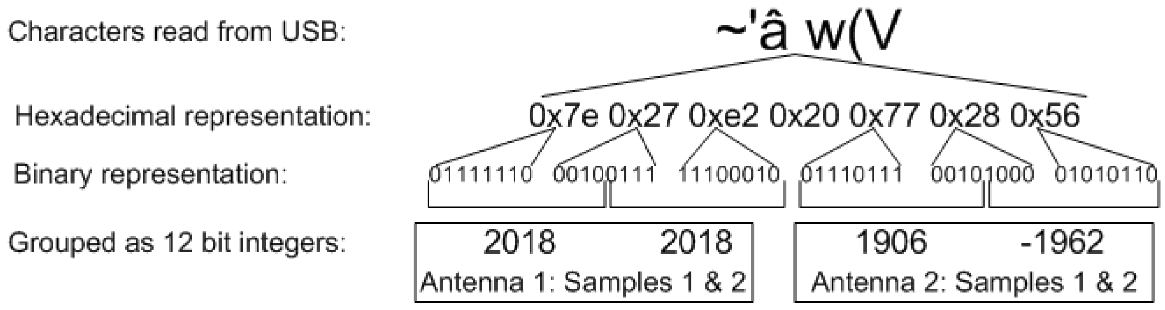


Figure 39: Sample data read in by the algorithm over the USB. The translation from USB read character to hexadecimal and binary is shown.

for the two channel mode. Adding a third antenna to the system requires the use of an additional board because there are only two ADCs on each FMC110 card. The host machine could communicate with this board in parallel with the first, so there would not be a large increase in the communications delay. To provide a means for three antenna input without the functioning board, an algorithm component was created to read signals in from text files. This component reads files of 12-bit integers and stores them as if they were the results of the parsed output. The MATLAB signal generator was used to write these files allowing us to use the three antenna signals simulated by the MATLAB model as inputs to the C system. Unlike a real environment, in which the system runs with no definitive endpoint, the text files have finite amounts of data which give the algorithm a fixed termination condition. The playback function provides absolute repeatably for testing the system.

5.2.2 Discrete Fourier Transform

The C algorithm uses the Fourier transform to determine the phase and frequency of the incoming signals; however, unlike MATLAB, C does not have a built-in FFT algorithm. The discrete Fourier transform requires a large number of computations, $O(N^2)$, so choosing an efficient implementation of the DFT is important to ensure that the algorithm runs within the time constraints. MATLAB uses a library called the Fastest Fourier Transform in the West (FFTW) to compute discrete Fourier transforms. FFTW is a free C library developed at MIT and licensed under the GNU General Public License (GPL) (Frigo and Johnson, 2005). In general, software under the GPL is not suitable for any sensitive applications because using a library licensed under the GPL requires the entire application to take on the GPL. Fortunately, Lincoln Laboratory is owned by MIT which grants non-GPL licenses. The FFTW algorithm is already used within Group 108 for other applications.

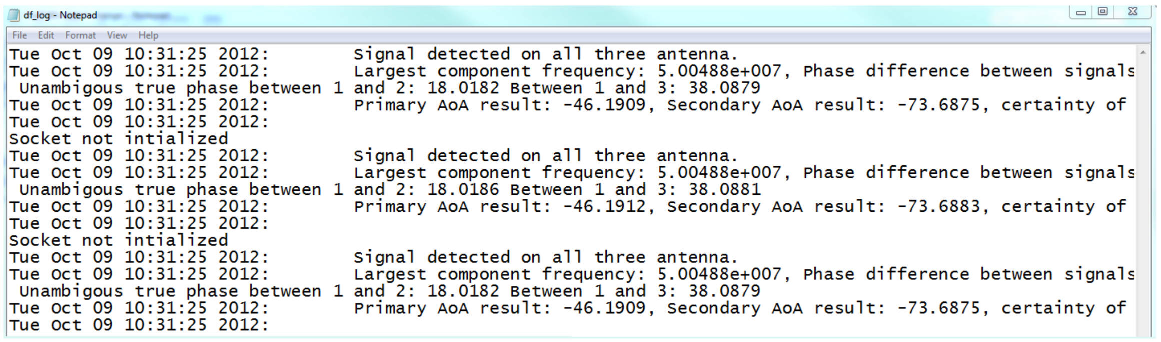
The FFTW library provides a number of features that make it ideal for this application. The most important of these features is processing speed. In a study of over 50 FFT algorithms, it was found that FFTW typically performs faster than most other free FFT libraries and is often comparable to those adapted for specific platforms, making FFTW one of the best options for rapid calculations (Frigo and Johnson, 2005). The FFTW derives its computational speed from its plan system. Before running the DFT, FFTW generates a plan dependent upon the size of the arrays to be transformed, the number of terms used, and the specific runtime environment. In order to determine the plan, the FFTW library runs a number of different DFTs using different algorithms. The primary FFT algorithm used by FFTW is the Cooley-Tukey algorithm. The core principle of the Cooley-Tukey algorithm is to split one DFT into n_1 transforms, each of size n_2. There is no obvious ideal choice for n_2, or for the order in which to decompose, calculate, and reassemble the separate DFTs. The best solutions to these problems are dependent on the particular hardware used and on the particular configuration at runtime. Therefore, FFTW runs Cooley-Tukey DFTs with different parameters and orders to determine which plan will run the fastest. There is a significant cost associated with this planning process, as each DFT runs in O(Nlog(N)) computations, but the plan generated can be used for any number of DFTs (Frigo and Johnson, 2005). The FFTW is meant to plan once, which can take several seconds, and then run many times on that plan. The system always runs DFTs of the same size (4096 samples), so the algorithm only needs to perform the expensive planning computation once. When the algorithm must compute a new plan, it uses FFTW's wisdom feature to save the old plan to avoid performing the same plan computation multiple times. When a plan is generated, the wisdom feature automatically saves the data for that plan in the library's shared memory. If another plan of the same size is requested later, FFTW uses the plan that was already generated and saved.

Although speed was the primary reason for selecting the FFTW, the FFTW's other features proved advantageous as well. The data input to the FFTW is always comprised of real numbers read in from the FPGA or from text files. FFTW allows for transforms on purely real input to save the effort of converting between real and complex formats. It is usable by any platform that can compile C, so it works on the Windows PCs used for development as well as any other machines which will eventually host the application. Finally, FFTW provides in depth online documentation and tutorials which made it easy to integrate into the system (Frigo and Johnson, 2005).

5.2.3 Logging

The logging component of our real time algorithm was very useful for testing and archiving purposes. It provided the ability to view the output of the system for each sample and after each stage of the algorithm. Furthermore, the logger will remain useful after the system is out of testing as it produces a persistent record of the algorithm's output. The logged outputs were used in MATLAB and Excel to confirm performance metrics.

In order to support a variety of uses, the logger was developed to provide a number of modes and options. The three modes for the system are human readable output, comma separated values (CSV) output, and raw data output. The human readable output is intended for manual interpretation and analysis as its primary function is aiding in debugging. The output in human readable format is comprised of a timestamp followed by a sentence explaining the data presented and the data itself. An example of the human readable output is shown in Figure 40. The CSV format can be processed in MATLAB and Excel, so it includes column headers and is organized in a table format. A sample of CSV output is shown in Figure 41. The raw data format prints out the byte-by-byte

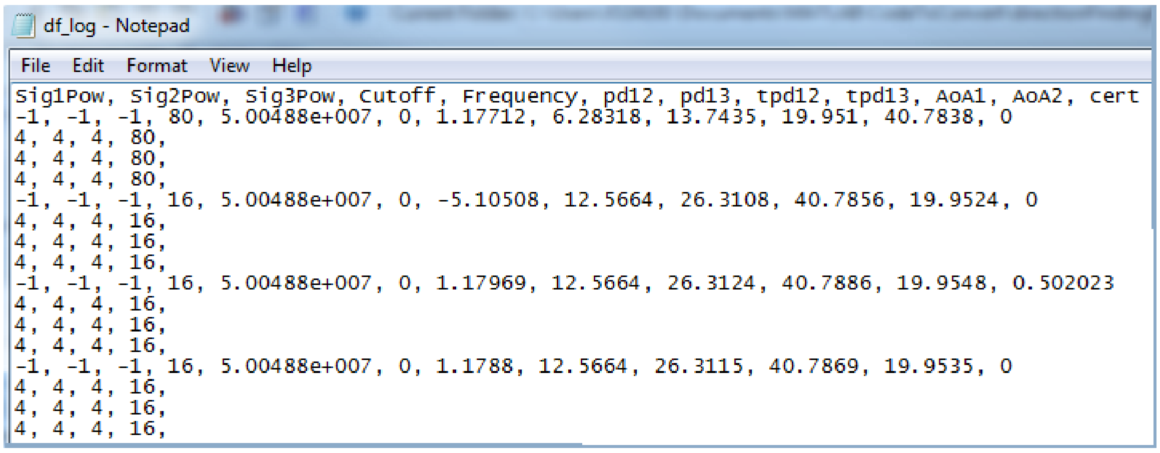


```
Tue Oct 09 10:31:25 2012:       Signal detected on all three antenna.
Tue Oct 09 10:31:25 2012:       Largest component frequency: 5.00488e+007, Phase difference between signals
 Unambigous true phase between 1 and 2: 18.0182 Between 1 and 3: 38.0879
Tue Oct 09 10:31:25 2012:       Primary AoA result: -46.1909, Secondary AoA result: -73.6875, certainty of
Tue Oct 09 10:31:25 2012:
Socket not intialized
Tue Oct 09 10:31:25 2012:       Signal detected on all three antenna.
Tue Oct 09 10:31:25 2012:       Largest component frequency: 5.00488e+007, Phase difference between signals
 Unambigous true phase between 1 and 2: 18.0186 Between 1 and 3: 38.0881
Tue Oct 09 10:31:25 2012:       Primary AoA result: -46.1912, Secondary AoA result: -73.6883, certainty of
Tue Oct 09 10:31:25 2012:
Socket not intialized
Tue Oct 09 10:31:25 2012:       Signal detected on all three antenna.
Tue Oct 09 10:31:25 2012:       Largest component frequency: 5.00488e+007, Phase difference between signals
 Unambigous true phase between 1 and 2: 18.0182 Between 1 and 3: 38.0879
Tue Oct 09 10:31:25 2012:       Primary AoA result: -46.1909, Secondary AoA result: -73.6875, certainty of
Tue Oct 09 10:31:25 2012:
```

Figure 40: Sample output from the logger in human readable format.

information obtained by the system. These data could be used by another program to replicate the exact information and structure produced by the algorithm.

The logger has four options, configurable through a text file, which control the quantity of information output to the log. The first option is simply used to turn logging on or off. In the cases where the absolute highest speed is preferable to a persistent log, or in low memory systems, turning the log off may be desirable. The second option is verbose or short log. The verbose log will print the intermediate calculations for the result. These intermediate calculations include signal power, mean noise level, signal frequency, phase differences, resolved true phase, and final angle of arrival output. The abbreviated log



```
Sig1Pow, Sig2Pow, Sig3Pow, Cutoff, Frequency, pd12, pd13, tpd12, tpd13, AoA1, AoA2, cert
-1, -1, -1, 80, 5.00488e+007, 0, 1.17712, 6.28318, 13.7435, 19.951, 40.7838, 0
4, 4, 4, 80,
4, 4, 4, 80,
4, 4, 4, 80,
-1, -1, -1, 16, 5.00488e+007, 0, -5.10508, 12.5664, 26.3108, 40.7856, 19.9524, 0
4, 4, 4, 16,
4, 4, 4, 16,
4, 4, 4, 16,
-1, -1, -1, 16, 5.00488e+007, 0, 1.17969, 12.5664, 26.3124, 40.7886, 19.9548, 0.502023
4, 4, 4, 16,
4, 4, 4, 16,
4, 4, 4, 16,
-1, -1, -1, 16, 5.00488e+007, 0, 1.1788, 12.5664, 26.3115, 40.7869, 19.9535, 0
4, 4, 4, 16,
4, 4, 4, 16,
4, 4, 4, 16,
```

Figure 41: Sample output from the logger in comma separated value format.

only prints out the final AoA results that are sent to the GUI. Both of the samples shown in Figures 40 and 41 are verbose logs. The third option toggles whether or not to record the signals received from the FPGA. These signals are logged to three separate files, one for each antenna. If verbose logging is also enabled then the DFT of the signals is logged as well. By default, logging is set to human readable mode with short output, and signal logging off.

5.2.4 Parallelization

The prototype system was programmed using linear software constructs. If the algorithm did not meet the processing time constraints, the algorithm was designed such that parts of the code could be reworked to run in parallel to decrease the processing time of the algorithm. The processes that benefit most from running in parallel are the ones that take the longest time to run, such as input-output operations or heavy computations. In this system, the largest sources of delay were the input operations with the FPGA, the communication and output to the GUI, and the computation of the DFT. The GUI runs on a separate machine and is essentially already in parallel, but both the input operations and the DFT computation could benefit from parallel processes.

The input communication with the FPGA was the largest single delay in our system, taking over a third of the time budget. The communication time scales with the amount of data being transferred and the data rate of the USB connection, so it could not be reduced dramatically without changing hardware. The time to finish transferring one data set could not be improved; however, the processing component can be launched on a separate thread while the algorithm requests the next set of data. This technique does not reduce the delay between signal arrival and final output because neither the input portion nor the processing portion will run faster. Launching separate threads, however, does increase the throughput of the system, or number of results computed per second. A diagram showing the differences between a computationally linear and multi-threaded system is shown in Figure 42.

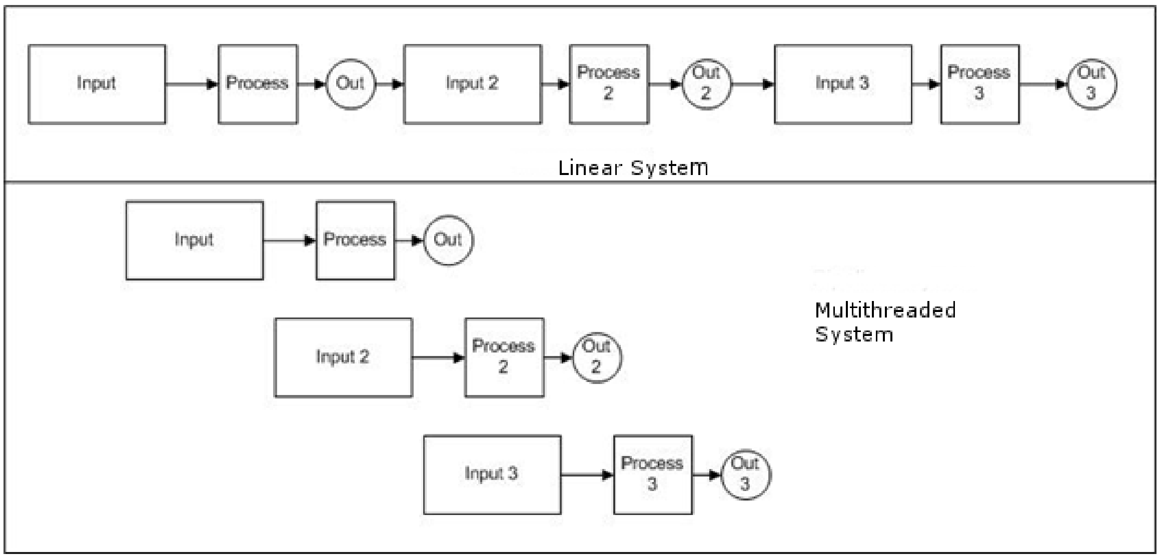


Figure 42: Diagrams representing the timing and control flow for linear and multi-threaded versions of the direction finding algorithm. Each control block represents a separate thread and movement to the right represents increasing time.

The linear system, shown in the top part of the diagram, waits for each output then requests the next input. If reading the input took 0.4 seconds and the processing step took 0.2 seconds, the linear system takes 0.6 seconds to produce output for each sample and 1.8 seconds for all three outputs. In the multi-threaded system, each output is still calculated in 0.6 seconds after the input arrives, but the inputs are read earlier. Instead of the second input being read at 0.6 seconds, after the whole first calculation, it is read after 0.4 seconds, immediately after the first input step is complete. Reading the input earlier means that after the first result there is a new output every 0.4 seconds, rather than every 0.6 seconds. The data are still 0.6 seconds out of date, but there is less time between refreshes, giving the operator a higher resolution view of the presence and movement of emitters. In reality the delay will be slightly more than 0.6 seconds because of the delays associated with creating and managing threads.

Assigning each batch to its own thread can help improve the throughput of our system, but does not help the delay, or latency, of it. In order to reduce the latency, one of the components must take less time to run. While the input component is hard to reduce without hardware changes, using parallel processing can help reduce the time to process the signal, specifically the time to take the DFT. The components of the processing algorithm are shown in Figure 43. The algorithm takes in three signals, one from each

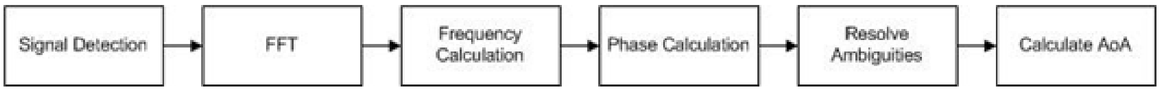


Figure 43: DF algorithm components.

antenna, and processes them. Parallelization can be used to speed up any processing step where the computation is independent for each signal. The three steps that meet these requirements are signal detection, the DFT, and the frequency calculation. These steps do not need any of the information contained in other signals to run. The other three steps compare multiple signals and therefore are not candidates for parallelization. The signal detection step is also not a strong candidate because if only noise is detected over the duration of a batch then the entire process should stop for all three signals. If this process is parallelized, the short circuit effect is lost. The frequency calculation can easily be parallelized, but because the DFT is run using an outside library, it is risky to assume that it is safe to run it in parallel. If the FFTW library uses shared or static resources during its DFT computations, then it cannot run in parallel without modifying the library. Fortunately, the FFTW can be used in parallel as long as separate plans are used. The FFTW plans are stored in shared memory so that they can be reused, but the actual calculation of the DFT can be run in parallel. Ideally, parallelizing these steps on each input would reduce their total runtime to the time it takes to process one input. The total runtime of the step, however, is the runtime of the slowest of the three computations. Managing the threads can also cause delays, meaning that the total runtime will be greater than the time to process one input. While the linear algorithm does meet the processing speed requirement on the machines used, these parallelization techniques could still be useful and may be necessary if the algorithm is run on a slower processor or with a larger sample size.

5.2.5 TCP Communication

In the system being designed by Group 108 the processing algorithm will be running on a separate machine from the one which the operator physically interacts with. Therefore, the C algorithm and the GUI were designed to run on separate computers with Ethernet connections. The system uses TCP to communicate across the Ethernet connection. TCP stands for Transmission Control Protocol and provides a reliable data stream between two computers. The other main method for Ethernet communication is User Datagram Protocol (UDP) which sends a collection of data, called a datagram. UDP is generally faster than TCP because TCP sends acknowledgement signals every time it receives a message. UDP, however, makes no assurances of delivery or order while TCP guarantees delivery in the order of transmission. The connection distance and message size, 44-120 bytes, are both small meaning that the latency will be low regardless of protocol used. Reliable communication is a necessity as each message contains information about an emitter which might only be found once. For these reasons we chose to use a TCP connection as opposed to UDP.

TCP requires a connection between two programs: the client and the server. The server listens on a specific network port, 64185 in the system, and waits for a client to connect to it. When the client connects to port 64185, the server chooses another free port and begins to communicate with the client on that port. In our application the C algorithm is the client and the GUI is the server. When the GUI is run, it opens and listens on port 64185 and waits for the algorithm to connect to it. When the algorithm runs, it connects to the server specified in its configuration file. If the algorithm successfully connects, it begins to send its results to the GUI. When the algorithm finishes running or is terminated it disconnects from the GUI which will continue running and wait for another connection. The GUI uses Java's built in TCP support through the ServerSocket class. C doesn't have any built-in TCP support, so the C algorithm uses the Winsock2 library provided by Windows to manage its connection.

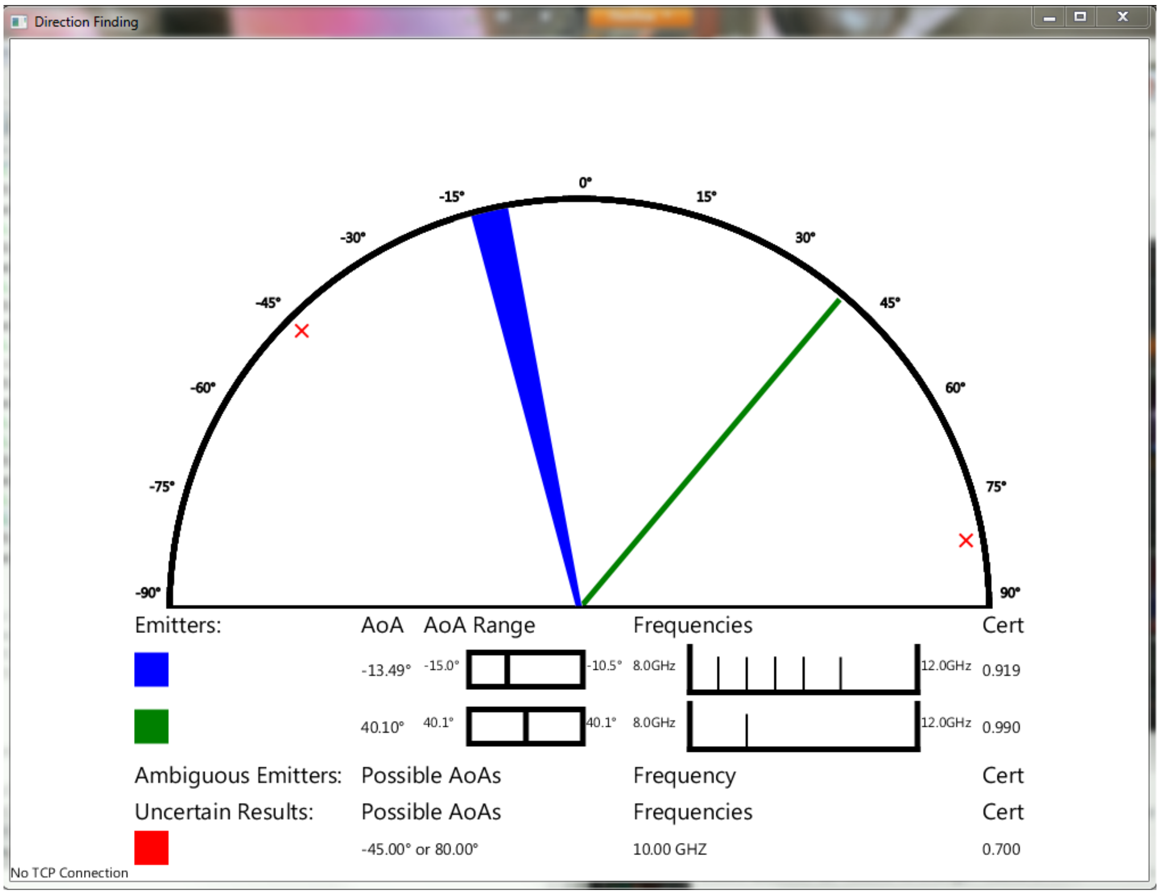
In order for the GUI to interpret the messages from the algorithm they must always be in the same format. To reduce message size the data are sent in raw byte format, rather than in text. The byte format is little endian as both the system are run on Windows machines. The first four bytes of the message contain a two's complement integer specifying the number of antennas used (two or three). Depending on the number of antennas used and the antenna separation, the number of angles calculated changes, so the next four bytes contain a integer specifying the number of angles calculated. This value is always two in a three antenna system. The next segment contains all the calculated angles stored in IEEE double precision floating point format (8 bytes). These values are followed by the certainty, the signal frequency, and the noise level all represented as doubles. The Windows operating system does not always send the message immediately, but using windows-API to change the socket properties, the system was configured to send results as soon as they were ready. By default, Windows stores small messages in a buffer until it reaches a certain size and then sends them all at once. The use of a buffer saves network congestion in systems communicating over large multi-user connections, but unnecessarily delays communication in this system. To avoid these delays, the buffer size was set to zero so that the messages would be sent immediately. Once the GUI receives the messages it unwraps them to access the angle information and then updates with the corresponding results.

5.3 Graphical User Interface (GUI)

The purpose of the GUI is to accurately represent the incoming information from the DF algorithm to a human operator while simultaneously providing additional processing. The GUI receives data from the C algorithm over TCP then identifies emitters and attempts to resolve ambiguous results. The GUI then displays both resolved and ambiguous results.

5.3.1 Features

A screenshot of the GUI is shown in Figure 44. The main display is the semicircle



Emitters:	AoA	AoA Range		Frequencies		Cert
🟦	-13.49°	-15.0°	-10.5°	8.0GHz	12.0GHz	0.919
🟩	40.10°	40.1°	40.1°	8.0GHz	12.0GHz	0.990

Ambiguous Emitters:	Possible AoAs	Frequency	Cert

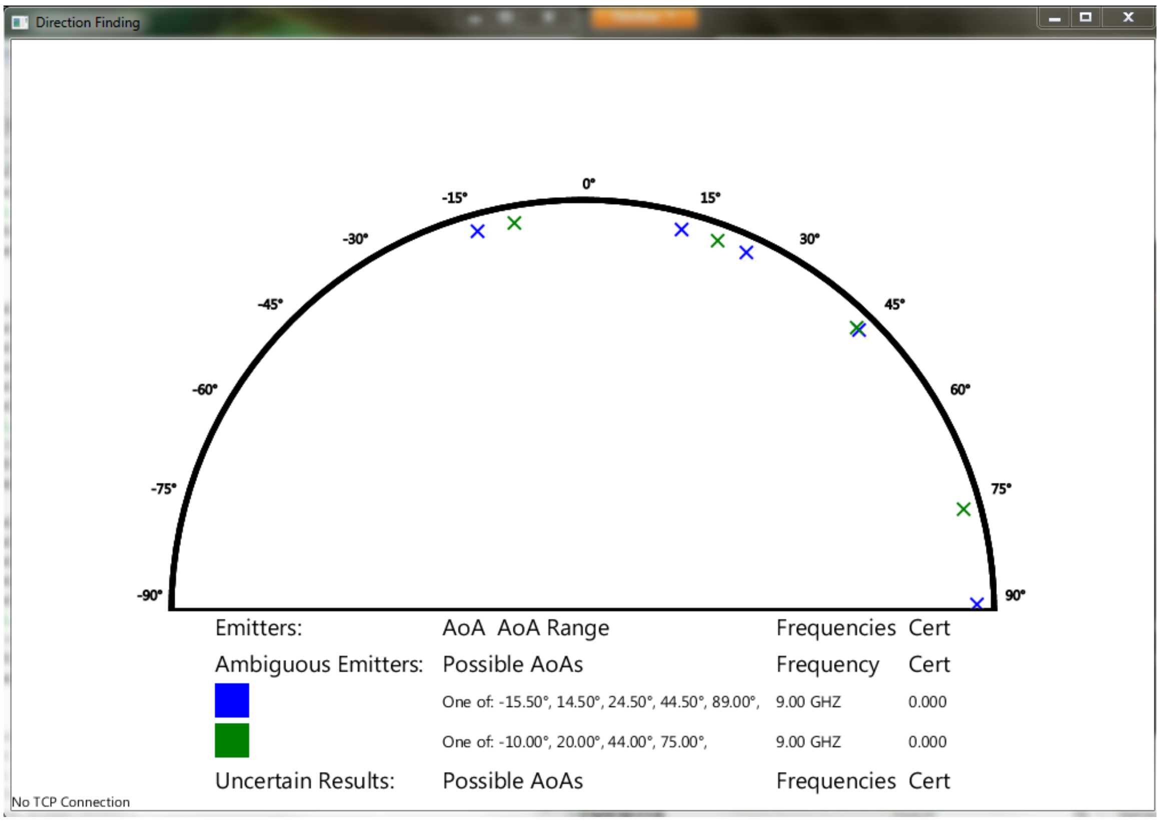
Uncertain Results:	Possible AoAs	Frequencies	Cert
🟥	-45.00° or 80.00°	10.00 GHZ	0.700

No TCP Connection

Figure 44: Screenshot of the system's GUI.

on the top half of the drawing. The blue and green lines represent the direction to two separate beacons. Beacons are identified by angle proximity; any results within 1° of each other will be grouped as one emitter. The angle of each emitter is based on the weighted average of the received AoAs of the signals. By weighting the most recent receptions more heavily, the GUI accurately follows the signal as it moves. The red x's along the edge of the semicircle represent a beacon that does not have enough certainty to resolve the signal to one of the two possible ambiguities. Red x's are used to visually inform the operator of the presence of an emitter without cluttering the screen with multiple lines for uncertain angles. The x's are also used to display results in two channel mode when there is no ambiguity resolution at all. A screenshot of the GUI displaying two channel results is shown in Figure 45. The emitter indication lines and the red x's are removed after 10 seconds.

The bottom portion of the GUI provides a more in-depth view of the signals shown in the main display. The description portion displays the full range of angles for the signals that have been condensed as well as the frequency of those signals. The frequency cannot be used to differentiate between emitters because emitters can periodically change frequencies, a technique known as frequency hopping. The full frequency display could allow the operator to recognize new emitters. Frequencies of compacted signals are shown as arrows between 8-12 GHz and disappear as the system stops detecting the signals. The final piece of information to display is the certainty of the AoA determination. An operator can use the certainty, provided on the far right of the GUI, to weigh the validity of the information displayed.



Emitters:	AoA AoA Range	Frequencies	Cert
Ambiguous Emitters:	Possible AoAs	Frequency	Cert
🟦	One of: -15.50°, 14.50°, 24.50°, 44.50°, 89.00°,	9.00 GHZ	0.000
🟩	One of: -10.00°, 20.00°, 44.00°, 75.00°,	9.00 GHZ	0.000
Uncertain Results:	Possible AoAs	Frequencies	Cert

Figure 45: Screenshot of the system's GUI displaying two input channel results.

In order to meet the display requirements, the GUI uses the JavaFX display library and a model-view-controller design pattern. The JavaFX library is built into the Java runtime, so it can be used on any computer that is able to run Java applications. JavaFX is used to draw the display because it provides high level layouts and drawing tools not available in the other built-in Java display libraries: Swing and AWT.

5.3.2 GUI Design

The GUI was split into three sections: the model which records and organizes data, the view which displays the information contained in the model, and the controller which adds to or changes the information stored in the model. These components were organized into three separate packages: processing, which is the model; display, which is the view; and communication, which is the controller.

The model stores all the information the GUI receives and organizes it for better display. The AoAs, certainty, frequency, and arrival time of each result are stored in a `SignalData` object. These results are organized into `SignalGroup` objects which contain a list of results that arrived within $\pm 1^\circ$ of each other and therefore could be from the same emitter. When a new `SignalData` is added to the model it is stored based on its AoA and certainty. The results are placed into one of three categories based on certainty. Results with 'high certainty' values, greater than 0.75, can be used to create an emitter line. Results with certainty values between 0.6 and 0.75 have 'low certainty' and are displayed as red x's. Results with certainty values below 0.6 are designated 'very low' certainty and can have their primary and secondary AoAs exchanged. Results with high certainty, greater than 0.75, will be added to an existing `SignalGroup` if possible, or a new `SignalGroup` will be created for it. A high certainty result is added to a `SignalGroup` if its primary result is within 1° of the range of angles stored in that group. If a result with low certainty, less than 0.75, is added the model will attempt to resolve the result to one of the two angles. First, the model will try to add it to an existing group. An uncertain angle can be added to a group if its primary angle is within 1° of the range of angles stored in that group and adding that result will not bring the average certainty of the group below 0.75. If the signal is very low certainty, less than 0.6, it may be that the correct result is the secondary result. In very low certainty cases the model will create a new result with the primary and secondary angles exchanged and a certainty of one minus the original certainty. It then attempts to add the new result to the `SignalGroup` objects, and if successful, stores the exchanged result rather than the original. If neither of these techniques can resolve the angle it is added to a list of uncertain results.

After a result is added, the model does some additional processing to clean up the results. To avoid very large angle ranges on the display, any groups with a range greater than 10° are split in half and any groups whose ranges intersect are merged into a single group. The model tries to resolve the uncertain angles into `SignalGroup` objects. A set of uncertain results are resolved to a group when there are at least four results with a common angle and at least two of those results contain angles which are separate from the rest of the set. When a number of results share an angle, 20° for example, it is clear there is an emitter present rather than simply false positives. A common angle by itself is not enough to determine the AoA. If all the results share the same ambiguities, 20° and -35° for example, then it is still not clear which angle to choose. If some results contain a different secondary angle, such as 50°, but still have the same primary angle, 20°, the correct result of 20° can be chosen.

Results received from the algorithm in two channel mode are processed differently. Rather than providing only the two most likely answers, as in three channel mode, the results received in two channel mode report all of the possible ambiguous angles. Two channel results are stored in `AmbigSignalData` objects and emitters are identified using the same angle proximity method as three antenna inputs. Only the four most recent ambiguous emitters are stored and only the most recent result for those emitters are displayed. In two channel mode there can be up to 12 results, and displaying more than four emitters would cause the display to become too cluttered. The organized information in the model is used by the view component to display the information.

The view uses the observer pattern to display the information in the model. Whenever the model changes, the display must change to match the model. The display components determine when to change by watching, or observing, the model. When the model changes, the display is informed of which parts changed and can update accordingly. The display components keep a record of what is currently displayed. When an update occurs, the display compares the stored record with the new data. The three main display components are the `GUIManager`, the `DescriptionManager`, and the `DrawingManager`. The `GUIManager` handles any data that must be maintained across both parts of the display. It assigns colors to groups, keeps track of when groups are added or removed, and keeps track of the uncertainties list. The `DescriptionManager` renders the detailed description of each group. When it receives an update it removes and redraws the corresponding row. The `DrawingManager` draws the main display image. When new information is received about a group, the old line for that group is drawn over in white, essentially erasing it. A new line is then drawn corresponding to the updated information. The technique of erasing the old line and drawing the new one minimizes the area changed on each update and reduces screen flicker. The display components use JavaFX to draw the images. JavaFX schedules its own redraws and therefore must run in its own thread. Pausing execution by waiting for input in that thread causes an unresponsive display. To prevent pausing execution in the display thread, the display does not wait for a change in the model directly, instead checking for changes whenever a refresh is requested by the controller.

The controller communicates with the algorithm, adds signals to the model, and requests display updates. The controller starts by setting up a TCP connection in a new thread separate from the display thread. Because it does not run in the display thread, the controller can wait the potentially long time required to accept a connection or receive inputs. Whenever it does receive an input the controller adds the result to the model and then schedules an update to run on the display thread. When the TCP connection drops, the thread restarts and waits for another connection. In addition to the communication thread, the controller runs a separate thread that refreshes the display every second. It removes all results older than 10 seconds from the model and then schedules a display refresh. The model-view-controller setup allows the GUI to organize and display the necessary results.

5.4 System Testing

Like the MATLAB model, the prototype system was tested to ensure that it met the system requirements. All of the frequency, field of view, and dynamic range tests run in MATLAB were run with the C algorithm to ensure that the C implementation also met the necessary requirements. In addition to testing the DF system, tests were also run using the connection to the FPGA and on the GUI. Several tests were run using two signal generators connected to the development boards. By specifying a phase offset in the signal generators, we were able to identify any biases present in the hardware and ensure that the parser would successfully reconstruct the signal. Finally, the GUI was tested for both its error checking and display capability. The error checking components were tested using JUnit. The display components were tested by running on both simulated data manually input to the GUI and data sent over TCP from the DF algorithm. These tests ensured that the prototype system functioned as intended.

6 Prototype System Results

There were three main types of tests run on the C-based prototype system: tests which compared results from the C and MATLAB algorithms, tests to ensure that the GUI would accurately accept and display information from the algorithm, and tests to ensure the prototype system could use real data from signal generators to calculate the AoA. The GUI and C algorithm were verified independently before testing the combined system with hardware because of the diverse functions each component performed and consequential risk of error. Testing with hardware verified that the system functions properly.

6.1 Direct Comparison Between C and MATLAB

When the MATLAB model and the C algorithm provide similar results, it provides confidence that the signal processing algorithm of the prototype system is working as expected. In order to test how closely the C algorithm aligned with the MATLAB model, we created a set of simulated signals under worst case conditions using MATLAB. The worst case conditions are: a RF of 12 GHz, an emitter distance of 1 km, a PW duration of 500 ns, and a PRI of 10 μs. To get the same waveform into both the MATLAB model and the C algorithm, we stored the data points resulting from sampling the simulated IF signal at each antenna in separate data files. With a known local oscillator frequency of 11.95 GHz for a wave with a carrier frequency of 12 GHz, both the MATLAB model and the C algorithm are able to read the data in from external files and then proceed as if the data were real. The two sets of results from this process were compared to see how closely the calculated AoAs match. The differences in calculated angle of arrival, shown in Figure 46, and in certainty, shown in Figure 47, were on the order of 10^{-5} degrees. The plots show that the C algorithm is in agreement with the MATLAB model.

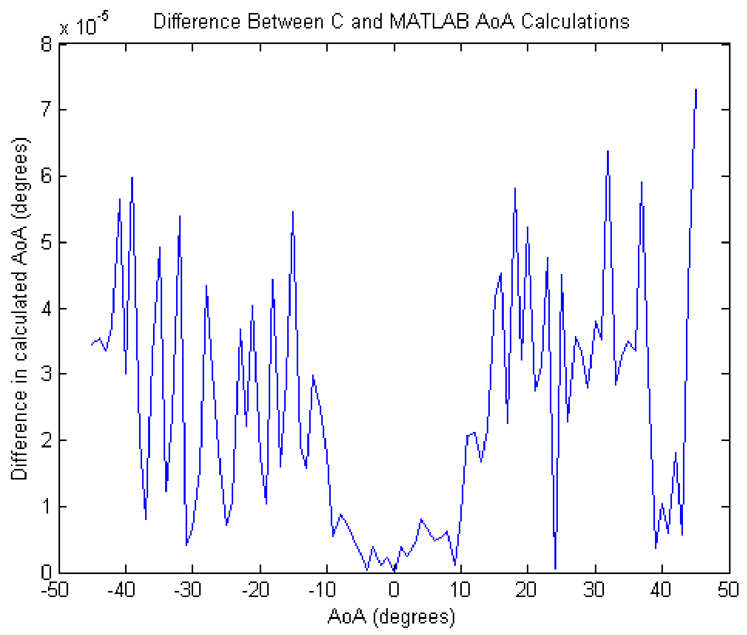


Figure 46: The differences in calculated AoA in the MATLAB and C algorithms for identical simulated signals. Differences of this magnitude show strong agreement between the MATLAB and C algorithm.

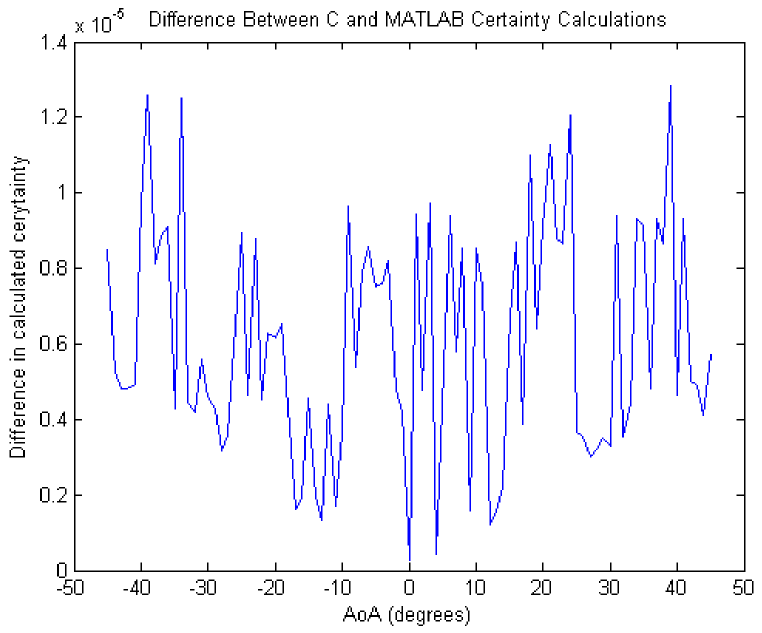
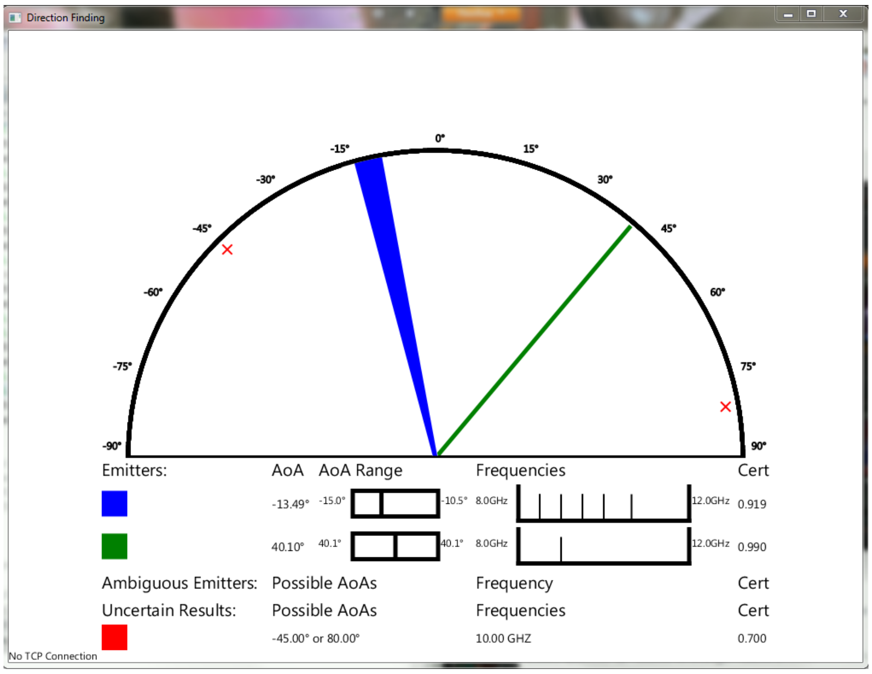


Figure 47: The differences in certainty reported the MATLAB and C algorithms for identical simulated signals. Differences of this magnitude show strong agreement between the MATLAB and C algorithm.

6.2 GUI Results

The GUI was tested to make sure it responded properly to various inputs and parameters. The GUI had two portions to test: the error checking portion and the graphical display portion. A Java testing framework, called JUnit, was used to insert simulated signals and automatically verify the output against expected results. The JUnit tests insured that the GUI correctly combined signals into display groups, stored separate uncertain signals, added uncertain signals to existing groups, exchanged the angle certainty in very low certainty cases, discarded old results from all locations, resolved multiple uncertain angles within the same angle range, merged groups with intersecting angle ranges, split groups with large angle ranges, and stored and discarded ambiguous results from a two antenna solution.

The GUI's error correction portion reduces the number of ambiguities displayed incorrectly. The GUI was fed the results of the C algorithm for data generated in MATLAB. Data was generated for each angle across the ±45° field of view with worst case system parameters. At each angle 10 pulses were generated so there were around 10 results per angle. In this test the GUI was able to resolve all of the results where the certainty of the correct angle was above 0.4. Less than 0.5% of all results were displayed incorrectly.



Emitters:	AoA	AoA Range		Frequencies		Cert
🟦	-13.49°	-15.0°	-10.5°	8.0GHz	12.0GHz	0.919
🟩	40.10°	40.1°	40.1°	8.0GHz	12.0GHz	0.990

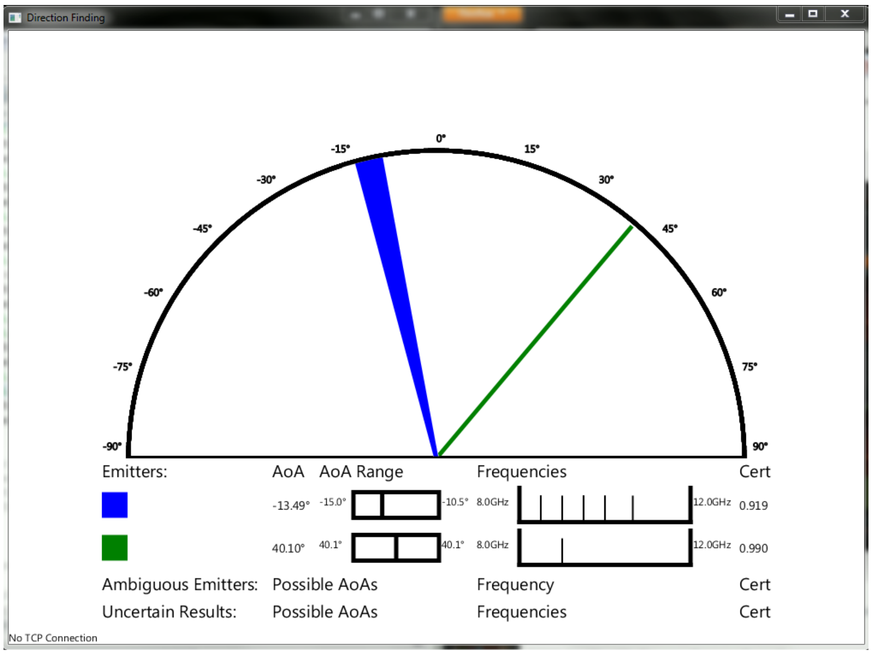
Ambiguous Emitters:	Possible AoAs	Frequency	Cert

Uncertain Results:	Possible AoAs	Frequencies	Cert
🟥	-45.00° or 80.00°	10.00 GHZ	0.700

Figure 48: Display resulting from inputting mixed certainty results from -11° to -15°, a single certain result at 40°, and an uncertain result of either -45° or 80°.

After the computational functionality was ensured, the display was tested to determine if it would display results correctly. The result of inputting mixed certainty results from -11° to -15°, a single certain result at 40°, and an uncertain result of either -45° or 80° is shown in Figure 48. In the figure, the blue wedge is relatively wide because it shows the entire range of -11° to -15° degrees where multiple signals have been detected. The narrow green wedge only represents a single result. Each different color on the GUI corresponds to a different emitter.

Figure 49 shows the result of inputting all of the same angles and then removing the certain 40° result. Removing the 40° result emulates the behavior of an angle that has not been updated in the previous ten seconds. The display performs as expected. The emitter at 40° is removed and the other emitters remain.



Emitters:	AoA	AoA Range		Frequencies		Cert
🟦	-13.49°	-15.0°	-10.5°	8.0GHz	12.0GHz	0.919
🟩	40.10°	40.1°	40.1°	8.0GHz	12.0GHz	0.990

Ambiguous Emitters:	Possible AoAs	Frequency	Cert
Uncertain Results:	Possible AoAs	Frequencies	Cert

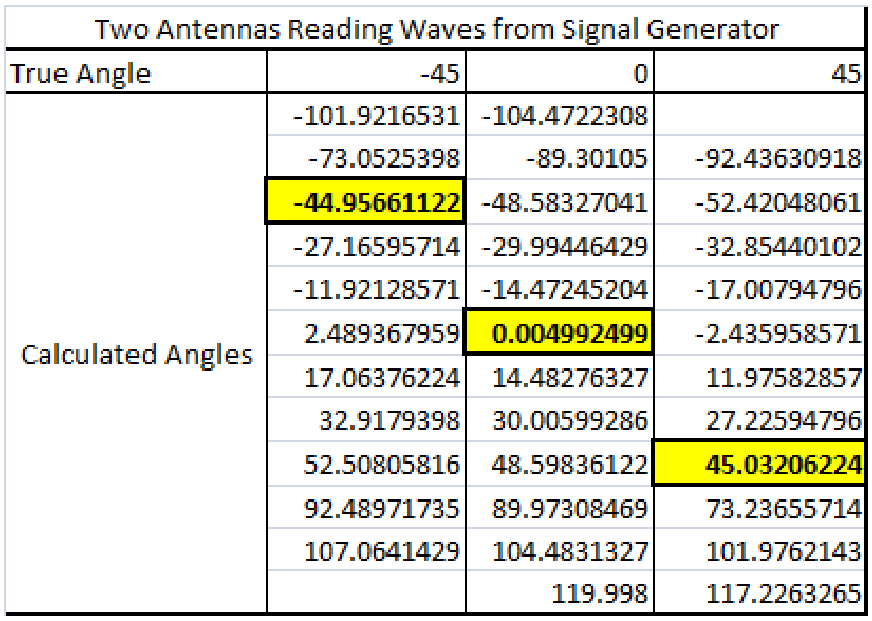
No TCP Connection

Figure 49: Display resulting from inputting mixed certainty results from -11° to -15°, a single certain result at 40°, and an uncertain result of either -45° or 80°. The 40° result has been removed because more than 10 seconds have passed since receiving a signal from that location.

In the majority of cases, the GUI will function properly. There are, however, a few remaining defects in the GUI that can cause the display to render incorrectly. The two main defects found were the GUI not drawing emitter lines correctly and the description pane overflowing the display. When there are many updates and refreshes over a short period of time the GUI will occasionally draw the emitter lines incorrectly. The GUI adds small fragments on the left, right, or top of the intended wedge. When the emitter line is erased, those unexpected fragments are not removed and can pollute the display. More testing would be required to determine the exact cause of these drawing errors as they occur sporadically. These errors could be mitigated, however, by changing the erasing technique to remove the bounding rectangle for the emitter line rather than drawing over the wedge directly. This technique would remove a much larger portion of the display, so it would require additional tracking to ensure that all data is redrawn correctly. The defect in the description pane arises when a large number of emitters are detected. The GUI will only display descriptions for up to four ambiguous emitters and for five uncertain results. It does not specify a limit, however, on the number of unambiguous emitters displayed. If enough emitters appear, the description pane will extend past the bottom of the screen. The defect could be fixed by specifying a maximum number of unambiguous emitters to display or by adding a scroll pane.

6.3 Hardware Tests

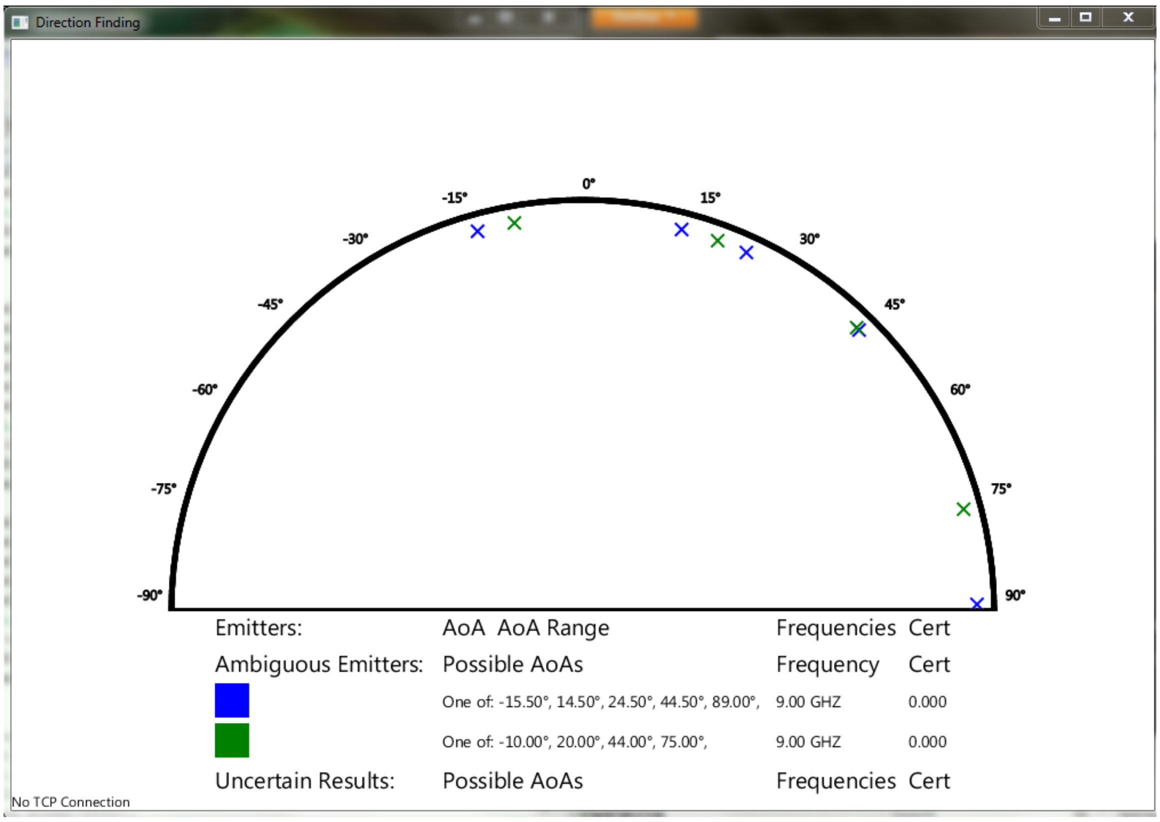
All tests run on the model, as well as the test to compare how closely the C algorithm followed the model, were created using simulated waveforms generated in MATLAB. Due to a defective ADC, three analog signal sources could not be tested. For hardware testing, therefore, we used two signal generators which represented antennas and down converters. With only two input channels, the algorithm calculated a number of possible AoAs but had no method to disambiguate them. Table 50 shows the average results from taking 100 samples on each of three signals whose true AoAs were -45°, 0°, and 45°. In each case, one of the angles was within 0.05° of the true angle, but 10-11 false angles were also reported which the two antenna system could not eliminate.



Two Antennas Reading Waves from Signal Generator			
True Angle	-45	0	45
Calculated Angles	-101.9216531	-104.4722308	
	-73.0525398	-89.30105	-92.43630918
	-44.95661122	-48.58327041	-52.42048061
	-27.16595714	-29.99446429	-32.85440102
	-11.92128571	-14.47245204	-17.00794796
	2.489367959	**0.004992499**	-2.435958571
	17.06376224	14.48276327	11.97582857
	32.9179398	30.00599286	27.22594796
	52.50805816	48.59836122	**45.03206224**
	92.48971735	89.97308469	73.23655714
	107.0641429	104.4831327	101.9762143
		119.998	117.2263265

Figure 50: Average of 100 measurements at the three indicated angles. Distinct values within each column provide all ambiguous angles. The highlighted angles are the results which most closely matched the true angles. A third input from a signal generator, which was unavailable for our tests, would determine which angle is the correct solution.

A sample of the GUI displaying two channel inputs is shown in Figure 51. With two input channels, the system does not disambiguate the results so all possible results are displayed. The sample contains results from two separate emitters so each set of possible results is shown in a different color.



Emitters:	AoA AoA Range	Frequencies	Cert
Ambiguous Emitters:	Possible AoAs	Frequency	Cert
🟦	One of: -15.50°, 14.50°, 24.50°, 44.50°, 89.00°,	9.00 GHZ	0.000
🟩	One of: -10.00°, 20.00°, 44.00°, 75.00°,	9.00 GHZ	0.000
Uncertain Results:	Possible AoAs	Frequencies	Cert

Figure 51: Display resulting from inputting ambiguous results from two input channels. All possible results for AoA are displayed.

6.3.1 Timing Tests

To be within specifications, the system must have the ability to capture, process, and display at least one batch of 4096 samples per second. The largest delays in the system are communication with the FPGA via a USB connection, and communication with the GUI over TCP. The minimum possible FPGA communication time, determined in Section 5.2.1 (FPGA Communication), was 270.3 ms. While running the physical system, communication times were observed ranging from 296 to 375 ms. Each data transfer from the FPGA transmits 8192 samples, two 4096 sample batches, per antenna.

Once the data are transferred to the processing PC, all of the computations, from parsing the data to calculating angles, for both batches takes place in under one millisecond. Communication with the GUI took 190 ms per batch of samples. Two batches are read in simultaneously so the second batch transferred has higher latency as it is sent to the GUI only after the first batch is finished transferring. The maximum latency of the system, therefore, is the sum of communication time with the FPGA, processing time, and two times the communication time with the GUI. These delays add up to a worst-case latency of 756 ms. Although the first batch reaches the GUI in less than 756 ms the second of the two batches is 756 ms out of date by the time it reaches the GUI. The throughput, results calculated per second, of the system is given by:

$$\frac{2\ batches}{756\ ms} = 2.65\ batches\ per\ second. \tag{37}$$

The measured latency is within the required latency of one second, but could be improved upon if the method communication is changed. One way to improve the throughput would be to communicate with the GUI and the FPGA simultaneously. Each time data is transferred from the FPGA, the two sets of solutions would be transmitted to the GUI for the previously processed data. The 380 ms delay associated with transferring both results to the GUI, is largest component in the latency, so all FPGA communication could take place during this time. Pipelining the communications would allow new results to reach the GUI every 190 ms, for a throughput of 5.26 batches per second.

7 Discussion

The purpose of this project was to develop a passive direction finding system that is capable of identifying the direction to a pulsed emitter to within ±2.5° across a 90° field of view, with a 40 dB dynamic range, a latency less than one second, and a minimum distance to an emitter of one kilometer. The MATLAB model was tested to ensure that the direction finding algorithm met these requirements. The prototype system was tested to ensure its agreement with the MATLAB model and that it met the one second latency requirement and to verify its operation on live data. The tests in Section 4.2 (MATLAB Results) demonstrate that an incorrect AoA calculation can come from either error in angle around the correct value or the selection of an incorrect ambiguity.

7.1 Analysis of MATLAB

The MATLAB model was used to determine how error in the AoA calculation was effected by various input parameters. The parameters tested in the MATLAB model were: antenna separation, AoA, signal strength, signal carrier frequency, and emitter distance. Differences between the MATLAB calculated AoA and true AoA were due to two sources, error due to simulated noise and resolving the phase ambiguity incorrectly. The simulated noise comes from adding -60 dBfs white Gaussian noise to the signals directly and from simulating the ADCs which produces quantization error. The value for the white Gaussian noise was chosen based on the performance defined in the ADS5400 ADC vendor data sheet. The quantization error of the system results from rounding the values of the signal to the ADC's bit values. The quantization error is always less than or equal to the maximum ADC step size, $2.44 * 10^{-4}$ V, and has an average value of half the maximum. The error from noise creates an error in phase calculation which induces mistakes when resolving the phase ambiguities. Selecting the wrong ambiguity creates a very large error in AoA. The magnitude of the error due to selecting the wrong phase is dependent on the antenna separation and the carrier frequency of the wave, but errors of 10° to 20° are typical for frequencies in the X band and our antenna setup. The likeliness of a phase ambiguity error occurring is dependent on the error in phase calculation and on the physical setup of the system. As the distances between antennas increase, so does the number of possible phase differences, causing the probability of selecting an incorrect phase to increase.

7.1.1 Choosing Antenna Separations

Increasing the antenna separation reduced the error due to noise in the AoA calculation and decreased the certainty. Larger antenna separations result in larger phase differences that are more resistant to changes due to noise but are more susceptible to errors in selecting the incorrect ambiguous phase. In worst case conditions, the error due to noise was 0.1° higher when the separation between antennas 1 and 2 was 5 cm than when the separation was 20 cm. The certainty dropped by approximately 0.15 over the same range.

After analyzing the results of testing the antenna separations between 5 cm, the minimum possible separation, and 20 cm, we decided that the spacing of 10 cm between antenna 1 and 2 would provide the best balance between average certainty and average error due to noise. The optimal antenna spacing corresponding to the 10 cm separation between antennas 1 and 2 was 21 cm. With the 10 cm separation, in worst case conditions, the ambiguity was resolved incorrectly in only 1% of the cases, which was an acceptable failure rate.

Non-optimal separations between antennas 1 and 3 cause dramatic reductions in certainty and, consequently, more phase ambiguity errors. Separations that are even 1 cm off the optimal spacing result in average certainties of the correct AoA of approximately 0.5. The system, therefore, will generally be uncertain of each calculation. Changes in separation on the order of 1 mm, however, did not have a drastic effect. The change in average certainty and error due to the separation difference was less than the typical variation between runs. Placement errors of this magnitude would therefore not cause system failure.

7.1.2 Effects of Different Parameters on AoA Error

The tests in Section 4 (MATLAB Results) determine the effects of AoA, signal strength, carrier frequency and emitter distance on errror in calculating the AoA. The true AoA and the signal strength had the greatest effect on the error of the AoA calculations. The error of the system tended to increase as the AoA moved away from 0°. For our minimum SNR, 20 dB, the difference in average error between the edges of the range (-45° and 45°) and 0° was 0.02° and the maximum error between ±45° was 0.415°, well below our ±2.5° accuracy requirement. The peak error at ±90° was 5.961°, and the difference in mean error between ±90° and 0 was approximately 1°. The peak errors at ±90° were above our 2.5° requirement, however, all values between ±85° fell within the requirements. As expected, when the signal strength was reduced, the SNR decreased and the error increased. The mean error at -40 dBfs and 45° was 0.17° while it remained 0.001° at 0 dBfs.

Changing the carrier frequency of the received signals had a small but noticeable effect on both the accuracy and the certainty of the system. As the RF increased, the ratio of distance between antennas to the signal's wavelength also increased, creating the same effects as keeping frequency constant while moving the antennas farther apart. Higher frequencies led to decreasing both certainty and error due to noise. The difference in average error due to noise between the two frequency extremes the system operates on, 8 GHz and 12 GHz, was 0.021° under worst case conditions. Under the same conditions, the average certainty was 0.019 higher at 8 GHz than at 12 GHz.

When the received signal strength was held constant, changes in emitter distance did not have a large effect on the mean error of the system for ranges above one kilometer. The error at one kilometer, the minimum expected range, was 0.0069°, while the error at 100 kilometers was 0.0073°. The error does increase at emitter distances of less than 100 meters as the assumption that the AoA is the same at each antenna no longer holds true. When testing with worst case conditions, the mean error due to noise was 0.097° while the maximum measured error due to noise was 1.03°, well below the ±2.5° accuracy requirement.

In order to determine the effects of parameter changes on the AoA error the emitter distance and signal strength were changed separately. In reality, the emitter distance and signal strength are linked. Emitters that are far away result in low signal strength at the antennas. This setup was tested in Section 4.5 (Emitter Distance). For an emitter with 100 dBm effective radiated power, the system was able to identify the angle within ±2.5° for all emitter distances from 1 km to 100 km. Decreasing signal strength increases AoA error so the error in the AoA calculation increased as the emitter moved farther away.

7.1.3 Error in Resolving Phase Ambiguities

Resolving the phase ambiguity incorrectly creates a larger error in the AoA calculation than noise does. When the algorithm resolves to the wrong ambiguous angle, the AoA error can be more than 10°. The chance of resolving to the correct angle is affected by the error in the phase calculation and the spacing between the two phase lines nearest to that calculation. The certainty of the results reflects the chance of resolving to the wrong angle. The certainty is defined as the distance of the measured phase result from the closest true phase line over the total separation between the two closest phase lines. A certainty of 0.5 is the lowest that is reported by the system and a certainty of 1.0 is the highest. A certainty value of 0.5 indicates that the system cannot determine which of the two reported solutions is more likely to be the correct angle. In the results sections, the certainty corresponding to the true AoA, ranging from 0 to 1, was shown. The error in the phase calculation is caused by noise and is therefore determined primarily by the SNR. The average certainty at -40 dBfs was 0.87 compared to 0.99 at 0 dBfs. The separation between the phase lines used to resolve the angles also factors into the certainty. Large separation between lines improves the certainty, but the line separation is not uniform (Jacobs and Ralston, 1981). The measured phase differences from both antennas determine which phase lines are used. AoA and carrier frequency partially determine the phase difference and therefore have sporadic effects on the certainty. With worst case input parameters, the average certainty was 0.87, meaning that on average the system was certain of the correct answer.

Between ±85°, the errors due to noise is ±2.5°, however, the noise can corrupt the phase measurements enough for the system to resolve the phase ambiguity incorrectly. The minimum separation between two phase lines in the proposed system is 0.27 radians, requiring a measured phase error of at least 0.135 radians to resolve the ambiguity incorrectly. The average line separation is 0.48 radians and the maximum is 2.7 radians. When every angle between ±45° with a 1° step size was tested 100 times under worst case conditions, 98 out of 9100 angles were resolved incorrectly. Therefore, under worst case conditions, the system selects the incorrect ambiguous angle approximately 1% of the time. There were only 16 times, 0.18%, where both of the reported angles were incorrect. The algorithm meets the project requirements in all but a few worst cases. In these cases the algorithm computes the AoA with data from only a small fraction of a signal pulse.

7.1.4 MATLAB versus C Results

When the algorithm was converted from MATLAB to C, identical signals were used as inputs in both the MATLAB and C tests to ensure that the C algorithm agreed with the model and still met the requirements. The results produced by the C implementation were within 10^{-4} degrees of the results from the MATLAB system for angle and 10^{-4} for certainty in all cases. Therefore, the C algorithm met the accuracy requirements to the same degree as the MATLAB implementation.

7.2 Analysis of Prototype System

Once we found that the MATLAB and C algorithm were in agreement, we tested the prototype system to ensure it met our requirements. The total latency between the signal arriving and the GUI reporting an AoA was inspected and the full prototype system was tested with MATLAB created data.

7.2.1 Latency Analysis

The C algorithm had an additional requirement of a total processing time less than one second. The two channel system took a maximum of 375 ms to read in two batches of samples over the USB and the entire algorithm took less than one millisecond to run on both batches. Windows takes roughly 190 ms to send the packet to the GUI for each batch, giving a maximum total run time of 756 ms. The run time of the algorithm from data read to display, therefore, is less than the required one second.

The DF system transfers 8192 samples of data per antenna from the FPGA on every data transfer, but only uses 4096 samples per antenna to compute the AoA. The data are processed in this manner in order to provide two independent sets of results every time data is imported from the FPGA. There are three scenarios in which this is useful. In cases where the same signal is present in both batches, the two results allows the GUI to better resolve ambiguities cases where the certainty for one set of results was low. A second set of results for the same emitter will likely clarify which angle of arrival is correct. In cases where the signal is only present in one of the two batches, processing time is saved because the algorithm will only process the data in the batch where a signal was present. The algorithm is also able to use the batch with no data in order to update its recent noise history. Batches with only noise occur more frequently with smaller batch sizes, so the noise history, and by extension the signal detector, are more accurate when smaller batch sizes are used. Finally, there can be cases where two distinct signals are found in the two batches. If two signals both received in one long batch, the system only identifies the stronger of the two signals. Utilizing two smaller batches decreases the likelihood that multiple emitters will be received as part of the same batch, having the effect of increasing the likelihood of identifying each emitter individually. Decreasing the batch size further would increase all of these effects, but would add more latency to the system, as each set of results takes 190 ms to transmit to the GUI.

The FFTW algorithm utilized in the prototype system is most effective when the number of samples is a power of two, so the next smallest sample size the system could effectively use would be 2048 samples. In this case, the maximum latency of a batch would be the sum of the communication delay between the FPGA and the C algorithm of 375 ms, the one millisecond for computations, and four times the 190 ms communication time to transfer results to the GUI for a total delay of approximately 1.2 seconds. This delay would exceed the one second maximum latency requirement.

7.2.2 Two Channel Input Mode

Using two signal generators to simulate antennas and down converters, the system was implemented and tested with two input channels. The three input channel system was tested with data from the MATLAB model. The error due to noise when using two signal generators to simulate antennas with 10 cm separation was 0.01° at -9 dBfs while MATLAB tests under similar conditions with simulated signals gave errors between 0.003° and 0.005°. The difference in error between simulated signals and physical signals from the signal generator is due to the precision of the signal generators, the phase delay from sending signals from the signal generators to the development board, and imperfect ADCs. The two input channel system does not resolve the phase ambiguities, so the two channel mode always produces multiple ambiguous solutions. With the two simulated antennas separated by 10 cm receiving 12 MHz signals, the system produced between 5 and 6 ambiguous answers on each batch. The test results show that the prototype system meets the system requirements to the same degree as the MATLAB model. As in the

MATLAB model, it has an error of 0.097° due to noise and resolves to the wrong angle approximately 1.08% of the time in the worst case conditions. The addition of the GUI's error checking system reduced the frequency of displaying the incorrect ambiguity. With a test of 10 pulses for each angle from $-45°$ to $45°$ with $1°$ increments and worst case inputs, the GUI reported the correct angle more than 99% of the time. The incorrect result was displayed only when the algorithm reported the incorrect result as certain or when both angles reported were incorrect. In the results section these values are displayed as less than 0.25 certainty and constitute less than half of the ambiguous results.

7.3 Limitations and Future Work

Although the prototype performs within the requirements, there are several steps required to bring our system to an operational solution. The prototype must be tested with actual antennas and radar signals. While the simulations and tests suggest that our system should function properly if antennas were used to replace the signal generators, there is always the risk of complications when adding true data to a system. Before the system can be implemented on an aircraft, the system must also be updated to both take into account the effects of a sweeping local oscillator window and run in real time. In addition to testing the system with three antennas, experimenting with interferometers using four or more antennas or a two antenna solution which can resolve ambiguities could improve the system.

7.3.1 Antenna Tests

As part of the project our group initially planned on testing the DF system with true data from antennas. Due to hardware malfunctions the DF system was only tested with signal generators simulating the antennas and down converters that would be used in an operation system. A true data test using these components and radar waves produced by an emitter would help verify that the system works as intended. Having a radar system at the ranges expected by the physical system, greater than one kilometer, would be impractical for a verification test. The hardware test planned for the system involved using a moveable emitter. The emitter and DF system would be placed at least 20 meters apart, perhaps in a parking lot. As shown in Section 4.5, the system does not experience significant errors until the distance drops below 20 meters. The test could still be run in a smaller setting such as a lab, but the system would ability to resolve ambiguities would deteriorate.

7.3.2 Sweeping Window

The system does not take into account the effects of sweeping the tunable local oscillator. In the operational system, the local oscillator would need to sweep across the

RF range in 100 MHz increments. Given the 0.3775 s processing time in the prototype system, it would take a minimum of 30.2 s to sweep across the entire 4 GHz bandwidth, capturing two batches of data with 16.38 μs of recording time, per 100 MHz band. The pulse width of emitted signals can vary from 500 ns to 1 ms, so recording for 16.38 μs does not guarantee that a pulse will be seen even when an emitter is present in the bandwidth which the system is monitoring. In addition, any signals that arrive outside the monitored window will be missed. The effect of the local oscillator sweep can be mitigated by increasing the IF bandwidth. A larger bandwidth means that the local oscillator takes larger steps across the IF band. The maximum width of the IF band is half of the ADCs' sampling frequency. The ADCs used in our system would allow an IF bandwidth of up to 250 MHz. If that IF bandwidth was used the oscillator would sweep the entire bandwidth in a minimum of 16 steps. If the system again took two batches of samples in each window it would take 12.08 seconds to sweep the 4 GHz bandwidth. Using a 250 MHz bandwidth would mean that the signals' IF can be at exactly the Nyquist rate. Sampling at exactly the Nyquist rate could lead to aliasing problems with signals at the edges of the range. The system missing signals due to the local oscillator sweep was not taken into account for this study.

7.3.3 Real Time Operation

Ideally, the DF system would run on inputs in real time, meaning that the algorithm should be able to process data as fast as it samples data. In order to remove the large communication delays, the entire algorithm should be implemented on an FPGA or an embedded system. The system is currently implemented with double precision floating point in both C and MATLAB. The algorithm would likely have to be converted to fixed point computations in order to be implemented on an FPGA or embedded processor. With fixed point calculation, values used in calculation can only be represented to a fixed number of decimal points, making the phase calculation less accurate. The FFTW library runs with double precision floating point, so a fixed point FFT algorithm would have to be used. By allowing the hardware to perform the calculations, the communication delay would no longer be a major factor and the computational speed would increase to the point where the system could potentially run in real time.

The timing results for the C algorithm were calculated using only one computer and using the built-in C timing functions. On different machines the latency could be different. The timing functions in C are limited to a resolution of one millisecond, so we were unable to determine an actual value for processing time other than the fact that it completed in under 1 ms. Although 1 ms is a short computation time when compared to the longer communication delays, 1 ms of processing time is very long compared to the 8.19 μs taken to sample the data. Measuring a more precise value for processing time would be beneficial in determining whether or not the C algorithm could be used for real time processing.

7.3.4 Additional Antennas

The system was designed to work with either two or three antennas. More could be theoretically be used in the system. Adding more antenna could increase the accuracy of the system as once the ambiguity is removed any pair of antennas could be used to make the final calculation. The farther apart the antennas used to make the final measurement are, the less of an effect noise has on the calculation. If a fourth antenna was added at a larger separation it would provide a more accurate phase measurement for the final calculation. In addition to increasing the accuracy of the system, adding additional antennas can also allow the system to measure the angle of elevation. By placing another three antenna interferometer offset in elevation angle from the original interferometer the system could determine the elevation angle as well as the azimuth angle (Jacobs and Ralston, 1981).

7.3.5 Resolving Ambiguities with two Antennas

The two channel input mode implemented in the system did not remove ambiguities from the phase calculations. A system that combines the amplitude comparison and phase interferometery methods could remove the ambiguities with only two antennas. The phase interferometry method produces a number of highly accurate but ambiguous results. The amplitude comparison method is used to calculate a rough value for the AoA and then that angle is used to choose which of the ambiguities is correct. The amplitude comparison method implemented in the Beacon Locator Project used two antennas placed 10 cm apart and had a peak error of 4° at the edge of the 90° field of view and less than 2.5° in the center of the range. The ambiguities between antenna 1 and 2, separated by 10 cm, were always more than 10° apart. As long as the accuracy of the of the amplitude comparison method is less than half of the separation between ambiguities the system would be able to choose the correct result. The error of the amplitude method could increase if the field of view was expanded to the range of ±85° leading to a chance of not resolving the ambiguities correctly. Implementing a system that combines phase interferometry and amplitude comparison could reduce the chance of ambiguity errors.

7.3.6 GUI Improvements

There are a number of improvements that could be made to the GUI to fix defects and increase functionality. Two of the defects found in the system were leftover line fragments and overflowing the description pane. The line fragments are caused by the GUI's redraw technique. The GUI attempts to erase only a small area so that the display will have as little flicker as possible. Expanding the area that is erased would make drawing errors less likely, but also increase the display flicker and require more complex redraw techniques to ensure that extra information is not erased. Determining a balance between the two techniques would improve the GUI. Overflowing the description pane can be easily solved by setting a maximum number of emitters to be described. Currently, no limit is imposed on the number of resolved emitters displayed.

The tracking algorithm used by the GUI could be improved to reduce the chance of displaying incorrect ambiguities. At the moment, the GUI will create a new emitter line for any result that has certainty greater than 0.75. The algorithm occasionally reports the incorrect angle with a high certainty value. In these cases, the GUI will always display the ambiguity incorrectly. If the system is likely to see an emitter more than once, the tracking algorithm could be changed to require multiple certain results to identify an emitter, requiring multiple ambiguity errors before an incorrect result is displayed.

The GUI could also be improved to add user interaction. Currently, the GUI is a read only system; the information is displayed without any user interaction. Adding the ability to configure system parameters, such as maximum angle range, certainty cutoffs, maximum emitters displayed, and minimum results required to identify an emitter would allow the user to customize the display. If the system received many low power signals, increasing the minimum results required to display an emitter would reduce the probability of displaying an incorrect angle. Adding the ability to manually resolve an ambiguous angle by clicking on it and the ability to clear the screen at any time would add more interaction and operator control. Another improvement to the GUI's functionality would be a separate input representing the plane's heading. With the current GUI setup, if the plane turns the results will remain displayed at the same angle for 10 s until they are removed. If the GUI knew the heading of the plane when it received its input, it could adjust the results as the plane turns. An emitter found at 45°, when the plane is flying straight, could then be moved to -45° after a 90° turn to the right.

8 Conclusion

The purpose of this project was to develop and implement a phase interferometry direction finding system for an airborne platform. Tests with simulated data show that the system successfully meets or exceeds all requirements of the project:

1. Accuracy: The system was required to calculate angle of arrival to within $\pm 2.5°$. Under worst scenario operating conditions, the average error due to noise for the system is less than $0.1°$ and the maximum error due to noise was $0.415°$. Phase ambiguities cause much larger error around 1% of the time under worst case conditions.

2. Airborne platform: The entire system requires 21 cm of antenna spacing, making the system a small enough size to be easily implemented on an aircraft.

3. Field of view: The system was required to determine AoA for a 90° field of view in front of the aircraft, ranging from -45° to 45°. Our system exceeds the field of view requirement and operates within specifications from -85° to 85° in front of the plane.

4. Latency: The system was required to compute the AoA for a set of data at least once per second. The maximum latency of the physical hardware system was 756 ms.

5. Frequency Range: The system is intended for operation in the X band range. Tests were conducted on frequencies across the full spectrum of X band, from 8 GHz to 12 GHz, and correct AoA results were calculated across the band. The system simulated a tunable local oscillator in order to down-convert the X band signals into a 100 MHz band from 15 to 115 MHz.

6. Dynamic Range: The system was required to process signals over a 40 dB dynamic range. System tests were evaluated at -40 dBfs to ensure the system met this requirement. In these conditions the system determined the correct AoA approximately 99% of the time. At higher powers the system was correct more often.

Although we were not able to test the system with three live input channels or with true antenna data, we successfully developed a C algorithm which can run on the final hardware setup with few modifications. The only change that will be required in the C algorithm for three channel operation is communication with a second FPGA in order to access data from the third antenna. We were able to process data from two signal generators using the two channel version of our system and accurately determine the correct angle of arrival, albeit the angle was one of a number of ambiguities which could not be resolved in our system using only two input channels. The C algorithm was able to process simulated data for three antennas and determine the correct AoA to a high degree of precision. The results from the C algorithm matched the results of the MATLAB model closely, within 10^{-4} degrees when given the same signals. The thoroughly tested MATLAB model provided strong evidence that when the third live data channel is added to the system, the system will produce accurate results.

Additionally, the system was able to successfully communicate the outputs to a GUI which actively displayed the angle of arrival results to the user. The GUI also displayed the frequency and certainty of each angle measurement to the user in order to provide more detailed and valuable information. In cases where an emitter is identified but, the algorithm was unable to confidently determine the location, the GUI displayed the information differently so that the operator was notified of the emitter's presence but not misled into believing a false AoA. When the uncertain detection could be resolved into a more certain angle due to additional measurements, the ambiguities were removed from the GUI and the correct angle appeared on the screen.

There are some improvements that we would suggest for the system. Moving the processing algorithm to the FPGA or an embedded processor would remove the communication delays and make the system operate much faster. The system created uses three antennas to resolve the phase, but there are methods for resolving ambiguities with only two antennas. These methods should be compared with the one currently implemented to see which one best removes the ambiguities. Additionally, introducing user controls and an enhanced tracking algorithm to the GUI would reduce the probability of false ambiguity selection. With these changes, and further testing with real antennas, the prototype created in this project could be converted into an operational system.

References

Bakshi, K.A., B. A. B. U. (2009). *Antennas and Wave Propagation*, pages 6–1 – 6–3. Technical Publications.

Candy, J. C. and Temes, G. C. (1992). *Oversampling Delta-Sigma Data Converters: Theory, Design, and Simulation*, volume 1, chapter Oversampling Methods for A/D and D/A Conversion, pages 1–29.

Fang, B. T.Zurek, R. W. and Martin, L. J. (1990). Simple solutions for hyperbolic and related position fixes. 26(5):748–753.

Frigo, M. and Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, 93(2):216–231. Special issue on "Program Generation, Optimization, and Platform Adaptation".

Holm, W. and Richards, M. (2010). *Principles of Modern Radar*, pages 737, 741. SciTech Publishing.

Jacobs, E. and Ralston, E. (1981). Ambiguity resolution in interferometry. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-17(6):766 –780.

Lathi, B. P. (2005). *Linear Systems and Signals.* Oxford University Press.

Silva, E., O'Connor, S., and Massa, C. (2011). *The Beacon Locator Project: A Passive Direction Finding System for Locating Pulsed Emitter Signals*, pages 25, 115.

Skolnik, M. I. (2001). *Introduction to Radar Systems*, pages 266, 540, 710. The McGraw-Hill Companies, third edition.

Smith, J. O. (2007). *Mathematics of the Discrete Fourier Transform (DFT)*, pages 35–37. W3K Publishing.

Wiley, R. G. (1985). *Electronic Intelligence: The Interception of Radar Signals.* Artech House.

Wolff, C. (1997). Radar tutorial. www.radartutorial.eu/09.receivers/ rx05.en.html.

Zhou, R., Zhang, H., and Xin, H. (2011). Improved two-antenna direction finding inspired by human ears. *IEEE Transactions on Antennas and Propagation*, 59(7):2691–2697.