Development of a Single-Channel Direction Finding Algorithm

Nathan M. Harter

Thesis submitted to the faculty of of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering.

Thesis Committee
R. Michael Buehrer, Chair
Jeffrey Reed
William Tranter

April 13, 2007
Blacksburg, VA

Keywords: Wireless Communications, Direction Finding, Signal Processing, Beamforming

Development of a Single-Channel Direction Finding Algorithm

Nathan M. Harter

Abstract

A radio direction finding (DF) system uses a multiple-element antenna array coupled with one or more receivers to estimate the direction-of-arrival (DOA) of a targeted emitter using characteristics of the signal received at each of the antennas in the array. In general, DF systems can be classified both by the number of receivers employed as well as which characteristics of the received signal are used to produce the DOA estimate, such as the signal's amplitude, phase, or time of arrival.

This work centers on the development and implementation of a novel single-channel direction finding system based on the differential phase of the target signal received by a uniform circular antenna array with a commutative switch. The algorithm is called the PLL DF Method and differs from older single-channel DF techniques in that it is a digital algorithm intended for implementation on a software-defined radio (SDR) platform with a custom-designed antenna array and RF switching network. It uses a bank of parallel software PLLs to estimate the phase of the signal received at each element of the multi-antenna array. Theses estimated phase values are then fed to a specialized signal processing block that estimates the DOA of the received signal.

This thesis presents the details of the initial version of the PLL algorithm which was used to produce a proof-of-concept system with an eight-element circular array. It then discusses various technical challenges uncovered in the initial implementation and presents numerous enhancements to the algorithm to overcome these challenges, such as a modification to the PLL model to offer increased estimator robustness in the presence of a frequency offset between the transmitter and receiver, revisions of the software implementation to reduce the algorithm's processing requirements, and the adaptation of the DF algorithm for use with a 16-element circular array. The performance of the algorithm with these modifications under various conditions are simulated to investigate their impact on the DOA estimation process and the results of their implementation on an SDR are considered.

For Sarah.

Acknowledgments

This work would not have been possible without the help of many people. First of all, I must thank my advisor, Dr. Michael Buehrer. His guidance was invaluable during and even after my time at Virginia Tech.

I would also like to thank DRS Signal Solutions, Inc. for funding this work and providing the receiver hardware as well as for offering me a job right out of school. Numerous engineers from DRS-SS also deserve my gratitude. Tom Potter and Jeff Silvey, the two points of contact for the University program, were helpful in either answering any questions I had about the receiver or directing them to someone who could. Guy Zaybekian was an incredible help when it came to my questions about how the radio operates, as well as a more than capable teacher when it came to working with him professionally on a new software defined radio during my tenure at DRS-SS. Also, the entire engineering staff was very helpful when the provided a responsive and supportive audience for presentations on our work.

Many of my colleagues at MPRG, all of whom I count as friends, also deserve their place here. Chris Anderson was indispensable when it came to solving problems relating to hardware. Swaroop Venkatesh and Jody Neel also knew were to point me when I had DSP questions and Max Robert was always around when I just needed to take a break. I must also thank John Keaveny, who was one of my good friends during both my undergraduate years at Saint Louis University as well as when I followed him out to Virginia Tech and took over his research project when he finished. I learned quite a lot from him in my first year at Tech.

Lastly, I must also thank my wife, Sarah. Without her love and support this would have been a difficult process, and not nearly as much fun.

Nathan Harter
April 13, 2007

Table of Contents

List of Figures

List of Tables

List of Acronyms

ADC	Analog to Digital Converter
AOA	Angle-of-Arrival, used interchangeably with DOA
AWGN	Additive White Gaussian Noise
BPSK	Binary Phase Shift Keying
CPLD	Complex Programmable Logic Device
CW	Continuous Wave
DF	Direction Finding
DOA	Direction of Arrival, used interchangeably with AOA
DRS-SS	DRS Signal Solutions, Inc.
DSP	Digital Signal Processor
GFLOPS	10^9 Floating Point Operations Per Second
FFT	Fast Fourier Transform
LO	Local Oscillator
LOS	Line of Sight
MA	Metric-Assisted
MSE	Mean Squared Error
PLL	Phase Locked Loop
RF	Radio frequency
RMS	Root Mean Square
SDR	Software Defined Radio
SNR	Signal to Noise Ratio

Symbol Glossary

ϕ	Angle of Arrival, azimuth
ζ	Angle of Arrival, elevation
$\hat{\phi}$	Estimated angle of Arrival, azimuth
$\hat{\theta}_m[i]$	Steady state output of *m*-th PLL
$\Delta\hat{\theta}_m[i]$	1st difference curve
$\Delta^2\hat{\theta}_m[i]$	2nd difference curve
λ	Wavelength
β	Phase propagation factor, $\beta = 2\pi/\lambda$
M	Number of elements in an antenna array
m	Index of elements in an antenna array, $m = 0, 1, \ldots, M-1$
D	Number of signals incident upon the antenna array
d	Index of incident signals, $d = 0, 1, \ldots, D-1$
K	Number of samples taken per array element
r	Array radius

Chapter 1

Introduction

Radio direction finding (DF) systems use a multiple-antenna array with one or more receivers to produce an estimate of the bearing angle or geographical coordinates of an intercepted signal of interest (SOI). The primary function of a DF system is to produce a direction-of-arrival (DOA). DF systems have numerous applications from amateur use in HAM radio "foxhunting" contests to emergency service to military defense and intelligence operations.

These systems can be categorized into two main types, *n*-channel DF systems using typically one receiver channel per antenna, and single-channel DF systems which use a single receiver with a multiple-antenna array along with some form of switching among the elements or combining them to present the receiver with a single signal. Single-channel DF systems offer obvious advantages over *n*-channel systems in terms of size, weight, power, and portability requirements but in general pay for these advantages with reduced processing power, accuracy, and robustness in adverse channel conditions. The challenges inherent in developing a single-channel DF system are precisely what make it an attractive field of study as the utility of a system that can offer performance characteristics that approach those of *n*-channel systems is undisputed.

This thesis is concerned with a specific single-channel DF algorithm known as the Phase Locked Loop Algorithm [1],[2],[3],[4]. It is the continuation of work started at MPRG by a previous student, John Keaveny, along with his advisor Dr. Michael Buehrer and fellow student Swaroop Venkatesh. His work focused on the initial development, simulation, and implementation of the algorithm. In short, the algorithm uses an 8 element circular array connected to a single-channel software defined radio (SDR) receiver through a digitally controlled 8-to-1 switching circuit that switches sequentially around the elements. This time divided signal is then fed into a bank of eight parallel phase locked loops (PLLs) which then track the phase of the signal at each antenna. This collected phase data is then differentiated and fed through a signal processing block to produce a single DOA estimate for one full array sweep (i.e. the time required to switch from the first antenna

element to the last).

This work extends the original algorithm development and implementation in a few ways. First, the initial implementation of the PLL model, while perfectly functional, was overly processor intensive. In order to add further enhancements to the algorithm implementation it was necessary to produce a more efficient implementation of the PLL model. Second, it was discovered during the initial implementation that a slight frequency offset occurs between the test signal generated by the transmitter and the receiver's local oscillator (LO). Since the DOA estimations process of the PLL algorithm is based on the phase of the received SOI, this frequency offset can wreak havoc with the estimate if it is of significant magnitude. This work provides a means of removing the frequency offset.

Third, a method for improving the error performance of the system was developed. This consists of producing a quality metric to accompany each DOA estimate as well as filtering the output DOA estimates over time. Finally, a method for scaling the algorithm to work with a 16 element array was also considered. This involved a new approach to dealing with the phase data generated by the PLLs as the original signal processing block would be computationally infeasible when the number of antenna elements is increased.

1.1 Project Equipment Description

1.1.1 Software Introduction

Practically all algorithm development was performed in Matlab. Matlab provides an excellent environment for quick evaluation of algorithms through its combination of built in functions and ease of development. For implementation purposes, the majority of the code was simply translated from Matlab into C. Only certain functions, such as the DFT implementation and switch control were prototyped in C before integration into the larger implementation. The C program was compiled using TI Code Composer Studio and downloaded to the SDR using proprietary software developed by DRS-SS.

1.1.2 Hardware Introduction

The SDR used in this project, as well as the funding, was provided by DRS Signal Solutions, Inc. of Gaithersburg, MD. This work is part of the Sunrise University project, which was established to fund implementation-based research at the graduate level at both Virginia Tech and Carnegie Mellon University. The radio used is the WJ-8629a software-definable surveillance

receiver. The 8629a has a frequency range of 20MHz to 2.7GHz, 22 selectable IF bandwidths from 200Hz to 1.23MHz, and 5 user-definable bandwidth slots. It also contains a floating point Texas Instruments C67 DSP capable of 1GFLOPS. DRS-SS also furnished their proprietary software for program development and download as well as radio control. As far as algorithm development is concerned, the actual specifications of the hardware used for implementation is unimportant as long as it conforms to certain basic assumptions, mainly that it provides a digital baseband version of the targeted RF signal relatively free of spurious signals generated by the receiver itself.

1.2 Thesis Description

Chapter 2 will provide a brief overview of direction finding techniques and will give a few examples of both single-channel and *n*-channel algorithms. Chapter 3 continues the discussion of DF algorithms by describing in detail the PLL Algorithm for single-channel DF. In Chapter 4, we will discuss the various enhancements to algorithm that encompass the bulk of the work. These enhancements include modification of the PLL model used for both more efficient implementation and robustness in the presence of a frequency shift on the signal of interest; modification of the curve-fitting algorithm to reduce computational complexity when used with antenna arrays consisting of a large number of elements; and the development of a quality metric to aid in the determination of estimation errors.

Chapter 5 presents simulations of the PLL algorithm and its enhancements discussed in Chapters 3 and 4. The simulation results were generated using Matlab and focus on the statistical performance of the algorithm in an AWGN channel with various operating parameters. The chapter will also investigate the operation of the algorithm with a moving target and simple multipath channel model. Chapter 6 will describe the implementation of the enhancements to the algorithm. The chapter will also present limited performance results of the algorithm operating under controlled laboratory conditions.

The final chapter will present our conclusions on the work. It will discuss the main contributions of this work to the PLL algorithm as well as a summary of the implementation successes and failures and lessons learned during the research. Finally, it will describe possible future directions for research.

Chapter 2

Introduction to DF Concepts

When one searches for DF literature, they will most likely find a vast assortment of algorithms and corresponding systems. However, much of this literature presupposes a familiarity on the part of the reader with fundamental knowledge of DF topics. In this chapter we will briefly discuss some of the basic concepts encountered in practically any DF discussion as well as introduce a few well-known DF algorithms in order to illustrate the basic concepts. Two of the algorithms discussed, the Watson-Watt and Doppler/Pseudo-Doppler methods, are single-channel techniques that . The third algorithm, Correlative Vector DF, is an *n*-channel method based on simple delay-and-sum beamforming. The fourth and final algorithm discussed, MUSIC, is a high-performance algorithm capable of operating in a multipath environment. We will revisit this algorithm in Chapter 7.

2.1 Direction Finding Fundamentals

2.1.1 Basic Assumptions

DF algorithms, apart from being classified by the number of receiver channels used, can also be classified by the manner in which they treat the signal received at an antenna array. Central to this is the response of the antenna array over DOA (in azimuth and possibly elevation) as well as frequency. Approaches can be categorized as amplitude-based, phase-based, or a combination of the two. Amplitude-based systems compare the received amplitude among the various elements in the antenna array to locate a point in an plane about the array that the signal originates. Phase-based systems determine DOA information from either the absolute or differential phase of the wavefront as it crosses the array. Systems that use a combination of amplitude and phase information tend to be more complex but also result in higher performance. Beamforming and superresolution systems fall into this category.

The basic antenna array can be envisioned as an arbitrary array of isotropic antennas arranged about an arbitrarily defined origin point as in Figure 2.1. For our purposes, we need to

make a few basic assumptions about the antenna array [5]:

- The difference in amplitude of a received SOI between antenna elements is negligible.
- There is a finite number of incident SOIs, each of which can be described by a plane wave.
- The SOIs are narrowband signals.

These assumptions follow the basic requirements for narrowband beamforming systems. The first point essentially means that the antenna array is small relative to the distance of propagation such that any measured path loss from one antenna to another is insignificant. This implies that the primary response of the antenna array to a signal is by modifying the phase of the signal. The second point allows for simplified modeling of the antenna array – the multipath environment can be described as the linear sum of a finite number of signals. Finally, the narrowband assumption implies that the phase response of the array is flat across the signal's bandwidth.

2.1.2 The Received Signal Model

Referring to the array in Figure 2.1 the phase difference between any antenna and a reference point at the origin of the coordinate system is given as

$$\Delta\psi_m = \beta\left(x_m \cos\phi \sin\zeta + y_m \sin\phi \sin\zeta + z_m \cos\zeta\right) \tag{2.1}$$

where $\beta=2\pi/\lambda$ is called the phase propagation factor, ϕ is the DOA in azimuth, ζ is the DOA in elevation, and the three-dimensional coordinates of the *m*-th antenna element are given as x_m, y_m, and z_m. In all analysis in this paper, the antenna arrays of interest are two-dimensional, meaning

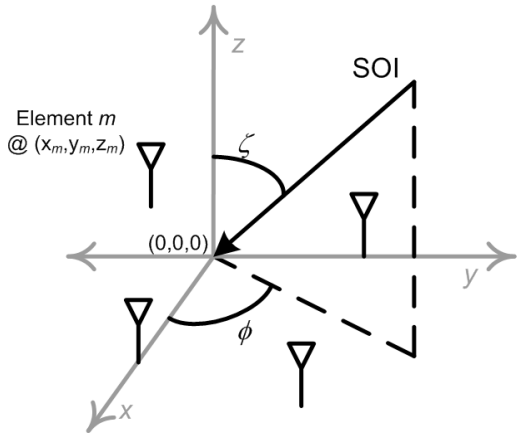

Figure 2.1. Example of an arbitrary antenna array for modeling purposes.

that the term relating to the z-coordinate of the array can be ignored. When modeling the signal received at the antenna element, the phase response of that element relative to a reference point along with the antenna's amplitude gain are seen as a complex scaling factor given by:

$$a_m(\phi,\zeta)=g_m e^{-j\Delta\psi_m}=g_m e^{-j\beta(x_m\cos\phi\sin\zeta+y_m\sin\phi\sin\zeta)} \tag{2.2}$$

where g_m is the gain for the m-th antenna, and the complex baseband output of the m-th antenna is given by:

$$x_m(t)=s(t)a_m(\phi,\zeta)=s(t)g_m e^{-j\beta(x_m\cos\phi\sin\zeta+y_m\sin\phi\sin\zeta)} \tag{2.3}$$

where s(t) is the SOI and $x_m(t)$ is the signal at the output of the m-th antenna. If we are concerned only with signals arriving on the same plane of the array then the elevation angle q is equal to 90°, resulting in:

$$x_m(t)=s(t)g_m e^{-j\beta(x_m\cos\phi+y_m\sin\phi)} \tag{2.4}$$

The collection of scaling factors into an Mx1 vector $\boldsymbol{a}$ is known as the array manifold vector or as the array steering vector in beamforming contexts:

$$\boldsymbol{a}=\left[a_0(\phi,\zeta)\;\; a_1(\phi,\zeta)\;\; \cdots\;\; a_{M-1}(\phi,\zeta)\right]^T \tag{2.5}$$

The received signal at the output of the array can then be expressed in matrix notation as

$$\boldsymbol{X}=\boldsymbol{a}\,s(t) \tag{2.6}$$

This approach can be scaled up to construct a received signal that is the combination of multiple signals. By creating a matrix S which represents the finite number of incident signals, D, as its rows:

$$\boldsymbol{S}(t)=\begin{bmatrix} s_0(t) \\ \vdots \\ s_{D-1}(t) \end{bmatrix} \tag{2.7}$$

and a second MxD matrix A:

$$A=\left[\boldsymbol{a}(\phi_{0,}\zeta_0)\;\; \cdots\;\; \boldsymbol{a}(\phi_{D-1},\zeta_{D-1})\right] \tag{2.8}$$

where $\boldsymbol{a}(\phi_d,\zeta_d)$ is the array manifold vector for the d-th received signal, we can then express the received signal as:

$$X=AS+N \tag{2.9}$$

where **X** is essentially the linear combination of D signals given by equation 6 and **N** is a noise vector representing AWGN in the received signal path that is assumed to be independent for each antenna and receiver path. This multiple signal model will be revisited in Chapter 7 during the discussion of the PLL algorithm's performance in the presence of multipath.

2.2 Classical DF Algorithms

2.2.1 Watson-Watt Algorithm

The Watson-Watt algorithm [6] is an old amplitude-based analog DF algorithm that can use either two orthogonally oriented loop or Adcock pair antennas. In an Adcock pair, the vector difference of the voltage outputs from the two antennas is taken to provide a single output from the pair. This difference operation effectively merges the circular gain patterns of the two discrete antennas into a figure-8 pattern with maximum gain along the baseline of the pair and a null along the perpendicular to the baseline midpoint. The perpendicular arrangement of two such pairs is shown in Figure 2.2 with the first pair antennas labeled A_N and A_S (for North and South antennas) and the second pair labeled A_E and A_W. This same gain pattern can be achieved with the use of two crossed loop antennas as the typical gain pattern for a loop antenna is a figure-8 with the gain pattern of the north-south pair shown as the solid gray line and the gain pattern of the east-west pair shown by the dashed black line.

In order to produce the DOA estimate, the voltage output from both antenna pairs (in the case of two Adcock pairs) is compared. The two outputs can be expressed as:

$$V_{NS} = V_N - V_S = e^{j\frac{\pi d}{\lambda}\sin\phi} - e^{-j\frac{\pi d}{\lambda}\sin\phi} = j2\sin\left(\frac{\pi d}{\lambda}\sin\phi\right) \tag{2.10}$$

$$V_{EW} = V_E - V_W = e^{j\frac{\pi d}{\lambda}\cos\phi} - e^{-j\frac{\pi d}{\lambda}\cos\phi} = j2\sin\left(\frac{\pi d}{\lambda}\cos\phi\right) \tag{2.11}$$

where d is the spacing between antenna elements. Given a sufficiently small antenna spacing, generally much less than half a wavelength at the frequency of the SOI, Equations (2.10) and (2.11) can be simplified to:

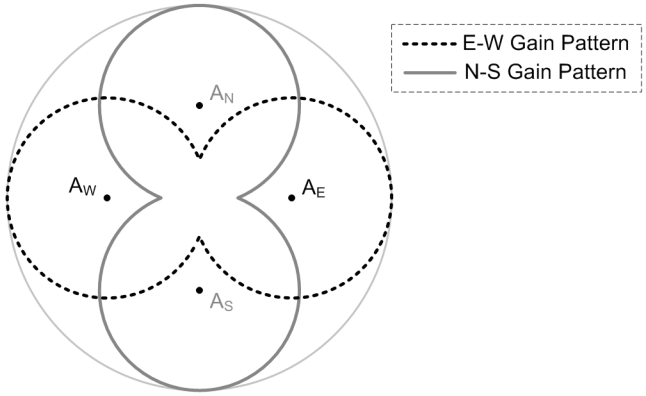

Figure 2.2. Watson-Watt antenna array gain pattern.

$$V_{NS} \approx \frac{2\pi d}{\lambda} \sin\phi \tag{2.12}$$

$$V_{EW} \approx \frac{2\pi d}{\lambda} \cos\phi \tag{2.13}$$

The north-south pair can be treated as generating the *y*-axis voltage while the east-west pair creates the *x*-axis voltage for the array's coordinate system. the arctangent of the quotient of the north-south voltage, V_{NS}, and east-west voltage, V_{EW}, as:

$$\hat{\phi} = \tan^{-1}\left(\frac{V_{NS}}{V_{EW}}\right) \tag{2.14}$$

This operation essentially uses the two voltage measurements to locate a point in an abstracted plane, the angle of this point corresponding to the DOA of the received signal.

This approach can use either a two channel receiver,with each channel devoted to receiving the combined signal from one of the Adcock pairs, or a single receiver channel that receives a multiplexed version of the signal. These two options trade twice the receiver hardware in the former case for a more complicated antenna hardware in the latter. One drawback to this system is that the gain pattern of each Adcock pair is symmetric about the pair's baseline, meaning that a signal arriving from one side of the pair is indistinguishable from a signal arriving from 180° away. This can be remedied through the use of an antenna at the center of the array to provide a reference signal, or by treating the axis voltages as complex numbers and using a four quadrant arctangent function.

2.2.2 Doppler and Pseudo-Doppler Methods

The Doppler and derivative Pseudo-Doppler algorithms [8] are single-channel algorithms that produce a DOA estimate based on the phase of the received signal. Originally, the doppler method used a single antenna that moved about the circumference of a circle at a fixed angular velocity. The pseudo-doppler method was developed using a multi-element circular array with a commutating switch that selects the antennas sequentially around the circle to approximate the circular motion of the doppler antenna. This algorithm seeks to measure the doppler shift induced on the received signal due to either the rotation of the single antenna or commutation of the switched antenna array. The received signal is modeled as:

$$r(t) = A\cos\left[2\pi f_o t + \frac{2\pi r}{\lambda}\cos\left(2\pi f_r t\right)\right] \tag{2.15}$$

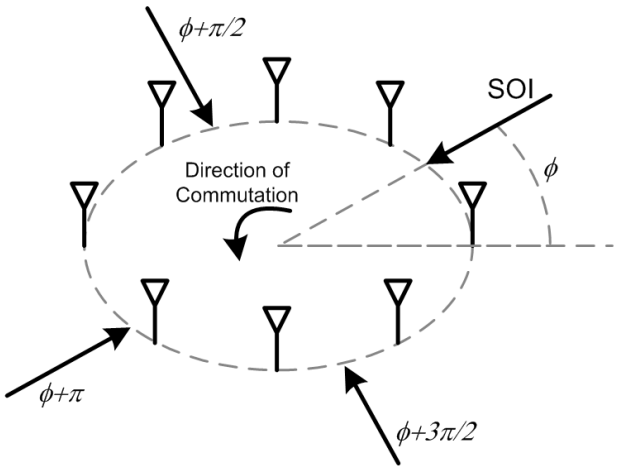

Figure 2.3. Doppler method frequency shift example.

where f_r is the inverse of the time taken to sample around the entire antenna array. For an illustrative example refer to Figure 2.3. As we sample around the array at a fixed rate, the doppler shift imposed on the signal will be directly proportional to the rate of sampling around the array. When the sampling approaches the DOA of the SOI as well as 180° away (the paths labeled ϕ and $\phi+\pi$) the measured Doppler shift will cross zero. Furthermore, the measured doppler shift will be at a negative and positive maximum when the array is sampled at 90° and 270° from the true DOA, respectively (paths $\phi+\pi/2$ and $\phi+3\pi/2$ in the example figure).

Therefore, the DOA estimation function can consist of a simple FM demodulator consisting of a frequency discriminator followed by a zero crossing detector shown in Figure 2.4. The phase of the signal at the output of the zero crossing detector will be directly proportional to the DOA. One of the drawbacks to this system is decreased listen-through capability due to FM and AM artifacts in the signal due to the sampling. Another drawback is the requirement that the speed of rotation or switching must be fast relative to any frequency modulation on the SOI as it can interfere with

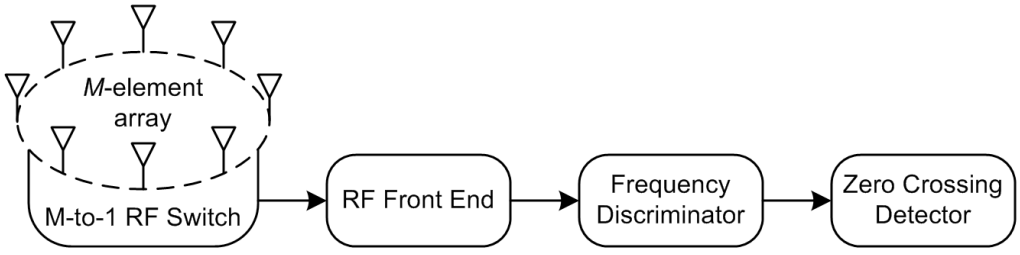

Figure 2.4. Pseudo-doppler system block diagram.

DOA estimation.

2.2.3 Correlative Vector DF

Correlative vector (CV) DF is one of the most commonly used DF methods of those based on beamforming algorithms due to its relative simplicity. It differs from the previous two algorithms in that it is an *n*-channel algorithm and it uses both the phase and amplitude of the received signal. With this approach, a database of array manifold vectors for the antenna array is generated for a set number or known DOA and frequency points, the distribution of which depends on how smooth the response is between points. This database can either be generated theoretically or through empirical measurements, depending on how closely the actual antenna array response matches the theoretical manifold. One of the main strengths of this algorithm is that it can accommodate any imperfections in array construction or installation as long as they are reflected in the database.

This algorithm is adapted from the beamforming algorithm known as delay-and-sum beamforming. Consider a generic beamforming system as shown in Figure 2.5, in which the output of each antenna from an *M*-element array is weighted by a complex number and then summed to produce a single output. The vector of complex weights is known as the steering vector, and the method of generating the steering vector is the main characteristic of any beamforming algorithm. For delay-and-sum beamforming, the steering vector is simply chosen to be the complex conjugate of a single entry from the array manifold database for a desired look direction ϕ. Weighting the received signal by this steering vector will essentially coherently combine the phase-shifted signals from each antenna in the array if the received signal's DOA corresponds to the look direction.

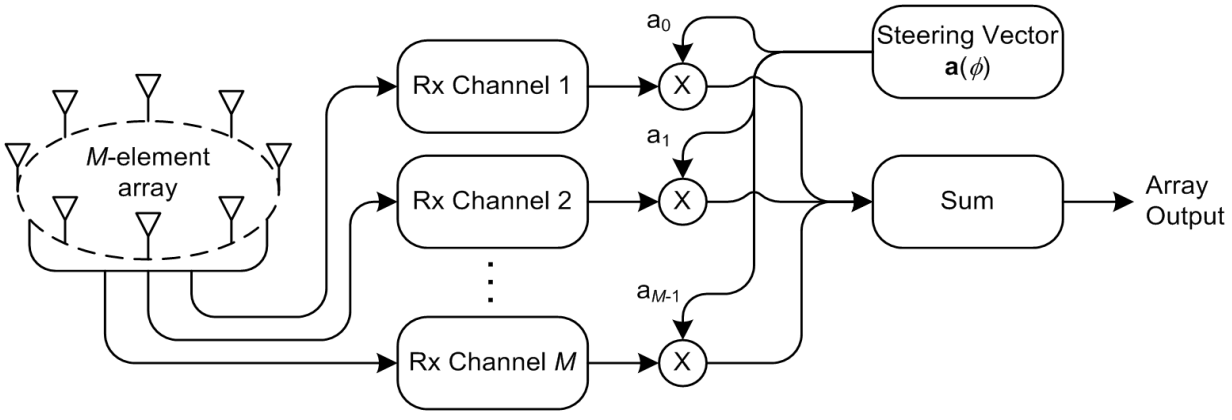

Figure 2.5. General beamforming system.

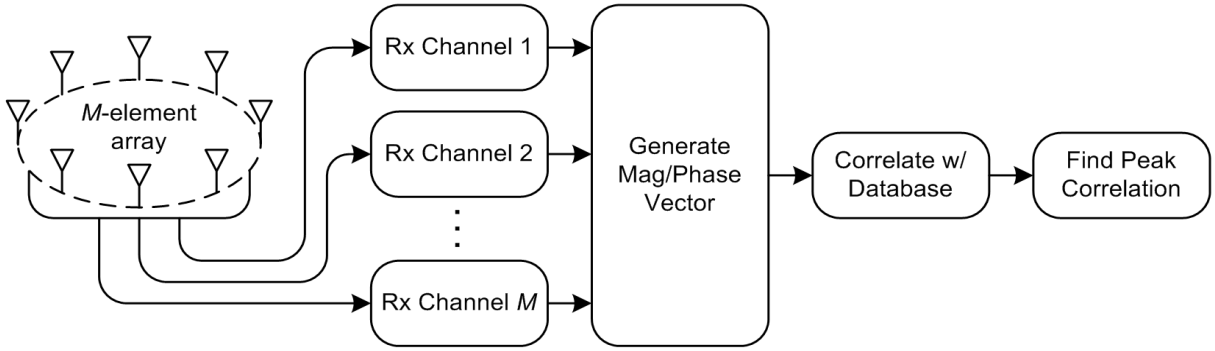

Figure 2.6. Correlative vector DF system block diagram.

When used as a DF algorithm, the DOA estimate is produced by correlating the measured array manifold with the each entry in the database at the operating frequency. The DOA at which the measured manifold exhibits maximum correlation with the database is chosen as the estimated DOA. Consider the block diagram of a general CV system in Figure 2.6. Given a measured manifold vector $\boldsymbol{v}$, the correlation with each database vector can be expressed as:

$$r(\phi)=\frac{\boldsymbol{v}\,\boldsymbol{a}(\phi)^T}{\sqrt{\left(\boldsymbol{v}\,\boldsymbol{v}^{\boldsymbol{T}}\right)\left(\boldsymbol{a}(\phi)\boldsymbol{a}(\phi)^T\right)}} \tag{2.16}$$

where $a(\phi)$ is the manifold vector in the database corresponding to the measured magnitude and phase vector at DOA ϕ relative to some defined reference point on the array. The collection of correlation values, $r(\phi)$, is known as the spatial spectrum. From the beamforming point of view, the correlation process can be seen as varying the steering vector applied to the measured received signal manifold and measuring the output power. Maximum output power will result when the steering vector corresponding to the received signal's DOA is chosen.

Figure 2.7 shows an example plot of a CV spatial spectrum (in red, dashed) for a 5-element circular array with a SOI DOA of 120°. Inspection of this plot shows that there is one definite peak at 120°, which shows that the received manifold vector has the strongest correlation with the database at that point. We can see that the main lobe of this spectrum is wide. If the azimuth points used in constructing the database are spaced too far apart to provide satisfactory results (e.g. 10 degree spacing when desired resolution is less than 1 degree), it is also possible to interpolate the resulting spatial spectrum in order to find the "true" correlation peak. Of course, processing time is traded for higher resolution when the spatial spectrum is calculated at an increasing number of

points.

2.2.4 The MUSIC Algorithm for Superresolution DF

The MUSIC algorithm, first proposed in 1979 by R. O. Schmidt [11],[12], falls into a class of high performance DF algorithms call superresolution, as the algorithm has the capability of simultaneously determining the DOA of multiple signals with a resolution of less than one degree, depending on antenna array geometry. This algorithm achieves its high resolution from the exploitation of an input covariance matrix derived from the input data model. The eigenvalues of this covariance matrix are determined and partitioned into two sets, called the signal and noise subspaces. The signal subspace is comprised of a set of array manifold vectors that correspond to the signals received at the array. The eigenvectors that form the noise subspace are completely orthogonal to the signal subspace. If used as the steering vectors in a beamforming system, the noise subspace vectors would effectively destructively combine the received signals because the noise vectors are orthogonal to the manifold of the received signals.

The MUSIC algorithm, a block diagram of which can be found in Figure 2.8, can be summarized in as follows [13]:

1. First, *K* samples from each receiver channel must be collected to form *M*x*K* array **X**. For simulation purposes, this array can be generated according to equation (2.9). Next, the covariance matrix R_{xx} must be estimated from the received data:

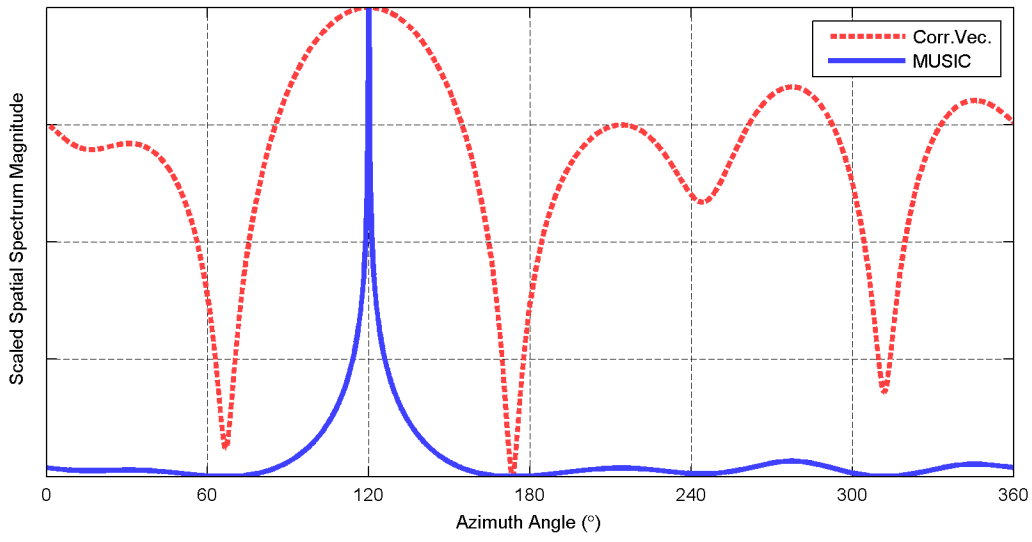

Figure 2.7. Example spatial spectra from the correlative vector and MUSIC DF algorithms with a single received signals at a DOA of 120°.

$$R_{xx} = \frac{1}{K} XX^H \tag{2.17}$$

2. Perform eigenvalue decomposition on R_{xx}:

$$R_{xx} V = V \Lambda \tag{2.18}$$

where $\Lambda = diag\{\lambda_0, \lambda_1, \ldots, \lambda_{M-1}\}$, $\lambda_0 \leqslant \lambda_1 \leqslant \ldots \leqslant \lambda_{M-1}$ are the eigenvalues and $V = [\boldsymbol{q_0}\ \boldsymbol{q_1} \cdots \boldsymbol{q_{M-1}}]$ are the corresponding eigenvectors of R_{xx}.

2. From the multiplicity $\hat{k}$ of the smallest eigenvalue λ_{min}, estimate the number of signals $\hat{D}$:

$$\hat{D} = M - \hat{k} \tag{2.19}$$

3. Form the noise subspace V_n from the eigenvectors corresponding to the $\hat{k}$ smallest eigenvalues. Determine the MUSIC spatial spectrum:

$$P_{MUSIC}(\phi) = \frac{a^H(\phi) A(\phi)}{a^H(\phi) V_n V_n^H a(\phi)} \tag{2.20}$$

where $a(\phi)$ is the manifold vector from the database corresponding to DOA ϕ.

4. Find the $\hat{D}$ largest peaks of $P_{MUSIC}(\phi)$. These correspond to the DOA of the received SOI(s).

When comparing the MUSIC spatial spectrum to that of the CV it is illustrative to consider the equations used for spatial spectrum calculation. Equations 2.16 and 2.20 are similar in that they both produce a correlation between some form of the received array manifold with a previously stored or measured manifold in the form of the database. The main difference is that the MUSIC algorithm is an inverted measure of the correlation as the vectors used in this algorithm are supposed to be orthogonal to the received signal, meaning that the correlation between the noise subspace is very nearly zero when correlated with the database at azimuth angle corresponding to

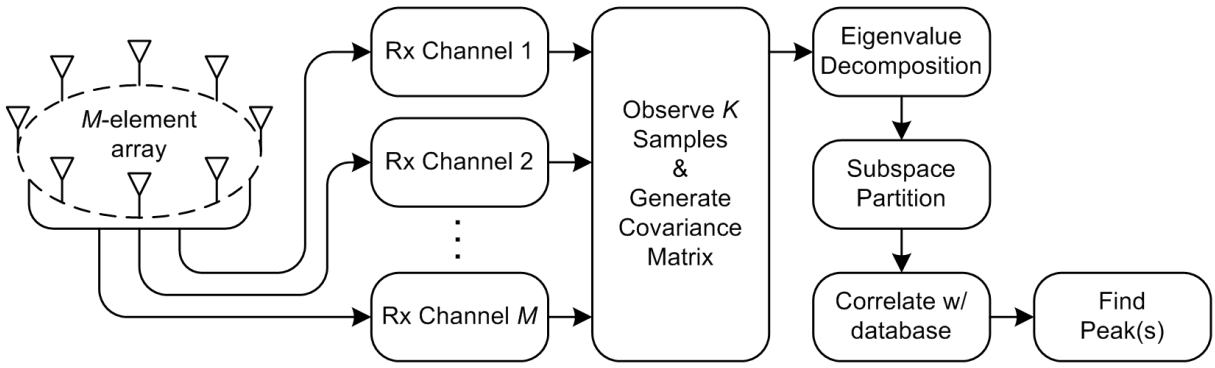

Figure 2.8. MUSIC algorithm block diagram.

the DOA of the received signal.

An example spatial spectrum for the MUSIC algorithm is shown in Figure 2.7 for a single received signal with a DOA of 120° and a five element antenna array. Compared to the CV spatial spectrum, the peak of the MUSIC spatial spectrum is obviously much narrower. This allows for the much higher resolution of the MUSIC algorithm – the wide peak of the CV spectrum can hide multiple signals arriving with a narrow angular spacing. If the noise subspace vectors were used as the array weights in a beamforming system, the resulting spatial spectrum would be a mirror image of the MUSIC spatial spectrum along the vertical axis – a nearly flat response across azimuth angles with sharp nulls at the DOA of the received signal.

A second comparison of the two algorithm's spatial spectra can be found in Figure 2.9. This is the result of an example simulation of two signals arriving at the array at angles of 120° and 200°, with a 3dB difference in the power of the first signal relative to the second. The CV spectrum clearly shows a peak at 120° but the second peak at 200° is nearly indistinguishable from the other sidelobes present in the spectrum because the CV spectrum is sensitive to the amplitude of the received signals. If one signal is significantly stronger than the other, the less powerful signal will essentially be hidden by the sidelobes of the spectrum. The MUSIC spectrum on the other hand shows two clearly defined peaks. This spatial spectrum is not dependent on the received signal strength of the signals because the power of the received signals is not reflected in the eigenvectors used to form the signal subspace – the eigenvalues contain this information.

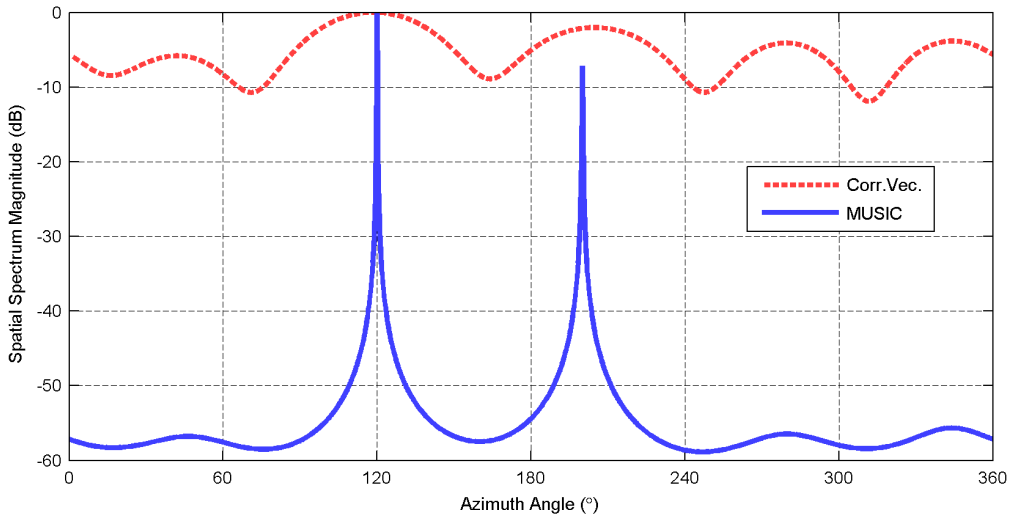

Figure 2.9. Example spatial spectra from the correlative vector and MUSIC DF algorithms with two received signals at DOAs of 120° and 200°.

Chapter 3

The PLL Algorithm For Single Channel DF

The previous chapter presented some of the basic concepts of DF systems and presented a few examples of algorithms. We will now move on to discuss the PLL Algorithm for single-channel DF, emphasizing at points where it draws from previously discussed algorithms. This chapter will first work through the mathematical background and lay out the assumptions we make about the system. This discussion will concern the basic version of the algorithm as originally developed for use with an 8-element array.

3.1 Algorithm Overview

The basic PLL DF system, as shown in Figure 3.1, consists of an *M*-element uniform circular array with an *M*-to-1 RF switching network and a software radio platform. Initial development of this algorithm assumed that it would be used with an existing 8-element array. The receiver filters and downconverts the RF input to a complex baseband signal which is then fed into a bank of *M* parallel software PLLs which are used to track the phase of the signal present at each antenna. After a single array sweep, i.e. the time it takes to switch around the array from the first to

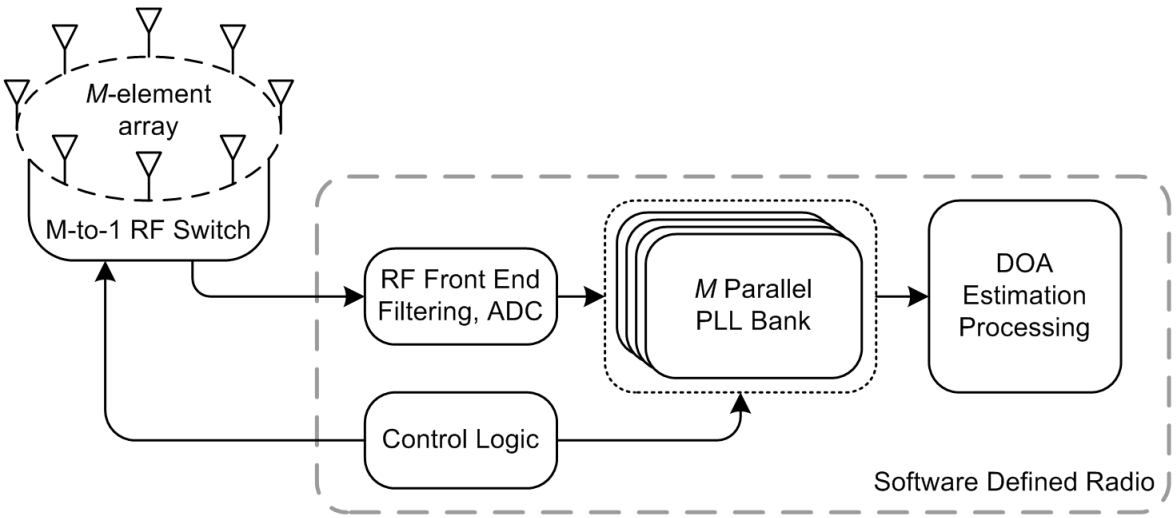

Figure 3.1. PLL DF system block diagram.

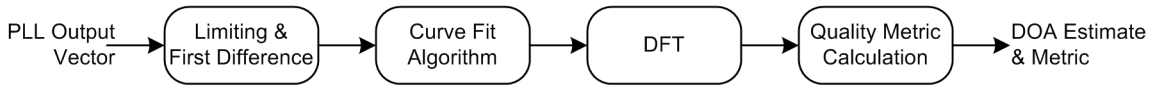

Figure 3.2. PLL DF method DOA estimation processing block diagram.

last element, the final value of each PLL is stored in an *M*x1 vector. This vector of phase values is then processed to produce a single DOA estimate for the array sweep.

This algorithm is based on the phase modulation induced on the incident signal by switching by the RF switching. The PLLs are used to measure the phase of the signal from each element in the array. The steady-state response of the PLLs then forms the measured phase data used by the DOA estimator to produce its estimate. A block diagram of the processing flow after the PLL is shown in Figure 3.2. The vector of phase samples from the PLL output, one measured value for each antenna, is first differentiated to remove any constant phase offset on the data and to limit the data's amplitude. It is then processed by a curve fit algorithm to remove any ambiguities on the PLL data related to data modulation on the received signal. A DFT is then performed on the PLL data in order to extract the DOA information as well as to produce a quality metric for the DOA estimate to assess the amount of noise and distortion present on the data output from the curve fit algorithm.

3.2 Mathematical Development

3.2.1 Antenna Array Model and Expression for PLL Input

Consider a uniform circular array of *M* elements with spatial coordinates x_m and y_m given by

$$\begin{aligned} x_m &= r\cos\left(\frac{2\pi m}{M}\right) \\ y_m &= r\sin\left(\frac{2\pi m}{M}\right) \end{aligned} \tag{3.1}$$

with the reference point of the array being the center of the circle. Working from the framework for describing the phase relationship between elements in an arbitrary antenna array as described in Chapter 2, we can simplify the array manifold given by equation 2.4 to

$$a_m(\phi) = g_m e^{-j\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)} \tag{3.2}$$

Furthermore, the signal present at the output of the *m*-th antenna can be given by

$$x_m(t) = s(t)\, e^{-j\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)} \tag{3.3}$$

where $s(t)$ is the complex baseband envelope of the the incident SOI. This envelope is assumed to take the form

$$s(t) = m(t)\, e^{-j\theta_o} \tag{3.4}$$

where $m(t)$ is the modulated data signal, either real or complex, and $e^{-j\theta_o}$ is a phase shift on the signal resulting from an arbitrary propagation distance of the signal. This phase shift is assumed to be constant across all elements of the antenna array, which is expected if the propagation distance is large compared to the radius of the array. The resulting signal presented to the PLL corresponding subsequently takes the form

$$x_m(t) = m(t)\, e^{-j\left[\theta_o+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)\right]} \tag{3.5}$$

We can see from equation (3.5) that the phase of the signal received at the m-th antenna element is related to the DOA of the incident signal by a sine curve with an angular frequency described given by the antenna element's angular displacement from the first antenna, amplitude given by the ratio of the array radius to signal wavelength, and phase given by the DOA itself, as:

$$\theta_m = \theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right) \tag{3.6}$$

This means that if the phase of the signal received at the array neglecting any data modulation on the signal itself is plotted versus the antenna index, one full period of a sine curve will result. It is this relationship that the DOA estimator seeks to extract from the PLL data. Figure 3.3 is an example plot of the PLL acquiring the phase of the received signal as the array is switching. In this example, the received signal is not modulated and the antenna array switches from one element to the next every 64 samples, which can be observed in the phase discontinuities at every 64th sample. The PLL's output value for the *m*-th antenna is taken as the final value of the PLL at the end of the data block corresponding to antenna *m*. Plotting the *M* final PLL values against the antenna index will produce an estimate of equation (3.6).

If the estimator is able to recover the sine curve traced by the switching-induced phase modulation, then it is possible to determine the DOA by measuring the phase shift of that sinusoid. Of course, data modulated onto the signal must be considered. For the development of this algorithm, it was assumed that the modulation format would be BPSK, with the signal $s(t)$ being

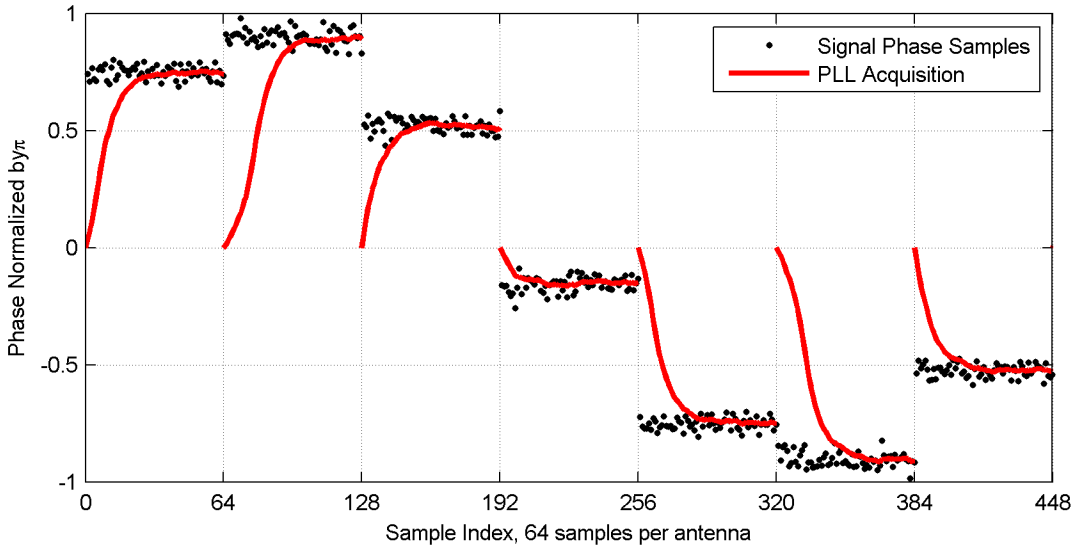

Figure 3.3. Example PLL acquisition on an unmodulated input signal.

expressed as

$$m(t)=Ae^{jn\pi}, \quad n=0,1 \tag{3.7}$$

where n represents the binary data symbols 0 or 1. Substituting this into equation (3.5) gives

$$x_m(t)=Ae^{-j\left[\theta_o+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)+n\pi\right]}, \quad n=0,1 \tag{3.8}$$

It is this data that will be fed into the software PLL for the m-th antenna.

3.2.2 Expression for PLL Output

The PLL is assumed to take the form of a carrier recovery loop for a BPSK signal. This carrier recovery loop will have the effect of tracking the phase of the signal's carrier while removing the BPSK modulation. One downside to this is that due to an arbitrary signal constellation rotation, the acquired phase can be shifted by π. This is because the PLL will only track angles in the first and fourth quadrants. With this in mind, we assume that the output of the PLL corresponding to the m-th antenna can be expressed as

$$\theta_m=\theta_o+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)+n\pi, \quad n=0,1 \tag{3.9}$$

If the data symbols and constant offset θ_o are known, we can compute the DOA using

$$\hat{\phi}=\cos^{-1}\left(\frac{\theta_m-\theta_0}{2\pi r/\lambda}\right)-\frac{2\pi m}{M} \tag{3.10}$$

where m is the antenna index as used above. Unfortunately this information is generally not known

a priori so an alternate approach needs to be devised.

Figure 3.4 illustrates this issue. The line labeled "Expected PLL Output" is the theoretical output of the PLLs without a phase shift or modulation effects plotted against antenna index. It is obviously a clean sinusoid. The line labeled "Actual PLL Output w/ offset" is the actual output of the PLLs for an unmodulated signal with a constant phase offset. This is still a sine curve, and evaluating equation (3.10) will still work as the constant phase shift appears simply as a DC offset on the data. The problem becomes more complex when modulation is added. The line labeled "Actual PLL Output w/ BPSK" is a plot of the PLL output data when the input signal is modulated. Notice that it only contains values in in the range of $-\pi/2 \leqslant \theta_m \leqslant \pi/2$ because as previously mentioned the PLL will only track phase values in the first and fourth quadrants. We can see that when this data differs from the actual PLL output without modulation the difference is exactly π. This ambiguity will obviously hinder any simplistic attempts to determine the DOA.

Furthermore, the amplitude of the phase curve, given above in equation 25 as $2\pi r/\lambda$, can take on values larger than π, which can cause problems with fitting the data to a sinusoid because due to the $\pm\pi$ on the data each element in the vector can have up to two additional possible true values to fit in the amplitude range. The examples in Figure 3.4 used an array radius equal to $\lambda/2$. Obviously, this assumption will only work for an antenna array operating at a single frequency. DF

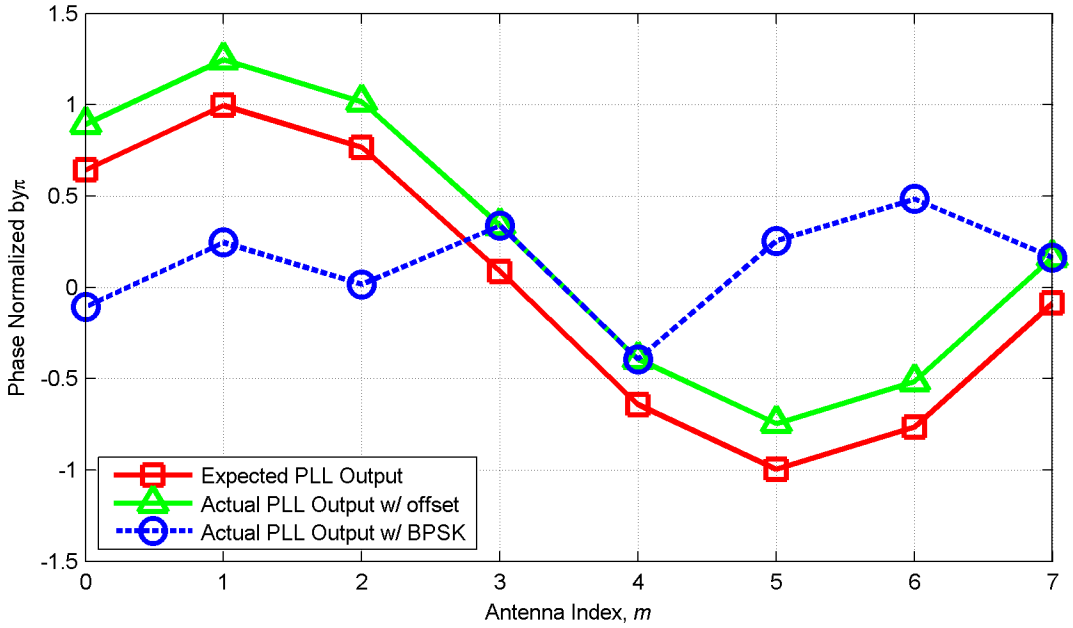

Figure 3.4. Example PLL output data for 8 antennas.

systems typically require an inter-element spacing of less than $\lambda/2$ in order to reduce any resulting ambiguities [8],[10]. For an 8 element circular array, this spacing will require an array radius of at most 0.65λ, which will result in the phase curve having an amplitude of 1.25π. This means that the phase tracked by the PLL will be wrapped into the range $-\pi \leqslant \theta_m \leqslant \pi$. Consequently, the curve fit algorithm must be able to handle phase wrapping on the PLL data in addition to ambiguities related to the modulation.

3.2.3 8-element Curve Fit Algorithm

We have thus far shown that the curve fit algorithm will need to deal with three major sources of phase ambiguity: phase wrapping due to a constant offset in the received signal, phase wrapping due to the phase curve's amplitude, and phase ambiguities due to the received signal's modulation. By recognizing that the first derivative of a sinusoid is another sinusoid, we are able to facilitate the process somewhat. If we define the first difference of the PLL data as

$$\Delta\theta_m = \theta_m - \theta_{m-1} \tag{3.11}$$

then the first difference of the PLL data can be expressed as

$$\Delta\theta_m = \left(\theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M} - \phi\right)\right) - \left(\theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{2\pi (m-1)}{M} - \phi\right)\right) \tag{3.12}$$

$$\Delta\theta_m = \left(\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M} - \phi\right)\right) - \left(\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M} - \frac{2\pi}{M} - \phi\right)\right) \tag{3.13}$$

$$\Delta\theta_m = -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\sin\left(\frac{2\pi m}{M} - \frac{\pi}{M} + \phi\right) \tag{3.14}$$

Note that differentiation has multiple effects: the constant phase offset θ_o is eliminated, the amplitude of the resulting sine curve has been scaled by $2\sin(\pi/M)$, and the DOA information is preserved as the phase offset of a sine curve. For the 8-element array with an inter-element spacing of $\lambda/2$ and subsequently a radius of 0.65λ, this will result in a sine curve with an amplitude of 0.995π.This result does ignore modulation ambiguities on the PLL data. If we take modulation into account while assuming that during the differentiation the resulting data points are limited to the region $-\pi \leqslant \Delta\theta_m \leqslant \pi$, we can express the resulting first difference data as

$$\Delta\theta_m = -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\sin\left(\frac{2\pi m}{M} - \frac{\pi}{M} + \phi\right) + n\pi, \quad n = 0,1 \tag{3.15}$$

The assumed limit of $\pm\pi$ on the first difference data is a valid assumption because the elimination of the DC component of the data means that the resulting curve will be centered about zero guaranteeing that the data will be less than π for element spacings of less than $\lambda/2$. This is illustrated in Figure 3.5. The "Expected 1st Diff. Data" represents the first difference of simulated PLL data without modulation. Reliance on the first difference of the phase data means that we are not necessarily interested with the exact value of the phase modulation induced by switching but with the differential phase modulation on the received signal.

This relation forms the basis of the curve fit algorithm. Given that each element of the first difference data vector can possibly be shifted from its expected value by π, this results in a total of 2^8 possible permutations from the measured first difference data, which are called the given difference curves. In order to pick the correct curve, these 256 possible curves are each compared to a database of theoretical first difference curves called the target difference curves. The mean squared error is determined between each given and target difference curve, and the given difference curve with the smallest MSE relative to any of the target difference curves is chosen as the true first difference curve from the set. The size of the set of target difference curves only needs to be large enough to provide a meaningful measure of MSE. In practice, it was found that only 16 to 32 target curves are necessary to provide good estimates. This operation is in a way similar to the correlation

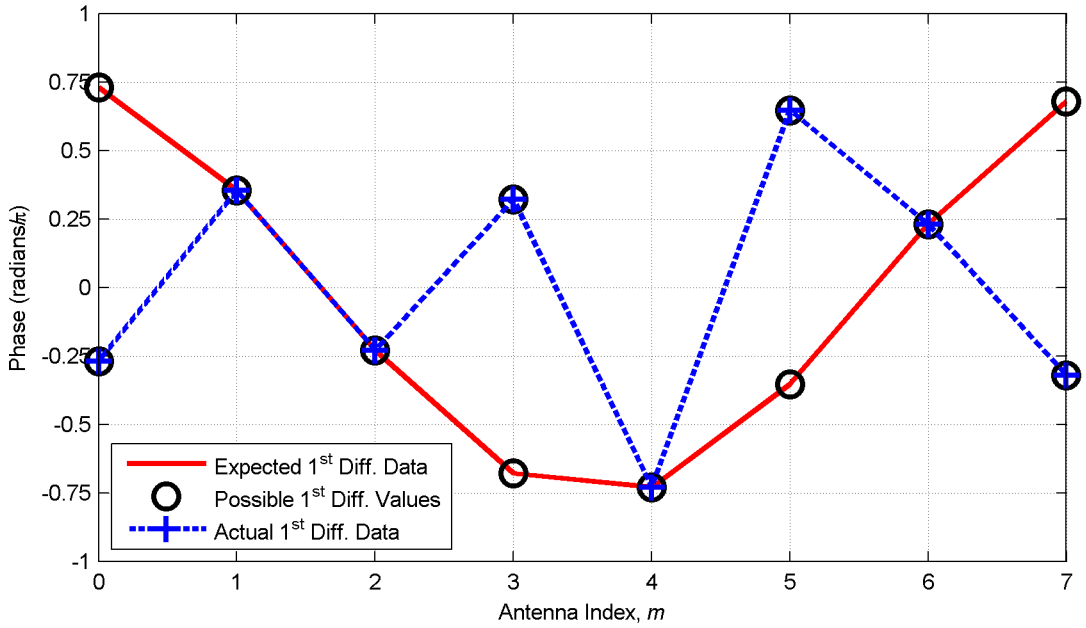

Figure 3.5. Example 1st difference data.

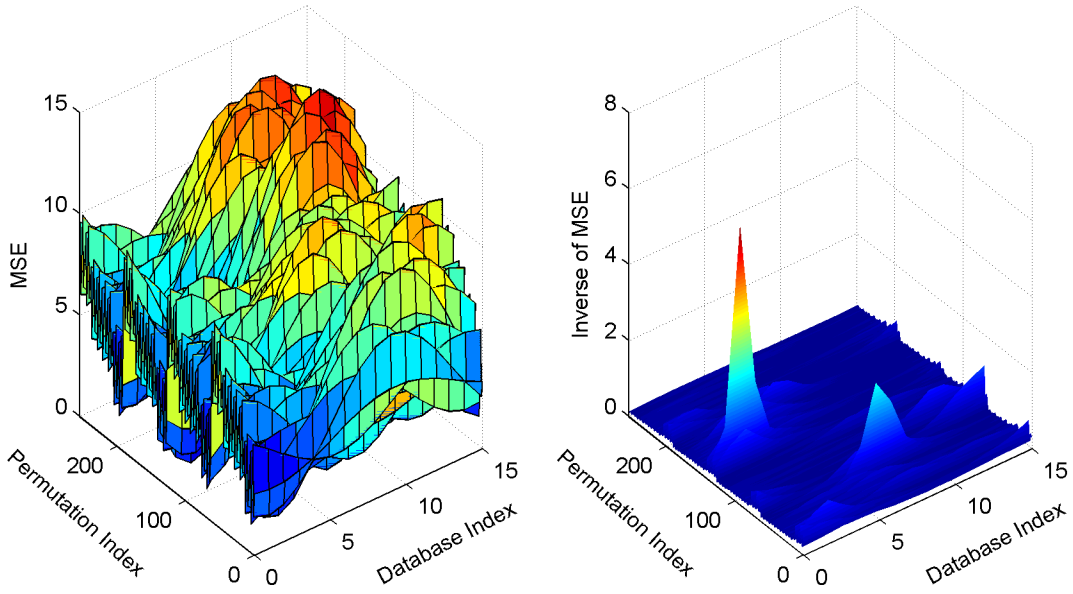

Figure 3.6. A.) left, MSE data from curve fit algorithm; B.) right, inverse of MSE data from curve fit algorithm.

process from the Correlative Vector algorithm except that the data set that is searched is two-dimensional instead of one-dimensional.

Figure 3.6a is an example plot of the measured MSE between all 256 permutations of a simulated first difference data set and a set of 32 target difference curves. The index of the first difference permutation that produces the smallest MSE is chosen. In order to demonstrate this concept better, Figure 3.6b plots the inverse of the data in Figure 3.6a. In Figure 3.6b, we can see that there is a single large peak, corresponding to a MSE that is close to zero, and that every other inverted MSE value is small.

3.2.4 DOA Estimation

Once the true first difference curve is selected, the final step in the DOA estimation is to determine the phase of the sine curve described by the data. This information can be retrieved by performing a DFT of the sequence. Recognizing that the difference curve can be represented as

$$f[m]=\begin{bmatrix} -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\sin\left(-\frac{\pi}{M}+\phi\right) \\ -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\sin\left(\frac{2\pi}{M}-\frac{\pi}{M}+\phi\right) \\ \vdots \\ -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\sin\left(2\pi-\frac{3\pi}{M}+\phi\right) \end{bmatrix} \tag{3.16}$$

the DFT of this vector can be evaluated as

$$F[k]=\sum_{m=0}^{M-1} f[m]e^{-j2\pi mk/M} \tag{3.17}$$

In the absence of noise, each bin of the DFT will be zero except for $k=1$. This is due to the fact that the data is still one full period of a sine wave. Windowing of the sequence is unnecessary since the sine wave is sampled such that all of the energy of that sine wave will be contained in the second bin of the DFT and that the sidelobes of its frequency transform will not appear in the DFT. Therefore, for $k=1$:

$$F[1]\propto\left(\left(\phi-\frac{\pi}{M}\right)+j\left(\phi-\frac{\pi}{M}\right)\right) \tag{3.18}$$

and the DOA estimate can be found as the angle of that complex number minus a constant offset:

$$\hat{\phi}=\measuredangle F[1]-\frac{\pi}{2}+\frac{\pi}{M} \tag{3.19}$$

Chapter 4

Enhancements to the PLL Algorithm

The previous chapter described the basic PLL Algorithm in detail for use with an 8 element array. In this chapter we will discuss the main improvements to the algorithm. First, we will discuss two modifications to the PLL: the change from a nonlinear to linear model, and the modification of the PLL for removal of a residual frequency component on the signal. We will then move to discuss two methods for improving the DOA estimate error performance: filtering the DOA estimates and defining a quality metric associated with the DOA estimates. Finally, we will introduce a new curve fit algorithm use with a 16 element array.

4.1 Modifications to the PLL

To preface the discussion of the PLL modifications, the general PLL model and its operation with antenna switching are considered. This leads into the discussion of the change from a nonlinear to linear PLL and the modification for the removal of a frequency component.

4.1.1 The PLL Model

The PLL model used to estimate the phase of the received signal is that of a software Costas PLL with decision directed feedback. The Costas PLL was originally proposed by J. P. Costas in 1956 [14] as an optimal demodulator for amplitude modulated (AM) double sideband suppressed carrier (DSB-SC) systems and is shown in Figure 4.1. In this PLL, the received RF signal $s(t)$,

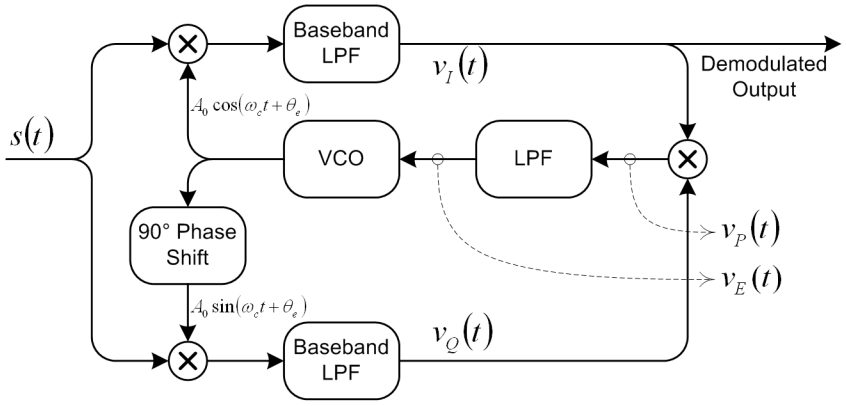

Figure 4.1. Block diagram of operation of Costas PLL for synchronous demodulation of DSB-SC signals.

$$s(t) = A_C \cos(\omega_c t) m(t) \tag{4.1}$$

is mixed with a quadrature oscillator produced by the PLL's voltage controlled oscillator (VCO) and both output signals are passed through lowpass filters in order to remove high frequency components, resulting in

$$v_I(t) = A_C A_0 \cos(\theta_e) m(t)/2 \tag{4.2}$$

$$v_Q(t) = A_C A_0 \sin(\theta_e) m(t)/2 \tag{4.3}$$

where A_C is the amplitude of the received signal, A_0 is the amplitude of the VCO, θ_e is the phase difference between the received signal's carrier and the VCO, and $m(t)$ is the modulating signal. When the phase error θ_e in the system is small, the in-phase signal $v_I(t)$ is proportional to the modulating signal and is used as the demodulated output of the system whereas the quadrature signal $v_Q(t)$ is practically zero. The in-phase and quadrature signals are then multiplied, resulting the the product voltage $v_P(t)$,

$$v_P(t) = \left(A_C A_0 m(t)\right)^2 \sin(2\theta_e)/8 \tag{4.4}$$

which is subsequently passed through a lowpass filter in order to produce a control voltage for the VCO, $v_E(t)$, as

$$v_E(t) = K \sin(2\theta_e) \tag{4.5}$$

which is a DC signal proportional to the sine of the phase error where K is a constant related to the filter and VCO settings. This error detector is known as a sinusoidal error detector, since the control signal to the VCO is sinusoidal function of the phase error.

This model for the Costas PLL can be abstracted for implementation in software and extended for the demodulation of PSK signals with the addition of decision directed feedback. The block diagram for this PLL is shown in Figure 4.2. This model assumes that the complex input

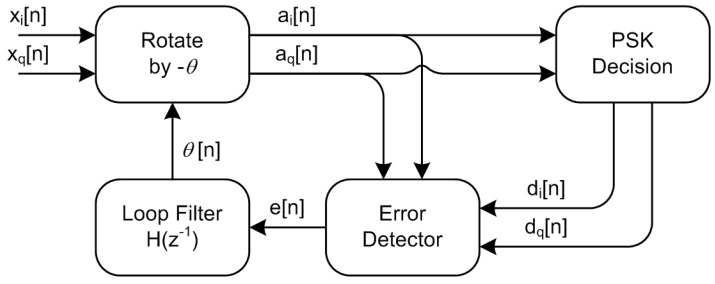

Figure 4.2. Block diagram of software Costas PLL with decision directed feedback.

signal $x[n]$ has already been downconverted, filtered, and sampled by the SDR's RF front end. Initially, we assume that the PLL is at phase $\hat{\theta}[n]$. The input signal $x[n]$ is rotated, via complex multiplication, to form intermediate signal $a[n]$.

$$a[n]=x[n]\cdot e^{-j\hat{\theta}[n]} \tag{4.6}$$

This is analogous to the quadrature mixer and baseband filters of the original Costas PLL. Complex signal $a[n]$ is then passed to the decision block that maps the intermediate signal to a demodulated PSK symbol $d[n]$. For a BPSK signal, this decision is formulated as

$$d[n]=\begin{cases} 1+j0, \Re\{a[n]\}\geqslant 0 \\ -1+j0, \Re\{a[n]\}<0 \end{cases} \tag{4.7}$$

where $\Re\{a[n]\}$ denotes the real part of $a[n]$. After this step, both $a[n]$ and $d[n]$ are used to create the error signal $e[n]$, by evaluating:

$$e[n]=\Im\{a[n]d^*[n]\} \tag{4.8}$$

where $d^*[n]$ is the complex conjugate of $d[n]$ and $\Im\{\cdots\}$ represents taking the imaginary part. Investigating further, we find that the argument of $\Im\{\cdots\}$ in equation (4.8) can be simplified as

$$\begin{aligned} a[n]\cdot d^*[n] &= |a[n]|e^{j\measuredangle a[n]}|d[n]|e^{-j\measuredangle s[n]} \\ &= |a[n]|\cdot|d[n]|e^{j(\measuredangle a[n]-\measuredangle d[n])} \\ &= |a[n]|\cdot|d[n]|\left[\cos(\measuredangle a[n]-\measuredangle d[n])+j\sin(\measuredangle a[n]-\measuredangle d[n])\right] \end{aligned} \tag{4.9}$$

which after isolating the imaginary part results in

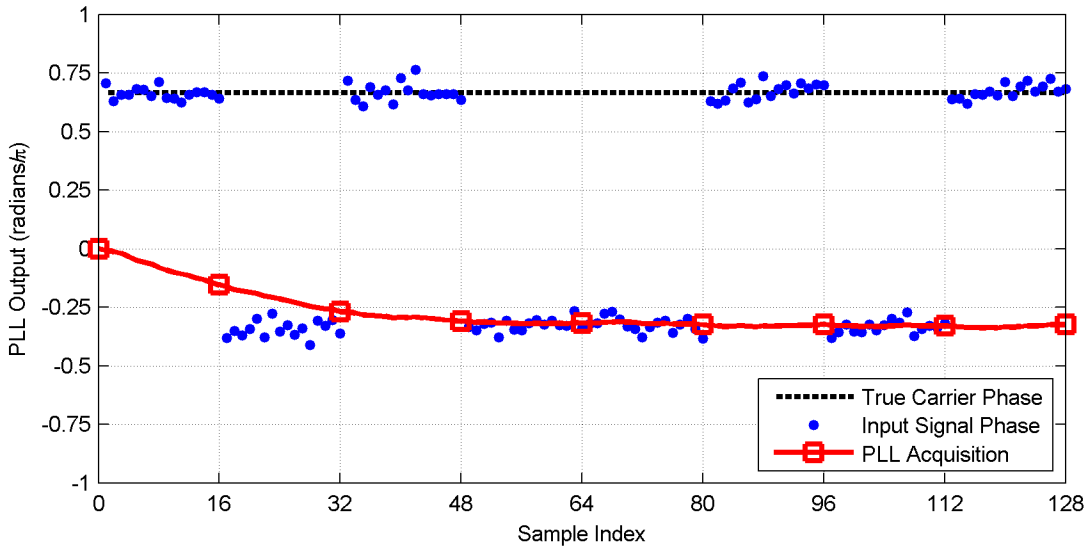

Figure 4.3. Example PLL acquisition with the presence of BPSK modulation.

$$e[n] = |a[n]| \cdot |d[n]| \cdot \sin\left(\measuredangle a[n] - \measuredangle d[n]\right) \tag{4.10}$$

where we find that the error signal is therefore proportional to the sine of the phase difference between the received symbol $a[n]$ and its demodulated counterpart $d[n]$. Note that the error signal is also influenced by the amplitude of the received signal, which means that unless the received signal is hard-limited $e[n]$ will be affected by noise on both the amplitude and phase of the input signal.

This error formulation is where the term "decision-directed feedback" stems from – unlike the original Costas PLL where the error signal was a function of the phase error between the received signal's carrier and the PLL's VCO, the error signal here is both a function of the phase error between the received signal and the PLL and the difference between the received signal's phase and the demodulated symbol, as follows.

$$e[n] \propto \sin\left(\measuredangle a[n] - \measuredangle d[n]\right) \propto \sin\left(\measuredangle x[n] - \hat{\theta}[n] - \measuredangle d[n]\right) \tag{4.11}$$

Decision-directed feedback is also what induces the $\pm\pi$ ambiguities on phase tracked by the PLL. When used with any PSK signal, the ambiguity on the output signal will be related to the rotational symmetry of the signal constellation. An example of PLL acquisition is shown in Figure 4.3 for a BPSK modulated signal. The line labeled "True Carrier Phase" represents the phase of the received signal without modulation, and the dots labeled "Input Signal Phase" represent samples of the actual modulated signal's phase. We can see that although the true signal's phase is at $2\pi/3$, the PLL tracks to $-\pi/3$ as the true carrier phase is in the second quadrant. Moreover, if we wished to use

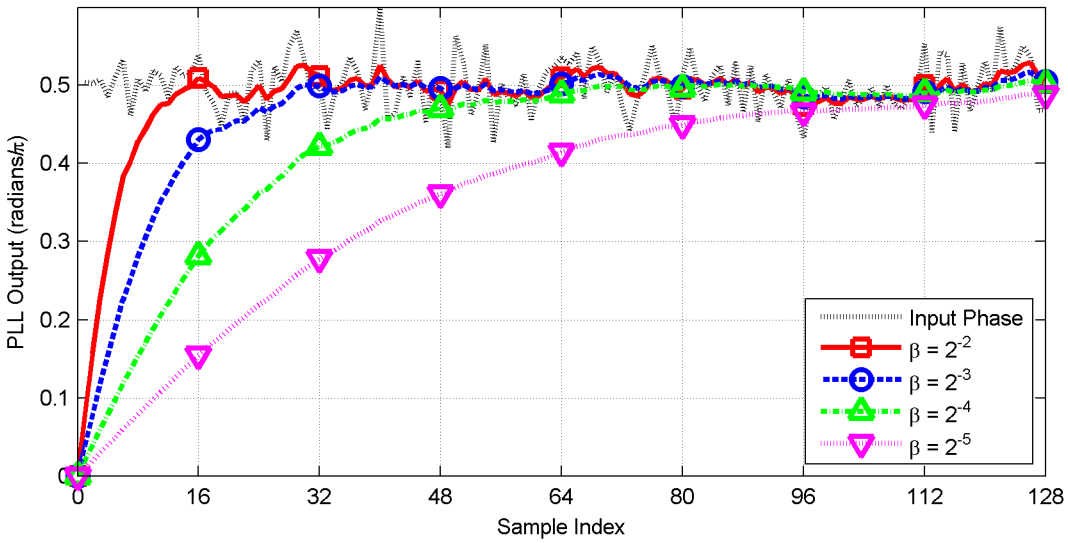

Figure 4.4. Example PLL acquisition with varying loop filter gain.

this PLL to demodulate an AM signal, we would simply force the output of the decision block, $d[n]$, to be a constant 1, which would simply eliminate the presence of the decision feedback resulting in

$$e[n] \propto \sin(\measuredangle a[n] - 0) \propto \sin(\measuredangle a[n]) \propto \sin(\measuredangle x[n] - \hat{\theta}[n]) \tag{4.12}$$

which is analogous to the VCO control signal given in equation (4.5) and is also a sinusoidal error detector.

After the error detector, the signal e[n] is passed through a first order IIR lowpass filter with the transfer function

$$H_{Loop}[z^{-1}] = \frac{\beta}{1 - z^{-1}},\ \beta < 1 \tag{4.13}$$

where β is the update parameter for the filter. It controls the filter's bandwidth and subsequently the time it takes to acquire the signal. This filter can be implemented in software by simply evaluating the difference equation:

$$\hat{\theta}[n+1] = \beta \cdot e[n] + \hat{\theta}[n] \tag{4.14}$$

Figure 4.4 provides an example of the PLL output phase when presented with an unmodulated signal with a phase of $\pi/2$ for values of β between 2^{-2} and 2^{-5}. We can see that as the filter gain decreases, the PLL takes a longer time to acquire the signal. Furthermore, we would expect that the variance of the PLL output would decrease with filter gain as the loop filter's bandwidth also decreases, thus reducing the noise power.

4.1.2 Parallel PLL Operation with Antenna Switching

One important consideration for the operation of the PLL is that the input to the PLL is discontinuous in time. As stated in Chapter 3, the signals from each antenna in the array are routed through an RF switching network which acts as a multiplexer in time that presents the receiver with a single RF signal. Fortunately, switching is completely under the DF system's control, so the dwell time on each antenna is known. In order to preserve the parallel operation of the PLLs, the state of the *m*-th PLL is stored at the end of that antenna's dwell period and the stored state of the next PLL is retrieved to initialize the PLL for the next antenna's data. This allows the PLL to track the phase of the received signal at each antenna over time – the PLLs do not have to reacquire at every array sweep, which allows for the selection of a PLL gain that results in an acquisition time that is long relative to each data block.

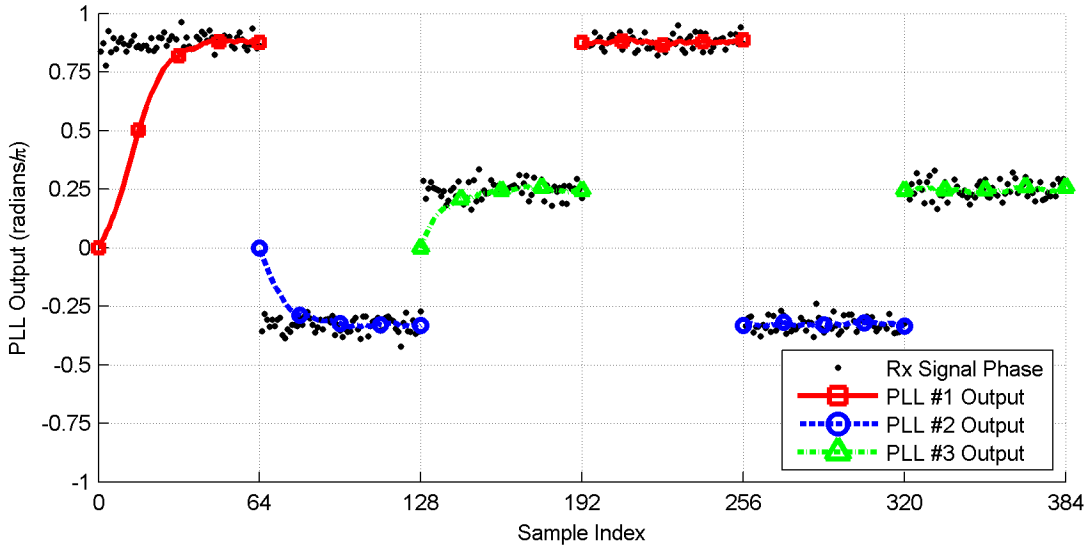

Figure 4.5. Example parallel PLL operation.

Figure 4.5 is an example plot of the PLL acquisition acquiring and tracking with an input signal from an arbitrary 3-element antenna array over two successive array sweeps. Consider the line labeled "PLL #1 Output," which is the phase output of the first PLL over time. We can see that during the first 64 samples, the PLL acquires the input signal phase, starting from its initial value of zero. At the 64th sample, the value of the PLL is stored and the PLL is reinitialized for the second antenna's signal. When the second data block from the first antenna arrives at sample 192, we can see that the value of the first PLL is then reloaded and the PLL is thus able to continue tracking the received signal's phase from its previous value instead of reacquiring the signal.

4.1.3 Nonlinear to Linear PLL

One drawback to the PLL model chosen is that although although it is conceptually simple, implementation in a real-time DF system can be problematic due to the number of mathematical operations required to update the PLL for each sample. For example, the PLL's loop filter outputs a single real phase value. In order to perform the initial phase rotation of the input signal, this phase value was converted to a complex value through through the CORDIC algorithm. During initial implementation of the algorithm, it was found that the curve fit algorithm requires a significant proportion of the processing and that in order to support sustained real-time DF estimation, the computational overhead related to PLL operation needed to be reduced. Therefore, the PLL needed to be either modified or replaced with a more efficient model to satisfy this goal. Fortunately, an overlooked aspect of the target implementation platform offered an attractive

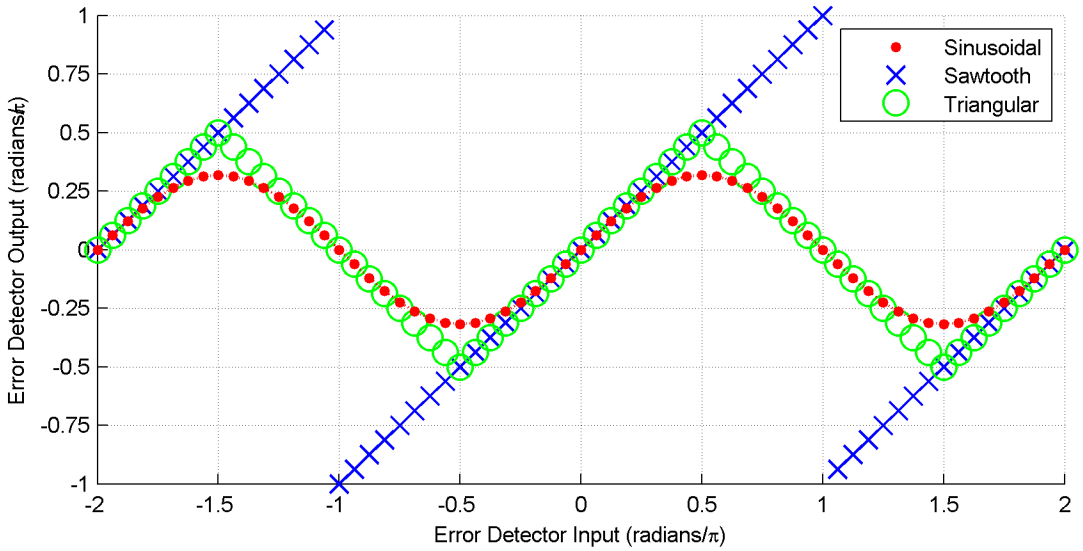

Figure 4.6. Sinusoidal and linear error detector characteristics.

solution – every complex data sample was available in both rectangular (I and Q) and polar (magnitude and phase) form. This meant that instead of using a PLL model that heavily relied on complex multiplication and a sinusoidal error detector a simpler model could be used with linear error detectors that considered only the received signal's phase.

To this end, two error detectors were considered: a sawtooth detector described by:

$$e_{saw}[n]=\begin{cases} e_{in}[n]+\pi, & -2\pi \leqslant e_{in}[n] < -\pi \\ e_{in}[n], & -\pi \leqslant e_{in}[n] \leqslant \pi \\ e_{in}[n]-\pi, & \pi < e_{in}[n] \leqslant 2\pi \end{cases} \tag{4.15}$$

and a triangular error detector described by:

$$e_{tri}[n]=\begin{cases} -\pi-e_{in}[n], & -3\pi/2 \leqslant e_{in}[n] < -\pi/2 \\ e_{in}[n], & -\pi/2 \leqslant e_{in}[n] \leqslant \pi/2 \\ \pi-e_{in}[n], & \pi/2 < e_{in}[n] \leqslant 3\pi/2 \end{cases} \tag{4.16}$$

The input-output characteristics for these two error detectors along with the sinusoidal error detector are plotted in Figure 4.6. Note that when the PLL's phase error is less than $\pi/8$, the three error detectors are nearly equivalent as the input-output relationship of the sinusoidal detector is approximately linear when the absolute value of the phase error is less than $\pi/8$. Figure 4.7 is a comparison of both the phase output and internal error signal (i.e. the signal presented to the loop filter) during the acquisition of an unmodulated input signal at a constant phase of $7\pi/8$ for a PLL with each of the three error detectors.

Observe that the sawtooth PLL acquires the signal's phase faster than both the sinusoidal and triangular detectors. This is due to the upper limit imposed on the error signal by the sinusoidal and triangular PLLs. We can see from the plot of the error signals that initially $e_{tri}[0]$ is equal to the input phase value of $7\pi/8$. Referring to the sawtooth characteristic in Figure 4.6, we can see that an initial PLL phase error will essentially pass through the detector unchanged whereas the triangular and sawtooth detectors will pass it as $\pi/8$ and $\sin(\pi/8)$, respectively. Furthermore, observe that the acquisition of the triangular PLL starts out similar to the sinusoidal PLL but the accelerates slightly. We can see that the triangular PLL's error signal is not bounded as tightly as that of the sinusoidal PLL and therefore can acquire signals slightly faster than the sinusoidal given the right conditions.

The revised model for the PLL with a linear error detector is shown in Figure 4.8. The triangular error detector was chosen due to its similarity to the sinusoidal error detector. In cases with excessive noise, the sawtooth error detector cause problems with acquisition and tracking due to its wider output range. It is still included in the PLL model for practical reasons – it simplifies the implementation of the triangular error detector by limiting the input to the latter to $\pm\pi$ instead of the possible $\pm 2\pi$. In practice, the performance of the PLL with the triangular error detector is not substantially different from that of the acquisition and tracking capabilities. The only noticeable difference is in the computational requirements. Note also in the block diagram that the initial step

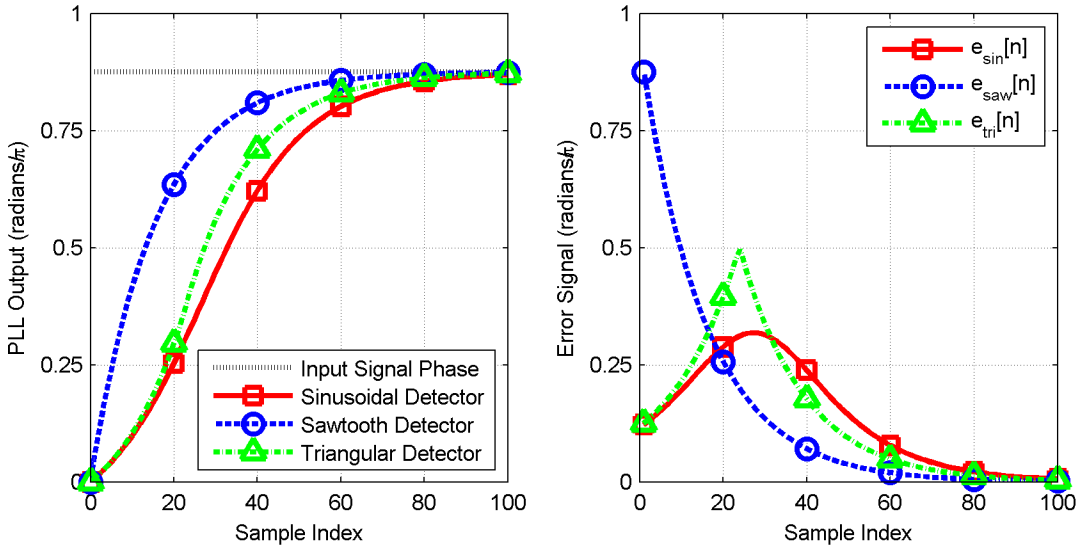

Figure 4.7. Comparison of PLL acquisition (left) and internal error signals for Costas PLL with different error detectors.

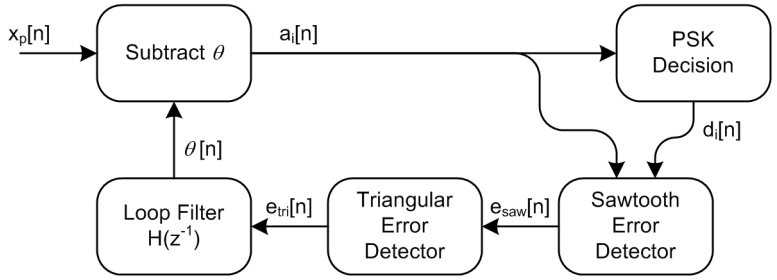

Figure 4.8. Linear Costas PLL block diagram.

in the PLL has changed to "Subtract $\hat{\theta}$" to reflect that the signal constellation rotation is now performed using polar coordinates. Likewise, the PSK decision block now produces its demodulated symbol outputs in polar form. These simple changes to the PLL can offer computational performance gains by removing all multiplication operations from the PLL, a measurement of which will be presented in Chapter 6.

4.1.4 Frequency Offset Removal

One other hurdle encountered in the initial algorithm implementation was the presence of a residual frequency offset between the transmitter and receiver due to calibration differences between the LOs of the equipment. This can cause two main problems when tracking the phase of the received signal. First, if the magnitude of the frequency offset is large enough, in general greater than tens of Hertz dependent on sampling frequency, the drift in phase of the received signal during

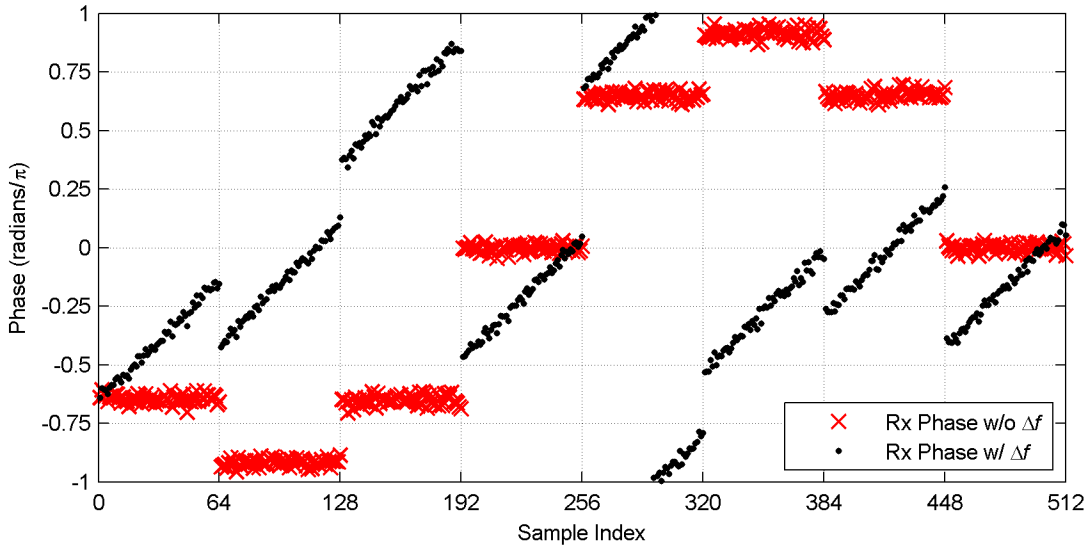

Figure 4.9. Example of phase at antenna array output with and without a 2kHz frequency offset.

one array sweep will skew the phase input to the PLL farther than can be adequately tracked by the PLL. If the frequency offset is sufficiently small it will appear simply as a constant offset in the PLL data, which can easily be dealt with. Second, the first order loop filter is in general insufficient to track a frequency shift due to the filter's poor ramp response. Initially, this was overcome through the use of a frequency reference signal between the transmitter and receiver that allowed the receiver to lock its internal LOs to a 10MHz reference generated at the transmitter. Unfortunately, this setup can not be expected in a real world situation so another solution needed to be devised.

We model the residual frequency offset on the signal as:

$$s(t)=m(t)e^{j2\pi\Delta f t}e^{j\theta_o} \tag{4.17}$$

where Δf is the frequency offset between the carrier frequency and receiver LO. The output of the *m*-th antenna is then:

$$x_m(t)=Ae^{-j\left[\theta_o+2\pi\Delta f t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)+n\pi\right]},\quad n=0,1 \tag{4.18}$$

Likewise, the corresponding phase input to the *m*-th PLL is:

$$\theta_m(t)=\theta_o+2\pi\Delta f t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi\right)+n\pi,\quad n=0,1 \tag{4.19}$$

which is similar to the original expression for the phase input to the PLL with the addition of the time-dependent frequency term $2\pi\Delta f t$. Figure 4.9 provides an example of the received phase at the output of the antenna array for two signals: one without a frequency offset and one with a

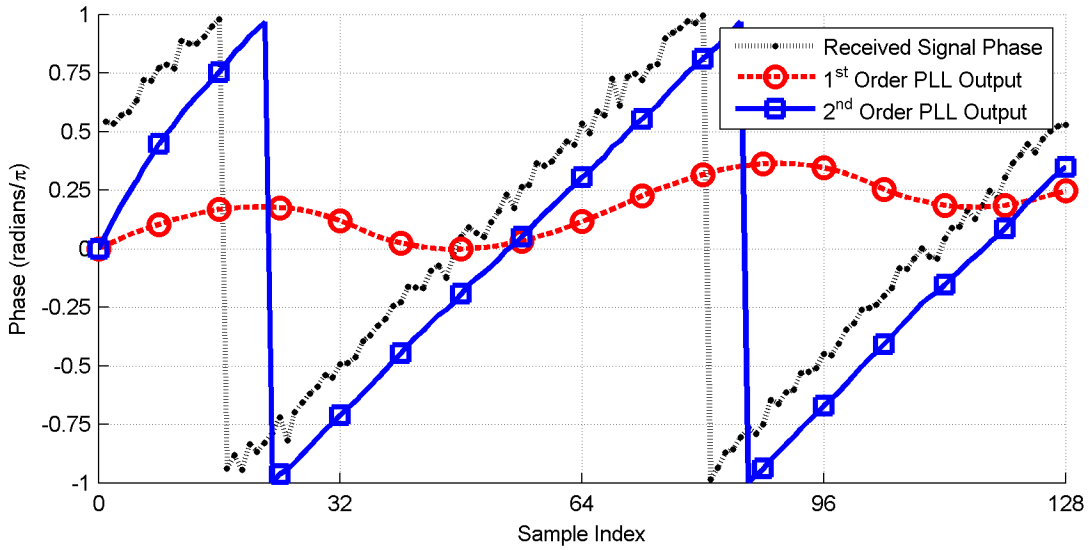

Figure 4.10. Example phase acquisition for a Costas PLL with 1st or 2nd order loop filter.

frequency offset of 500Hz, both with the same DOA. The antenna array switches every 64 samples. Observe that in the signal with the frequency offset, the discontinuity at the point of antenna switching (i.e. Every 64th sample) is equal in magnitude the the corresponding discontinuities in the signal that does not contain a frequency offset. This means that although the received phase is time-varying, the differential phase shifts imposed on the signal by the antenna switching are preserved. Therefore, if we can remove the time varying component of the received signal's phase we should still be able to measure the differential phase shifts.

In order to remove the frequency offset on a signal, the first order PLL loop filter can simply be replaced with a second order filter, which will successfully track the frequency offset. Figure 4.10 provides an example of the acquisition of an unmodulated received signal with a 2KHz frequency offset of two separate PLLs, one with a first order loop filter and the other with a second order filter. It is obvious that the first order PLL can not track the frequency offset whereas the second order PLL has no problem. Unfortunately, substituting a second order filter in the PLL does not solve the entire problem. The second order PLL will remove the frequency shift on the signal but it will also hide the constant phase information in which we are interested.

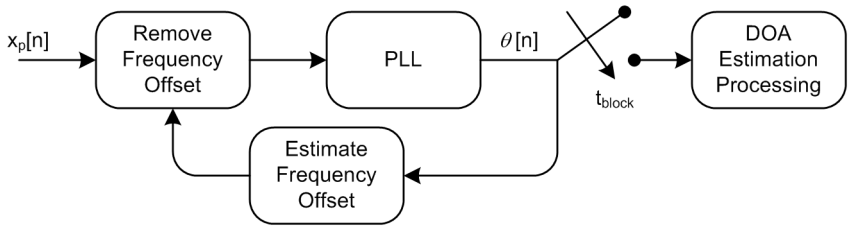

Figure 4.11. Frequency removal overview.

Therefore, a method must be devised which will allow for removal of the frequency offset without destroying the underlying phase relationship between the antennas. The selected approach is to estimate the frequency offset on the signal using the PLL output data from an entire array sweep and then use that estimate to preprocess the received data before PLL processing to accommodate the algorithm's original design, as shown in Figure 4.11. Figure 4.12 illustrates the intended result of the frequency estimation process. The line labeled "Δf Removal Vector" is a plot of the vector used to remove the frequency offset on the data given perfect knowledge of the frequency offset. Note that it is continuous across the entire array sweep – this is necessary to preserve the phase discontinuities as the antenna array is switched. When this vector is subtracted

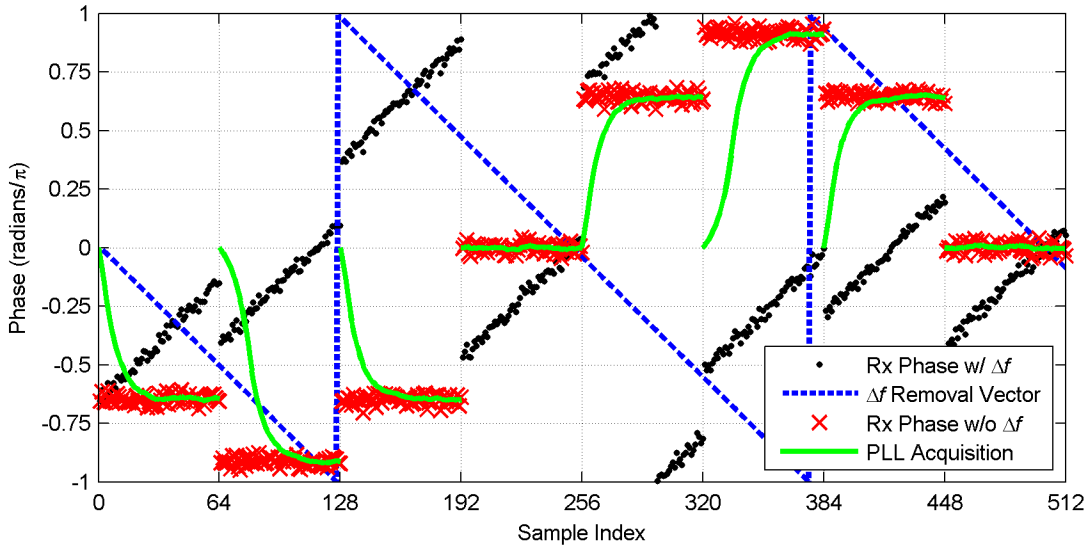

Figure 4.12. Example PLL acquisition of signal with frequency offset removed given perfect knowledge of offset.

from the received signal's phase, we are left with the signal labeled "Rx Phase w/o Δf," which contains the differential phase information in which we are interested. Furthermore, a second order PLL is not needed to track the resulting signal's phase as there is no longer a frequency component.

The method for estimating the frequency offset can be summarized as follows:

1. Initially, use a second order filter in the PLL with a wide bandwidth.
2. Record PLL output for an entire array sweep.
3. Run offset estimator:
   a) After allowing for PLL acquisition, differentiate the PLL output to determine slope of received signal phase.
   b) Filter with moving average filter to determine average slope during entire array sweep.
   c) At the end of the array sweep, update frequency offset accumulation filter, which is another moving average filter with a very narrow bandwidth.
4. Create frequency removal vector from current frequency offset estimate.
5. Repeat steps 1-4 for a predetermined number of array sweeps to allow for frequency estimator acquisition.
6. After frequency estimate stabilizes, use a first order filter in the PLL with a narrow bandwidth for more precise phase measurements. Continue evaluating steps 1-4 for fine tuning of the frequency offset estimate.

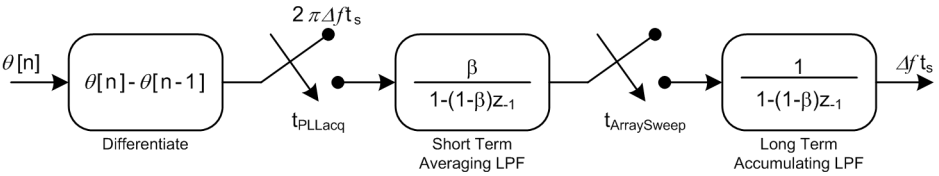

Figure 4.13. Detail frequency estimator block diagram.

Figure 4.13 shows the detailed block diagram of the frequency offset estimator. As discussed previously, frequency offset on the received signal will appear as a constant slope to the received signal's phase. In order to estimate that slope, we first differentiate the PLL's output after it has acquired the frequency offset. Since frequency is the derivative of phase, differentiating the phase will return a value proportional to the frequency offset present. Specifically, after differentiation we get:

$$\hat{\theta}_m[n]-\hat{\theta}_m[n-1]=2\pi\Delta f\left(t[n]-t[n]\right)=2\pi\Delta f\, t_s \tag{4.20}$$

where t_s is the time between samples. The first filter, labeled "Short Term Averaging LPF," is an exponential weighted moving average filter, the purpose of which is to average the output of the differentiator during the entire array sweep. Its transfer function is:

$$H(z^{-1})=\frac{\beta_1}{1-(1-\beta_1)z^{-1}} \tag{4.21}$$

which can alternatively be expressed by the difference equation:

$$y[n]=\beta_1 x[n]+(1-\beta_1)y[n-1]=\beta_1\left(x[n]-y[n-1]\right)+y[n-1] \tag{4.22}$$

where β_1 is the gain of the filter. We can see that the updated state of this filter is a function of the difference between the filter's previous state and the input value, $x[n]-y[n-1]$. The second filter, labeled "Long Term Accumulating LPF," is another averaging lowpass filter with the transfer function:

$$H(z^{-1})=\frac{\beta_2}{1-z^{-1}} \tag{4.23}$$

and corresponding impulse response:

$$y[n]=\beta_1 x[n]+y[n-1] \tag{4.24}$$

where β_2 is again the gain of this filter. This filter takes a slightly different form that the short term averaging filter because its goal is to keep track of the frequency offset during the entire time the system is running. This means that the first filter is actually estimating the "leftover" frequency

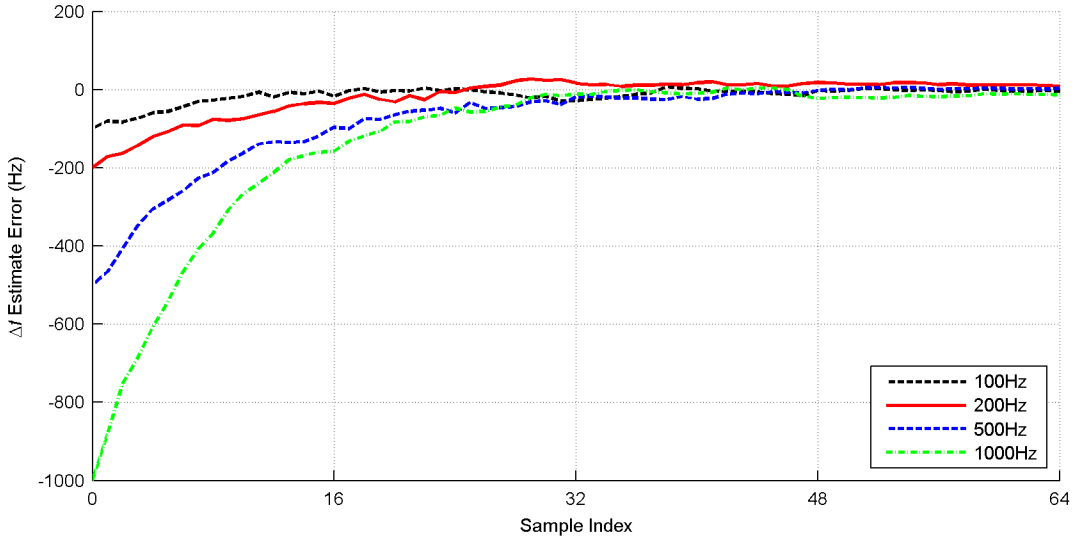

Figure 4.14. Example plot of error in frequency offset estimate per array sweep index.

component of the received data after it has been preprocessed to remove the estimated frequency offset and is providing the second filter with the difference between the frequency offset estimate and the leftover offset. During steady state operation, this will allow for constant fine-tuning of the offset estimate to account for any possible long term drift in the frequency of the received signal relative to the receiver's LOs.

Figure 4.14 plots the frequency offset estimator's output error, which is the difference between the output of the long term lowpass filter and true frequency offset, plotted per array sweep for a total of 256 array sweeps and four different frequency offsets with an SNR of 10dB. For the first 64 sweeps, the PLL is operated in second order mode to provide proper acquisition of the frequency component of the signal. After the 64th iteration, the PLL is switched to the original first order mode for more reliable phase tracking. Observe that after the 64th sample, the frequency estimator stays within ±20Hz independent of the true magnitude magnitude of the frequency offset.

4.2 16 Element Curve Fit Algorithm

4.2.1 Motivation for a New Curve Fit Algorithm

Originally, this algorithm was designed to be used with an existing 8-element uniform circular array. When it was decided to investigate the use of a 16-element uniform circular array with this DF algorithm to lay the groundwork for future research into multipath issues, we realized that the curve fit algorithm used with the 8-element array does not scale nicely to 16 elements.

Therefore, a new algorithm was designed to take advantage of certain properties of the phase relationships between antennas when there are 16 elements in the array.

The original curve fit algorithm compared the set of 2^8 possible first difference curves from the 8 given data points against a target database of 16 to 32 theoretical curves by determining the mean squared error between each given and target difference curves. When scaling this algorithm to work with a 16 element array, we first recognize that the possible number of permutations of the measured first difference data is 2^{16}, or 65536. This represents an exponential increase in the number of calculations to be performed in order to select the true first difference curve. In order to further explore the effect of increasing the number of antennas used in the system, we can estimate the number of additions and multiplications the system would need to perform to produce its MSE surface as

$$\#\ of\ Additions = (2\mathrm{M}-1)2^M N \tag{4.25}$$

$$\#\ of\ Multiplcations = (M+1)2^M N \tag{4.26}$$

$$Total\ \#\ of\ operations = 3M\ 2^M N \tag{4.27}$$

where M is the number of antennas and N is the number of curves in the target database. From equation 4.27, we can compare the total operation count required for the two array configurations to find that the curve fit algorithm with 16 antennas requires 512 times as many operations, as derived below:

$$\frac{\#\ of\ Operations,\ M=16}{\#\ of\ Operations,\ M=8} = \frac{3\cdot 16\cdot 2^{16}\cdot N}{3\cdot 8\cdot 2^8\cdot N} = \frac{2^{20}}{2^{11}} = 2^9 \tag{4.28}$$

Table 4.1. Curve fit MSE operation count.

	N=16			N=32		
M	# of Additions	# of Mults.	Time Required @ 160MHz (ms)	# of Additions	# of Mults.	Time Required @ 160MHz (ms)
8	61.4e3	36.9e3	0.5886	122.9e3	73.7e3	1.177
16	32.5e6	17.8e6	314.5	65.0e6	35.7e6	629.1

Table 4.1 is a summary of the operation count for two values of both M and N. We can see that that the total number of operations required to evaluate the algorithm with 16 antennas is three orders of magnitude higher than required with 8 antennas. Furthermore, if we assume that each operation would require one clock cycle to perform, the 16-element version would take 512 times

longer than the 8 element version, as predicted by equation 4.28. To put these values into perspective, the algorithm as implemented takes 64 samples per antenna at a sample rate of 125ksps, resulting in a dwell time on any one antenna of 0.512ms and a total time of 4.096ms to sample 8 antennas or 8.192ms to sample 16 antennas. From this estimation, the curve fit algorithm for 8 antennas with a target database of 16 curves would take approximately 0.5886ms, the computation of which can be easily distributed over one array sweep. The same situation with 16 antennas would require 314.5ms, which is considerably greater than the sweep time. This algorithm would take over 38 array sweeps, and would drop the total number of DOA estimates produced per second to 3.2. Obviously, a new approach needs to be found.

One factor that must be taken into account when comparing the 8 and 16 element versions of the original MSE-based curve fit algorithm is the amplitude of the first difference curve. Recall from equation 29 that the amplitude of the first difference curve takes the form

$$|\Delta\hat{\theta}|_{max} = \frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right) \tag{4.29}$$

We must also realize that the array radius is related to the spacing of the elements by

$$r = x\frac{\sin(\pi/2 - \pi/M)}{\sin(2\pi/M)} \tag{4.30}$$

where x is the inter-element spacing. The maximum amplitudes of the PLL output and first difference curve for both the 8 and 16 element arrays are plotted in Figure 4.15. We can see that for the same antenna spacing, the maximum amplitude of the first difference curve for the 8 and 16-element arrays is nearly identical. This means that with similar inter-element spacing, the differential phase between antenna elements will stay the same regardless of the number of antennas and that increasing the number of antennas while maintaining the same inter-element spacing merely has the effect of increasing the sampling frequency of the first difference curve. With this in mind, we can conclude that the ambiguities related to modulation will not affect the data any differently than with the 8-element array for the same antenna spacing.

4.2.2 The 16-element Curve Fit Algorithm

The first step in designing the new algorithm was to consider the second difference of the PLL output data (or, the first difference of the first difference data), given as

$$\Delta^2\hat{\theta}_m = \Delta\hat{\theta}_m - \Delta\hat{\theta}_{m-1} \tag{4.31}$$

which is exactly the same form as the expression for the first difference (refer to equation 26). By

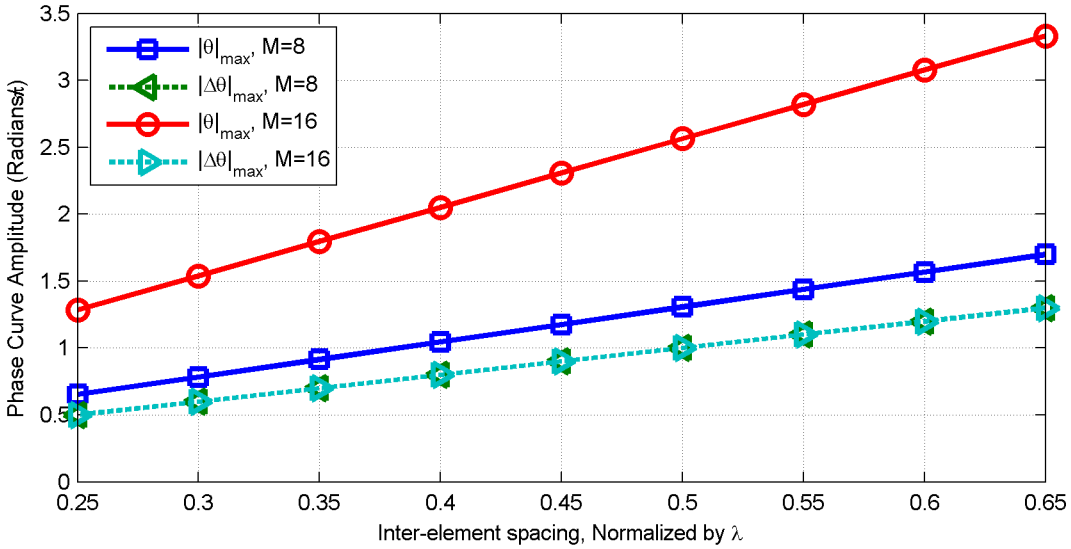

Figure 4.15. PLL output and first difference curve amplitudes relative to antenna array inter-element spacing.

substituting the expression for the first difference curve given in equation 29, we get:

$$\Delta^2\theta_m = -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\left[\sin\left(\frac{2\pi m}{M}-\frac{\pi}{M}+\phi\right)-\sin\left(\frac{2\pi(m-1)}{M}-\frac{\pi}{M}+\phi\right)\right] \tag{4.32}$$

$$\Delta^2\hat{\theta}_m = -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\left[\sin\left(\frac{2\pi m}{M}-\frac{\pi}{M}+\phi\right)-\sin\left(\frac{2\pi m}{M}-\frac{3\pi}{M}+\phi\right)\right] \tag{4.33}$$

By using the trigonometric relation:

$$\sin\alpha-\sin\beta = 2\sin\left(\frac{\alpha-\beta}{2}\right)\cos\left(\frac{\alpha+\beta}{2}\right) \tag{4.34}$$

we can further simplify equation 57 and express it as:

$$\Delta^2\hat{\theta}_m = -\frac{8\pi r}{\lambda}\sin^2\left(\frac{\pi}{M}\right)\cos\left(\frac{2\pi m}{M}-\frac{2\pi}{M}-\phi\right) \tag{4.35}$$

As expected, the second difference curve again takes the form of one full period of a sinusoid the phase of which is a function of the DOA of the received signal. From this derivation, we find that the maximum amplitude of the second difference curve is:

$$|\Delta^2\hat{\theta}|_{max} = \frac{8\pi r}{\lambda}\sin^2\left(\frac{\pi}{M}\right) \tag{4.36}$$

This relation, along with the maximum amplitude for the first difference curves, is plotted in Figure 4.16 as a function of inter-element spacing. Even though the first difference curves measured from both the 8 and 16 element arrays are nearly identical, the higher effective sampling rate of the 16 element difference curve translates into a much smaller amplitude for the second difference curve.

Moreover, the amplitude of the second difference curve with M=16 is less than 0.4π for inter-elements spacings of less than 0.5λ. This information is the basis for the new curve fit algorithm.

As stated previously, every measured element of the first difference data vector has a possible ambiguity of $\pm\pi$ due to the acquisition of the PLL in BPSK modulation. But for the 16 element array, we know that the m-th element of the first difference curve can be no more than 0.4π away from the (m-1)-th element. The new curve fit algorithm is as follows:

1. For the second through 16th elements in the first difference vector, calculate and store those values' possible replications at $\pm\pi$ and $\pm 2\pi$.
2. Starting with the first element of the measured first difference vector:
   a) Measure the difference between it and the five possible values for the second element ($\Delta\theta_2$, $\Delta\theta_2 \pm \pi$, and $\Delta\theta_2 \pm 2\pi$). These are the temporary second difference values.
   b) Take the first difference value corresponding to the smallest temporary second difference value and select it as the "true" measured first difference value.
3. Repeat step 2 for elements two through 15.

Figure 4.17 shows an example of this algorithm. The black line labeled "$\Delta\theta$, Pre Curve Fit" represents the 16-element measured first difference data. There are obvious discontinuities in this data, the first of which occurring between sample 2 and 3. The values labeled "$\Delta\theta$ Possible Values", plotted in green, are the 4 extra possible values for the first difference data, determined in step 1 of the algorithm. The lines labeled "Valid $\Delta\theta$ Range" reflect the second difference

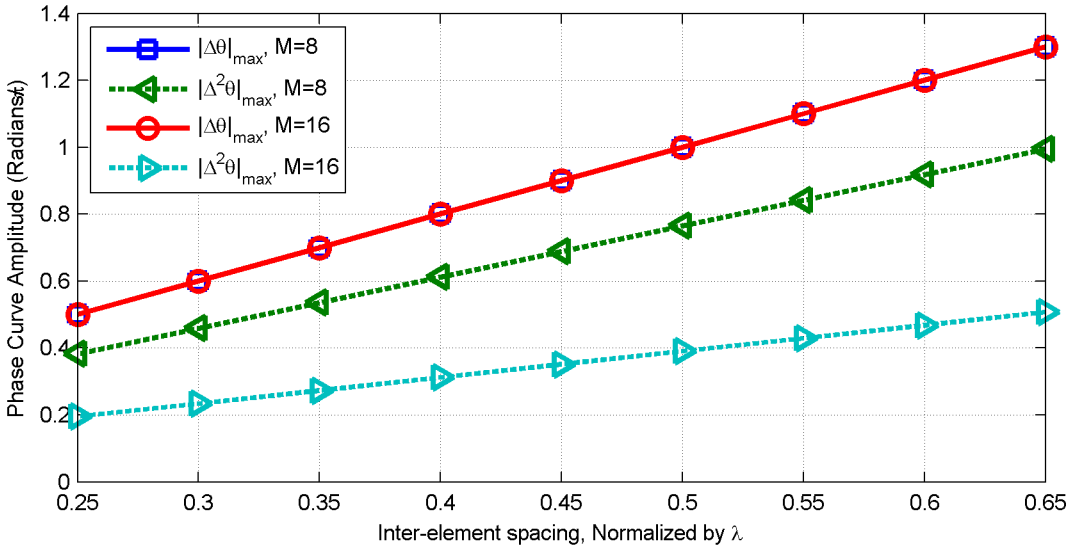

Figure 4.16. Plot of maximum amplitude of first and second difference curves relative to inter-element spacing.

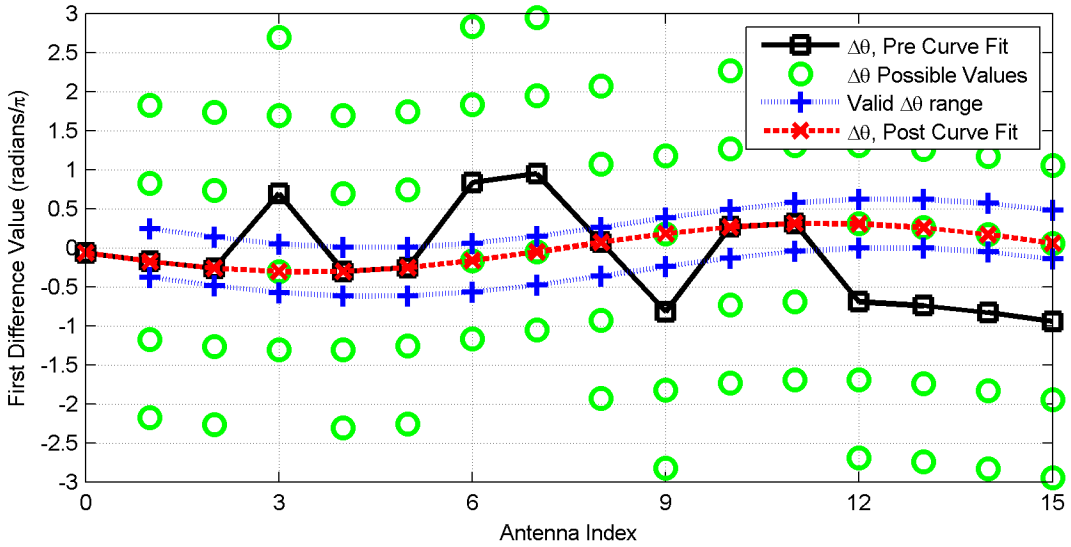

Figure 4.17. Example first difference data before and after processing by the 16-element curve fit algorithm.

tolerance of $\pm 0.4\pi$ on the first difference data. The first difference values that lie in this range will satisfy the requirements for selection in step 2b of the algorithm, namely have the smallest associated second difference. This algorithm is similar to tracing through a trellis, as in the Viterbi algorithm, measuring the minimum distance between between measured data points to determine the final difference curve.

To evaluate this algorithm, 150 additions and subtractions must be performed along with the sorting of 15 5-element arrays to produce the final first difference curve. This is obviously an incredible reduction in the computational requirements of the DF algorithm overall. This does not come without a price though – reliance on successive differentiation of the phase data will introduce an increasing amount of noise in the DOA estimates. If we assume that the PLL output values are independent identically distributed random variables, each differentiation of the data will double the variance of the result. This means that the first difference data will have a variance twice that of the PLL output values, and the second difference data will have a variance four times the PLL output. At moderate to high SNRs the resulting variance of the second difference data would still be rather small, but low SNR conditions can possibly hurt this curve fit algorithm more than the original 8-element version. We will explore the impact of this through simulation in chapter 5.

4.3 DF System Error Performance Enhancements

4.3.1 Motivation

It is an unfortunate reality that any digital communication system will make errors, and great effort goes into the design of these systems to reduce the effect of these errors on overall system performance. Consider Figure 4.18, which is a set of three scatter plots of the estimated DOA samples for the PLL algorithm with an 8 element array for a SNR values of 3dB. These scatter plots are produced by simulating the DF algorithm to produce a number of DOA estimates for various true DOA values from 0º to 360º, and the error in each of these DOA estimates is plotted against the true DOA value. For this plot, 256 estimates were produced for true DOA values that range from 0º to 357º in 3º increments. In this example, we can see that there is a significant amount of DOA estimates with little error, indicated by the band of points about zero on the y-axis. However there are numerous estimates with an error magnitude that is quite large. From these bands of erroneous DOA estimates, we find that under poor SNR conditions, the PLL algorithm with 8 elements is not only likely to make errors, but those errors will significantly deviate from the true value. In this plot, the majority of the erroneous estimates cause errors in excess of 90º or more in magnitude. This raises two questions. First, what can be done to decrease the influence of DOA estimates with significant error on the RMS error of the system, and second, is it possible to detect when the system has made such an error in order to flag that estimate as erroneous?

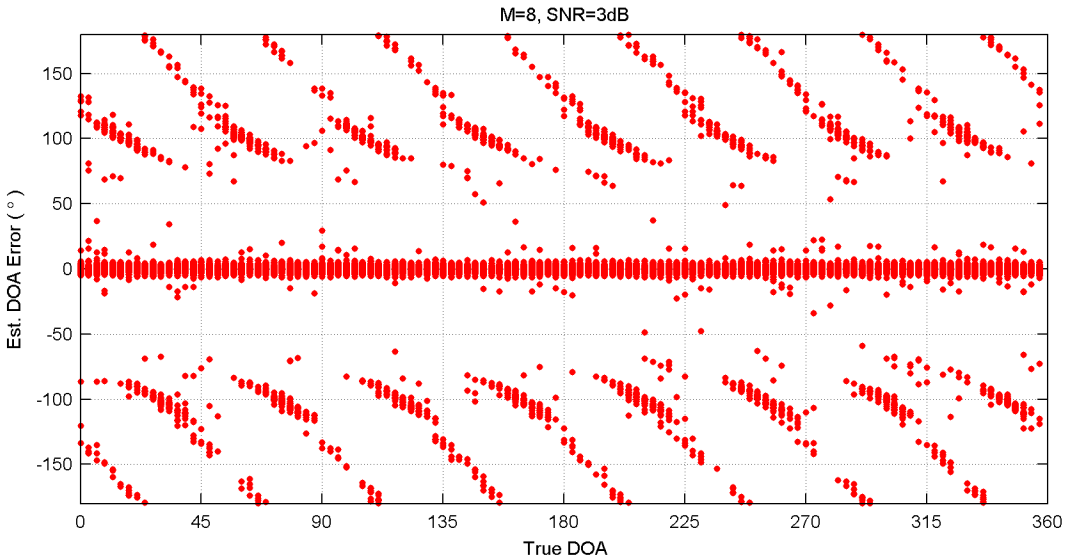

Figure 4.18. Example DOA Estimate Error Scatter plots for the 8 element PLL algorithm at 3dB SNR.

4.3.2 DOA Estimate Filtering

In order to enhance the RMS error performance of the DF system, a lowpass filter was added to the output of the DOA estimator. Any lowpass filter would suffice, but for implementation an exponential weighted moving average filter was chosen due to its simplicity and ease of implementation. As stated previously, such a filter can be described by the transfer function

$$H(z^{-1}) = \frac{\beta}{1-(1-\beta)z^{-1}} \tag{4.37}$$

From a practical perspective, this filter is implemented by evaluating:

$$\hat{\phi}_{OUT}[n+1] = \beta\,\hat{\phi}[n] + (1-\beta)\,\hat{\phi}_{OUT}[n] \tag{4.38}$$

This has the effect of reducing the influence on RMS error measurements of DOA estimates produced with a large error magnitude because that error will effectively be scaled down by the value of β chosen.

4.3.3 DOA Estimate Quality Metric

Although filtering will help reduce the effects of large estimate error, it still does not help us make reasonable decisions about which estimate was actually "good." Consequently, a metric was developed that attempts to measure the integrity of the DOA estimation process. Consider the final step in the DOA estimation process, in which the DFT of a data series is calculated. For a DOA estimate in the absence of noise, this data series will describe one full period of a sine wave, but if there is a large amount of noise it is possible that the first difference data will be a noisy, distorted version of the true data. Therefore, the DOA estimate quality metric was defined to measure the amount of noise and distortion present on the first difference data using the magnitude of a subset of the DFT bins of that data set:

$$Met_{\hat{\phi}}[n] = \frac{|F[1]|}{\sum_{m=3}^{M/2} |F[m]|} \tag{4.39}$$

where $F[1]$ is the DFT bin used to produce the DOA estimate and the denominator is the sum of the rest of positive frequency portion of the DFT. Since the data set in question is purely real, the negative frequency portion is disregarded as it is a mirror image of the positive frequencies. Furthermore, the first DFT bin is also ignored as it only corresponds to a DC offset on the first difference data, which has no bearing on the DOA estimate in any case.

This metric calculation can be viewed in two ways – on one hand it is simply a simplified

calculation of the SNR of the first difference data, and on the other hand it is also a measure of how closely the first difference data resembles a sinusoid. Figure 4.19 depicts the first difference and DFT data used in the metric calculation for two different first different curves produced by the 16 element curve fit algorithm. The DFT plots were normalized by the magnitude of their largest bin for comparison purposes. The data labeled "Good $\Delta\theta$ data" show the first difference and subsequent DFT data for a simulation that produced a DOA estimate with very low error (<1º). We can see that in the DFT of this sequence, only the bin corresponding to $k=1$ is of a significant magnitude, whereas the rest of the bins are negligibly small. On the other hand, the data labeled "Bad $\Delta\theta$ data" produced a DOA estimate with a very large error (>45º) for the same true DOA. When comparing the two data sets, it is obvious that the "bad" data barely resembles a sinusoid whereas the "good" data certainly does. The DFT of the "bad" data contains significant energy in the entire frequency spectrum relative to the DFT of the "good" data, evidence of the distortion present in the first difference data.

These two examples produced metric values of approximately 200 and 4 for the "good" and "bad" data respectively. From observing a large number of DOA estimates and their associated quality metrics, either simulated or from implementation, a metric threshold, T_{Met}, can be determined that will provide an indication of whether a DOA estimate can be considered "good" or not. If a DOA estimates metric falls below this threshold then that estimate is simply ignored and

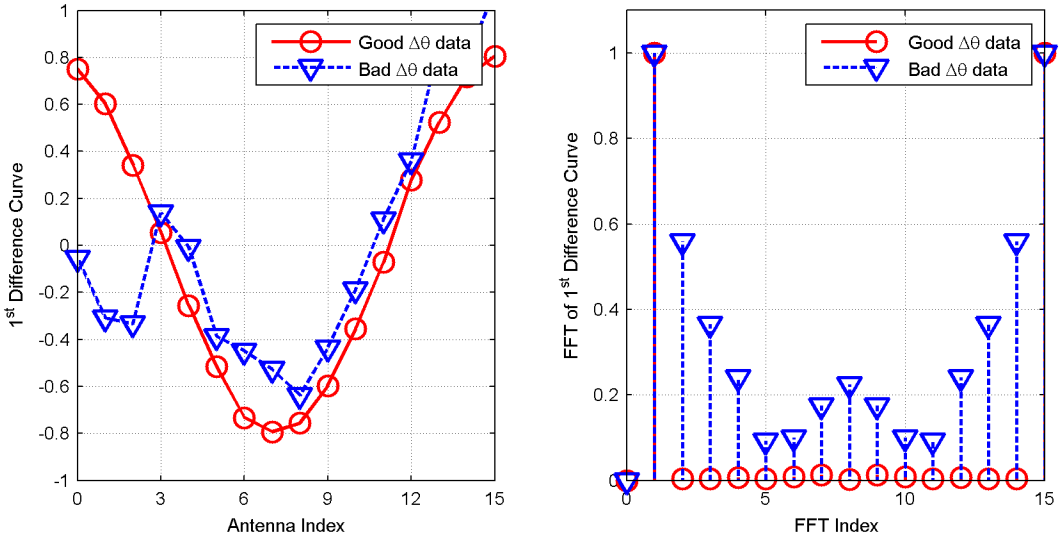

Figure 4.19. Example 1st difference curves (left) and associated frequency transforms (right) to illustrate metric calculation.

will not be considered when calculated performance statistics. We call this approach metric-assisted (MA) filtering.

4.3.4 Combined Lowpass and Metric-Assisted Filtering Approach

The final step in improving the performance of the DF system seems obvious – if lowpass or MA filtering the data provides an increase in error performance, why not combine them? To accomplish this, we first form a decision based on the value of the quality metric and a set threshold for the metrics:

$$d_{Met}[n]=\begin{cases}1, & Met_{\hat{\phi}}[n]\geq T_{Met}\\ 0, & Met_{\hat{\phi}}[n]<T_{Met}\end{cases} \tag{4.40}$$

This threshold can either be a predetermined set value or modified by the operator on the fly to produce acceptable results. With this decision, we can then decide to either evaluate the lowpass filter expression in equation (4.38) or let the previous filter value remain:

$$\hat{\phi}_{OUT}[n+1]=\begin{cases}\beta\hat{\phi}[n]+(1-\beta)\hat{\phi}_{OUT}[n], & d_{Met}[n]=1\\ \hat{\phi}_{OUT}[n], & d_{Met}[n]=0\end{cases} \tag{4.41}$$

This approach, which we will call MA-LP filtering, will provide the benefits of both the lowpass and MA filtering approaches – estimates deemed "bad" by the MA approach will be ignored while the error related to any "bad" estimates that cross the metric threshold will be reduced by the lowpass filter. These various approaches along with error trends in the 8 and 16 element algorithms will be explored in the next chapter.

Chapter 5

Simulation Analysis of the PLL DF Algorithm

Presenting the details of any DF system undoubtedly leads to one question: how well does it work? This chapter seeks to answer that question by presenting the results of a simulated DF system built on the PLL algorithm in an AWGN channel. The simulations were constructed to demonstrate the performance of the basic version of the algorithm while varying parameters such as array radius and PLL gain as well as the effect of the frequency offset estimation and estimate filtering enhancements presented in chapter 4. This chapter also investigates the impact of motion and a basic two-ray multipath channel model on the DOA estimation process.

5.1 Simulation Overview

Simulation, data collection, and statistical processing of the PLL algorithm were carried out using a combination of scripts and functions in Matlab. Except for generic operations such as the DFT and AWGN generation, custom functions were written to model the antenna array and switched output signal, Costas PLL, frequency offset estimation (if applicable), and the curve fit algorithm. A basic simulation setup is depicted in Figure 5.1. A master simulation control script was used to determine runtime system parameters as well as to coordinate calls to the DF system simulation function and to store the data from numerous repeated trials of the DF algorithm. Typically, a single script would gather estimated DOA and associated quality metric data for a single version of the algorithm for a set of true DOAs ranging from 0º to 360º and a set of SNR values. The simulated data is then stored in a series of Matlab data files categorized by parameters such as number of antenna elements, 1st order PLL gain, antenna spacing, and frequency offset magnitude

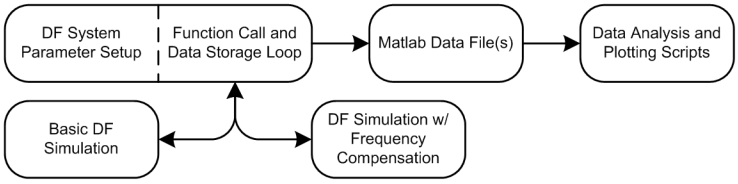

Figure 5.1. Overview of Matlab simulation setup.

for future numerical and graphical analysis.

While any number of statistics pertaining to the performance of the system can be determined from the simulated data files, the most important value is the RMS error performance, defined for a single SNR value as:

$$E_{RMS} = \sqrt{\frac{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(\hat{\phi}[i,j]-\phi[i]\right)^2}{I\,J}} \tag{5.1}$$

where $\hat{\phi}[i,j]$ is the *j*-th DOA estimate resulting from a simulation with true DOA $\phi[i]$, and *I* and *J* are the number of true and estimated DOAs, respectively, for a single SNR value. The RMS error of the system is essentially a measure of the standard deviation of the DF algorithm.

When evaluating DF systems, one must determine an acceptable value for the RMS error of the system. This value is somewhat subjective and depends on numerous factors, such as the number of RF channels, installation type, and application. For example, multi-channel military DF systems installed in a fixed location typically achieve RMS values of less than 1 degree while a mobile single-channel system is considered acceptable if it offers RMS error performance of 10°. For the purposes of analysis, we will consider RMS error values of 10° for moderate SNR values adequate. We also will consider "moderate" SNR values to be in the range of 6dB to 12dB. Most of the simulations were performed with SNR values that range from 12dB down to 0db in order to investigate the system performance in low and moderate SNR conditions.

Heretofore the performance differences between the 8-element and 16-element versions of the PLL algorithm have not been mentioned. For all simulations in this chapter, both versions of the algorithm were simulated under the exact same conditions so that while we are investigating the effect of varying parameters on the PLL algorithm in general we can simultaneously contrast the performance of the two versions.

5.2 Error Performance vs. PLL Gain

First we will consider the effect of varying the gain in the PLL's first order loop filter on both versions of the PLL algorithm. To preface this discussion, we will also look at simulation results that measure the input-output relationship of the PLL with respect to received SNR as well as the variance of the first difference data relative to the PLL output. After that we will proceed to the first real results of simulations designed to measure the RMS error performance of the PLL DF

algorithm.

5.2.1 PLL Output Statistics

In order to determine the variance of the PLL output for a signal received with AWGN, simulations were used instead of theoretical analysis. This arises from the problem of determining the distribution of the phase of the received signal. The phase of a complex Gaussian signal is essentially a function of the quotient of the real and imaginary parts of that signal. This quotient is given by a Cauchy random variable, the mean and variance of which are undefined. Further complicating this problem is the fact that the arctangent of this quotient is bounded – using a 4 quadrant arctangent function with this quotient will yield an output bounded by $\pm\pi$. But our investigation does not necessarily require a rigorous mathematical expression for this distribution, a simulation experiment will suffice. To this end, the PLL itself was simulated at each SNR and filter gain value by passing 2 parallel sequences of around 4 million complex input samples with a known expected phase in order to produce a reasonable estimate of the variance of the PLL output with an adequate allowance for PLL acquisition. Furthermore, the difference between the two parallel output sequences was taken to provide an estimate of the expected variance increase incurred by the differentiation of the PLL output data to produce the first difference curve. If we assume that the outputs of two parallel PLLs are identical independently distributed random variables, at least with respect to variance, then the difference of those two outputs will have a variance equal to twice that of the PLL output.

Figure 5.2a is a plot of the standard deviation of the PLL output against the SNR of the received signal for the first-order Costas PLL with varying filter gain. It is expected that as the PLL gain decreases, the standard deviation of the filter output will decrease due to the reduction in bandwidth and resulting power out of the filter. We can see that as the filter gain is decreased, the standard deviation of the PLL output undergoes a corresponding decrease as expected. Figure 5.2b is a plot of the mean error of the PLL output for each value of the filter gain. We can see the mean is consistently close to zero for all values of SNR and filter gain. This is to be expected, as the PLL loop filter is a lowpass filter that will effectively average the input signal. Figure 5.3a is a plot of the standard deviation of the PLL output and the subsequent first difference data. Figure 5.3b plots the variance of the first difference data normalized by the PLL output. We see that these normalized values are consistently near the value 2, which is what we would expect as the result of the addition

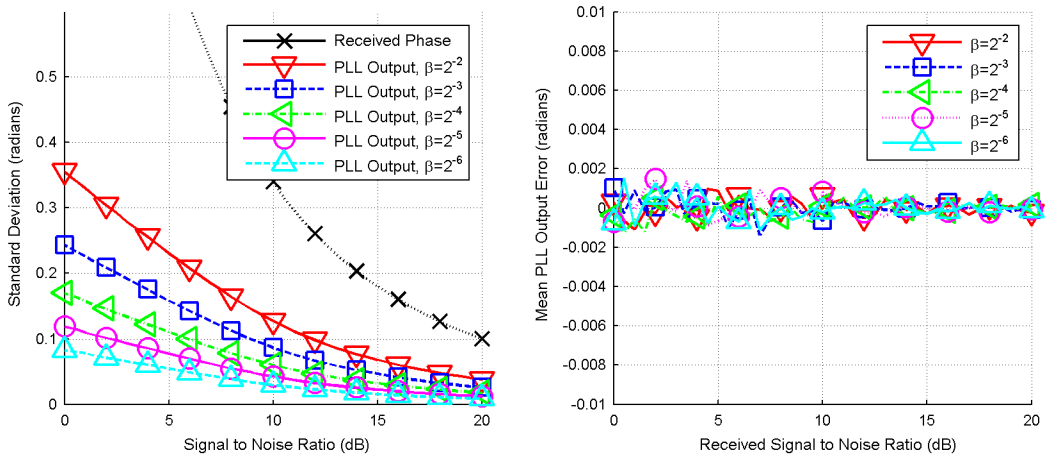

Figure 5.2. PLL Performance plots: PLL output standard deviation vs. received SNR (a, left) and mean error of the PLL output (b, right).

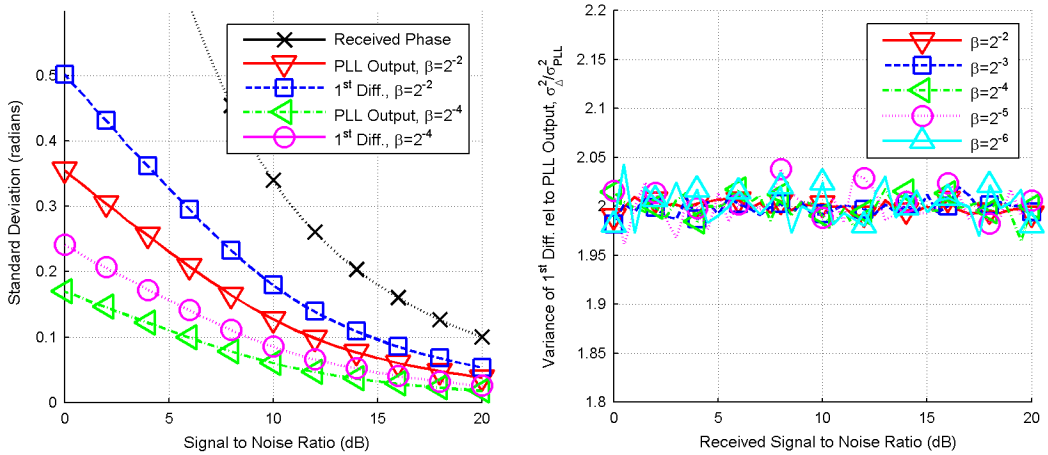

Figure 5.3. Further plots of simulated PLL performance: PLL output and 1st difference standard deviation relative vs. received SNR (a, left) and 1st difference variance relative to PLL output variance vs. SNR (b, right).

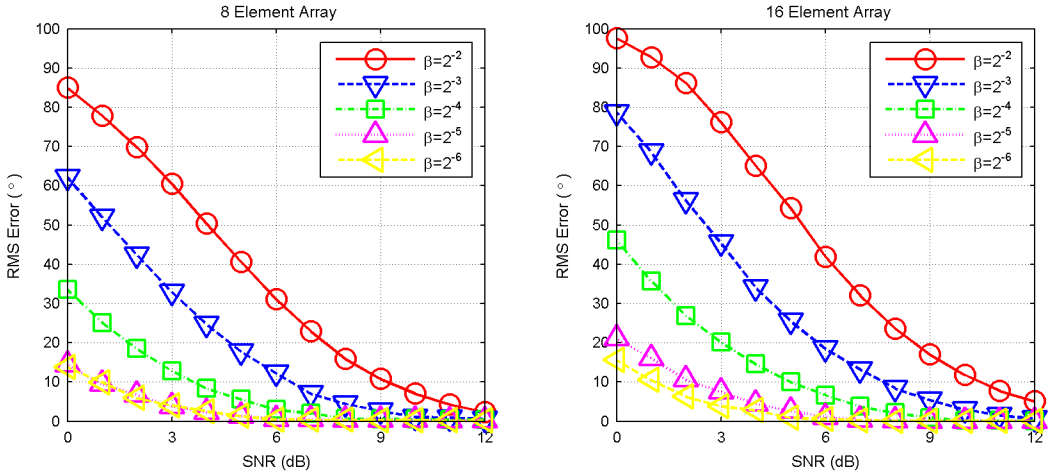

Figure 5.4. Simulated DOA estimate RMS error of the 8-element (a, left) and 16-element (b, right) versions of the PLL DF algorithm with varying PLL gain.

or subtraction of two independent identically distributed random variables will have twice the variance.

5.2.2 Effect on RMS Error Performance

Figures 5.4a and 5.4b plot the DOA estimate RMS error in both versions of the PLL DF algorithm for with varying PLL filter gain and an inter-element spacing of 0.4λ. We can see that as the gain of the filter decreases, the RMS error of both DF systems decreases. Also note that for all but the smallest values of β the 8-element algorithm outperforms the 16-element version. This is contrary to what is typically expected of DF systems, that antenna arrays with more elements in general offer higher performance because the system has more information about the received signal. For this algorithm though, the curve-fit algorithm for the 16-element version is actually a drawback because the nature of its operation is more sensitive to noise because it relies on both the first and second difference of the PLL data. As seen in the previous section the variance of the phase data will double with each successive differentiation, which means that the information used by the 16-element curve fit algorithm is subject to a higher noise level.

When considering the general RMS error requirement for a single-channel DF system laid out at the beginning of this chapter, we can conclude that the 8-element algorithm meets expectations when the PLL gain is lower than 2^{-3}. For the 16-element algorithm, a filter gain of less than 2^{-4} is acceptable. Therefore, in the selection of filter gain we must weigh the benefits of reduced estimate error variance with PLL acquisition time. Later in this chapter we will see if the addition of DOA estimate filtering enhances system performance for larger values of the PLL filter gain.

5.3 Error Performance vs. Antenna Spacing

The second parameter considered for variation was that of the inter-element spacing in the antenna array relative to wavelength. Initial implementation of the algorithm [1],[2],[3],[4] utilized a preexisting 8-element array that was originally designed for operation at 2GHz [17]. At this frequency, the elements were spaced at exactly $\lambda/2$, resulting in an array radius of 0.65λ. Unfortunately, the performance of the implemented system was considerably worse than that of the simulated system even even when the SNR was 20dB or more. Recall from Figure 4.14 that the maximum amplitude of the first difference curves grows linearly with inter-element spacing. As the spacing approaches $\lambda/2$, the first difference amplitude approaches π. During the differentiation

process, calculated first difference points are restricted to the region $-\pi \leqslant \Delta\theta_m \leqslant \pi$ and for the curve fit algorithm we only consider that calculated point and only one other possible value, either $\Delta\theta + \pi$ or $\Delta\theta - \pi$. With any amount of noise in the system, it is highly likely that the calculated first difference point and its secondary value will both be in error if the true first difference value is close or equal to π. Therefore, if neither of the calculated values for $\Delta\theta$ are correct the curve fit algorithm will obviously not produce a correct estimate of the first difference curve. Early simulations compensated for this by allowing for up to three separate values for each first difference point whereas the implemented version of the algorithm did not in order to reduce the computational load. This was compensated for by reducing the frequency used for testing. As the operating frequency is reduced, the wavelength of the signal increases and the corresponding inter-element spacing and array radius increase because the physical dimensions of the array have not changed. The final frequency used was 1.5707GHz because at that frequency the array radius is equal to $\lambda/2$. The corresponding inter-element spacing was therefore reduced to 0.38λ.

Figures 5.5a and 5.5b show the simulated RMS error performance of both versions of the PLL algorithm for inter-element spacings that range from 0.35λ to 0.65λ with a PLL gain of 2^{-3}. For the 8-element version, performance obviously degrades as the spacing increases and that for spacings greater than $\lambda/2$, the system appears to be practically useless. Moreover, a spacing of $\lambda/2$ definitely appears to be a critical point, where it begins to cross from acceptable to unacceptable performance. This represents a model of the initial implementation's operating conditions. In contrast, the 16-element version experiences modest performance decreases as the spacing grows.

These performance differences result from the difference between the two versions' curve fit algorithms and their treatment of the first difference data. As described in the preceding paragraph, the 8-element curve-fit algorithm can fail when the amplitude of the first difference curve approaches π. Once the expected amplitude exceeds π it is guaranteed that the curve fit algorithm will not produce meaningful results. This can be compensated for by expanding the number of curves considered during the curve fit algorithm. For example, we could consider the measured first difference point for each antenna and that point shifted by both $+\pi$ and $-\pi$. This would increase the number of candidate first difference curves from $2^8 = 256$ to $3^8 = 6561$, which represents a significant increase in the amount of time needed to evaluate the curve fit

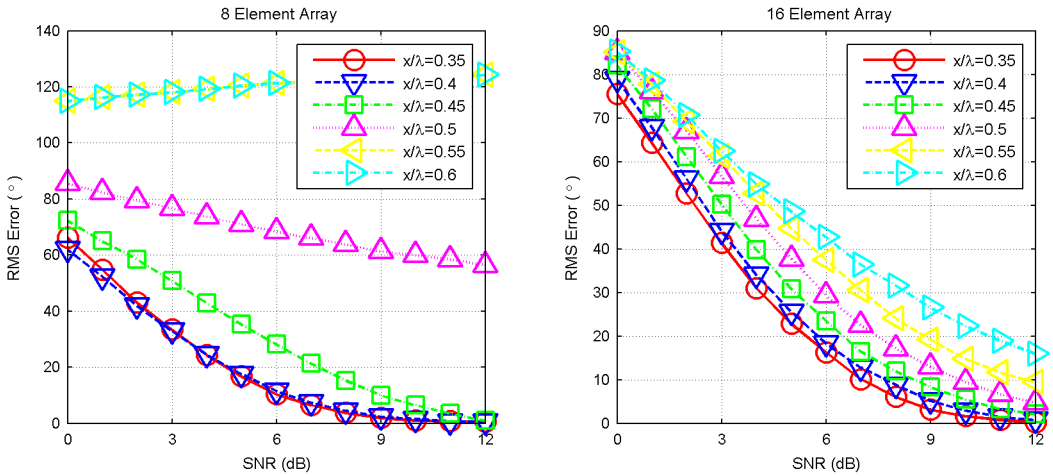

Figure 5.5. Simulated RMS error performance of the 8-element (left) and 16-element (right) versions of the PLL DF algorithm with varying inter-element spacing.

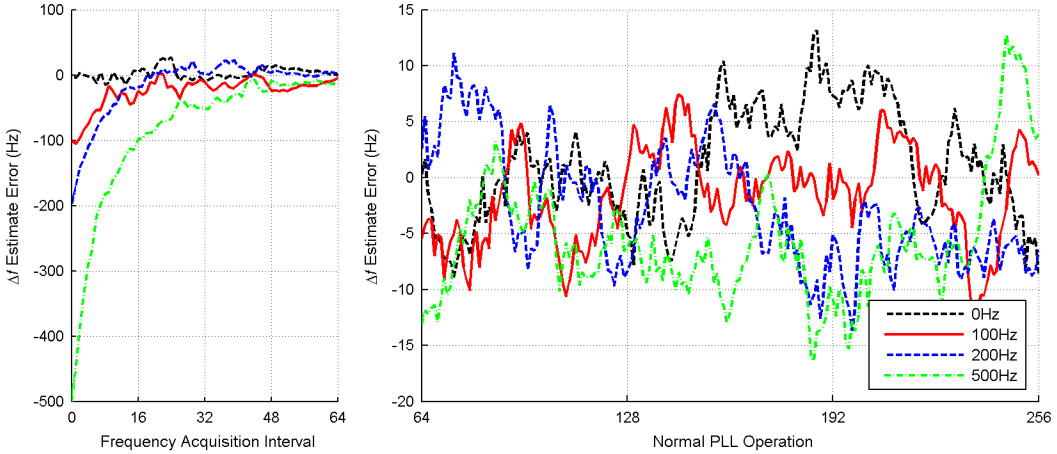

Figure 5.6. Example frequency offset estimate acquisition and tracking for various frequency offsets for a signal received with 10dB SNR.

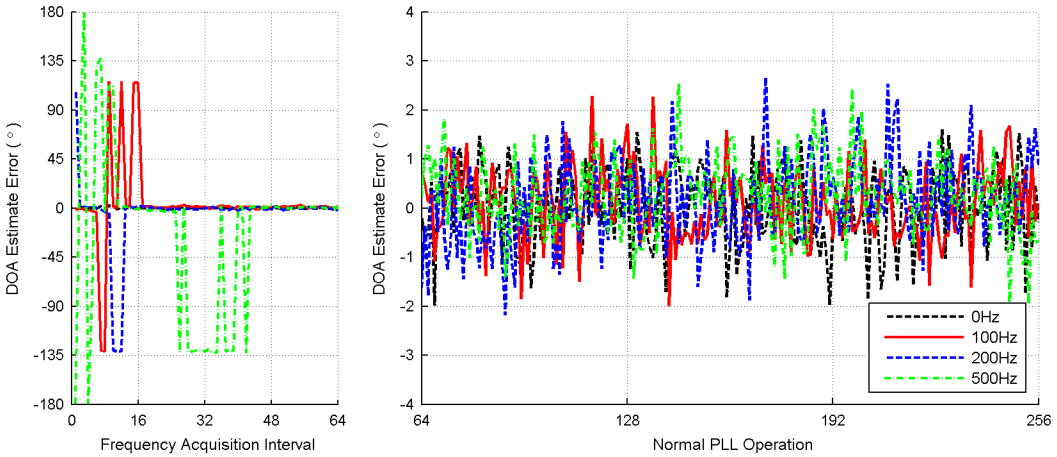

Figure 5.7. Example estimated DOA for PLL algorithm with frequency offset estimation during offset acquisition and tracking for various frequency offsets for a signal received with 10dB SNR.

algorithm. The 16-element curve fit algorithm is not necessarily subject to these same constraints. While the first difference data for this version is bounded to within $\pm\pi$, the maximum amplitude of the second difference data does not approach π until the inter-element spacing approaches 0.65π at which point this version would be unable to resolve the $\pm\pi$ ambiguities due to the BPSK modulation.

5.4 Error Performance vs. Frequency Offset

Similar to the discussion of system performance with respect to PLL filter gain, we will begin the discussion of system performance with the frequency offset estimator included by looking at the performance of the estimator itself. This will then flow into a discussion of the impact of this addition on DOA estimate error.

5.4.1 Frequency Offset Estimator Performance

Figures 5.6 and 5.7 provide examples of the frequency estimator output and DOA estimates for the DF system with frequency offset values of 0Hz, 100Hz, 200Hz, and 500Hz during the acquisition and tracking phases of the estimator. For these simulations, the estimator was allowed 64 array sweeps for acquisition of the frequency offset. This means that the PLL was operated with a second order loop filter for the first 64 array sweeps following initialization and then switched to a first order loop filter after this period.

We can see that during this acquisition phase, the error in the frequency estimator output approaches zero in each case, indicating that the alloted time period is sufficient. Of course, if it is found that the estimator fails to converge during the given acquisition duration, either the allowed time can be increased to accommodate the estimator or the bandwidth of the second order filter can be increased to accelerate acquisition. During the tracking phase of the estimator, the error in the frequency offset is kept to within ±20Hz, which is an acceptable deviation. As stated in chapter 4, if the frequency offset on the received signal is small relative to the sampling frequency, it will appear as a constant shift across all of the phase values measured by the parallel PLLs. Also, observe in Figure 5.7 that during the acquisition phase of the estimator, the DOA estimates produced by the system are often in error but once the estimator converges the estimates show very little error. This gives us some measure of confidence that the estimator will not have a large affect the RMS error of the DOA estimates.

Figures 5.8a and 5.8b plot the variance and mean of the simulated frequency offset estimator

for frequency offset values of 0Hz, 100Hz, 200Hz, and 500Hz during after the initial acquisition period of the estimator. These plots demonstrate that the error statistics of the estimator are not dependent on the actual magnitude of the frequency offset as they are reasonably consistent. Much of the variation between individual traces stems from performing the simulation with limited input sample sizes. It is expected that given a larger number of simulated received samples, the traces would be grouped tighter. Furthermore, note that the estimator in the presence of no offset reasonably matches the statistics of the estimators when a frequency offset is actually present.

5.4.2 Effect on RMS Error Performance

Figure 5.9a and 5.9b plot the RMS error of the PLL DF algorithm with frequency offset values of 100Hz, 200Hz, and 500Hz. These values were chosen in order to determine whether or not the RMS error of the DF system is dependent on the magnitude of the frequency offset. As we saw in the previous section, the mean and variance of frequency estimator is apparently not dependent on the magnitude of the offset. For these simulations, the first order filter gain is 2^{-3} and the inter-element spacing is 0.4λ. In these plots, the trace labeled "Basic Version" refers to the performance of the system without the estimator included for the same array and PLL parameters and no frequency offset. These two plots show that the addition of the frequency offset estimator incurs some performance penalties for the 8-element algorithm but not for the 16-element algorithm. Upon close inspection, we note that with the frequency estimator, the 8-element algorithm actually seems to match the performance of the 16-element algorithm. The residual offset left on the first difference data with the 8-element algorithm essentially has the same effect on the resulting DOA estimates that noise has on the first difference data with the 16-element algorithm. Furthermore, the 16-element curve fit algorithm appears to perform equally well whether the estimator is included or not. This is because the RMS error overhead resulting form the increased sensitivity of the curve fit algorithm to noise in the phase estimates from the PLL masks any increase in the RMS error due to a residual frequency offset.

5.5 Effect of DOA Estimate Lowpass and Metric-Assisted Filtering

5.5.1 DOA Estimate Error Distributions

In order to fully understand the operation of the PLL DF algorithm it is necessary to examine the distribution of the DOA estimate errors in an attempt to discern any patterns or tendencies that the DOA estimation process exhibits. These error patterns are the impetus for the

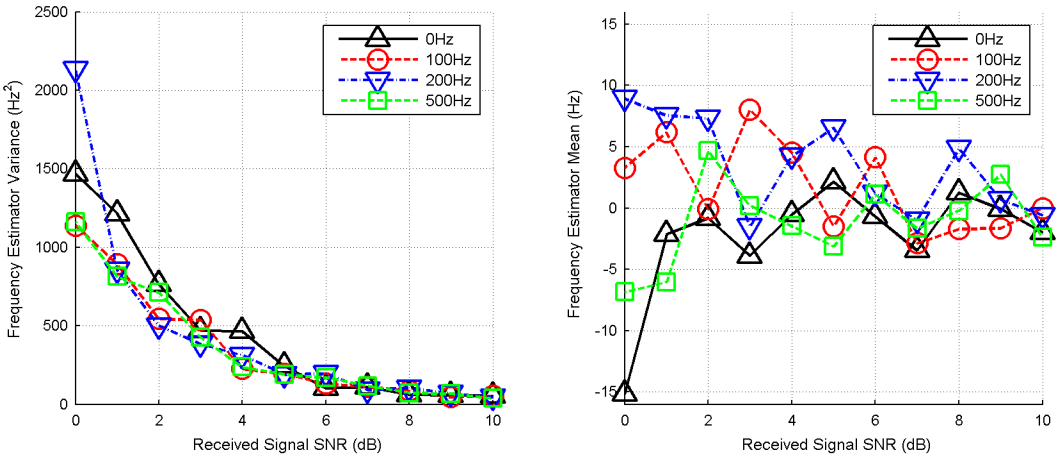

Figure 5.8. Variance (left) and mean (right) of frequency offset estimator in simulation.

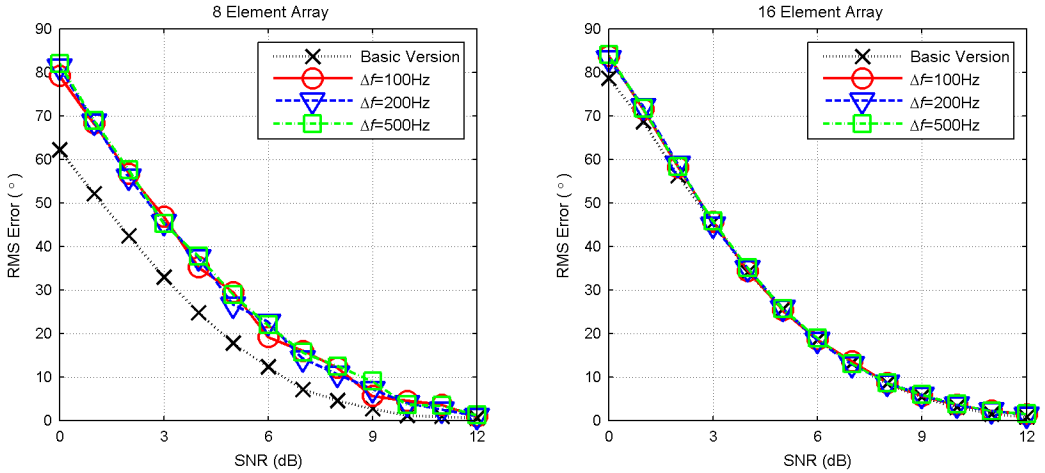

Figure 5.9. Simulated RMS error performance of the 8-element (left) and 16-element (right) versions of the PLL DF algorithm for varying frequency offsets.

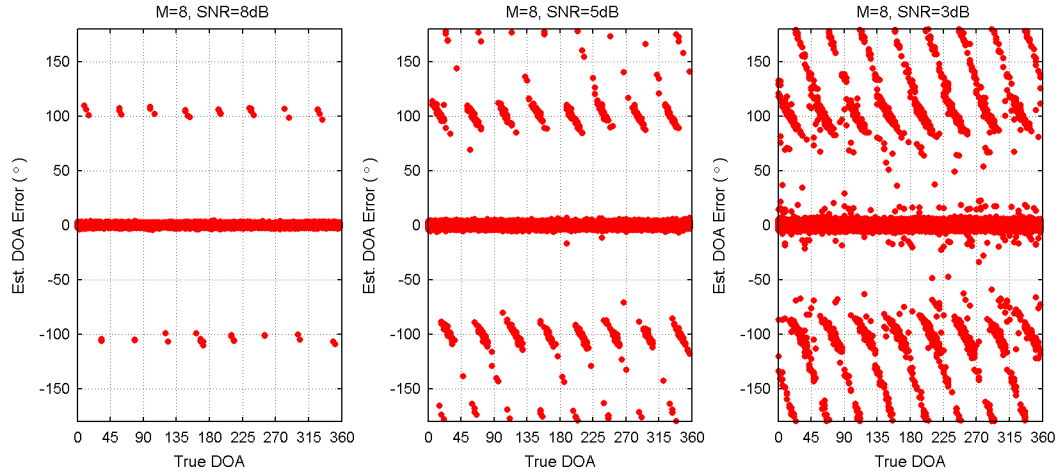

Figure 5.10. Example DOA estimate error scatter plots for the 8-element PLL DF algorithm with an SNR of 8dB (left), 5dB (center), and 3dB (right) with no frequency offset.

development of the lowpass and MA filtering approaches, and began with the initial analysis of the 8-element PLL algorithm at the start of this research. Figure 5.10 contains three scatter plots of DOA estimate error samples for simulations run in an AWGN channel with received SNR values of 8dB, 5dB, and 3dB. To produce these plots, the error in DOA estimates for numerous simulation trials is plotted against the true DOA value. This allows for the detection of any obvious bias in the DOA estimator. In the scatter plot for the 8dB SNR case, we find that the majority of the DOA estimates carry very little error and that the few erroneous estimates have an error magnitude at or near 100°. As the SNR is decreased in the remaining two cases, we notice that while the number of obviously erroneous estimates increases there is still a consistent cluster of DOA estimates with very little error. This means that when the 8-element DF algorithm produces an incorrect DOA estimate, that estimate will in general represent a large error magnitude of 85° or greater. The 8-element algorithm appears to have no "middle-ground" when it makes a mistake – the DOA estimate is either fairly close to the true value or quite far off. In other words, the performance of the DOA estimator does not degrade gracefully as SNR decreases.

Compare the 8-element scatter plots with those in Figure 5.11, which are similar scatter plots produced at the same SNR values for the 16-element DF algorithm. We see that even at 8dB SNR, the 16-element algorithm produces DOA estimates that can be up to 50° off from the true value. And in a manner similar to the 8-element algorithm, as SNR decreases we find an increasing number of DOA estimates with larger error magnitudes. This explains the overall higher RMS error curves for the 16-element algorithm displayed in Figure 5.4. Even though as SNR decreases the 8-element algorithm has a tendency to produce DOA estimates with significantly large error magnitude, those large errors will have a lesser effect on the overall RMS error as the DOA estimates with small error magnitudes are consistently small and thus help to reduce the mean squared error. Since the 16-element algorithm even at moderate to high SNR values produces DOA estimates with a larger spread about zero error it has an effective RMS error overhead relative to the 8-element algorithm that is further deteriorated by DOA estimates produced with large error magnitude as SNR decreases.

As described in chapter 4, the DOA quality metric was devised to help combat the error tendencies of the DF algorithms by using it to flag estimates as either "good" or "bad," but in order to do this we need to determine a suitable threshold for the metric values. Figures 5.12a and 5.12b

plot histograms of the quality metrics associated with the DOA estimates for a set of SNR values ranging from 12dB down to 0dB. Note that the histograms were taken on the logarithm base-10 of the actual metric value so that their range is compressed. We find that as SNR decreases for the 8-element algorithm, the general shape of the distribution does not change but the mean value decreases from over 30 (1.5 in the plotted log scale) to less than 10 (1.0 in the plotted scale). For the 16-element array, we see that in general the observed metric values fall into one of two groups – those with magnitudes greater than 10 and those with magnitudes less than 10 – and that there is an obvious separation between these two groups. As SNR decreases, the grouping with a magnitude greater than 10 shrinks. Figure 5.13a and 5.13b show the distribution of the DOA quality metric for a single SNR of 4dB with the metrics grouped according to the actual error magnitude in their associated DOA estimates. We see that for the 8-element case there is significant overlap between the two distributions whereas for the 16-element algorithm the vast majority of the estimates with small error have a large metric value. This shows that we can more easily set a metric threshold for the 16-element algorithm than the 8-element algorithm because the distribution of the metrics for the 16-element algorithm definitely fall into two easily separable categories. Setting a threshold for the 8-element algorithm is not as easy for the 8-element algorithm because the distributions show no clear separation.

The difference between these two sets of histograms can be explained by considering the operation of both versions' respective curve fit methods. The first difference curve selected by the 8-element curve fit algorithm will still appear somewhat sinusoidal because it was the data set that most closely resembled a sinusoid. Therefore, the DFT associated with the selected first difference curve will always contain some amount of energy in the bin used to produce the DOA estimate. We see this in the histograms. The overall variance in the metric values remains roughly the same as SNR is decreased while the overall noise and distortion level in the first difference curve, which affects the denominator of the metric calculation, will grow thus creating shifting the mean of the metric distribution lower. Conversely, the first difference data sets produced by the 16-element curve fit algorithm is less likely to resemble a sinusoid as noise level decreases because the distance measurements between first difference points, i.e. the second difference, will be more affected by received noise. In fact, a data set resembling a line of zero slope would be more likely picked than a sinusoid by this curve fit algorithm because the difference between successive data points would be

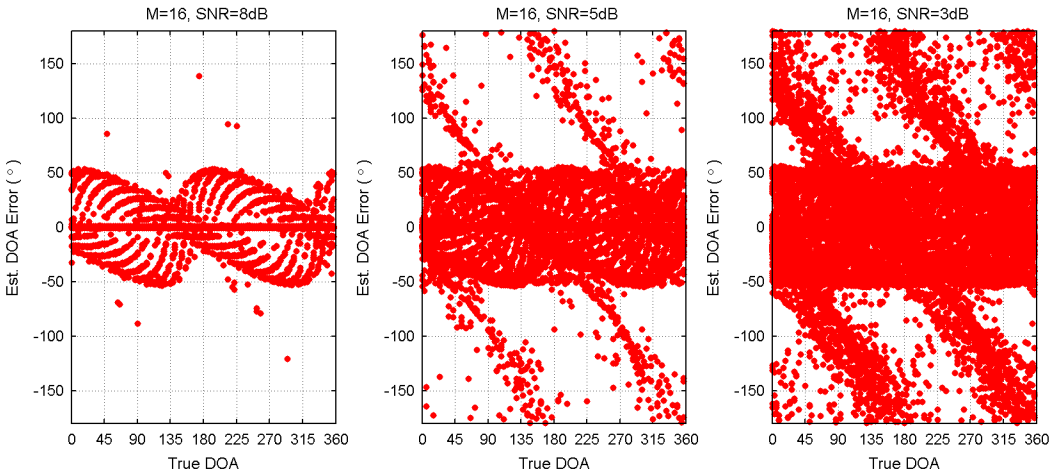

Figure 5.11. Example DOA estimate error scatter plots for the 16-element PLL DF algorithm with an SNR of 8dB (left), 5dB (center), and 3dB (right) with no frequency offset.

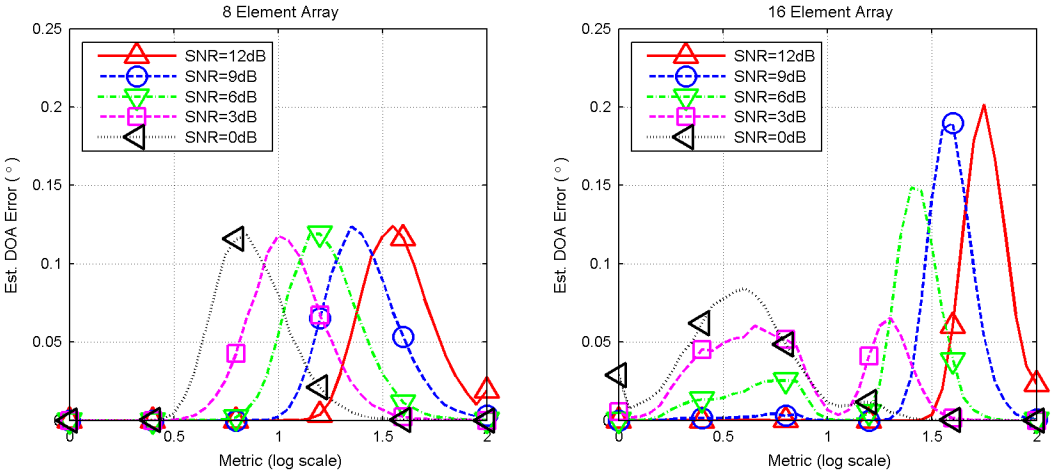

Figure 5.12. Histograms of DOA estimate quality metrics showing distribution of the metric relative to SNR for a.) (left) the 8-element and b.) (right) 16-element versions of the PLL DF algorithm.

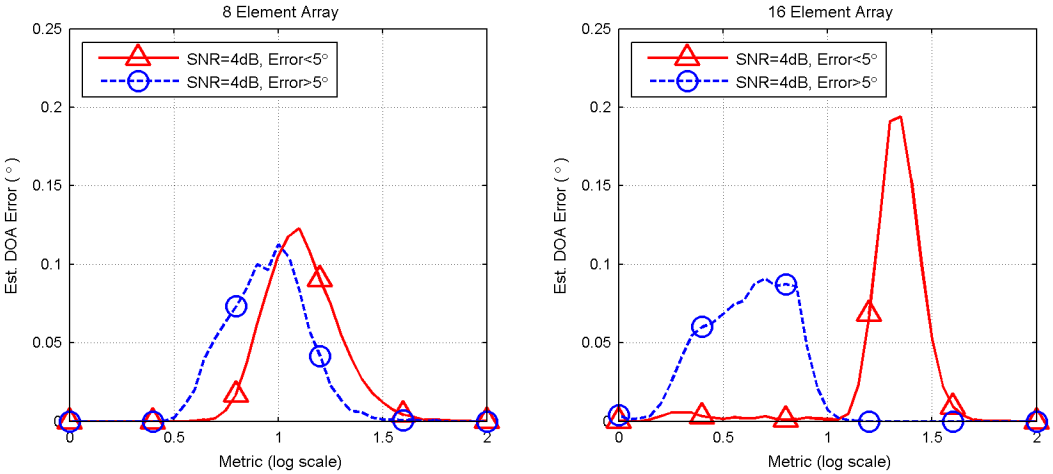

Figure 5.13. Histograms of DOA estimate quality metrics showing distribution relative to DOA estimate error magnitude for a.) (left) the 8-element and b.) (right) 16-element versions of the PLL DF algorithm.

zero.

5.5.2 Effect of Filtering Approaches on RMS Error Performance

Figures 5.14a and 5.14b plot the RMS performance of both versions of the PLL algorithm for a system with a PLL gain of 2^{-3} and an inter-element spacing of 0.4λ. For MA and MA-LP filtering, the threshold for the metric was set to 10. This threshold value was based on the metric distributions for the 16-element version of the algorithm and was used for the processing of the output for both versions. The gain of the DOA estimate lowpass filter was 2^{-2}. The line labeled "Raw" denotes the RMS error in the system without filtering (compared to Figure 5.4) while LPF, MA, and MA-LPF denote the RMS error in the system with lowpass filtering, MA filtering, and MA-LP filtering, respectively.

When comparing the effect of lowpass to MA filtering on RMS error measurements, we see that lowpass filtering has a greater effect on the 8 element algorithm than MA filtering because in many of the cases where the quality metric is above the set threshold for a DOA estimate with a large error magnitude. Figures 5.15a and 5.15b offer a clue as to why this happens. Figure 5.15a is a scatter plot of the DOA estimate error samples after metric filtering for the simulation with 3dB SNR. The points labeled "Rejected" denote DOA estimates with metrics that fell below the threshold of 10 while the points labeled "Accepted" denote samples processed either by the lowpass filter or taken as-is for RMS error calculation. We see that although a large number of DOA estimates with large error magnitudes are thrown out, quite a few remain and are included in the RMS error calculation while a number of DOA estimates with very little error are thrown out as

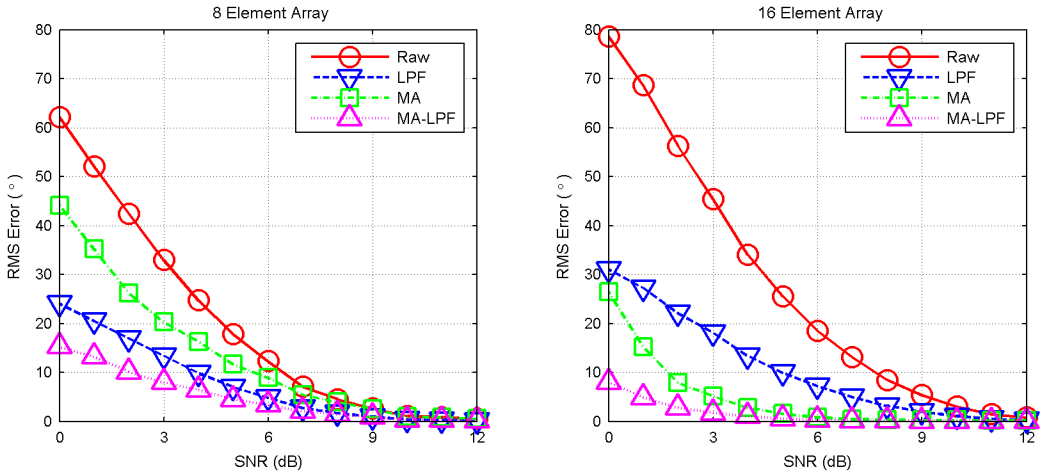

Figure 5.14. Simulated RMS error performance of the 8-element (left) and 16-element (right) versions of the PLL DF algorithm with DOA estimate filtering.

well. This will hurt the RMS error measurement because not only do estimates remain that carry a significant amount of error but also because the distribution of error estimates will be diluted by a loss of estimates clustered about zero error. Fortunately, the number of estimates remaining with little error is sufficient to maintain a low overall mean squared error and the system still achieves a lower overall RMS error than without any filtering.

Figure 5.15b is a histogram of the observed estimate error to accompany the scatter plot. The line labeled "Total" represents the error distribution before MA filtering, while the "Accepted" and "Rejected" lines represent the distributions of the accepted and rejected DOA estimates' error, respectively. This confirms that although approximately 15% of the estimates rejected had error magnitudes larger than 20°, the rest of the rejected estimates could actually have been used as "good" estimates as they fell within ±10° of the true DOA. Lowpass filtering outperforms MA filtering simply because the overall percentage of "bad" DOA estimates is small relative to the overall estimate distribution so that by reducing their error contribution the overall mean squared error is reduced. For this version, combining MA with lowpass filtering only reduces the RMS error slightly from the lowpass-only case in low SNR situations because it helps flag some of the bad estimates.

We find that that filtering approaches have a different effect on the 16-element version. MA filtering offers a larger reduction in RMS error than lowpass filtering. Figures 5.16a and 5.16b show the DOA estimate error scatter plot and distributions for the 16-element version with a SNR of 3dB (similar to Figure 5.15). In this case we find that metric does fulfill its intended goal – the overwhelming majority of rejected estimates were obviously in error while a relatively small number of "good" estimates were thrown away and a very minor amount of "bad" estimates were retained. This shows why MA filtering outperforms lowpass filtering for this version of the DF algorithm. The lowpass filter has to contend with DOA estimates that have a consistently large error variation while that variation is almost totally eliminated with the use of the quality metric. As expected, MA-LP filtering offers the greatest performance gains because the few "bad" estimates that remain are attenuated.

Overall we see that with the addition of the filtering approaches the 16-element algorithm performs better than the 8-element algorithm because it exceeds the 10° RMS error criterion down to very low SNR values by a larger margin. Recall from section 5.1.2 that for the PLL filter gain

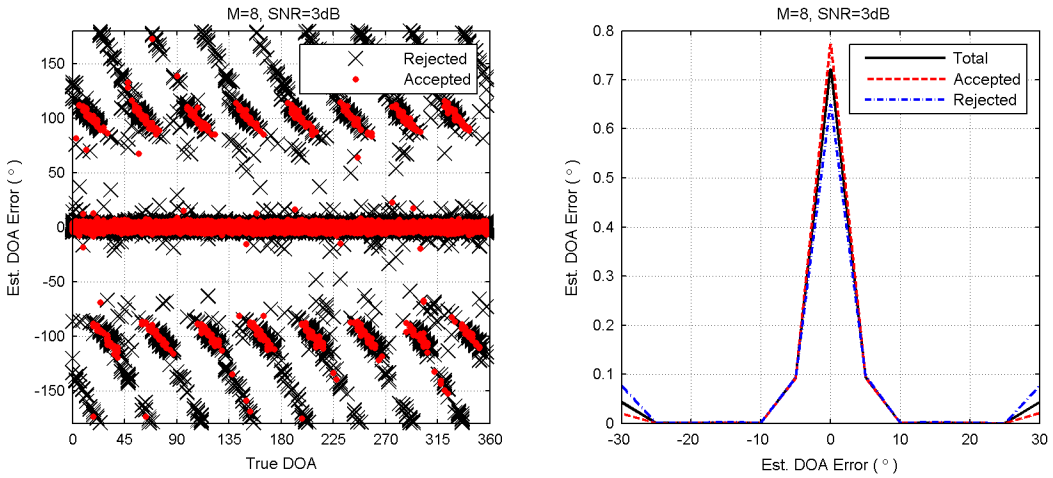

Figure 5.15. DOA estimate error scatter plot (left) and associated error histogram (right) of DOA estimate error for 8-element PLL DF algorithm after MA filtering with SNR=3dB.

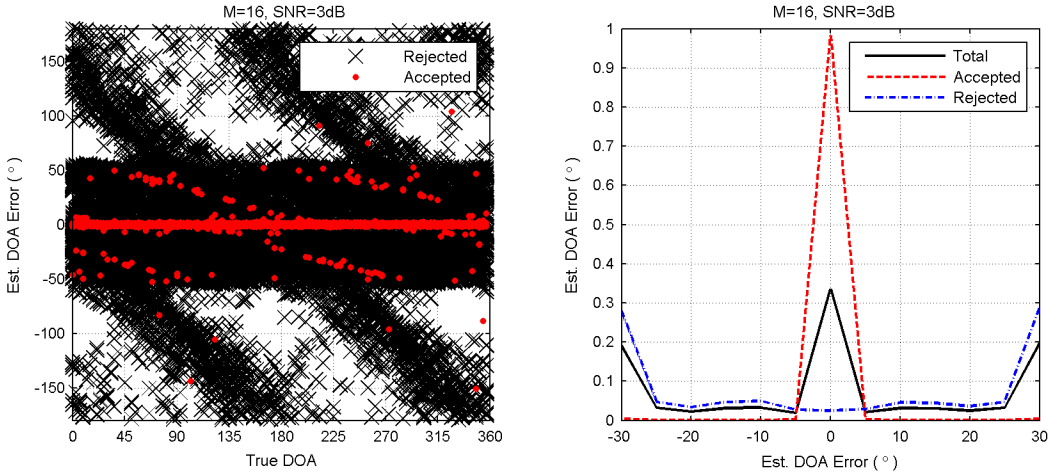

Figure 5.16. DOA estimate error scatter plot (left) and associated error histogram (right) of DOA estimate error for 16-element PLL DF algorithm after MA filtering with SNR=3dB.

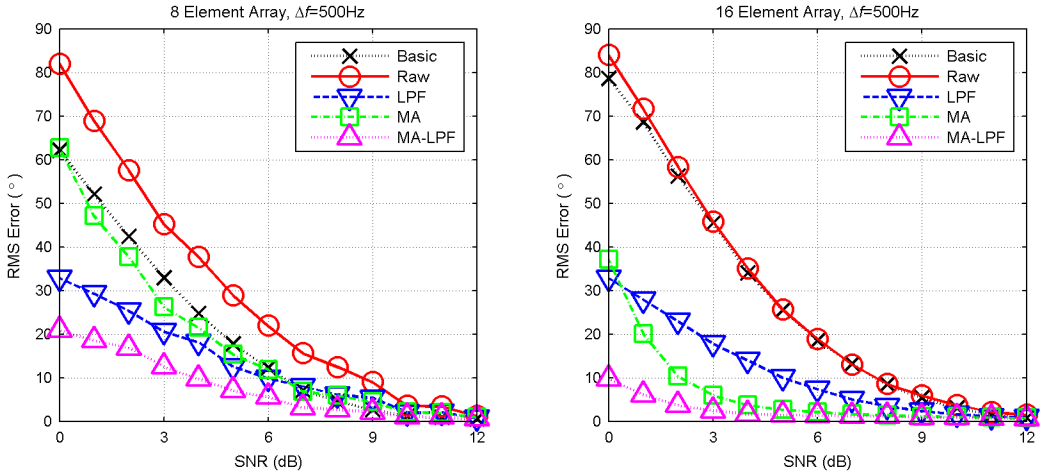

Figure 5.17. Simulated RMS error performance with DOA estimate filtering of the 8-element (left) and 16-element (right) versions of the PLL DF algorithm with the addition of frequency offset removal and a 500Hz offset.

used in this system, the 16-element version did not meet the desired RMS error value for this PLL gain. Of course, we also find that the filtering additions to the 8-element algorithm definitely enhance the 8-element version and also make it a viable DF algorithm according to the same criterion. This version almost met the RMS error target value before filtering.

Figures 5.17a and 5.17b present the simulated RMS error results for for both versions of the algorithm with DOA estimate filtering and frequency offset removal. The data labeled "Basic" refers to the unfiltered performance of the system without the presence of a frequency offset. For the 8-element version, MA filtering reduces the system's RMS error to approximately the RMS error of the basic algorithm while lowpass filtering introduces a significant RMS error reduction and MA-LP filtering performs the best of the three. Compared to the filtered RMS error data in Figure 5.13a, we see that the addition of the frequency offset estimator still incurs some RMS error penalties but the addition of estimate filtering does reduce them. As discussed earlier, the frequency offset estimator has no major effect on the performance of the 16-element version, which is further confirmed in Figure 5.17b. When compared to the results in Figure 5.13b, there are no major differences between the effects of filtering on the 16-element system with or without the frequency offset estimator.

These performance gains do not come without a price though. Figure 5.18 shows the percentage of estimates discarded in the MA and MA-LP filtering processes for the both the 8-element and 16-element versions of the algorithm without the inclusion of the frequency offset estimator. This shows that although the system experiences some dramatic performance gains, the number of useful DOA estimates output over time is reduced. For example, with 64 samples taken per antenna at a sampling rate of 125ksps, the total number of estimates produced per second by the 8-element and 16-element versions is 244 and 122, respectively. At an SNR of 3dB, the number of samples that pass through the two algorithms drops to 108 and 25 per second, respectively. As SNR approaches 0dB, we see that less than 25% of the estimates produced by the 8-element algorithm are passed while less than 10% of the 16-element algorithm's estimates are allowed. Of course, the viability of these theoretical rates is subjective and depends on application. Generally speaking, this algorithm does provide attractive performance characteristics and can be considered a feasible basis for a DF system based on its performance in AWGN. Also, remember that even if the MA filter does not consider the DOA estimates deemed "bad," those estimates are still available at the output

of the system. This allows the user to view all of the DOA estimates and use the entirety of the information produced by the estimator.

5.6 DOA Estimation on a Moving Target

One question that arose during this study of the PLL DF algorithm was that of its performance when tracking a moving target because any change in position of the target will effectively change the manifold of the signal received by the DF antenna array. According to the assumptions laid out in chapter 2, it is expected that any change in the DOA of the received signal will appear as a change in the phase of the received signal. If these changes are relatively small the PLL will have no problem in tracking them, but large changes may prevent succesful DOA estimation.

Figures 5.19a and 5.19b show scatter plots of the estimated DOA and error associated with the DOA estimates over time, respectively, for a target that is moving with respect to the DF antenna array. The SNR was 10dB and the received signal had a frequency offset of 250Hz. The DF system sampled the received signal 64 times per antenna at a sampling frequency of 125kHz. The 16 element algorithm with frequency offset removal was used. In the scatter plots, estimates labeled "Rejected" had associated quality metric values less than 10. We can see from Figure 5.19b that the accepted DOA estimates fell within ±5° of the true DOA and that there is no noticeable lag between the DOA estimate and the true DOA.

To put this example in perspective, the actual velocity of the target and distance from the array must be considered. When considering a moving target, it is obvious that the DOA is no longer static. Therefore the received signal at the output of the *m*-th antenna can be expressed as:

$$x_m(t) = m(t)e^{-j\left[\theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M} - \phi(t)\right)\right]} \tag{5.2}$$

where $\phi(t)$ is the time dependent DOA which for this example is described by:

$$\phi(t) = \frac{\pi}{4} + \frac{\pi}{3}\sin(\pi t) \tag{5.3}$$

The rate of change in DOA over time for this target can be expressed as:

$$\frac{\Delta\phi}{t} = \frac{\pi^2}{3}\cos(\pi t) \tag{5.4}$$

Assuming that the target maintains a constant distance from the antenna array, the rate of change of

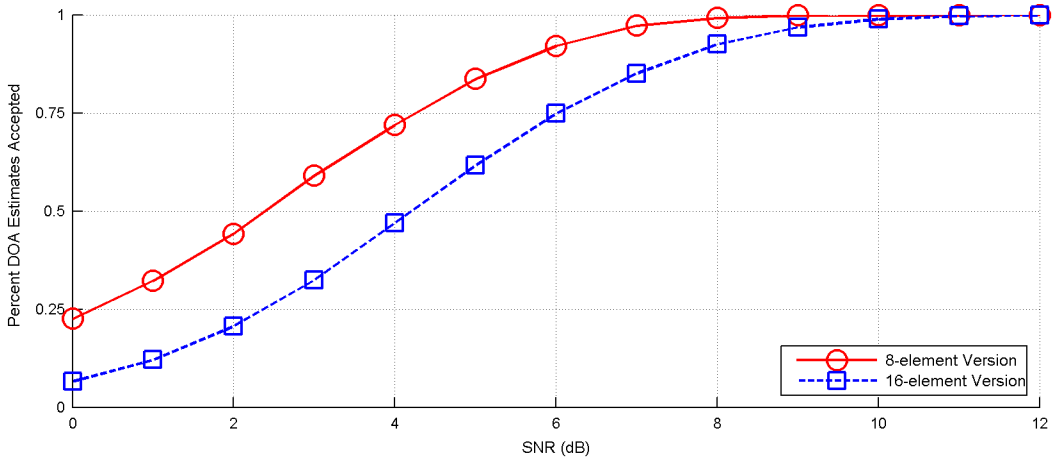

Figure 5.18. Simulated rate of acceptance of DOA estimates using the DOA estimate quality metric for both versions of the PLL DF algorithm.

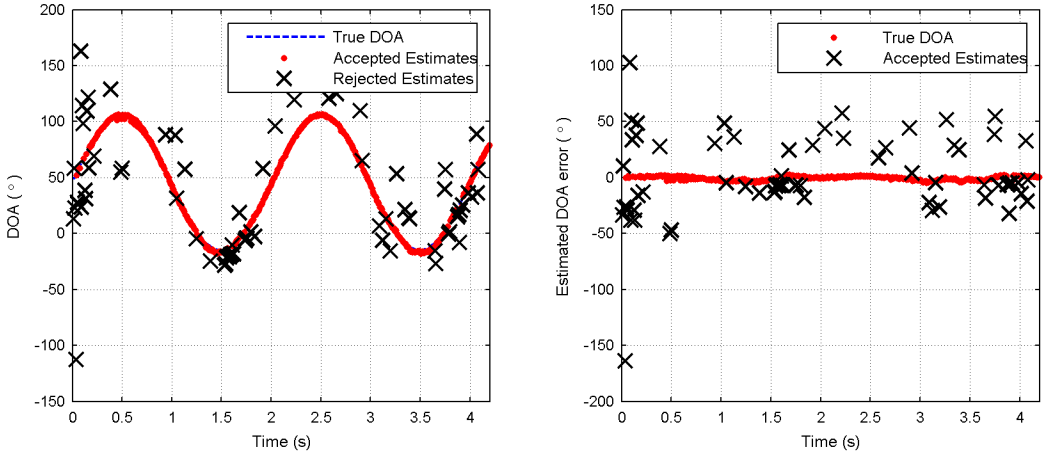

Figure 5.19. Example plots of DOA estimation on a moving target in a 6dB AWGN channel with a frequency offset of 250Hz: a.) (left) DOA estimates over time and b.) (right) DOA estimate error over time..

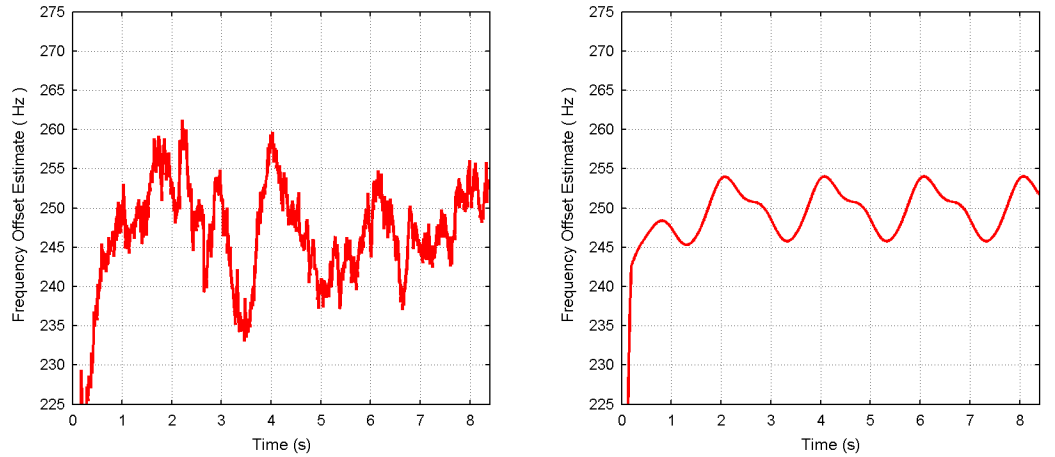

Figure 5.20. Example output of frequency offset estimator for DF on the moving target from Figure 5.19 with a.) (left) 10dB SNR and b.) (right) 100dB SNR and a frequency offset of 250Hz.

the DOA over time can be related to the velocity of the target by:

$$\frac{\Delta\phi}{t} = \frac{v}{r} \tag{5.5}$$

where v is the speed of the target in meters per second and r is the distance from the receive antenna. For this example, the maximum velocity of the target is equal to $r\pi^2/3$, which for a distance of 1 kilometer results in a target velocity of 3290 meters per second, just over 9.5 times the speed of sound. From this result, it is obvious that this example is a fairly unrealistic simulation in terms of rate of change of DOA, yet the algorithm is still able to track the DOA.

Figure 5.20a shows the output of the frequency offset estimator from the example in Figure 5.19. The frequency offset on the signal was set to a constant 250Hz. We see that after the initial half-second acquisition period the offset estimator tracks the frequency offset reasonably well but has noticeable deviations from the true value. Figure 5.20b is the output of the frequency offset estimator for the same system and target setup with an SNR of 100dB, which effectively eliminates noise on the received signal. We can see from this Figure that the output of the estimator oscillates about the true value of 250Hz and that the oscillations are kept with ±5Hz of that value. This oscillation is due to the offset estimator tracking the change in DOA over time as part of the frequency offset on the signal, but the magnitude of that offset is fairly insignificant.

As stated previously, small frequency offsets have very little effect on the DF estimation process and Figure 5.6 showed that with an SNR of 10dB the estimator will vary ±10Hz from the true value of the signal anyway. This example shows that when considering the change in DOA due to a moving target we should not expect the DOA estimation process to be negatively affected because the apparent frequency offset induced on the signal due to the rate of change of the DOA is negligible. The frequency offset estimator does not hamper the ability of the DOA estimator to track the phase of the received signal because although it removes the short term frequency shifts related to the target's motion it does not stop the PLL from tracking the long term change in the manifold of the received signal.

5.7 DOA Estimation with a Two-Ray Multipath Model

Previous work on the PLL algorithm [1],[2],[3],[4] has shown that although superior in performance to other single-channel DF algorithms, specifically the Watson-Watt and pseudo-doppler methods, in an AWGN channel the PLL algorithm's performance in fading channels leaves

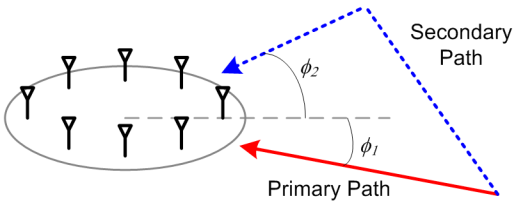

Figure 5.21. Two-ray multipath model.

much to be desired. Here we will investigate the problems that arise when evaluating the PLL DF algorithm in a multipath channel that follows a simple two path model shown in Figure 5.21. While this model is obviously quite simple and does not take fading into account, it does help illustrate some aspects of the problem of determining the DOA of a received signal in the presence of multipath.

Figure 5.22a is a scatter plot of the estimated DOA for the two-ray multipath model for a primary DOA of 0° and secondary DOA of 80° as the ratio of the power of the primary signal to the power of the secondary signal varies from 20dB down to -20dB. Figure 5.22b plots the percentage of DOA estimates accepted using the quality metric as the power ratio of the two paths changes. This plot shows that if one path is 10dB stronger than the other, the DF algorithm will reliably pick the angle of that path as the estimated DOA. If the one path's power is within 1dB of the other path, the DOA estimated by the algorithm indicates that the signal appears as if it is coming from directly between the two paths.

This is to be expected from the PLL. When one signal is significantly stronger than the other, the PLL will track the strongest signal. If one signal is the line-of-sight signal, it should be significantly stronger than the secondary signal because the other signal will have propagated over a longer distance. However, if the secondary signal is generated by a reflection that is in close proximity to the antenna array the two signals may be close in amplitude. The remainder of the analysis work centered on any possible approaches that can be taken to improve the algorithm's performance in this case.

5.7.1 Derivation of Signal Representation for Two Signals with Equal Strength

In order to determine whether anything can be done to help mitigate multipath issues arising from the two-ray model, a representation of the signal received by the array must be determined. First, the two separate paths are modeled as two distinct waveforms impinging upon the array:

$$r_{1,m}(t)=A_1\cos\left[\omega_c t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{Na}-\phi_1\right)+\theta_1\right] \tag{5.6}$$

$$r_{2,m}(t)=A_2\cos\left[\omega_c t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{Na}-\phi_2\right)+\theta_2\right] \tag{5.7}$$

where ϕ_1 and ϕ_2 are the DOAs and θ_1 and θ_2 are constant phase offsets of the primary and secondary paths, respectively. The signal received at the antenna array is simply modeled as the sum of the two separate paths:

$$R_m(t)=r_{1,m}(t)+r_{2,m}(t)=\Psi_m\cos\left(\omega_c t+\Theta_m\right) \tag{5.8}$$

where Ψ_m and Θ_m are the resulting amplitude and phase of the signal received at the *m*-th antenna. It is these resultant amplitude and phase expressions that are of interest. To solve for them the trigonometric identity:

$$\cos(x)+\cos(y)=2\cos\left(\frac{x+y}{2}\right)\cos\left(\frac{x-y}{2}\right) \tag{5.9}$$

is used where:

$$x=\omega_c t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi_1\right)+\theta_1 \tag{5.10}$$

$$y=\omega_c t+\frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\phi_2\right)+\theta_2 \tag{5.11}$$

The sum of these two terms can be expressed as:

$$(x+y)=2\omega_c t+\theta_1+\theta_2+\frac{2\pi r}{\lambda}\left[\cos\left(\frac{2\pi m}{M}-\phi_1\right)+\cos\left(\frac{2\pi m}{M}-\phi_2\right)\right] \tag{5.12}$$

$$(x+y)=2\omega_c t+\theta_1+\theta_2+\frac{4\pi r}{\lambda}\cos\left(\frac{2\pi m}{M}-\frac{\phi_1+\phi_2}{2}\right)\cos\left(\frac{\phi_2-\phi_1}{2}\right) \tag{5.13}$$

while their difference can be reduced to:

$$(x-y)=\theta_1-\theta_2+\frac{2\pi r}{\lambda}\left[\cos\left(\frac{2\pi m}{M}-\phi_1\right)-\cos\left(\frac{2\pi m}{M}-\phi_2\right)\right] \tag{5.14}$$

$$(x-y)=\theta_1-\theta_2-\frac{4\pi r}{\lambda}\sin\left(\frac{2\pi m}{M}-\frac{\phi_1+\phi_2}{2}\right)\sin\left(\frac{\phi_2-\phi_1}{2}\right) \tag{5.15}$$

The amplitude, Ψ_m, of the signal received at the *m*-th antenna, R_m, is then found to be:

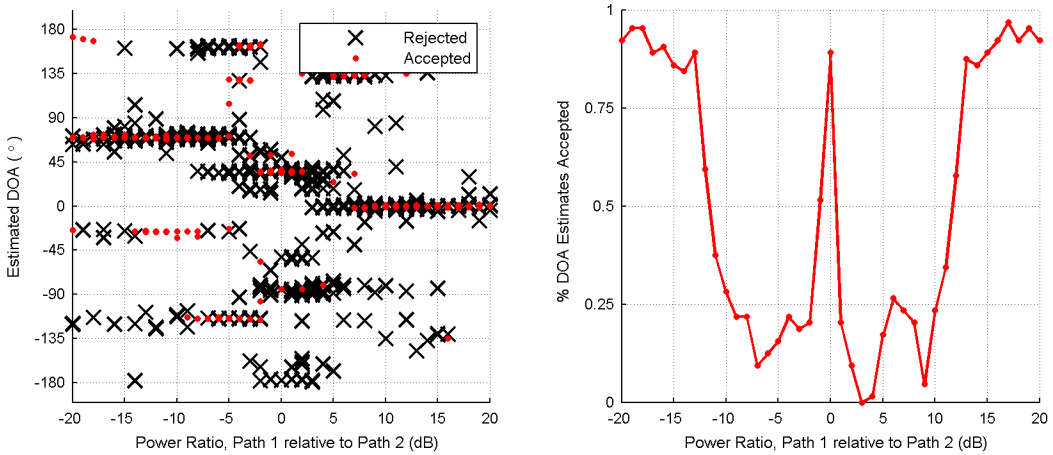

Figure 5.22. PLL DF algorithm in a two-ray multipath environment showing a.) (left) scatter plot of DOA estimates and b.) (right) percentage of accepted DOA estimates vs. power ratio of the two unmodulated signals.

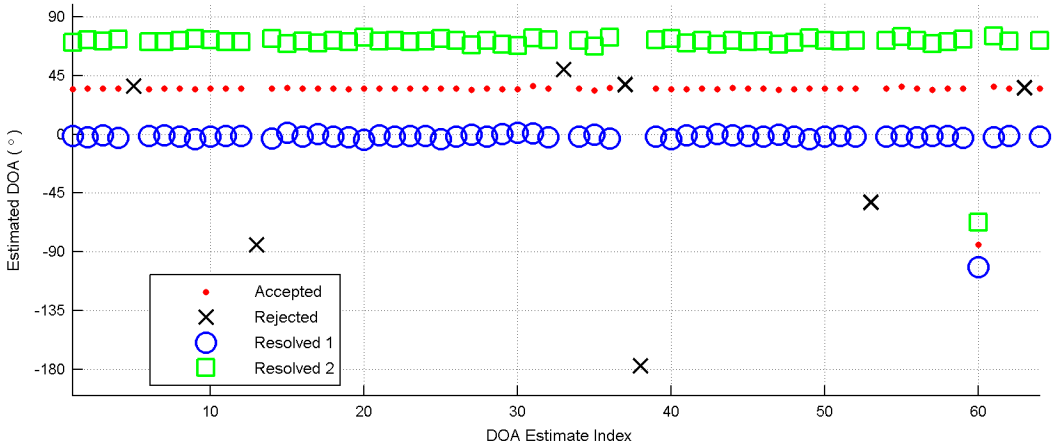

Figure 5.23. Example plot of angle resolution for two-ray model with equal signal powers with true DOAs of 0° and 80°.

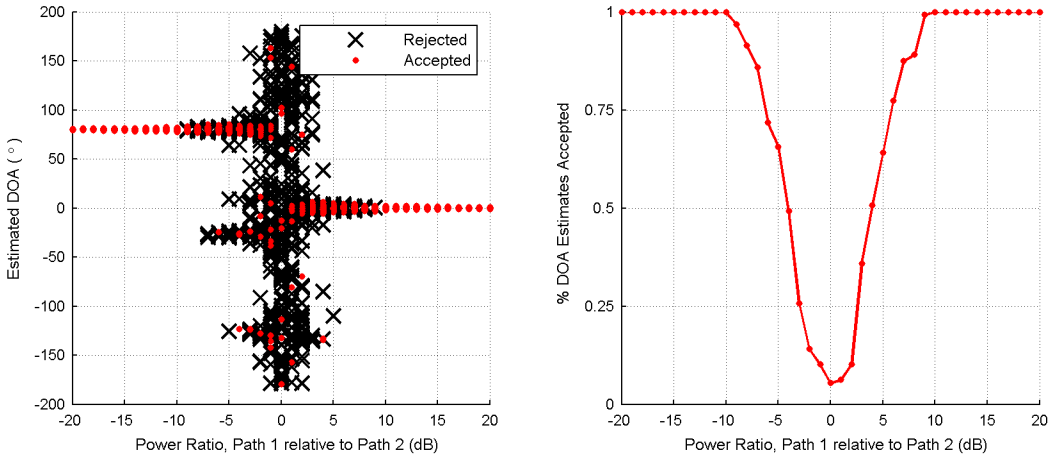

Figure 5.24. PLL DF algorithm in a two-ray multipath environment showing a.) (left) scatter plot of DOA estimates and b.) (right) percentage of accepted DOA estimates vs. power ratio of the two modulated signals.

$$\Psi_m = 2\cos\left(\frac{x-y}{2}\right) = 2\cos\left[\left(\frac{\theta_1-\theta_2}{2}\right) - \frac{2\pi r}{\lambda}\sin\left(\frac{2\pi m}{M} - \frac{\phi_1+\phi_2}{2}\right)\sin\left(\frac{\phi_2-\phi_1}{2}\right)\right] \tag{5.16}$$

from the $(x-y)$ term because it is not a function carrier frequency ω_c. The phase, Θ_m, can be determined from the portions of the $(x+y)$ term that are not a function of the carrier frequency:

$$\Theta_m = \frac{x+y}{2} - \omega_c t = \left(\frac{\theta_1+\theta_2}{2}\right) + \frac{2\pi r}{\lambda}\cos\left(\frac{\phi_2-\phi_1}{2}\right)\cos\left(\frac{2\pi m}{M} - \frac{\phi_1+\phi_2}{2}\right) \tag{5.17}$$

Therefore, in the absence of noise the estimate of the received phase at the output of the *m*-th PLL can be expressed as:

$$\hat{\theta}_m = \theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{\phi_2-\phi_1}{2}\right)\cos\left(\frac{2\pi m}{M} - \frac{\phi_1+\phi_2}{2}\right) \tag{5.18}$$

This differs from the expected phase as given in chapter 3 as:

$$\hat{\theta}_m = \theta_o + \frac{2\pi r}{\lambda}\cos\left(\frac{2\pi m}{M} - \phi\right) \tag{5.19}$$

in two main ways. First, the amplitude of the sinusoid traced by the output of the collection of PLLs is modified by a function of the angular separation of the two paths, $\cos[(\phi_2-\phi_1)/2]$, and second, the phase of the sinusoid is also a function of the angular separation of the two paths. This means that the signal will appear to come from between the two paths when the amplitude of the two paths is equal. If follows that the first difference curve is given by:

$$\Delta\hat{\theta}_i = -\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\cos\left(\frac{\phi_2-\phi_1}{2}\right)\cos\left(\frac{2\pi m}{M} - \frac{\pi}{M} - \frac{\phi_1+\phi_2}{2}\right) \tag{5.20}$$

and the final DOA estimate produced by the DFT after the curve-fit algorithm is:

$$\hat{\phi} = \frac{\phi_1+\phi_2}{2} \tag{5.21}$$

This shows that according to the model for the received signal, the first difference data generated from the output of the parallel PLLs will cause the DOA estimation block to estimate the DOA as the average of the two paths' DOAs and since the data is still sinusoidal the associated quality metric should still be able to determine whether the curve fit process was succesful. However, the first difference curve does contain extra information that can possibly be used to resolve the angles of the two paths.

5.7.2 Resolution of Two Directions of Arrival for Equal Amplitude Case

From equation 5.20, we see that the maximum amplitude of the first difference curve is

given by:

$$|\Delta\hat{\theta}_m| = \frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)\cos\left(\frac{\phi_2-\phi_1}{2}\right) \tag{5.22}$$

which is now a function of the DOAs of the two separate paths. By defining a value $\hat{A}$:

$$\hat{A} = \cos\left(\frac{\phi_2-\phi_1}{2}\right) = \frac{max_m\{\Delta\hat{\theta}_m\} - min_m\{\Delta\hat{\theta}_m\}}{2\frac{4\pi r}{\lambda}\sin\left(\frac{\pi}{M}\right)} \tag{5.23}$$

as a scaled measurement of the amplitude of the first difference curve selected by the curve fit algorithm, the two paths' DOAs can be

$$\phi_1 = \hat{\phi} + \cos^{-1}\hat{A} \tag{5.24}$$

$$\phi_2 = \hat{\phi} - \cos^{-1}\hat{A} \tag{5.25}$$

Figure 5.23 plots an example scatter plot of the estimated DOA for the two-ray channel model along with the resolved DOAs of the two signals. For this example, the two signals had true DOAs of 0° and 80°. We see that when the estimate is accepted based on its quality metric, the two DOAs are successfully determined from the measured first difference data. However, this process will be very sensitive to noise because the variance of the first difference data will increase as SNR decreases.

5.7.3 Final Comments on Performance in Multipath Channels

The preceeding development does break down when modulation is included. Figure 5.24 represents a repeat of the simulation that generated the data for Figure 5.22 with the addition of BPSK modulation. This shows that if the power of one signal is within 10dB of the other signal's power, the DF algorithm exhibits a definite tendency toward error. But notice that although the number of obviously erroneous estimates increases, the algorithm is still capable of producing a decent proportion of correct estimates of the DOA of the stronger signal down to a power ratio of roughly 3dB. This means that although the two-ray resolution will not work in general, if one path is at least twice as strong as any other reflections, the algorithm will still be able to determine the proper DOA, although not necessarily in a reliable manner. Furthermore, as the number of multipath components grows, it will be ever more likely to make mistakes.

We can conclude that from this simple model, the algorithm in its current form can not be expected to offer robust performance in multipath channels. Keep in mind that this investigation

did not consider most of the effects of small-scale fading, such as time-varying amplitude or phase of the channel, only the effect of the reception of two signals with a large angular separation and large-scale fading. With small-scale fading, a time-varying amplitude on the received signal should not adversely affect the DOA estimation process as the PLL is designed such that it will be able to maintain a decent estimate of the received signal's phase even when the signal is experiencing a deep fade. If the angular separation between paths is small such that the received signal appears to emanate from a single DOA, the system should perform as if the SNR is temporarily degraded during the fading. However, a time-varying channel phase will definitely prove problematic because if the phase of the channel varies quickly relative to the switching rate of the array. If the channel phase is not relatively constant over the time it takes to perform one full array sweep, then the basic assumptions on the operation of the algorithm will be violated and the DOA estimation process will break down.

Chapter 6

Algorithm Implementation

The final part of this work was concerned with the proof-of-concept implementation of the additions to the PLL algorithm discussed in chapter 4. Initial work on the algorithm [1],[2],[3],[4] yielded a working version of the basic PLL DF algorithm as presented in chapter 3. This initial implementation provided the starting point for the implementation of the additions to the 8-element algorithm. In this chapter, we present the implementation of the algorithm including both the hardware and software platforms, the test environment, as well as the successes and failures encountered in the process.

6.1 Hardware Overview

6.1.1 WJ-8629a Software Defined Receiver

The WJ-8629a Software Definable Receiver, provided by DRS Signal Solutions, Inc., is a DSP based software radio with a frequency range of 20MHz to 2.7GHz, a maximum computational rate of nearly 1GFLOPS, and communication and data interfaces over both RS-232 and VXI. For data processing, it provides either samples directly from the analog-to-digital converter or from the digital down-converter in both rectangular (I and Q) and polar (magnitude and phase) form along with frequency and automatic gain control (AGC) information. There are 22 predefined IF filter bandwidths from 200Hz to 1.23MHz and 5 slots for user customization. Along with its included set of demodulators and audio/video filters, it allows the user to develop custom algorithms in either C or assembly for its TMS320C6701 DSP through the use of the included Sunrise DSP software developer's kit (SDK).

Figure 6.1 shows a block diagram of the processing flow in the WJ-8629a. The RF input

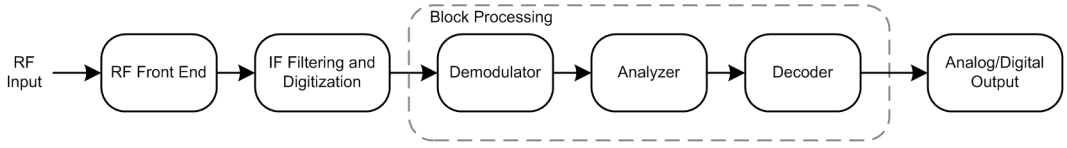

Figure 6.1. Overview of processing flow in the WJ-869a.

signal is first processed by the analog RF front end which performs the initial wideband filtering and downconversion to a 70MHz IF signal. The IF signal is then processed by the digital downconverter (DDC), which digitizes the signal, filters it with a selectable narrowband filter, and converts the digitized signal to baseband. The WJ-8629a offers 22 different IF bandwidths from 200Hz to 1.23MHz with data sampling rates and block sizes varying accordingly. It is possible for the user to create up to 5 custom IF filters with the Sunrise SDK. The data output from the DDC is then available for block processing within the DSP.

Three separate software processing slots, labeled the demodulator, analyzer, and decoder, are available for the use of predefined or custom processing modules. The WJ-8629a includes a set of common demodulation functions, such as AM, FM, CW, LSB, USB, ISB, and FSK, for use in the demodulator slot. The analyzer and decoder both have the options for audio/video filters as well as an FSK decoder. Each slot allows for up to 4 custom DSP algorithms to be downloaded and activated. During normal operation, the digital data from the DDC is first passed to the demodulator slot and processed by the selected demodulation function. The data output from the demodulator is then passed to the analyzer, the output of which is in turn passed down to the decoder. One or more of the slots can be bypassed at any one time, resulting in the data blocks simply being passed through the bypassed processing block without change. The final output of the decoder block is then available as either analog or digital data through one of many interfaces, either as a stereo audio signal from the headphone jack, as a mono video output from a SMA connector, or as digital data through both the VXI bus and RS-232 port.

Receiver control is either performed locally by a PC controlling the VXI bus or remotely through an RS-232 connection. All commands and queries to the radio, such as demodulator selection and tuned frequency, are sent using ASCII. For example, to tune the radio to 94.7MHz the string "FRQ 94.7" is sent and to determine the tuned frequency of the radio the string "FRQ?" is sent, to which the radio responds with "FRQ 94.7". The WJ-8629a used in our implementation was installed in a VXI chassis with a National Instruments VXI PC as the bus controller. Software supplied with the receiver allowed for control of the receiver through a GUI. This GUI simplified the tasks of downloading our custom algorithms as well as recording data output from the DSP.

For our implementation, an IF bandwidth of 100kHz was selected which uses a sampling rate of 125kHz and a data block size of 64 samples per block. The noise floor of the receiver at this

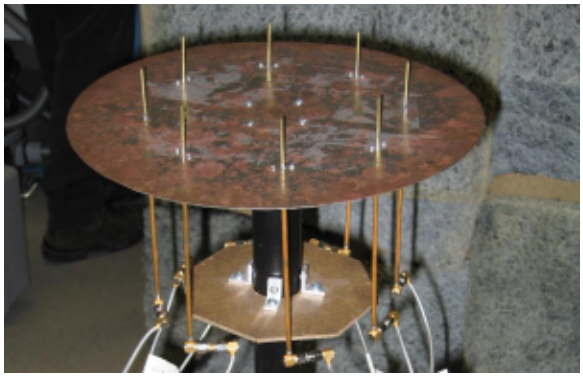

Figure 6.3: MPRG 8-element uniform circular array.

bandwidth is -110dBm. This bandwidth was selected because it represented a good trade off between noise level and block size – the PLL with a filter gain of 2^{-3} is able to acquire the target phase of the signal completely within one data block if the antenna switch has a temporary malfunction resulting in a loss of data for one or more array sweeps and the noise level on the received data allowed for low power transmission of the target signal. As testing occurred for the most part in an indoor environment, low transmission power was desired to reduce the power of any possible multipath reflections. The DF function was written in C and consisted of a single module that was downloaded to the demodulator slot in the receiver. The analyzer and decoder slots were set to bypass mode as they were unused.

6.1.2 8-element Antenna Array

The MPRG antenna array [17], shown in Figure 6.3, is a uniform circular array of 8 monopole antennas with a diameter of approximately 19.5cm. At 2.0GHz, this gives a radius of 0.65λ and inter-element spacing of $\lambda/2$. As discussed in chapter 5, this antenna spacing led to an unacceptable level of error in the initial implementation as it increases the amplitude of the first difference curve to a critical point beyond which the curve-fit algorithm can not successfully resolve

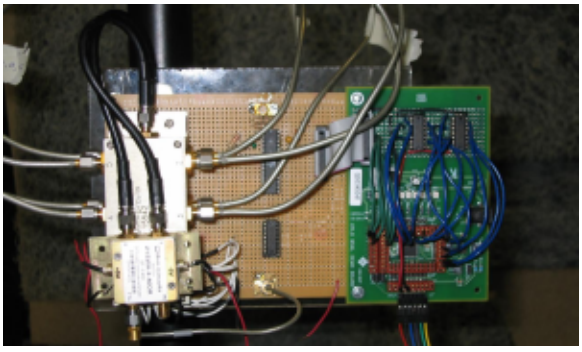

Figure 6.2. 8-element array switching Circuit.

the $\pm\pi$ ambiguities on the observed first difference data. Subsequently, the frequency of operation was reduced to 1.5707GHz. This reduced the radius of the array to $\lambda/2$. Conceptually this frequency was chosen because the target signal will go through one complete period as it propagates across the array. The corresponding inter-element spacing for the array at this frequency is approximately 0.38λ.

6.1.3 Antenna Switching Circuit

As this array was originally designed for use with a multi-channel receiver system, it did not have the necessary switching hardware to use with a single-channel receiver. Therefore, a switch was developed to take the 8 RF outputs from the array and select one for routing to the receiver based on a control signal generated by the receiver. The antenna switching circuit developed, shown in Figure 6.2, consists of two main parts: a set of RF switches suitable for the frequency range and number of antennas, and a control circuit to . The switches selected for the project were a combination of single-pole four-throw (SP4T) and single-pole double-throw (SPDT) gallium-arsenide (GaAs) switches from Mini Circuits (models ZSWA-4-30DR and ZYSWA-2-50DR, respectively). These switches were selected primarily due to their wide frequency range (DC-3.0GHz), fast switching times (45ns typical), high port isolation (32dB minimum at 2.0GHz), and TTL control.

The switch control circuit, a block diagram of which is shown in Figure 6.4, primarily consists of a Texas Instruments ADS7805 16-bit analog-to-digital converter (ADC) for reception of the control signal and a Xilinx complex programmable logic device (CPLD) for ADC control and

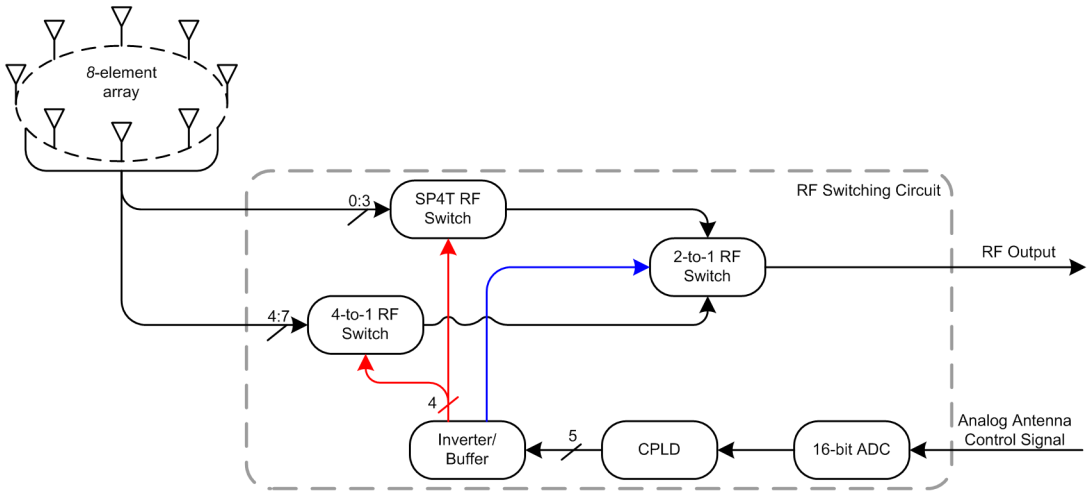

Figure 6.4. Switching circuit for the 8-element array.

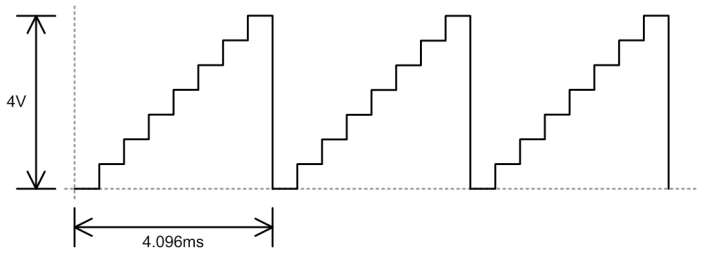

Figure 6.5. Analog control signal for 8-element switching circuit.

switch signal processing. The design was a joint product of both this work and the previous student's work. The analog control signal output by the receiver, shown in Figure 6.5, essentially consisted of a "staircase" function in which each of the eight voltage levels corresponded to the desired antenna selected by the switching circuit. This control signal is output at the receiver from the analog video output port, which is an SMA connector. It is generated in the DSP code by simply setting the output data array for an entire processing block to a single value.

The ADC samples this signal at 100kHz under the control of the CPLD. The CPLD then latches the ten most significant bits of the 16-bit output upon receiving the "Ready" signal from the ADC. The CPLD then converts the measured voltage level to the proper combination of TTL signals to drive the switches. The SPDT switch requires a single control signal to select between its two inputs while the SP4T switches used 4 control signals each. As the output from only one SP4T would be selected by the SPDT at any instant in time, these switches shared control signals in order to reduce the complexity of the wiring. These five switch control signals are latched and the latched outputs are then wired from the CPLD's interface pins through inverter ICs to the switch control ports.

6.1.4 Test System Setup

Testing of the DF system implementation took place primarily in the hardware design lab in the MPRG student office space located in Durham Hall. Figure 6.6 is a block diagram of the basic test scenario. These test scenarios were set up to approximate an AWGN channel. To attempt this, the target signal was transmitted at a low enough power so that any multipath reflections would hopefully be weak compared to the primary signal but the primary signal would be received at a reasonably high SNR to provide a relatively "clean" received signal for algorithm verification. In addition, a log-periodic antenna was used at the transmitter to direct most of the target signal towards the DF antenna array and reduce the number of secondary reflected paths. The transmitter

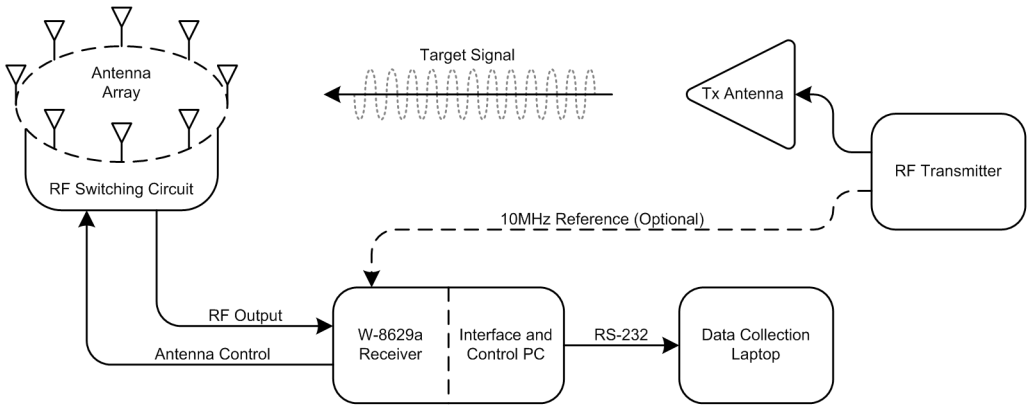

Figure 6.6. DF system testing setup.

and antenna array were usually separated by at least two to three meters so that the target signal appears to come from the far field of the array.

6.2 Implementation of 8-Element Algorithm Additions

Overall, the implementation of the various processing modules proved to be straightforward and did not represent a significant risk as the underlying hardware and software platforms for the basic 8-element algorithm were already shown to have worked. Once the simulation blocks were working in Matlab, all that was needed was to convert the Matlab functions into C code and include them in the C source code for the DSP. Of course, certain restrictions applied to the algorithm as implemented on the DSP that do not exist in the Matlab environment. First, the majority of the C code was written using fixed-point variables instead of the double precision floating point variables used by default in Matlab, primarily to save processing time as fixed-point arithmetic in general takes fewer processing cycles to perform that floating point. In fact, during the initial implementation of the algorithm it was discovered that although the DSP was designed to perform floating point operations the use of floating point math for the entire implementation exceeded the maximum number of processing cycles allowed per data block by a large margin. The use of fixed point variables also required that special attention be paid to the operation of the filters in both the PLL as well as the frequency offset estimator. Second, the implementation needs to coordinate the antenna switching with the PLL block processing. There will be a delay between the output of the switching signal the reception of the corresponding antenna output due to, among other things, the receiver's operating system overhead and the time the signal takes to propagate through the various filters in the RF front end and digitization section.

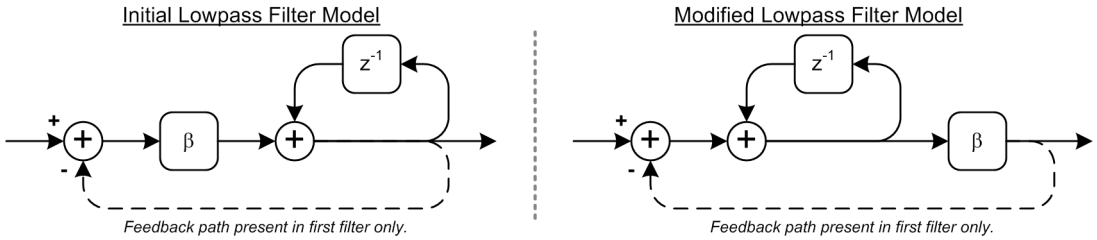

Figure 6.7. Implementation model for lowpass filters in frequency offset estimator showing a.) (left) initial model and b.) (right) modified model..

6.2.1 Linear PLL and Frequency Offset Estimator Implementation

As the nonlinear to linear PLL transformation and the addition of the frequency offset estimator would require two major overhauls of the PLL operation, it was decided to completely replace the existing PLL with the frequency offset estimation version from Matlab simulation. Implementation was relatively straightforward and required only minor alterations to translate the Matlab code to C. It was discovered during initial testing of this new PLL that the frequency offset estimation circuit would fail to fully acquire the frequency offset, leaving a considerable residual offset on the received data. The cause of this was the first filter in the frequency offset estimation process. Figure 6.7a shows the initial signal flow model used to implement the filter. Remember that the difference equation describing the filter is given by:

$$y[n]=\beta_1 x[n]+(1-\beta_1)\,y[n-1] \tag{6.1}$$

In order to reduce the number of gain elements needed for the filter, it was instead implemented using:

$$y[n]=\beta_1(x[n]-y[n-1])+y[n-1] \tag{6.2}$$

which is accomplished by presenting the filter with the difference between the current input and output (i.e. $x[n]-y[n-1]$) rather than the input $x[n]$. The gain parameter, β_1, selected for this filter was 2^{-5}.

It turns out that with the phase quantization in the DDC this filter gain would essentially reduce the fixed-point input to zero if the residual frequency offset fell below 60Hz. The phase values output from the DDC are 16-bit integers which means that $\pi=2^{15}-1=32767$. Recall from the discussion of the estimator in chapter 4 that the estimator does not explicitly track the value of the frequency offset in Hertz, but the quantity $2\pi\Delta f\, t_s$, where Δf is measured in Hertz and t_s is the sampling frequency measured in seconds. In fixed point notation, this value becomes

$(2^{16}-2)\pi\Delta f t_s$, and a frequency offset of 61Hz in turn is represented by the integer 32 (100000 in binary). To simplify the implementation, all filter gains were set as powers of 2 so that they could be implemented using right shifts instead of divisions. Because of this, a frequency offset of 61Hz will be reduced to the integer 1 by this filter. Smaller values of the frequency offset will be effectively ignored by the filter.

One solution to this problem would be to increase the filter gain so that it could track a much smaller residual offset but that would increase the variance of the estimator due to an increase in the effective noise power. A quick return to elementary discrete time filter theory provided the answer – simply switch the places of the accumulator and gain element in the filter, as shown in Figure 6.7b. This will result in an equivalent filter while simultaneously solving the problem as the accumulator will then be able to grow to a much larger value. This accumulated value is then scaled by the filter gain. With this minor modification, the frequency offset estimator performed its intended purpose extremely well.

Figures 6.8a and 6.8b show an example of the received phase and corresponding PLL output during the acquisition and tracking stages of the frequency offset estimator. Each plot shows the output of the PLL for each antenna for two full array sweeps with 64 samples taken per antenna.

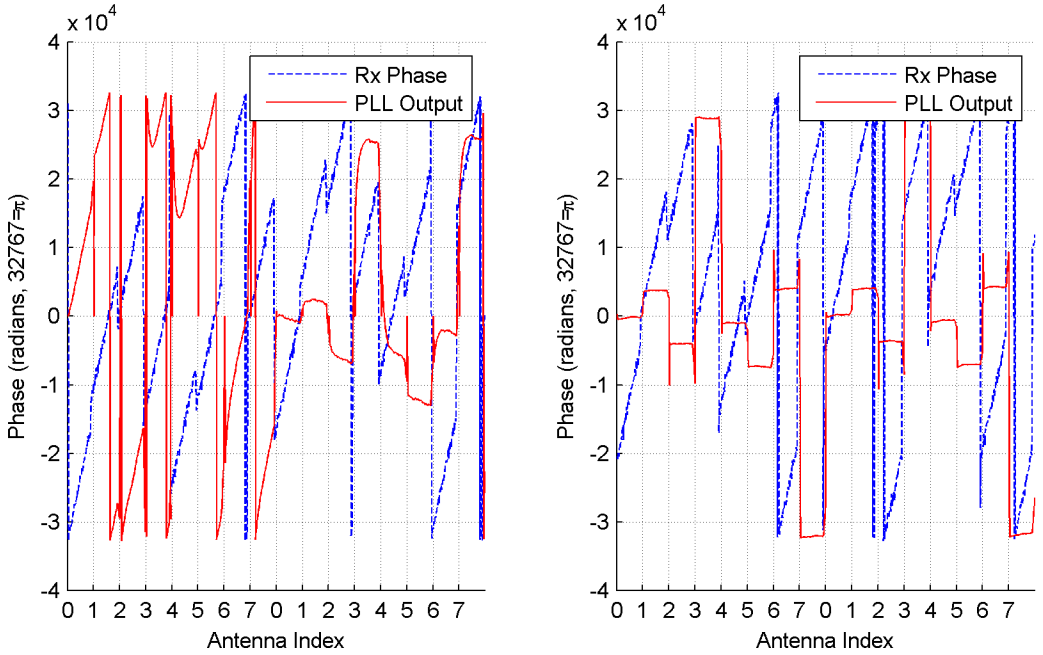

Figure 6.8. Example plot of implementation of PLL output with ~560Hz frequency offset during a.) (left) frequency offset acquisition and b.) (right) frequency offset tracking phases.

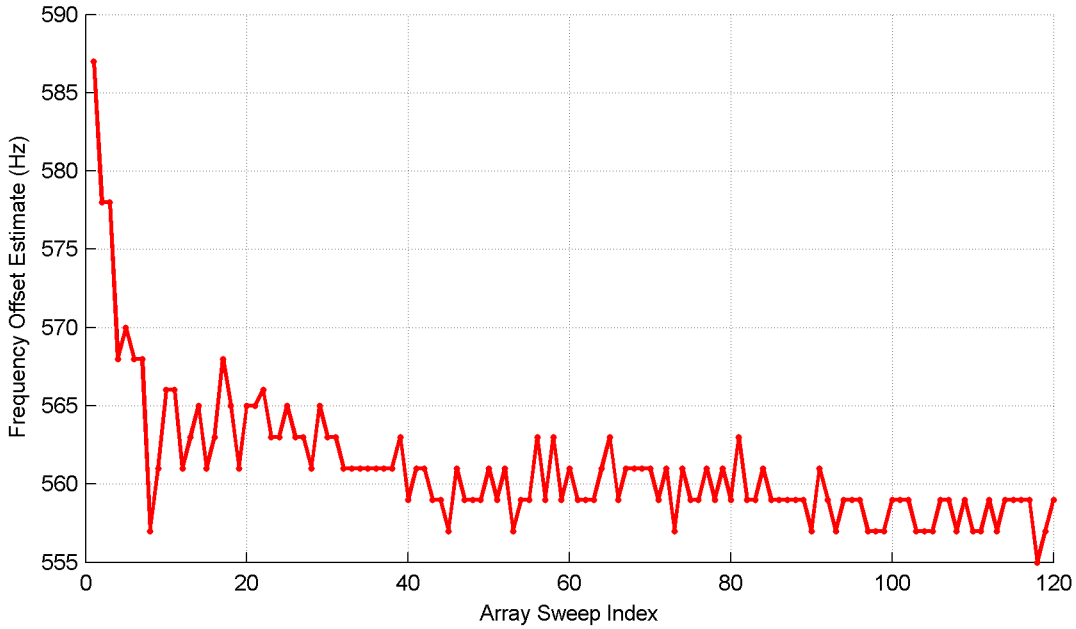

Figure 6.9. Example frequency offset estimator output from implementation for a received signal with a true frequency offset of approximately 560Hz.

We see in Figure 6.8a that initially the PLL has a slope nearing that of the received phase, although with a definite phase offset. But after the first array sweep, the PLL outputs somewhat level off due to the offset estimator beginning to acquire the frequency offset. Figure 6.8b, which is taken from the same recorded data approximately 60 array sweeps after the first plot, shows that the frequency offset estimate has stabilized as the phase tracked by each PLL is relatively constant across each data block.

Note that the phase tracked by the first PLL (during antenna index 0) starts out near zero at every array sweep. Because the received signal has a time-varying phase, in the absence of perfect knowledge of the frequency offset the exact value of the phase of the signal at the start of each array sweep is unknown. For a stationary target, it is assumed that the constant phase difference between elements will remain the same and this assumption must remain valid in order to facilitate parallel operation of the PLLs. Otherwise, the PLL for each antenna would have to reacquire the signal's phase with each array sweep. To compensate for this time-varying phase, the first phase sample from the first antenna is used as a constant offset for every other phase sample for the entire array sweep. This ensures that the first PLL will always track near zero while every other PLL will track a constant phase value.

Figure 6.9 shows the output of the frequency offset estimator from the same data set the example PLL plots were taken from. We see that after approximately 35 array sweeps the estimator has acquired the frequency offset. The "true" value of the frequency offset was estimated to be 560Hz from recorded data received by a single antenna over a period of approximately 2 seconds and postprocessed in Matlab using a large FFT. Note that the offset estimate moves in 2Hz steps – this is the smallest frequency offset that can be measured due to the system's phase quantization.

6.2.2 DOA Estimate Quality Metric Implementation

Implementation of the DOA estimate quality metric consisted of two steps. First, the method of DFT calculation needed to be addressed as only one frequency bin was calculated in the initial implementation. Second, a method needed to be devised to implement the actual metric calculation as division is not supported by the WJ-8629a due to peculiarities in the compiler used. Specifically, the division and square root functions offered by the TI FastRTS library were not supported on the receiver. Since the metric calculation relied on determining the magnitude of a set of complex numbers another approach needed to be found.

In order to produce each DOA estimate, the phase of one bin of the DFT of the first difference data needs to be determined. The original algorithm implementation managed this by directly evaluating the DFT for the single frequency required. In order to calculate the DOA quality metric, the algorithm requires half of the frequency spectrum to be calculated. In order to accomplish this, a few options were considered in addition to direct evaluation of the DFT, such as the Goertzel algorithm and other advanced FFT techniques, but in the end direct DFT evaluation was the method chosen because it was recognized that the data set in question consists of only 8 or 16 elements. While the Goertzel algorithm and decimation-FFT algorithms are shown to be more efficient than direct evaluation of the DFT, the processing cycle reductions offered by these options is minimal when the actual data series is small. For the 8-element algorithm, determining four frequency bins instead of just one added a nearly trivial amount of processing time relative to the entire algorithm implementation. For the 16-element algorithm, the massive reduction in the processing requirements of the curve fit algorithm compensated for the calculation of eight frequency bins.

Determining the magnitude of the complex numbers representing the frequency bins calculated by the DFT without the availability of a square root function at first glance appears

daunting. After searching for various numerical approximations to determine the magnitude of a complex number, such as a polynomial approximation of a square root function or successive approximation utilizing a binary search, a much simpler approximation was found [18] that simply estimates the magnitude of a complex number by using a linear combination of the real and imaginary parts:

$$|X| \simeq \alpha \cdot max(|I|, |Q|) + \beta \cdot min(|I|, |Q|) \tag{6.3}$$

where I is the real part of the complex number X and Q is the imaginary part and α and β are coefficients which can be altered based on the desired error characteristics of the estimator. This approximation works by essentially restricting the phase of the complex number to a region in which a linear combination of the two parts is a reasonable approximation of the magnitude. First, taking the absolute value of the real and imaginary parts of X forces the phase of the complex value to the first quadrant. Second, the *max* and *min* operations further restrict the phase of the number to be less than $\pi/4$. Table 6.1 shows an abbreviated list of possible values for the coefficients α and β and their associated error statistics.

Table 6.1. List of coefficients and associated error statistics for magnitude estimation function.

Name	α	β	Error Characteristics		
			Average (linear)	RMS (dB)	Peak (dB)
Min. RMS Error	0.94754	0.39245	0.00055	-32.6	-25.6
Min. Peak Error	0.96043	0.39782	-0.01305	-31.4	-28.1
Min. RMS w/ Avg. = 0	0.94806	0.39270	0.0	-32.6	-25.7
1, Min. RMS Error	1.0	0.32326	-0.02087	-28.7	-23.8
1, Min. Peak Error	1.0	0.33598	-0.02561	-28.3	-25.1
1, 1/2	1.0	0.5	-0.08678	-20.7	-18.6
1, 1/4	1.0	0.25	0.00646	-27.6	-18.7
1, 3/8	1.0	0.375	-0.04016	-26.4	-23.4

Selection of the coefficients depends on the acceptable amount of error in the estimated magnitude. As the quality metric denominator is the sum of 3 or more numbers produced by this approximation, the "Min. RMS w/ Avg.=0" version of the coefficients was selected for use in the algorithm. To perform the final metric calculation, the numerator and denominator values were produced and the denominator value was scaled by the desired metric threshold.

6.2.3 *Data Collection and Filtering Using a Matlab Graphical User Interface*

Initially the DOA estimates were recorded using the debug interface to the DSP from the Sunrise SDK tools. The memory location that held the estimated DOA variable was periodically probed for its value and written down. Obviously, this process is very tedious and time-consuming and a more efficient approach could be found. As all control of the receiver during development and testing took place using the VXI bus, the RS-232 port was free for other uses. Therefore, that port could be used to output the DOA estimates as well as their associated quality metrics. The approach taken was to:

1. Convert the floating-point DOA estimate to a string of ASCII characters in a fixed format with 3 digits before and 4 digits after the decimal point.
2. Output DOA estimate string followed by carriage return and line feed characters.
3. Output decision based on the calculated quality metric followed by carriage return and line feed characters. The string "+1" is sent for a good estimate and "-0" for a bad estimate.

The receiver will output these values for every array sweep following system initialization. Since the DOA estimate and metric decision values are sent as plain text values instead of binary encoded, any terminal program can be used to display this data during system operation. This also facilitated the development of a Matlab GUI to collect and display DOA estimate data in real time.

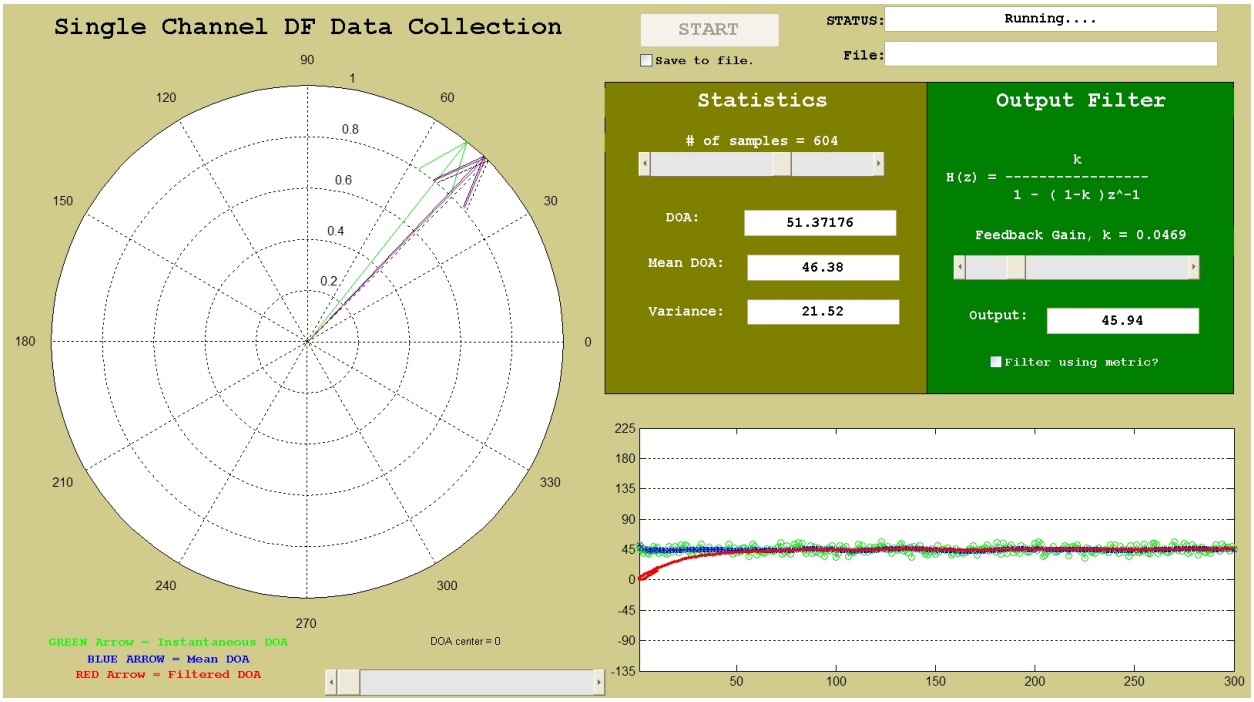

Figure 6.10. Screen capture of Matlab data collection GUI.

A screen shot of the GUI is shown in Figure 6.10. The underlying Matlab program opens the serial port on the host PC to listen for the receiver serial output, retrieves each DOA estimate value, and then calculates various statistics for a sliding window of estimates as well as filters the data. The polar plot in the left of the GUI plots the current DOA estimate as well as the current mean DOA and filter output. The plot in the lower right of the GUI plots a sliding window of the previous 300 DOA estimates. The GUI also allows the user to save the collected DOA estimates to a Matlab data file for future processing.

16 Element Array Construction

In order to test the implementation of the 16-element algorithm, a new antenna array was required as the 8-element array and switching circuit would obviously be insufficient. Consequently the design of a new uniform circular array was undertaken based on the design of the 8-element array.

6.2.4 Array Materials and General Design.

Table 6.2. Parts list for major items involved in construction of 16-element array switching circuit.

Part	Model #	Manufacturer/Supplier	Quantity	Price
Spartan-2 FPGA Board	B5-Spartan2e+	Burch Electronic Designs	1	$300
Flash PROM	B5-FlashProm	Burch Electronic Designs	1	$80
DIP Switches Add-on	B5-DipSwitch	Burch Electronic Designs	1	$20
SP4T Switch	ZSWA-4-30DR	Mini-Circuits	5	$120
Hex Inverter	SN54AC04	Texas Instruments	4	Free
1' Phase Matched Cables	-	RF Products	20	$10
SMA Connectors	-	Mouser.com	16	$2.50

The goal was to copy the manufacturing aspects of the 8-element array while simply scaling the size up to accommodate 16 antennas. The intended inter-element spacing was 0.4λ, which translates to an array radius of 1.03λ. At 2.0GHz, this would require an array diameter of 30.76cm (12.1in.) but the implementation is not necessarily tied to that specific frequency. For convenience sake, the array diameter was set to 10in. (25.4cm), giving an inter-element spacing of 0.4λ at an operating frequency of 2.422GHz, which is well within operating frequency range of both the receiver and RF switches. Table 6.3 is a list of the major parts used in construction of the 16-element array.

As this array will be digitally controlled, a Xilinx Spartan2 field programmable gate array (FPGA) from Burch Electronic designs was chosen. The development board from BurchED provides access to all 230 I/O pins on the FPGA and contains its own voltage regulators and a programmable oscillator that can provide a stable clock signal from 1MHz to 100MHz. A Flash PROM add-on board from BurchED was selected to load the FPGA program upon power-up. This is required because the circuitry in FPGAs is RAM-based and will lose its programming when it loses power. An additional add-on board containing 16 DIP switches was included to provide flexible input for the selection of multiple possible switching algorithms in the FPGA as well as for future expandability.

The SP4T RF switches from Mini-Circuits were the same switches used in the switching circuit for the 8-element array. They require both +5V and -5V power supplies and 5V TTL signals for control. The hex inverters are CMOS based inverters used to buffer the input and output pins on the FPGA in order to protect the FPGA from current spikes. The inverters are also used to convert both the the digital control signals from the receiver as well as the control signals to the switches. Both the receiver's digital outputs and the switch control signals operate at 5V while the FPGA operates at 3.3V. The FPGA is technically capable of sinking and sourcing 5V logic signals, but the inverters were added as a failsafe measure.

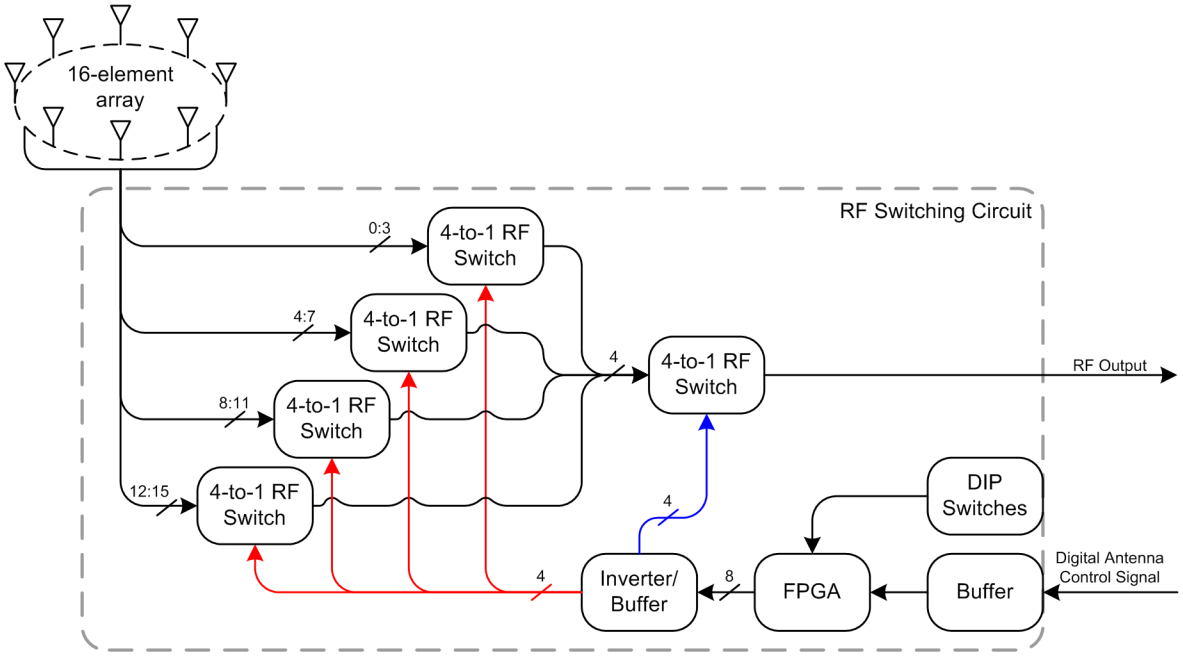

Figure 6.11. Block Diagram of switching circuit for 16-element array.

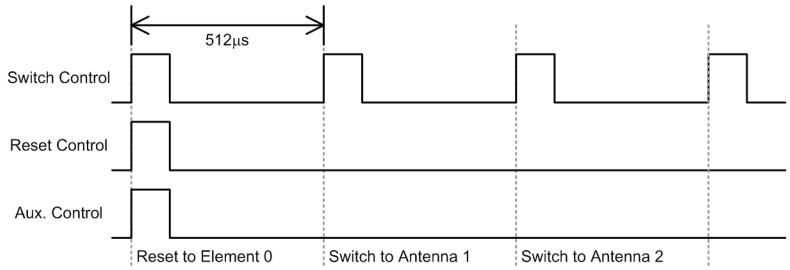

Figure 6.12. Digital control signals for new switching circuit.

The antennas and ground plane were made of the same materials as the 8-element array. The antennas were made from rigid brass tubing with a diameter of 1/16in. and sanded to a length of $\lambda/4$ and soldered to the end of the SMA panel mount receptacles. The ground plane consisted of a piece of sheet metal to which the SMA receptacles were attached.

6.2.5 Switching Circuit Design

A block diagram of the new switching circuit design is shown in Figure 6.11. Thanks to a special modification by an engineer at DRS-SS, the WJ-8629a used for this implementation also includes 3 digital output signals which can be set from the DSP code by simply setting bits in certain memory locations. These digital outputs were routed to unused pins on one of the receiver's front-panel connectors. The original scheme for antenna control was to use one of the digital liines as a data clock and the other two lines as parallel data streams that would send the binary number corresponding to the desired antenna element two bits at a time. Unfortunately, timing for the signals proved difficult to consistently provide and the FPGA was unable to properly receive the data.

Consequently a simpler signaling scheme was devised and is shown in Figure 6.12. One of the control lines is used as a "switch control" signal. When the FPGA detects a short pulse on this line it simply switches to the next element counter-clockwise from its current position. The second control line is used as a "reset control" signal. Sent concurrently with the "switch control" signal, this signal tells the FPGA to reset to the first antenna in the array regardless of its current position. The third control line, called "auxiliary control" is unused but still sent to the FPGA to provide future expansion capabilities. In the current implementation it simply mimics the "reset control" signal but is ignored by the FPGA. The one drawback that this circuit has in relation to the 8-element switching circuit is that it is only possible to easily select the first antenna in the array instead of any arbitrary antenna. For testing purposes, this was solved by the addition of the DIP

switches for use as manual control inputs.

The RF switch network takes a two-tiered approach. The first tier of 4 SP4T switches are connected directly to the antennas by a set of 16 phase-matched SMA cables. The outputs of the four first tier switches are then connected to the inputs of the single second tier SP4t by another set of 4 phase-matched SMA cables. The output of the second tier switch provides the ssingle RF output for the array.. A total of 8 digital control signals are routed from the FPGA through inverters to the SP4T switches. The first tier switches all share the same 4 control signals. As only one of the first tier switches can be selected by the second tier switch at a time, the state of the other three first tier switches is unimportant.

Table 6.3. Tabular description of manual switching circuit state control via DIP switches.

DIP Switch State (Logical)								**Switching Circuit State**
0	**1**	**2**	**3**	**4**	**5**	**6**	**7-15**	
1	X	X	X	X	X	X	X	Reset State (Non-Operational)
0	0	0	X	X	X	X	X	Operational, Switch through 16
0	1	0	X	X	X	X	X	Operational, Switch through 8 (even indexed)
0	X	1	0	0	0	0	X	Static, Antenna 0 selected
0	X	1	1	0	0	0	X	Static, Antenna 1 selected
0	X	1	0	1	0	0	X	Static, Antenna 2 selected
							• • •	
0	X	1	1	0	1	1	X	Static, Antenna 13 selected
0	X	1	0	1	1	1	X	Static, Antenna 14 selected
0	X	1	1	1	1	1	X	Static, Antenna 15 selected

In order to provide backwards compatibility between the new switching circuit and the original 8-element antenna array as well as to facilitate array calibration, an expansion board for the FPGA containing 16 DIP switches was used. Table summarizes the various control options available for the switching circuit. The reset state is asynchronous and activated by the first switch (switch 0). During reset, the FPGA ignores all other switch and signaling inputs and directs the RF switches to select the first antenna. The first operational state, labeled in the table as "Switch through 16," switches sequentially through all 16 antennas based on the control signaling from the receiver. The second operational state, labeled "Switch through 8," switches sequentially through

the even-indexed switch ports to support the use of an 8-element array. The "static" states allow for manual selection of the antenna elements based on the binary encoding of their antenna index using switches 3 through 6.

Figure 6.13 is a picture of the assembled switching circuit. The entire assembly was mounted on a 1/4" Plexiglass base. Wiring for the RF switch power supply lines was routed underneath the base. The power supply for the inverters came from the FPGA board – in addition to a 3.3V regulator to power the FPGA, the development board also had a 5V regulator to supply any possible 5V peripherals. The Flash PROM and JTAG programming boards also drew their power from the FPGA board. The +5V and -5V power supply connections for the RF switches were provided by a desktop power supply in the MPRG lab.

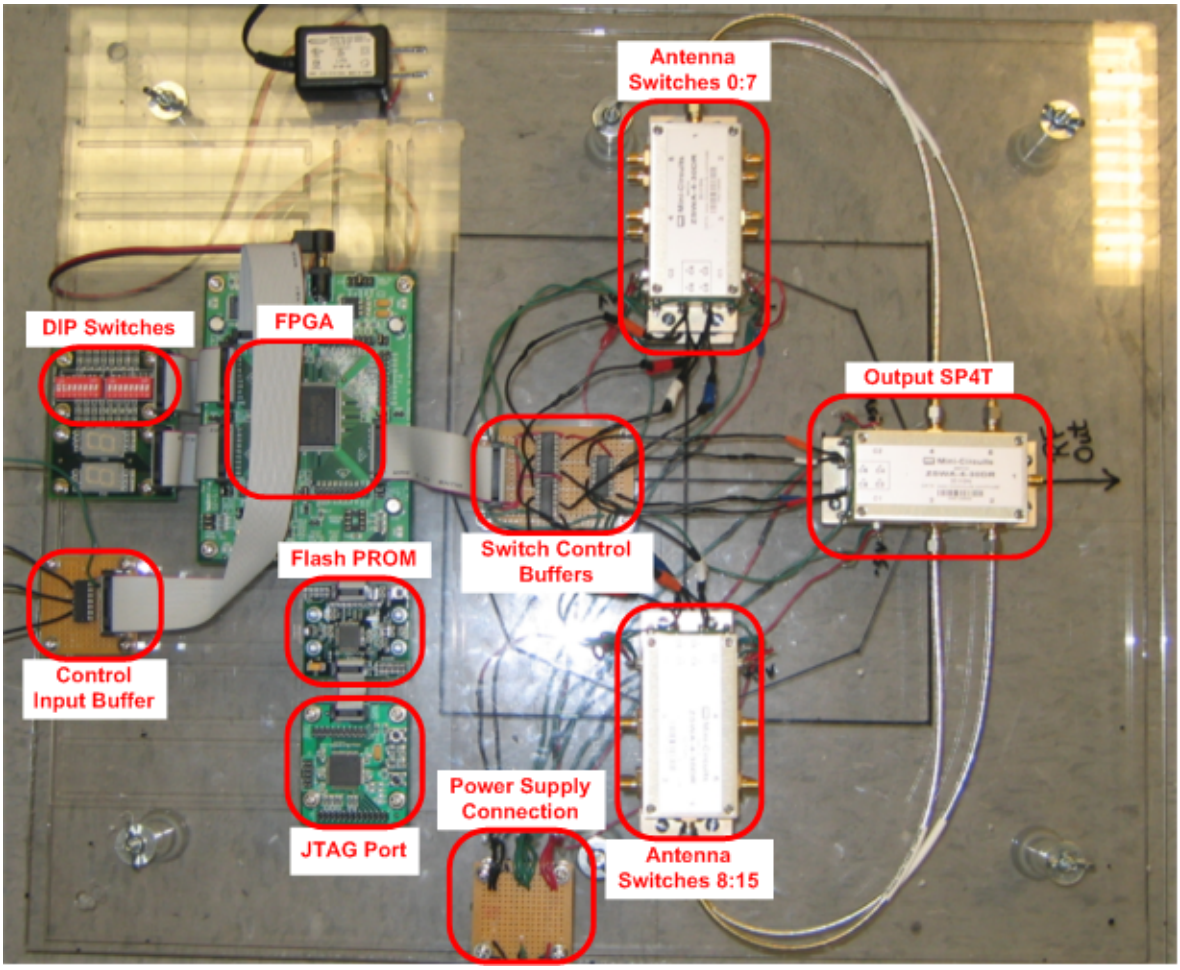

Figure 6.13. Picture of the assembled 16-element switching circuit.

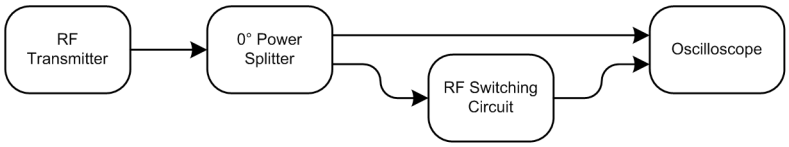

Figure 6.14. RF switching circuit phase calibration setup.

6.2.6 Antenna Array Calibration and Operation

After construction of the antenna array, the final two steps before algorithm testing were to first produce a calibration table to compensate for any phase differences between the different paths through the switching network and second to measure the propagation delay between the generation of the switch signal at the receiver and arrival of the desired antenna's data in the DSP. Figure 6.14 provides a block diagram of the phase calibration setup. An RF signal at the desired freqeuncy offset is generated by and RF signal generator and passed through a 0° power splitter. One output from the splitter is connected straight to an oscilloscope to be used as the reference signal. The other output from the power splitter is then connected to each of the 16 RF ports on the switching circuit and the output of the switching circuit is connected to the second oscilloscope input. The phase difference between the reference signal and switching circuit output is then recorded for each RF switch setting. This set of 16 phase values is then used in the DF processing at the output of the PLL to remove the constant phase shifts on the received signal due to the separate RF paths

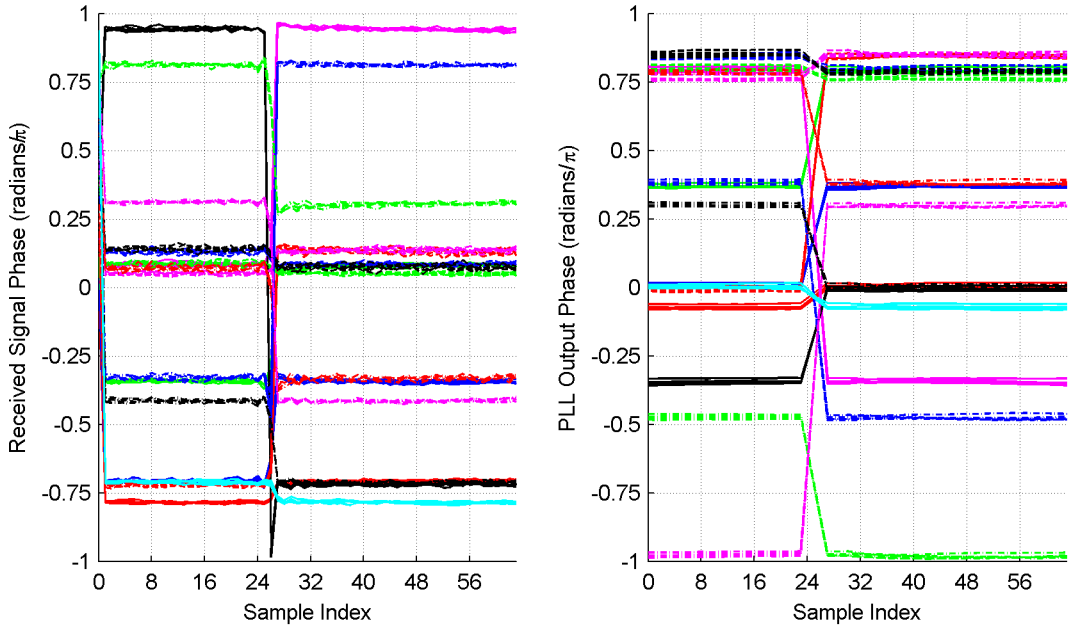

Figure 6.15. Eye diagram of 16-element array output and corresponding PLL used to measure switch transition timing.

through the switching circuit.

The final step in array calibration was to measure the timing of the switching transitions.. Figure 6.15a is an eye diagram of the phase of the array output signal plotted against sample index for each processing block. This plot shows that the RF switching transitions consistently occur during samples 26 through 28. This is markedly different from the original 8-element array – the switch transitions for this array coincided with the beginning of each data block, which definitely simplified the implementation of the parallel PLLs. For the new switching circuit, the PLL state will need to be loaded and stored near the middle of each processing data block instead of at the beginning and end of each block but that is of minor consequence. Figure 6.15b shows the eye diagram of the parallel PLL tracking plotted against the sample index for each data block. To account for the switching transitions during samples 26 through 28, the PLL state at sample 24 is stored to the relevant memory slot for that antenna and the PLL for the next antenna is not initialized until sample 31. The 5 samples (index 25 through 30) containing the switching transitions are ignored. To measure the delay between switching signal generation and arrival of the relevant antenna's data, the first antenna was disconnected from the RF switching circuit and the number of processing blocks between the generation of the reset control signal and arrival of a data block consisting purely of noise was counted. It turns out that it takes a total of 4 data blocks.

6.3 Implementation Performance

6.3.1 Processing Cycle Requirements

Figure 6.16 is a bar graph of the number of processing cycles used by each version of the algorithm relative to the maximum available cycles for the IF bandwidth used. With an IF

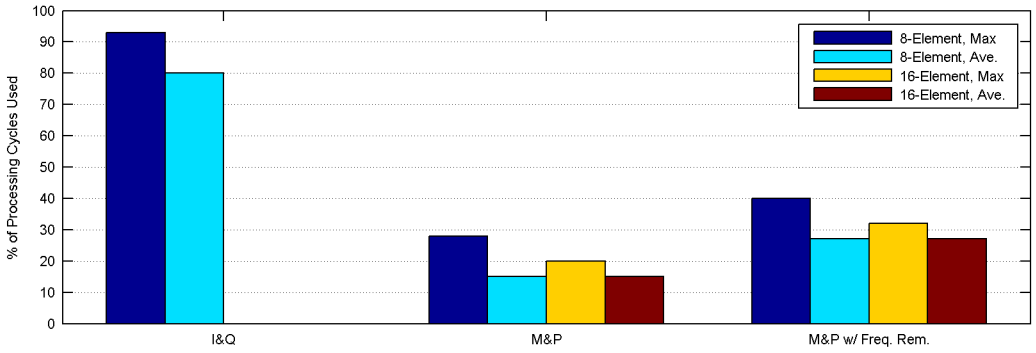

Figure 6.16. Processing cycles required by various implementations of the PLL DF algorithm as a percentage of total available cycles.

bandwidth of 100kHz, each processing block has 20480 processing cycles available. The DSP OS does introduce a slight overhead, approximately 1% of the total on average. This OS overhead includes AGC operations as well as coordination and control of the RF and digitization portions of the receiver and memory management. The "I&Q" version of the implementation represents the processing requirements of the algorithm as originally implemented using the nonlinear PLL and original 8-element array. This version required 80% of the avaliable processing cycles on average and peaked at over 90% during the evaluation of the curve fit algorithm. The "M&P" version of the algorithm with the 8-element array represents the requirements of the algorithm after the nonlinear PLL was replaced with the linear PLL and the original 8-element array. This shows that the replacement of the PLL introduced a drastic reduction in the amount of time needed to evaluate the DF algorithm. After the introduction of the frequency offset estimator, represented by the category "M&P w/ Freq. Rem.," the addition required a modest increase in the average number of processing cycles required – from 15% to 28%.

When comparing the processing requirements of the 16-element algorithm to that of the 8-element algorithm, the measurements show that the requirements for both algorithms are comparable even though the 16-element curve fit algorithm should take less time to perform. The

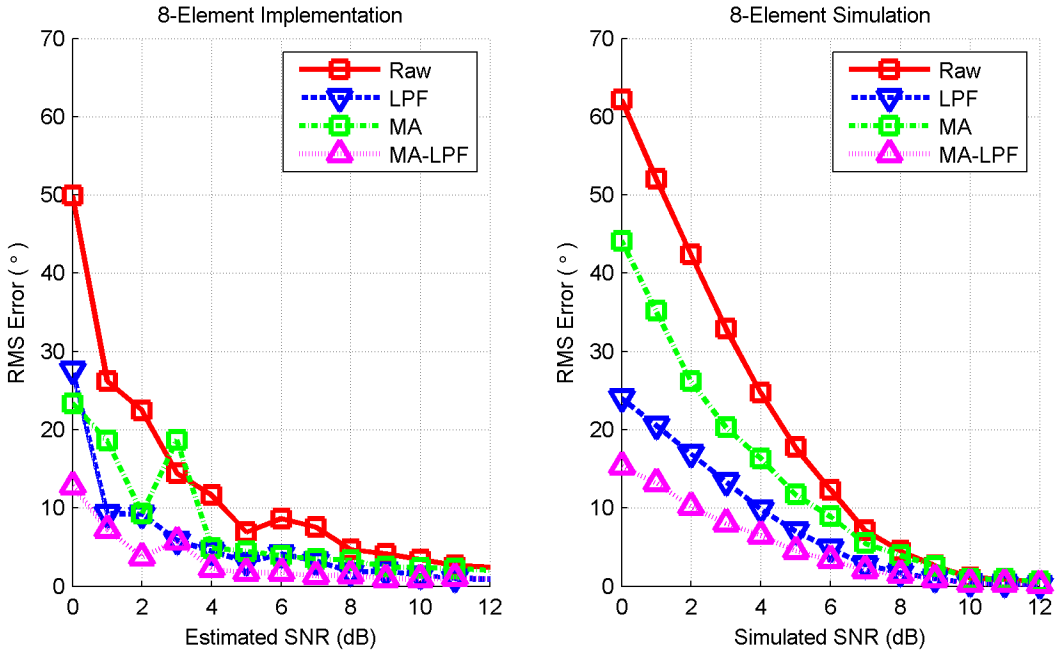

Figure 6.17. Performance plot of the implementation of the 8-element PLL DF algorithm with frequency offset removal showing a.) (left) RMS error vs. SNR and b.) (right) simulated RMS error vs. SNR for comparison.

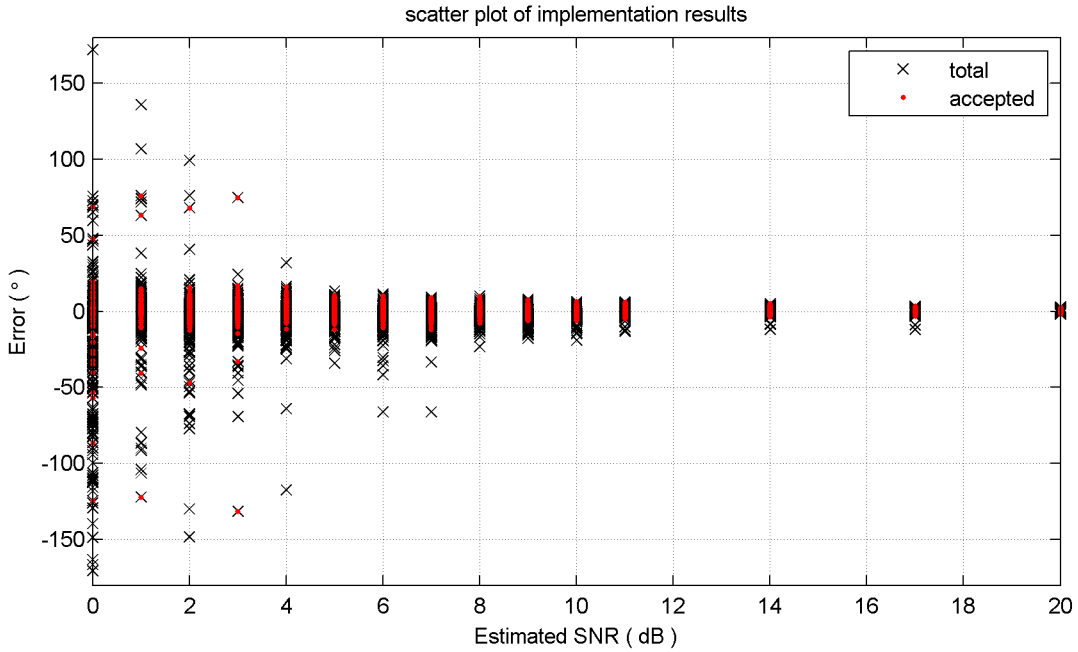

Figure 6.18. Scatter plot of error in estimated DOA samples from implementation of the 8-element algorithm.

reason the two versions have similar requirements stems from the implementation of control signal generation for the 16-element array. The generation of the digital control signaling is implemented in the DSP code by setting the necessary register bits and then using a loop of "no-operation" commands to create a delay before clearing the register bits. This delay loop runs longer than is necessary for the FPGA to detect the signaling pulses. As it was expected that the 16-element algorithm would have a significant amount of free processing cycles to accommodate a lax implementation.

6.3.2 Performance of the 8-element Algorithm with a Frequency Offset

Figure 6.17a is a plot of the measured RMS error plotted against estimated SNR on the received signal for the 8-element PLL DF algorithm with the frequency offset estimator. For this trial, the frequency offset was approximately 550Hz. This plot shows that the implementation of the algorithm offers excellent performance, i.e. less than 10° RMS error, down to an SNR of 5dB. The addition of DOA estimate filtering further enhances this error performance. Figure 6.18 is a scatter plot of the error in the recorded DOA estimate samples for each estimated SNR value. We can see that the majority of the samples are clustered about zero error and that the width of the distribution of the samples increases as SNR decreases.

When comparing these plots to the simulated performance of the 8-element algorithm in Figure 6.17b, we find that the implementation appears to perform better than the simulated system. This may not actually be the case. The SNR for these DF trials was estimated using the signal strength reported by the receiver. To determine the noise floor at the bandwidth used, the RF input was terminated and the reported signal strength was recorded. Then during the trials, the reported signal strength during algorithm operation was recorded and the difference between this value and the measured noise floor was used as the estimated SNR. This was only a rough estimate of the signal strength as the amplitude of the received data noticeably varied as the array switched. The signal strength reported by the receiver was a long term average and essentially hid the fluctuations in the strength of the received signal from each antenna.

6.3.3 Performance of the 16-element Algorithm

Unfortunately, DF trials with the 16-element array ultimately were not as successful as with the 8-element array. Irregularities in the array construction resulting from the irregular shape of the ground plane relative to the shape of the array as well as difficulties producing the actual antenna elements contributed to the received signal manifold deviating from the theoretical model enough to cause breakdowns in the curve fit algorithm. Figures 6.19a and 6.19b plot the phase curves traced

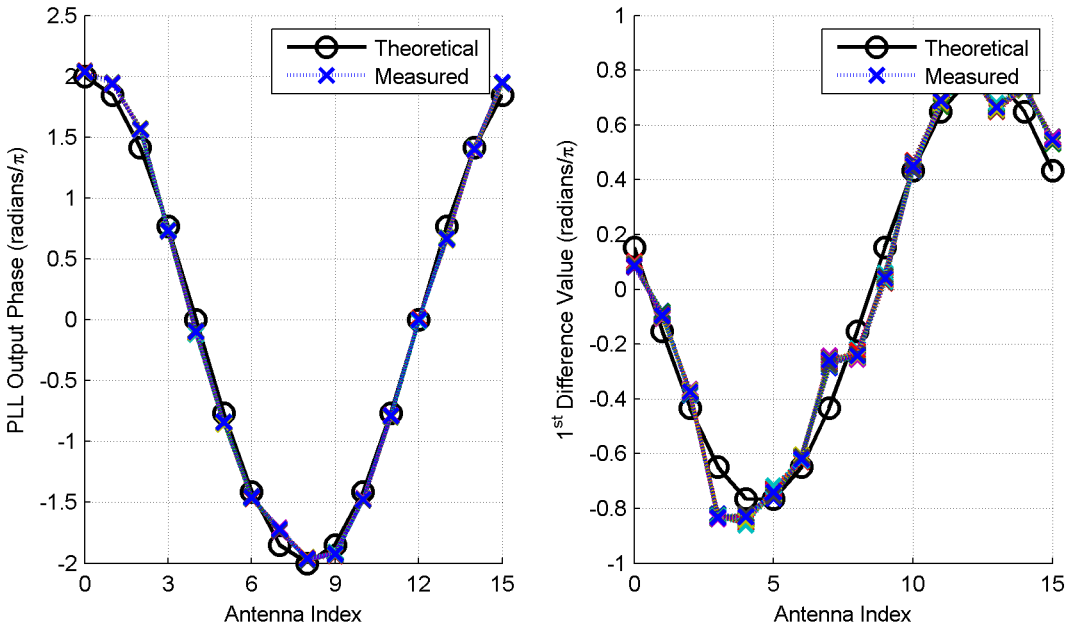

Figure 6.19. Plots of phase curves measured using the 16-element antenna array showing the a.) (left) PLL output and b.) (right) first difference curves for 64 consecutive array sweeps.

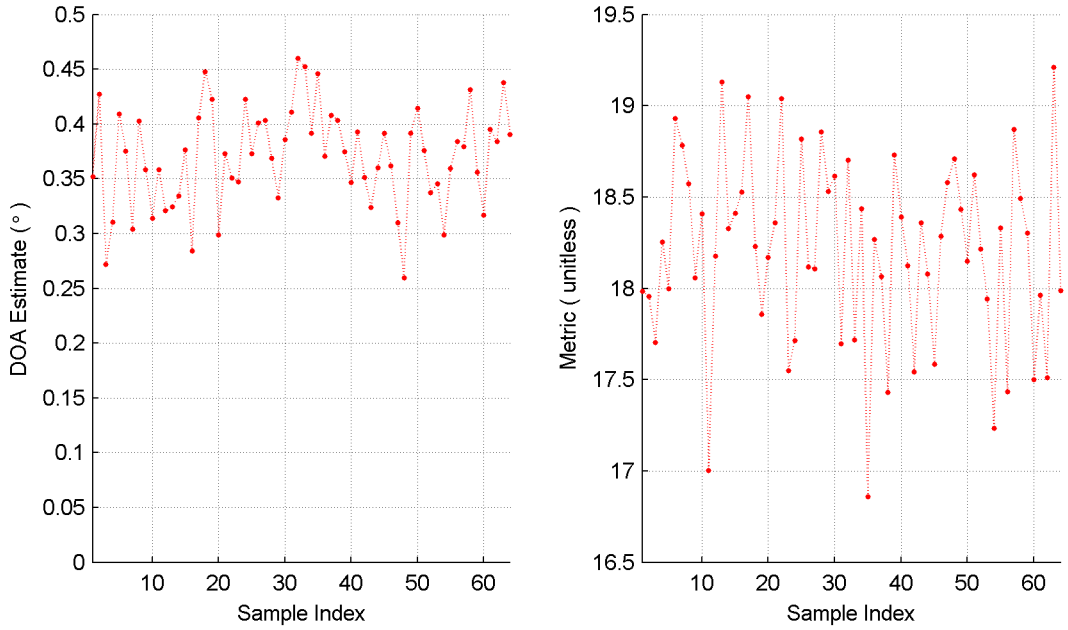

Figure 6.20: Plots of data from the implementation of the 16-element algorithm showing a.) (left) Estimated DOA and b.) (left) associated quality metric magnitude over time for a true DOA of 0°.

by the parallel PLL output and the first difference from implementation data and compare them to the theoretically expected curves. This data was taken during the only successful trial of the 16-element DF algorithm. The true DOA was approximately 0° and the estimated SNR was over 20dB. We can see that although the PLL output data appears relatively close to the expected value, the first difference curve resulting from the PLL output data appears to differ significantly from the expected value, yet the curve fit algorithm is still successful in producing the correct data set. As the SNR was a fairly large value, noise on the received signal should not have much of an effect on the phase estimates produced by the PLL.

Figures 6.20a and 6.20b plot the estimated DOA and associated quality metric data resulting from the phase data plotted in Figure 6.19. The estimated DOA was consistent over the observed time duration and the quality metrics show that the measured first difference curve produces acceptable results. Unfortunately this was not repeatable. Array imperfections as well as possible mutual coupling between elements distorted the estimated phase curves for other DOAs used in testing. As the DOA of the target transmitter was changed, the deviations in the measured phases from the expected values were not constant or repeatable, and therefore they became impractical to easily take into account in the DSP software. This means that the main contribution of the work on

the 16-element algorithm was the theoretical development and the design of the new switching control circuit.

Chapter 7

Conclusion

This thesis has provided an overview of the development of a single-channel algorithm from the design of the initial basic version through its implementation on a software-defined receiver. This final chapter will first summarize the main contributions of this work to the algorithm's development. Second, technical challenges and problems faced in the implementation of the algorithm will be discussed. Lastly, possible future directions for this project to follow will be discussed.

7.1 Summary of Results

The basic version of the PLL DF algorithm as presented in Chapter 3 is primarily an adaptation of classic single-channel DF concepts for use with digital modulation schemes and implementation on a software defined receiver. Past single-channel DF systems were designed to work with analog modulated signals and the implementation relied heavily on custom analog hardware. This algorithm broke with that tradition by requiring a simple antenna array and RF hardware but powerful and flexible DSP hardware.

Chapter 4 discussed the main contributions of this work in the form of modifications to the PLL algorithm to overcome various challenges encountered in the initial implementation of the algorithm. First, the nonlinear Costas PLL used to estimate the phase of the signal received by each antenna element was changed to a linear Costas PLL. This change did not introduce any apparent differences in the phase estimation process but was intended to reduce the computational burden imposed by the PLL on the DSP hardware. The nonlinear PLL required numerous multiplications and calls to a CORDIC algorithm function in its processing loop. The linear PLL eliminated those requirements by performing the exact same functions with simple addition and subtraction. Second, a modification of the PLL to estimate and remove a frequency offset while preserving the differential phase relationships between the signals tracked by each PLL was introduced. Third, a new curve fit algorithm was developed to allow the algorithm to operate with a 16-element antenna

array without requiring an exponential increase in computational complexity that would otherwise result from scaling up the original curve-fit algorithm which was designed for an 8-element array. Finally, a DOA estimate quality metric was established to aid in the filtering of erroneous DOA estimates produced by the DF system in low SNR conditions.

Chapter 5 presented the simulation analysis of the performance of the algorithm in an AWGN channel with the various algorithm modifications. Simulations of the algorithm with varying PLL gain showed that as PLL gain decreases and the PLL loop filter consequently decreases the RMS error of the DOA estimates decreases due to reductions in the variance of the PLL phase output error. The simulation of the frequency offset estimator showed that the introduction of the estimator introduces a noticeable increase in the RMS error for the 8-element algorithm but the 16-element algorithm's performance does not similarly degrade because the 16-element curve-fit algorithm already introduces an inherent error overhead due to the sensitivity of the curve-fit algorithm to noise in the phase estimates. The inclusion of lowpass and MA filtering on the output DOA estimates does provide a considerable reduction in the RMS error of both the 8 and 16-element algorithms. Due to the nature of the two algorithm's respective curve-fit processes, it was found that lowpass filtering has a greater effect on the 8-element algorithm whereas MA filtering has the greatest effect on the 16-element algorithm. In the end, both the 8 and 16-element PLL DF algorithms were found to have very good performance in an AWGN channel.

Chapter 6 presented the implementation of the DF algorithm on an SDR platform with a custom design 8-element uniform circular antenna array and RF switching circuit. The implementation of the modifications to the 8-element algorithm were described and the results were quite favorable. The switch to a linear PLL provided a drastic reduction in processing requirements while the frequency offset estimator performed its intended purpose and the resulting RMS error reasonably conformed to expected values determined by simulation. The implementation of the 16-element algorithm required both a new antenna array and switching circuit. A new FPGA-controlled circuit was designed that accepted digital control signals from the SDR as well as manual controls from a bank of switches. Unfortunately, numerous imperfections in the sizing and construction of the antenna array prevented the 16-element algorithm from performing well. As expected, the algorithm required less processing power than the 8-element algorithm to perform its computations but inconsistencies in the phase of the received signal due to problems with the

manufacturing of antennas and possible mutual coupling between antenna elements did not allow the curve-fit algorithm to reliably converge to the expected first difference curve. In the end, the 16-element algorithm may prove to not be a viable path for future development due to the difficulties in manufacturing an antenna array with a large number of elements for use in the UHF band. In fact, the antenna array is the most difficult part of the implementation of most DF systems due to the sensitivity of RF hardware to slight differences in manufacturing processes over time.

7.2 Future Work

As this DF system is primarily implemented in software as a DSP algorithm, the addition of future adaptations and modifications to the signal processing layer is limited only by the practical limitations of signal processing techniques and DSP hardware abilities. Work on the PLL DF algorithm can extend it for use with different modulation schemes and to attempt to combat the problems arising from DOA estimation in fading channels. As far as modulation is concerned, this algorithm was designed to work only with BPSK modulated or unmodulated target signals. The curve-fit algorithm in particular bases its operation on the assumption that BPSK modulation is present. Higher order PSK modulation schemes as well as other digital modulation types such as QAM will present problems relating to the method of carrier recovery employed. The Costas PLL used in this algorithm can be very easily modified to accommodate higher-order PSK modulation, but the ambiguities on the estimated carrier phase for each antenna will definitely increase the complexity of the curve-fit algorithm. For example, the estimated carrier phase of a QPSK modulated signal can have ambiguity of $n\pi/4$, where $n \in \{0,1,2,3\}$. This will lead to a dramatic increase in the permutations of the measured first difference data used to resolve the ambiguities by the MSE-comparison method for the 8-element algorithm. It may turn out that the 16-element curve-fit algorithm will not work as the amplitude of the second difference of the PLL output approaches the amplitude of the phase ambiguities. A generalized version of the algorithm that can handle many digital modulation schemes will represent an attractive option for a modern single-channel DF system. Furthermore, modifications to the algorithm that allow for an antenna array that differs from the theoretical model would definitely make it a general purpose tool.

Fading channels and other harsh RF environments also present numerous problems that may be insurmountable for a single-channel system. Intelligent filtering and statistical modeling of

the received signal may provide innovative solutions to the problem but may also lead to dead-ends for the single-channel system.. One way in which this work can continue is by increasing the scope of the project to involve the design and implementation of existing multi-channel DF algorithms on SDR platforms with reduced channel counts. Initial work on the adaptation of the MUSIC algorithm for use with a two-channel receiver and multi-element antenna array [19] shows great promise. The ability of the MUSIC algorithm to spatially resolve multiple received signals in fading channels is well known but has typically required a one-to-one receiver-to-antenna ratio as well as significant processing power. An efficient implementation of the algorithm with a small two-channel receiver that can approach the performance of the full-blown multi-channel algorithm represents a significant leap forward in DF technology.

7.3 Final Thoughts

This thesis proved to be a valuable experience. Not only was it theoretically challenging, but it was also technically challenging in that it required the inclusion of real-world concerns as implementation of the algorithm was the main goal. Being involved with the project from basic algorithm design to the design of antenna switching hardware and implementation as a real system was very rewarding. Often times research work can seem divorced from reality in that the assumptions relied upon to investigate any problem may not always be valid and the violation of them can have serious implications for the viability of the work in a real implementation. While closely related, industry and academia can ask very different questions of this work. Whereas a professor might be more interested in the theoretical underpinnings of a DF system, an engineer in the field will always be concerned with how well it performs in practice. This project attempted to satisfy both and in that attempt was an invaluable learning experience not only because of the successes but also because of the failures. It was the perfect primer for a career as an engineer in the wireless industry.

References

[1] J. J. Keaveny, "Analysis and Implementation of a Novel Single Channel Direction Finding Algorithm on a Software Radio Platform," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 2005.

[2] N. Harter, J. J. Keaveny, S. Venkatesh, and R. M. Buehrer, "Analysis and Implementation of a Novel Single-Channel Direction Finding Algorithm," *Proc. Of 2005 IEEE Wireless Communications and Networking Conference*, Vol. 4, pp 2530-2533, March 2005.

[3] N. Harter, J. J. Keaveny, S. Venkatesh, and R. M. Buehrer, "Analysis and Implementation of a Novel Single-Channel Direction Finding Algorithm," DRS Signal Solutions Technical Symposium, May 2005.

[4] N. Harter, J. J. Keaveny, S. Venkatesh, and R. M. Buehrer, "Development of a Novel Single-Channel Direction Finding Algorithm," *Proc. Of 2005 IEEE Military Communications Conference*, Vol. 4, pp 2720-2725, October 2005.

[5] J. C. Liberti and T. S. Rappaport, *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications*, Upper Saddle River: Prentice Hall PTR, 1999.

[6] RF Products, Web Note WN-002 "Basics of the Watson-Watt Radio Direction Finding Technique."

[7] D. Peavey and T. Ogunfunmi, "The Single Channel Interferometer Using a Pseudo-Doppler Direction Finding System," *Proc. Of 1997 IEEE Conference on Acoustics, Speech, and Signal Processing*, Vol. 5, pp 4129-4132, April 1997.

[8] RF Products, Web Note WN-004 "A Comparison of the Watson-Watt and Pseudo-Doppler DF Techniques."

[9] R. S. Smith, H. Anderson, and L. Jugler, "Correlative Vector Direction Finding," Watkins-Johnson Technical Symposium, Gaithersburg, MD, 1995.

[10] L. Jugler, "Direction Finding Techniques for TDMA Signals," DRS Signal Solutions Technical Symposium, Gaithersburg, MD, 2005.

[11] R. O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *Proc. of RADC Spectrum Estimation Workshop*, Griffiss AFB, NY, pp. 243-258, 1979.

[12] R. O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *Proc. of IEEE Trans. On Antennas and Propagation*, Vol. AP-34, No. 3, Mar. 1986.

[13] R. Xu, "Evaluation of MUSIC for Position Location in a Cellular System with Multipath," M.S. Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1996.

[14] Costas, J.P., "Synchronous Communications," *Proceedings of the IRE* , vol.44, no.12pp.1713-1718, Dec. 1956.

[15] L. W. Couch, *Digital and Analog Communications Systems 6th Edition*, Upper Saddle River, NJ: Prentice Hall, 2001.

[16] W. H. Tranter, K. S. Shanmugan, T. S. Rappaport, and K. L. Kosbar, *Principles of Systems Simulation with Wireless Applications*, Upper Saddle River, NJ: Prentice Hall PTR, 2004.

[17] R. B. Ertel, "Antenna Array Systems: Propagation and Performance," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, 1999.

[18] R. G. Lyons, *Understanding Digital Signal Processing*, Upper Saddle River, NJ: Prentice Hall PTR, 1997.

[19] N. Harter, "Superresolution Direction Finding with the MUSIC Algorithm," DRS Signal Solutions Technical Symposium, Gaithersburg, MD, 2006.

Vita

Nathan Harter was born in Maryville, Illinois on December 29, 1980. He attended Althoff Catholic High School in Belleville, IL and later enrolled at Parks College of Engineering and Aviation at Saint Louis University in the Fall of 1999. While at Saint Louis University, he pursued a Bachelor of Science in Electrical Engineering degree, graduating summa cum laude.

After graduation from Saint Louis University, Nathan enrolled at Virginia Polytechnic Institute and State University in the Fall of 2003, where he was awarded the Bradley Graduate Research Fellowship by the department of Electrical and Computer Engineering. While pursuing his Master of Science degree he became a member of the Mobile and Portable Radio Research Group. During his time at MPRG he focused on direction finding systems and the implementation of communication systems on software defined radios. His research interests include wireless communications, direction finding algorithms and their implementation, software defined radio, and communication system simulation. He defended his master's thesis on Friday, April 13th, 2007.

After his time with MPRG, Nathan accepted a position with the Direction Finding and Cellular Systems Group at DRS Signal Solutions in Gaithersburg, Maryland where his work involved the design and implementation of DF and beamforming systems and the design and testing of SDR hardware and algorithms. He is currently a Systems Engineer at G3 Technologies, Inc. in Mt. Airy, Maryland. His work at G3 focuses on cellular networks and on the design of direction finding and intelligence systems. Nathan now lives outside Frederick, Maryland with his wife Sarah and their portly cat Sylvia. His hobbies include reading, playing the guitar, snowboarding, cooking, and biking.