Summer 7-2007

Source Localization within a Uniform Circular Sensor Array

Danny Zhu
danny.z.zhu@gmail.com

7-1-2007

Source localization within a uniform circular sensor array

Danny Zhu

SOURCE LOCALIZATION WITHIN A UNIFORM CIRCULAR SENSOR ARRAY

by

Danny Z. Zhu

A Thesis Submitted

in

Partial Fulfillment

of the

Requirements for the Degree of

MASTER OF SCIENCE

in

Electrical Engineering

Approved by:

PROF. _____
(Dr. Sohail Dianat – Advisor)

PROF. _____
(Dr. Jayanti Venkataraman – Co-Advisor)

PROF. _____
(Dr. Vincent Amuso – Committee Member / Department Head)

DEPARTMENT OF ELECTRICAL ENGINEERING

COLLEGE OF ENGINEERING

ROCHESTER INSTITUTE OF TECHNOLOGY

ROCHESTER, NEW YORK

JULY 2007

Preface

Not too long ago, while attending my last Military Ball at RIT, the guest speaker, Colonel George M. Bilafer emphasized to the future lieutenants of the United States Army before him what he considered to be the most important thing in his career: people. Before I graduate, I would like to do the same by acknowledging the contributions of many people, without whom I would not have prevailed. First and foremost are my parents, James and Kathy, and my sister, Helene, whom have raised and nurtured me to become the young man I am today. No amount of tuition, good grades or publications could ever amount to the hard work and unconditional support they have shown me. For that I am eternally thankful and indebted.

In the five years I spent away from home, I would not have succeeded in many of my academic endeavors without some sense of guidance and mentorship. For providing their continued aid, support and mentorship, I thank my academic advisor, Dr. Sohail Dianat, whom always provided truly valuable insight at every interaction, and my co-advisor, Dr. Jayanti Venkataraman, for her excellent advice and lab facilities. Both of these professors played pivotal roles in assisting with my research; without them, this publication would not have been possible. Furthermore, I would like to thank Dr. Richard Reeve and Dr. Vincent Amuso, who were indispensable in providing me the guidance and insight I needed to succeed as an engineering student.

Moreover, for their tireless efforts in supporting the students of the Kate Gleason College of Engineering, I would also like to recognize Ken Snyder and Jim Stefano for their material support and valuable technical knowledge. Additionally, I would also like

to thank the RIT Honors Program, Student Government, and Office of Student Affairs for providing travel grants to attend the IEEE 2007 Antenna Propagation Society International Symposium. For their superb equipment and support of my research, I would also like to recognize Agilent and Hewlett Packard for loaning equipment to the department of Electrical Engineering.

Also among those I wish to recognize are my peer engineering students, whom I hold in highest regard: Brian Argauer, David Blonski, Eric Ernst, Andrea Wyant and Christopher Urban. You are some of the finest and most intelligent engineers I have met; thank you for being the ambitious and hardworking people you are—many of you were the focal point of my inspiration and motivation and I wish you all the best in your future endeavors.

To the non-engineering teachers and professors whom have made my time at RIT truly worth while, I would like to thank: Dr. Sean Sutton (Political Science), Dr. David Suits (Philosophy), Dr. Carl Lutzer (Mathematics), and Dr. Michael Richmond (Physics). You are all among the best professors RIT has to offer, and it has been a privilege to be a pupil in your classes, even if it was only one.

Finally, I would also like to thank the Department of the Army and Cadet Command for extending my scholarship benefits to accommodate the schedule of the BS/MSEE program, through the support of Lieutenant Colonel Donald Beattie Jr., Lieutenant Colonel Dale Watson, and Command Sergeant Major (R) Gary Mastroleo. They have done more than teach me the importance of setting up your soldiers for success; they have shown me.

Abstract

Traditional source localization problems have been considered with linear and planar antenna arrays. In this research work, we assume that the sources are located within a uniformly spaced circular sensor array. Using a modified Metropolis algorithm and Polak-Ribière conjugate gradients, a hybrid optimization algorithm is proposed to localize sources within a two dimensional uniform circular sensor array, which suffers from far field attenuation. The developed algorithm is capable of accurately locating the position of a single, stationary source within 1% of a wavelength and 1˚ of angular displacement. In the single stationary source case, the simulated Cramer-Rao Lower Bound has also shown low noise susceptibility for a reasonable signal to noise ratio. Additionally, the localization of multiple stationary sources within the array is presented and tracking capabilities for a slowly moving non-stationary source is also demonstrated. In each case, results are presented, analyzed and discussed. Furthermore, the proposed algorithm has also been validated through hardware experimentation. The design and construction of four microstrip patch antennas and a wire antenna have been completed to emulate a circular sensor array and the enclosed source, respectively. Within this array, data has been collected at the four sensors from several fixed source positions and fitted into the proposed algorithm for source localization. The convergence of the algorithm with both simulated data and data collected from hardware are compared and sources of error and potential improvements are proposed.

Table of Contents

List of Abbreviations

B	Magnetic Flux Density
BM	Birgin Martinez
CG	Conjugate Gradient
CRLB	Cramér Rao Lower Bound
CW	Continuous Wave
DF	Direction Finding
DNA	Deoxyribo Nucleic Acids
DOA	Direction of Arrival
E	Electric Field
ESPRIT	Estimation of Signal Parameters via Rotational Invariance Techniques
f	Frequency
FIM	Fisher Information Matrix
FR	Fletcher Reeves
GA	Genetic Algorithms
GPS	Global Positioning System
H	Magnetic Field
HFSS	High Frequency Structure Simulator
Hz	Hertz
HPBW	Half Power Beam Width
IEEE	Institute of Electrical and Electronics Engineers
MATLAB	Matrix Laboratory
MSE	Mean Squared Error
MUSIC	Multiple Signal Classification
PR	Polak Ribière
RMS	Root Mean Squared Error
SA	Simulated Annealing
SDMA	Space Division Multiple Access
SNR	Signal to Noise Ratio
SOLT	Short Open Load Thru
SWR	Standing Wave Ratio
T	Temperature
VSWR	Voltage Standing Wave Ratio

List of Tables

List of Figures

1 Introduction

1.1 Motivation

Traditional source localization problems have been considered with linear and planar antenna arrays, but do not involve the attenuation of signals as a function of distance in their mathematical models. Under these realistic conditions, the arrays described suffer from signal attenuation for sources that are far away, thereby making parameter estimation difficult using current algorithms. Additionally, signals considered by these algorithms are often impinging on the arrays externally, however the sensor geometry considered in this paper focuses on sources enclosed by the array. As a result of this, subspace methods such as Multiple Signal Classification (MUSIC) [1] and Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) [2] can not be applied. Using the proposed circular sensor array, signals emitted far from one sensor are still close to an opposing sensor. Thus, even under range attenuating conditions, it is possible to estimate and localize the spatial parameters of the source.

Furthermore, the proposed localization algorithm does not rely on time delay techniques for spatial estimation, as is required in multi-dimensional trilateration. In trilateration, three or more known points, and their corresponding distances from the receiver are required. These distances are calculated by determining the delay involved in receiving radio waves. However, the difficulty in this calculation is ensuring accuracy in timing. Global Positioning System (GPS) satellites, for example, are equipped with atomic clocks to stay in sync with one another. Not only is this expensive, but group

velocity dispersion may occur during wave propagation, making temporal resolution, and hence estimation of spatial parameters, even more difficult. For these reasons, a new algorithm will be presented as a less complex approach to source localization.

1.2 Objectives

Goals of this research are to develop an optimization algorithm which can efficiently localize a stationary source placed in the far field and enclosed in a circular sensor array. Additionally, methods to localize multiple fixed sources are explored and proposed, and the potential of tracking a source slowly moving along an azimuthal trajectory is identified. Localization of an enclosed source in a uniform circular sensor array is simulated via a hybrid heuristic optimization technique implemented in MATLAB, and shown in hardware as a proof of concept.

1.3 Literature Review

As a subset of signal processing, the history of array processing has had many uses throughout the years, particularly in the field of communications. An excellent review of such applications is provided in [3]. With recent developments in adaptive antennas (also known as smart antennas) [4], array processing has been of great interest in wireless communications and networking applications. In this context, array processing has first been used for spatial filtering or beamforming [5]. This lead to the eventual development of adaptive beamforming [6]-[8] and time delay estimation techniques [9] to improve the resolution of proximate signals. Among these applications is also the detection of objects, such as land mines [10], tracking energy sources that are moving in space, and use of Space-Division Multiple Access (SDMA) in cellular

networks. While other signal parameters such as frequency can be estimated using Capon's Maximum Likelihood Estimate (MLE) [11] and Pisarenko's harmonic decomposition [12], the focus of this thesis is on the estimation of spatial signal parameters from a signal enclosed by a circular sensor array.

In the estimation of signal parameters, unbiased estimates of the number of signals, directions of arrival (DOA), strengths and cross correlations among waveforms, polarizations and noise/interference strength are generally of interest [1]. Previously published research on DOA and direction finding (DF) applications can be categorized into conventional, subspace, maximum likelihood, and integrated techniques [13]. Conventional methods include delay-and-sum beamforming, which has poor spatial resolution, and is highly dependent on the directivity of the antenna array as well as SNR. Subspace methods such as MUSIC and ESPRIT exploit the structure of the received data. In the former algorithm, the noise and signal subspaces can be separated by eigendecomposition, subsequently allowing the spatial parameters of the signal to be determined with great resolution. However, the disadvantages of this method are its reliance on accurate modeling of the antenna array, and the costly search of that array. Additionally, it requires that the number of sources impinging on the array is less than the number of sensors in the array. On the other hand, ESPRIT offers an improved resolution over the modal MUSIC algorithm and reduces computation costs significantly. Maximum likelihood methods exploit the statistical parameters of the data to determine the DOA as in [14]; these techniques tend to be computationally intensive though they perform better in low SNR environments.

1.4 Problem Formulation

Consider a sensor array consisting of M sensors, uniformly spaced on the boundary of a circle with radius r (normalized to wavelength, λ) and angle θ_m. Each sensor captures signals transmitted from I stationary sources with unknown polar coordinates, $\{r_i, \phi_i\}_{i=1}^{I}$ within the array.

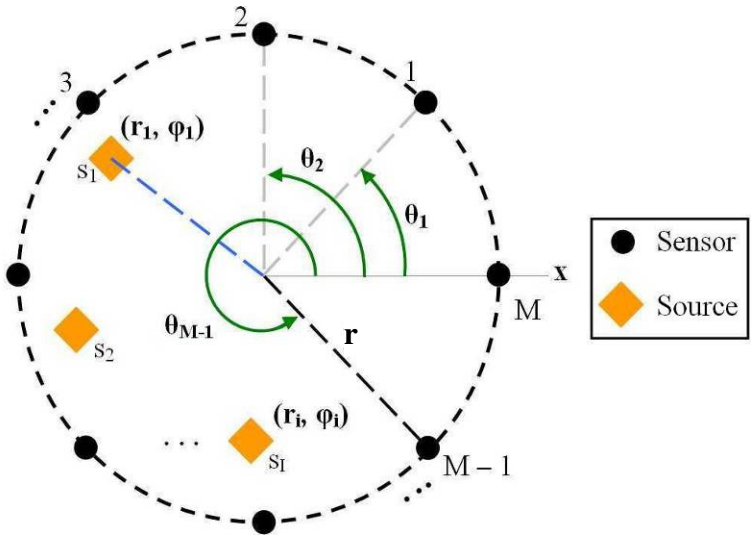


Figure 1.1. Uniform circular sensor array with M sensors and I sources.

Each source is assumed to be in the far field or Fraunhofer region with respect to the sensors of the array. To meet this criteria, the following equation must be satisfied:

$$R > \frac{2D^2}{\lambda} \tag{1-1}$$

where R represents the minimum radial distance required between the source and the sensor, D represents the largest length of the source antenna, and λ represents the free space wavelength. For an isotropic source, which propagates omni-directionally and uniformly in free space, the wavefronts of an emitted traveling wave can be represented as a sphere with radius R. This is significant because the level of signal attenuation as

seen at the receiver is proportionate to the distance from the emitter. In the far field zone, all wavefronts at a distance R from the emitter appear as uniform plane waves, and the major and minor lobes of antenna directivity patterns are well formed. Under these circumstances, the received signal at the m^{th} sensor, $x_m(k)$, is given by:

$$x_m(k) = \left[\sum_{i=1}^{I} s_i(k) \frac{e^{-j\frac{2\pi}{\lambda}\sqrt{r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_m - \varphi_i)}}}{\sqrt{r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_m - \varphi_i)}} \right] + W_m(k) \tag{1-2}$$

where λ is the wavelength, $W_m(k)$ is the zero mean additive white Gaussian noise at the m^{th} sensor and $s_i(k)$ is the narrowband signal transmitted by the i^{th} source given by:

$$s_i(k) = A_i e^{j(\omega_0 k + \psi_i)} \tag{1-3}$$

where A_i is the signal amplitude and $\{\psi_i\}_{i=1}^{L}$ is the signal phase assumed to be independent, and identically distributed with a uniform distribution between $-\pi$ and π. The cross correlation between signals detected at the m^{th} and n^{th} sensor is defined as:

$$r(m,n) = E\left[x_m(k) x_n^*(k)\right] \tag{1-4}$$

Under the noise free case, the cross correlation function is given by:

$$r(m,n) = \sum_{i=1}^{I} \sum_{l=1}^{I} E\left[s_i(k) s_l^*(k)\right] \frac{e^{-j\frac{2\pi}{\lambda}\sqrt{r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_m - \varphi_i)}}}{\sqrt{r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_m - \varphi_i)}} \frac{e^{j\frac{2\pi}{\lambda}\sqrt{r^2 + r_l^2 - 2r \cdot r_l \cos(\theta_n - \varphi_l)}}}{\sqrt{r^2 + r_l^2 - 2r \cdot r_l \cos(\theta_n - \varphi_l)}} \tag{1-5}$$

where the cross correlation between sources is given by $E\left[s_i(k) s_l^*(k)\right] = A_i A_l E\left[e^{-j(\psi_i - \psi_l)}\right]$. Since $A_i A_l E\left[\cos(\psi_i - \psi_l) + j \sin(\psi_i - \psi_l)\right]$ becomes 1 or 0 depending on i and l, $E\left[s_i(k) s_l^*(k)\right] = \begin{cases} 0 & i \neq l \\ A_i^2 & i = l \end{cases}$, and (1-5) reduces to:

$$r_s(m,n) = \sum_{i=1}^{I} \frac{A_i^2 e^{-j\frac{2\pi}{\lambda}\left[\sqrt{r^2+r_i^2-2r\cdot r_i\cos(\theta_m-\varphi_i)}-\sqrt{r^2+r_i^2-2r\cdot r_i\cos(\theta_n-\varphi_i)}\right]}}{\sqrt{\left[r^2+r_i^2-2r\cdot r_i\cos(\theta_m-\varphi_i)\right]\left[r^2+r_i^2-2r\cdot r_i\cos(\theta_n-\varphi_i)\right]}} \tag{1-6}$$

The actual covariance matrix can be written as:

$$r(m,n) = r_s(m,n) + \sigma^2\delta(m,n) \tag{1-7}$$

In practice, $r(m,n)$ is not available and must be estimated from the observed data. Assuming that N data samples are collected at each sensor, the cross correlation is estimated using an ensemble data time averaging and is given by:

$$\hat{r}(m,n) = \frac{1}{N}\sum_{k=1}^{N} x_m(k)x_n^*(k) \tag{1-8}$$

where * denotes a complex conjugate. To find the location of the source, the normalized mean squared error (MSE) between the true and estimated cross correlation function is formed. The normalized MSE is given by:

$$J(r_i,\varphi_i) = \frac{\sum_{m=1}^{M}\sum_{n=1}^{M}\left|r(m,n)-\hat{r}(m,n)\right|^2}{\sum_{m=1}^{M}\sum_{n=1}^{M}\left|\hat{r}(m,n)\right|^2} \tag{1-9}$$

For M sensors and $k = 1,2,3,...,N$ data samples, an M x N matrix can be generated:

$$X = \begin{bmatrix} x_1(1) & x_1(2) & \cdots & x_1(N) \\ x_2(1) & x_2(2) & \cdots & x_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_M(1) & x_M(2) & \ldots & x_M(N) \end{bmatrix} \tag{1-10}$$

and (1-8) becomes:

$$\hat{r}(m,n) = \frac{1}{N}X^H X \tag{1-11}$$

where ^{H} denotes Hermitian transpose. For a high signal-to-noise ratio (SNR), the non-linear surface defined by (1-9) contains minima at the location of the sources. Therefore, to localize them, the normalized MSE cost function must be minimized with respect to (r_i, φ_i). As shown below, for sensor arrays with small radii, this is trivial.

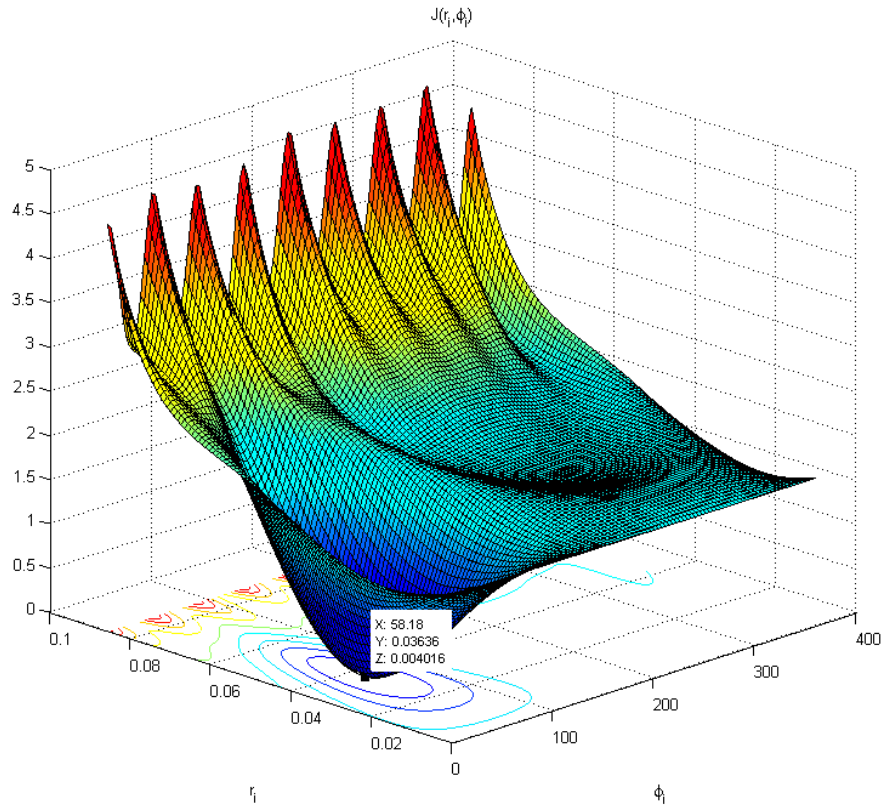


Figure 1.2. Normalized MSE performance surface of a 20 dB SNR source for r = λ.

In this case, to determine the location of these minima, typical minimization algorithms, such as steepest descent, Newton's method and conjugate gradients, discussed in detail later, are sufficient. In general, each of these methods uses basic calculus principles, which require the partial derivatives of (1-6) with respect to r_i and φ_i to be calculated to minimize the function. Using the following substitutions, a closed form expression for the instantaneous gradient of (1-6) can be found:

$$A = r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_m - \varphi_i) \qquad B = r^2 + r_i^2 - 2r \cdot r_i \cos(\theta_n - \varphi_i)$$

$$dA_{r_i} = \left[2r_i - 2r\cos(\theta_m - \varphi_i)\right] \cdot dr_i \qquad dB_{r_i} = \left[2r_i - 2r\cos(\theta_n - \varphi_i)\right] \cdot dr_i \tag{1-12}$$

$$dA_{\varphi_i} = -2r \cdot r_i \sin(\theta_m - \varphi_i) \cdot d\varphi_i \qquad dB_{\varphi_i} = -2r \cdot r_i \sin(\theta_n - \varphi_i) \cdot d\varphi_i$$

$$\frac{\partial r(m,n)}{\partial r_i} = \sum_{i=1}^{I} \frac{A_i^2 e^{-j\frac{2\pi}{\lambda}\left[\sqrt{A}-\sqrt{B}\right]}}{2\sqrt{AB}} \left[-j\frac{2\pi}{\lambda}\left(\frac{dA_{r_i}}{\sqrt{A}} - \frac{dB_{r_i}}{\sqrt{B}}\right) - \left(\frac{dA_{r_i}}{A} + \frac{dB_{r_i}}{B}\right)\right] \tag{1-13}$$

$$\frac{\partial r(m,n)}{\partial \varphi_i} = \sum_{i=1}^{I} \frac{A_i^2 e^{-j\frac{2\pi}{\lambda}\left[\sqrt{A}-\sqrt{B}\right]}}{2\sqrt{AB}} \left[-j\frac{2\pi}{\lambda}\left(\frac{dA_{\varphi_i}}{\sqrt{A}} - \frac{dB_{\varphi_i}}{\sqrt{B}}\right) - \left(\frac{dA_{\varphi_i}}{A} + \frac{dB_{\varphi_i}}{B}\right)\right] \tag{1-14}$$

Unfortunately, a sensor array with a radius of one wavelength is impractical. Suppose the radius is increased from λ to 6λ; the performance surface would then look like:

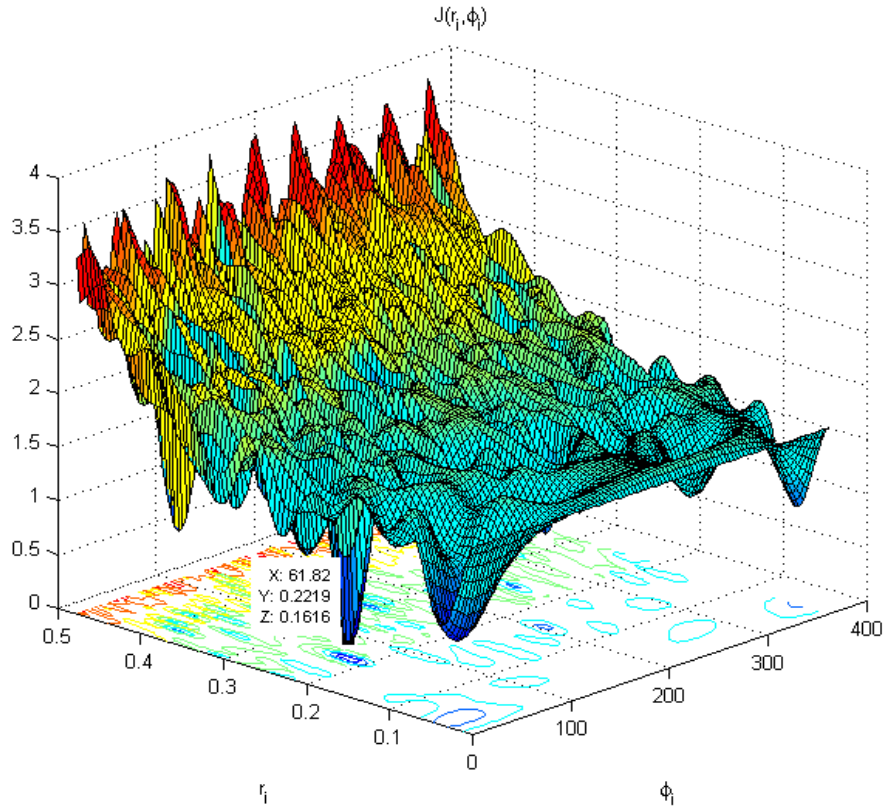


Figure 1.3. Normalized MSE performance surface of a 20 dB SNR source for r = 6λ.

In this case, the locations of several minima will cause the minimization approaches mentioned earlier to converge to a sub-optimal solution. In the next chapter, the theory behind these approaches is reviewed and heuristic techniques to solve the problem of source localization are introduced.

2 Optimization Techniques

2.1 Overview

Several types of optimization problems exist, however this chapter will review non-linear constrained optimization and various techniques that apply to it. For the problem formulated in Section 1.4, particular interest is placed in the minimization of the multivariate normalized MSE cost function in (1-9). There are several approaches that can be used to accomplish this; each technique presented carries its own advantages and disadvantages, however, typical criteria used to evaluate the quality of optimization algorithms are speed, cost and memory use. For example, while it is possible to find the minimum value of the normalized MSE cost function by calculating it for every combination of r_i and φ_i, for large scale problems where the solution space is enormous, this approach would be slow, and require much memory for storage.

To review, the simplest function to minimize is one of quadratic form. In $\Re^2$, this is represented by a parabola, and in $\Re^3$, a circular paraboloid. The goal then, is to reach the bottom of the bowl as quickly and accurately as possible. However, in most algorithms, these two criteria are typically inversely related. That is, the tradeoff for an algorithm which converges to the minima faster is a less accurate result. Likewise, algorithms which take longer to converge to the minima are slower, but more accurate. To better illustrate the quadratic form, the contour plot of a quadratic function with a minimum at $\{x_1, x_2\} = (-2,2)$ taken from [23] is shown in Figure 2.1 on the following page:

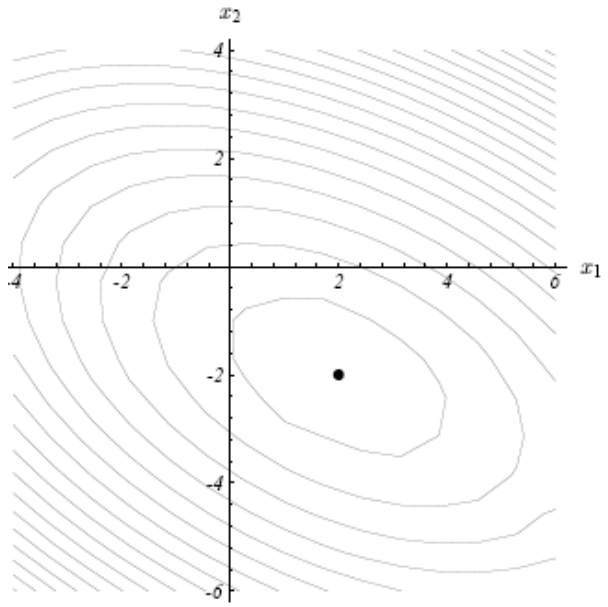


Figure 2.1. Contour plot of the quadratic form.

Note that each elliptical curve of the contour plot represents a constant height or $f(x)$ value. The corresponding gradient directions of the function in Figure 2.1 are shown below:

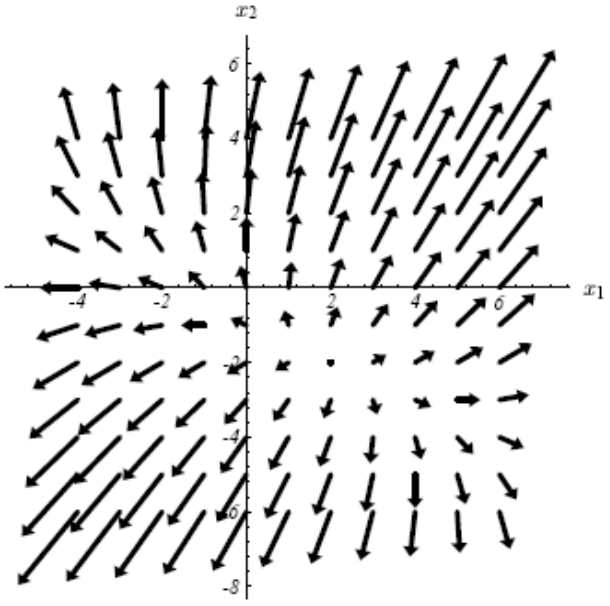


Figure 2.2. Corresponding gradients of the quadratic form.

These gradient directions point in the direction of steepest increase of $f(x)$, with the steepness represented by the magnitude of the vectors. It is also significant to recognize that the orientation of the vectors is orthogonal to the contour lines shown in Figure 2.1.

2.2 Newton's Method

Also known as the Newton-Raphson method, this technique is a root-finding algorithm, which also has applications in optimization. Newton's method starts by calculating the line tangent to a function at a particular point, and then determining the root of that line (where it crosses the x-axis), using the zero-crossing of the tangent line as the next iteration in the process. This process continues until the function evaluated at the current value of x becomes zero. This can be better visualized in Figure 2.3 below:

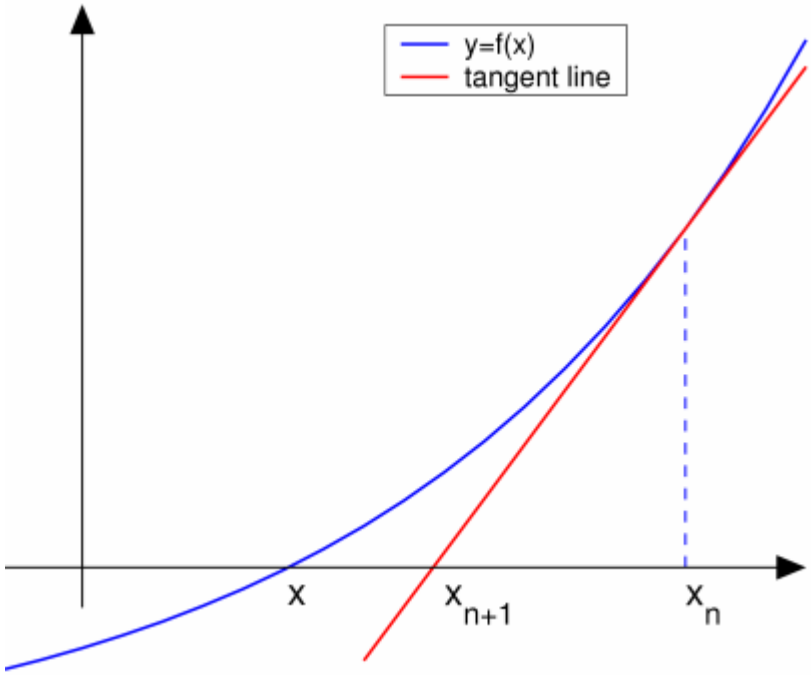


Figure 2.3. Newton-Raphson algorithm for root-finding.

The corresponding equation governing Newton's method for root finding at the k-th iteration is given by

$$x_{k+1} = x_k + \frac{f(x)}{f'(x)} \tag{2-1}$$

However, in the optimization sense, the objective is not root-finding, but rather minimization. From elementary calculus, it is known that the minimum of a quadratic function occurs at the critical point of the function, where the derivative is equal to zero. Therefore, this algorithm can be applied to the derivative of a quadratic cost function, thereby minimizing it. Similar to (2-1), the corresponding iterative equation for this algorithm is given by

$$x_{k+1} = x_k + \frac{f'(x)}{f''(x)} \tag{2-2}$$

Notice that the derivative of the cost function must be available and the cost function itself must be twice differentiable. Alternately, the second derivative of the function may be approximated by using the first derivative. Under these conditions, so long as the cost function is quadratic, this method will always minimize it. Unfortunately, in many practical applications, this assumption does not always hold true as seen in the performance surfaces of Figure 1.2 and Figure 1.3.

2.3 Steepest Descent

Similar to Newton's method, the method of steepest descent also utilizes the derivative of the cost function. In multi-dimensional space, this derivative is referred to as the gradient. From Figure 2.2, it can be seen that the direction of steepest descent is taken to be opposite that of the gradient. The arrows represented there are projections

onto the z-plane of the rate of increase of the function; notice that at the minimum, no projection exists. As in Newton's method, this algorithm exploits the fact that the minimum of the objective function is discovered when $\nabla_k = 0$. The difference in this algorithm from Newton's method is that the function is traversed along the direction of the gradient, as in

$$x_{k+1} = x_k + \mu(-\nabla_k) \tag{2-3}$$

The selection of the step-size, μ, is significant because it will determine the effectiveness of the search process. Typically, small values will converge slower but more accurately, while larger values will converge more rapidly but less accurately. From [15], it is known that in the univariate optimization case, the stability and rate of convergence are governed by:

Stability	Step Size	Geometric Ratio
Stable	$0 < \mu < \frac{1}{\lambda}$	$\lvert r \rvert < 1$
Stable, Overdamped	$0 < \mu < \frac{1}{2\lambda}$	$0 < r < 1$
Stable, Critically damped	$\mu = \frac{1}{2\lambda}$	$r = 0$
Stable, Underdamped	$\frac{1}{2\lambda} < \mu < \frac{1}{\lambda}$	$-1 < r < 0$
Unstable	$\mu \le 0$ $\mu \ge \frac{1}{\lambda}$	$\lvert r \rvert > 1$

Table 2-1. Effect of μ on convergence of one-dimensional gradient searches..

where λ represents the eigenvalues of the objective function, and the geometric ratio, r, is defined as

$$r = 1 - 2\mu\lambda \tag{2-4}$$

2.4 Conjugate Gradients

The technique of conjugate gradients uses concepts also seen in the method of steepest descent, however it utilizes multiple search directions. As in previous cases, the minimum of the line occurs when the gradient is orthogonal to the search direction. Applications of these techniques have been prevalent in engineering design, neural networks, and nonlinear regression [23]. As in Newton's method and the method of steepest descent, the method of conjugate gradients also requires that the cost function is continuously differentiable and the gradient, which can be calculated from (1-13) and (1-14), exists. The variations in search directions explored in this publication are the Fletcher-Reeves (FR), Polak-Ribière (PR) and Birgin-Martinez (BM) [24], [25]. For more variations, the reader is referred to an excellent overview of conjugate gradients, provided in [26].

In this algorithm, the surface of the cost function is traversed in a specific direction, which changes at each iteration, and is given by

$$d_{k+1} = r_{k+1} + \beta_{k+1} d_k \tag{2-5}$$

where β is determined by the variation of conjugate gradients being used (shown later), and r represents the residual or the direction of steepest descent, defined as

$$r_k = -f'(x_k) = -\nabla_k \tag{2-6}$$

The key to the residuals and the directions in this algorithm are that each consecutive residual and search direction is orthogonal to all previous ones, in order to eliminate searching in a direction that has already been exhausted. Next, a line search is performed along the direction of (2-5) using

$$x_{k+1} = x_k + \alpha_k d_k \tag{2-7}$$

where the step size, α_k, is found to minimize $f(x_k + \alpha_k d_k)$ using any style of line search. Some line searches available to use include quadratic or cubic interpolation, golden section searches, Fibonacci search, and so forth. Recall from Section 2.3 that the residual at the minimum of $f(x_k + \alpha_k d_k)$ will be orthogonal to the search direction, and hence will have no projected component onto the search direction, as shown below:

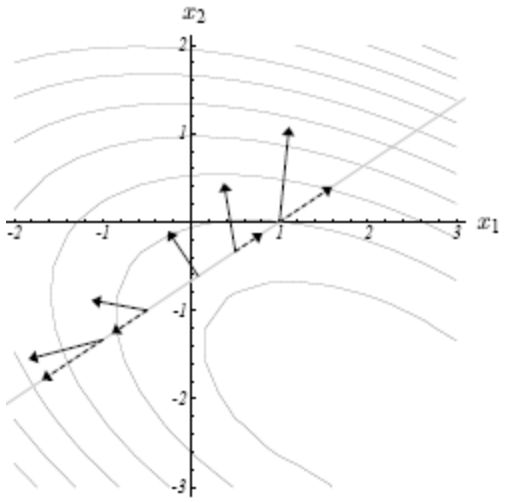


Figure 2.4. Gradients and their projections along a search direction.

Referring back to (2-5), the variants of β corresponding to the FR, and PR methods are given by

$$\beta_{k+1}^{FR} = \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k} \tag{2-8}$$

$$\beta_{k+1}^{PR} = \frac{r_{k+1}^T (r_{k+1} - r_k)}{r_k^T r_k} \tag{2-9}$$

While both require the second derivative of the function to be minimized, the Fletcher-Reeves method also requires the unnecessary and complex calculation of an inverse

Hessian matrix. Thus, for this particular application, the Polak-Ribière technique will be applied. Alternately, in the BM spectral conjugate gradient method, the search direction is computed by

$$d_{k+1}^{BM} = -\theta_{k+1} g_{k+1} + \beta_k^{BM} s_k \tag{2-10}$$

where

$$\beta_k^{BM} = \frac{\left(\theta_{k+1} y_k - s_k\right)^T g_{k+1}}{y_k^T s_k} \tag{2-11}$$

$$\theta_{k+1} = \frac{s_k^T s_k}{y_k^T s_k} \tag{2-12}$$

$$s_k = x_{k+1} - x_k \tag{2-13}$$

$$y_k = g_{k+1} - g_k \tag{2-14}$$

$$g_k = \nabla f\left(x_k\right) \tag{2-15}$$

Regardless of the search direction variant, repeated trials showed that convergence of the conjugate gradient method was highly dependent on the initial guess. Thus, for a performance surface similar to the one described in Figure 1.3, being "trapped" by local minima is typical. This was expected, since the method of conjugate gradients is known not to be globally convergent—in fact, it may not even be locally convergent if no lower bound exists [23]. As mentioned previously, this becomes an issue with sensor arrays of practical sizes, which prompted the use of more viable, heuristic methods.

2.5 Heuristic Methods

By now, it should be apparent that the optimization techniques previously discussed are sufficient for simple problems with local minima, but struggle in situations

where multiple minima exist. As shown in Section 1.4, for $r > \lambda$ and one source, multiple minima occur in the performance surface of (1-9). In a practical networking application using IEEE 802.11b frequencies, the free space wavelength, λ_0 is approximately 12.25 cm. Therefore, for practical applications, $r >> \lambda$, and hence many local minima will exist, making the previous techniques not immediately feasible. Nevertheless, at the location of the source, the surface described by the normalized MSE is ideally zero—a global minimum. Thus, localization is still possible, albeit much more sophisticated. The use of heuristic approaches discussed in this section, will allow large, non-linear and non-convex optimization problems to be solved.

2.5.1 Genetic Algorithms

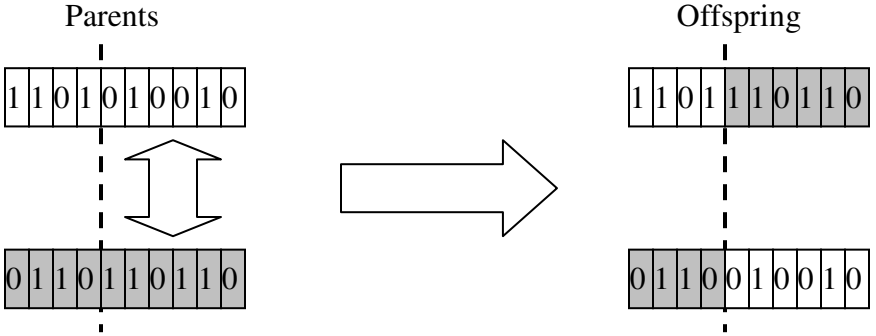
Genetic Algorithms (GA) fall under the category of Evolutionary Algorithms. As the name suggests, these types of algorithms simulate the evolution of individual structures through selection, mutation and reproduction [27]. Each process depends on the performance of each evolution in a pre-defined environment. The algorithm itself models the science of human evolution. Each individual has unique genetic characteristics, known as Deoxyribonucleic Acids (DNA). This DNA comes in the form of chromosomes, which are structured as double helixes. Each segment of this structure represents a specific characteristic, such as eye color or skin tone for humans. For different organisms, a different number of characteristics or chromosome segments exist. For example, frogs have 26 chromosomes (13 pairs), whereas humans have 46 chromosomes (23 pairs). The significance of the pairing is that each individual organism

is a product or offspring of two other individuals (parents), also with unique traits. After mating, traits from each parent are combined and passed on to the offspring, considered to be the next generation or evolution. However, some traits are naturally dominant, while others are recessive, which dictates the likelihood of passing on to future generations. Furthermore, each trait carries with it a possibility of mutation in passing to the offspring. In GA, the user can establish the likelihood of traits passing as well as the probability of mutation by assigning a fitness level to each individual.

The fitness level for each individual structure or solution is a key element of GA; it determines the quality of the evolution and is analogous to the normalized MSE cost function previously described, though a larger fitness level is usually desired over a smaller one. The individual with the highest fitness is generally considered the optimum solution, and is desired. This fitness level, like the normalized MSE cost function, is a function of some parameters—or the variables we are optimizing with respect to. In combinational optimization, several parameters may exist. For example, in the design optimization of an antenna by GA, the antenna dimensions are considered, as well as its performance parameters, like half power beam width (HPBW) or perhaps maximum gain and directivity. In this application of source localization, the parameters of interest are spatial ones: namely, (r_i, φ_i). For these parameters, a bit resolution must be defined. For example, the first five bits might represent the r parameter, while the next five might be used to represent the φ parameter. In this particular example, the bit size would accommodate $2^5 = 32$ different values of r. The resulting ten bit string would then represent the chromosome of one possible solution.

In the first evolution, multiple solutions are randomly attempted and their corresponding fitness levels are evaluated. Among this first evolution, a mating pool is created such that individuals with the highest fitness are more likely to be included, while individuals deemed unfit are less likely to be accepted. One implementation of this is the binary tournament method, where candidates are selected pairwise, and the individual with the higher fitness is selected into the pool. More customizability is available to the user here by determining the size of each tournament as well as implementing a probability of selecting the higher fitness individual for mating in each pairwise selection.

Once the mating pool reaches the same size as the population, it will then be used to generate the next evolution, by randomly pairing mates together and mixing their traits to create offspring. The combination of traits from both parents can be chosen in several ways, and is referred to as "crossover." The probability of crossover in the mating pool is typically selected to be between 0.6 and 1.0; some mates will trade traits, while others will not and will be passed into the next phase as is. This portion of the algorithm allows the search space to be broadly traversed, ensuring that the solution will not be trapped by local minima. Different types of crossovers exist, including single point, 2-point and uniform crossover.



(a)

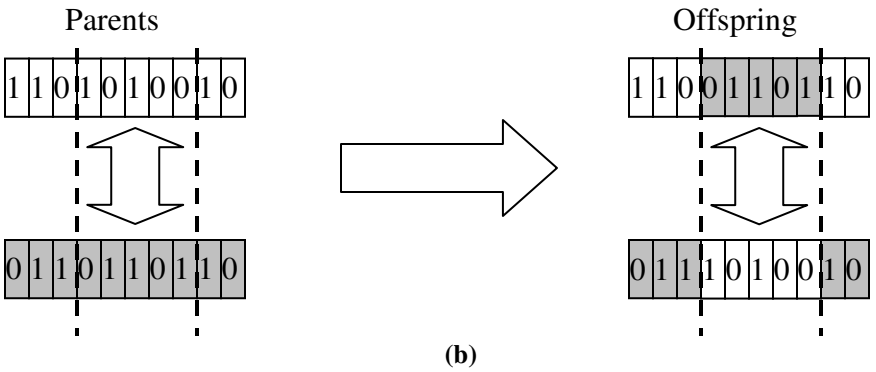


Figure 2.5. Example of (a) single point, and (b) two-point crossovers.

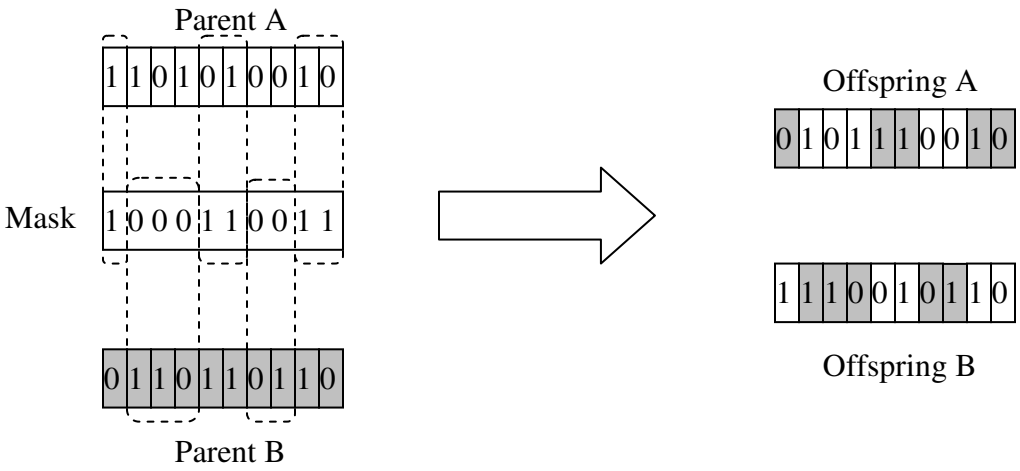


Figure 2.6. Example of a uniform crossover using a crossover mask.

Once crossover has taken place, mutation occurs; in this process, a string of bits is assigned a user defined probability of inverting. As in crossover, mutation allows breadth of search, however as evolutions progress, mutation will not be limited by the traits of the population (whereas crossover would). This process reiterates until the termination conditions are met, which are usually when an individual with high enough fitness is produced or after a certain number of generations has been completed.

2.5.2 Metropolis Algorithm

The Metropolis Algorithm, first introduced in [28], is an algorithm which has its roots in statistical mechanics and the process of annealing. It was first used in [29] as a viable solution to the famous "traveling salesman" problem, and is commonly referred to as Simulated Annealing (SA). In metallurgy and materials science, the process of annealing involves heating a material in order to change its structural properties, namely strength and hardness. The analogy used in optimization is this: if a liquid substance cools and anneals too quickly, its atoms solidify into a sub-optimal arrangement—that is, one with an energy level that is greater than the minimum or ground state. Alternately, if the liquid cools gradually, the atoms will solidify into the optimum energy state. This energy state corresponds to the normalized MSE cost function, as in (1-9), that needs to be minimized. The cost function, however, is only one of the key components required to effectively implement a Metropolis Algorithm. Other elements include having system configurations, perturbations in the configurations, and a control parameter, T, which is analogous to temperature and used in an annealing schedule. In this application, the system configurations are defined by various (r_i, φ_i) values. Similarly, the perturbations to the system are made randomly within a constrained subset of positions, $(\Delta r_i, \Delta \varphi_i)$, as determined by the cross correlation matrix of (1-8). Some popular applications of this algorithm are in physical design of computers, and optimization of integrated circuit layouts (i.e. optimal placement and wiring of components).

First, an initial system configuration is selected and the cost function is evaluated. This configuration is then rearranged until a new configuration that reduces the cost

function is identified. This rearranged configuration becomes the starting point of the next iteration and the process continues until no further improvements can be found. During each step of the process, the cost of the configuration is evaluated. If the change in energy, $\Delta E = E_{k+1} - E_k$, experienced is less in the new state, the perturbation is accepted. However, to facilitate momentum in the search space, if the energy level increases in the new arrangement, it is accepted with the following probability:

$$P(\Delta E) = e^{-\Delta E / k_B T} \tag{2-16}$$

where $k_B = 1.38x10^{-23}\, J/_{\circ K}$ represents the Boltzmann constant. This probability can easily be implemented through generating a uniformly distributed random variable between 0 and 1 at each iteration and comparing it to the probability. Note that (2-16) will reduce as the control parameter, T , decreases, as illustrated in [30]:

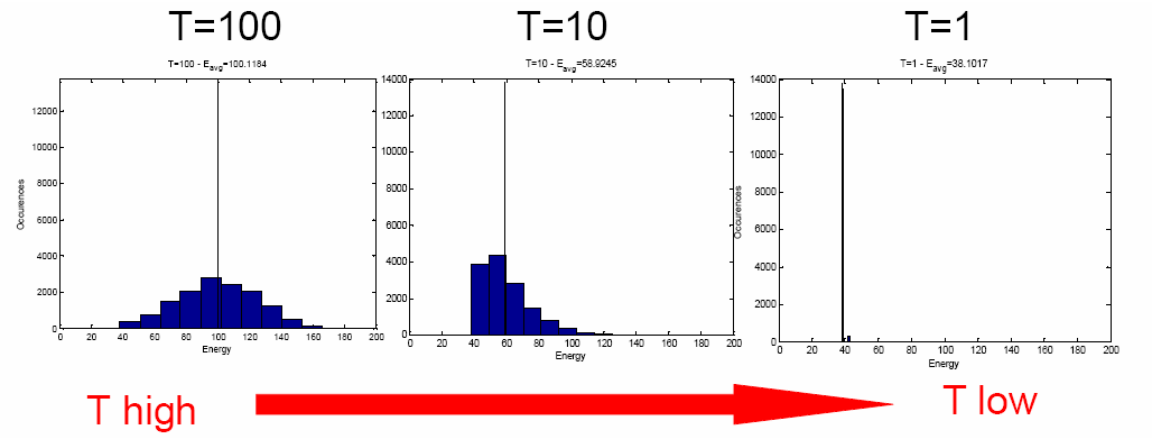


Figure 2.7. Boltzmann probability as temperature ranges from high to low.

How quickly it reduces is dictated by the next integral element: the annealing schedule. The annealing schedule dictates the rate of change of the temperature, which controls the probability to accept an uphill move. This allows the algorithm to avoid getting

"trapped" by being able to traverse out of local minima on the normalized MSE performance surface described by (1-9). After perturbation and cost evaluation, the temperature is reduced and the configuration is perturbed again. This process continues until a user defined stop condition is met. To date, several readings suggest that an exponential decay of temperature over time typically yielded the best results, however an interesting paper on improving annealing schedules can be found in [31]**.** For a more detailed summary on SA, the reader is referred to [32] and an outline of MATLAB implementation in [33].

More recently, an updated SA algorithm has been proposed in [34] to expedite the convergence of the original Metropolis algorithm. In this version, the parameter space is constrained, a "valve value" for the inner loop is given, and inner loop memory is also incorporated. For this application of source localization, the search space is reduced to the area contained by the central angle between two adjacent sensors. Thus, the more sensors there are in the circular array, the smaller the search section. These additions are summarized in the following flowchart, taken from the same paper:

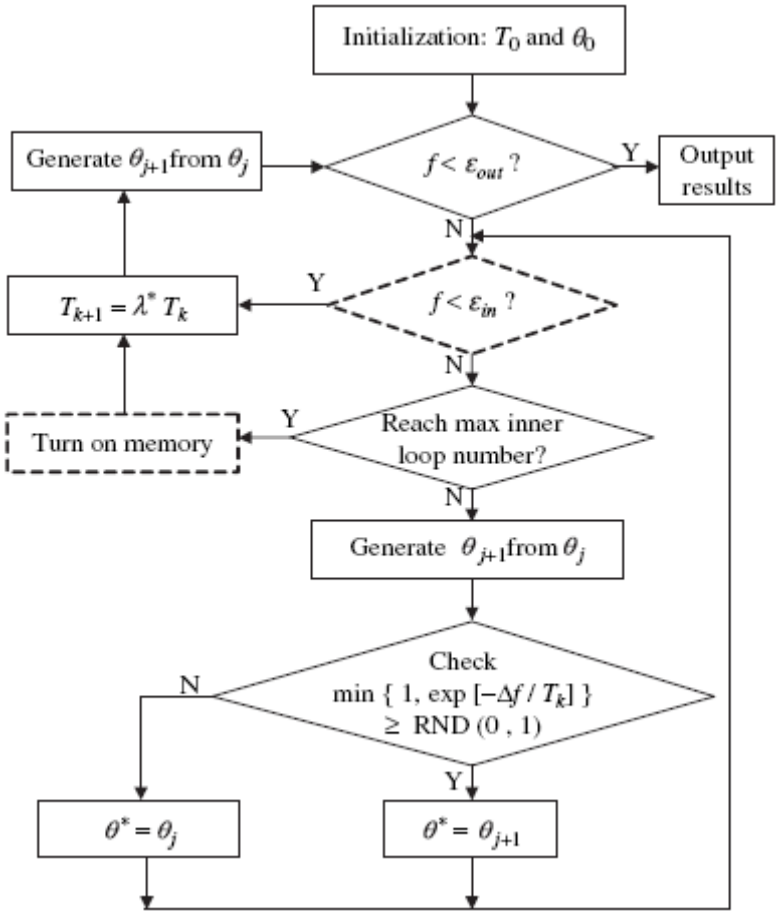


Figure 2.8. Flowchart of modified simulated annealing algorithm.

Here, the solid lines represent the original algorithm while the dashed lines represent the added features of the revised algorithm. Note that θ_j represents the j-th configuration, and f represents the objective cost function. The control parameter, T, remains the same, but is reduced by a factor of λ for $0 < \lambda < 1$ during each annealing schedule step (to avoid confusion with wavelength, this step size will be referred to as α herein). The inner and outer valve values are represented by ε_{in} and ε_{out} respectively. Once the cost function reaches either of these values, the loop will halt execution. Another feature of this algorithm makes use of Markov chains, which are the set of energy costs evaluated at

each temperature step. As shown in Figure 2.8, a maximum Markov chain length (the number of perturbations, j, within each k-th temperature state) is used as a stopping condition for the inner loop.

2.6 Summary

In this chapter, a brief review of several optimization algorithms has been reviewed. For this particular application of source localization, a construction method of heuristics will be used to localize the stationary user enclosed by the circular sensor array. First, a feasible solution in the vicinity of the global minimum will be found through the modified simulated annealing algorithm of Figure 2.8. Once a solution in the vicinity of the true location is determined, the conjugate gradient algorithm will be used to more closely resolve the true location. In the next chapter, it will be shown through simulation that application of this hybrid-heuristic approach will effectively locate the source within 0.01λ and $1°$ from its true position.

3 Simulation Results

3.1 Algorithm

It was shown in Section 1.4 that convergence to the source location by conjugate gradients is highly dependent upon the initial conditions for radial and angular displacement; for an initial point in the vicinity of the source, convergence will always occur. Therefore, an "educated guess" approach that relies on the known cross correlation approximation in (1-8) is adopted to determine the starting position of the radius and angle.

First, the sensor where the maximum cross correlation occurs is identified, and its adjacent sensors are analyzed. Because the sensor positions are known, an excellent estimate of what quadrant the source resides in can be provided in this manner. By identifying the adjacent sensors as boundaries, the unnecessary and costly searching of other quadrants of the array is avoided, hence constraining the parameter space as in Figure 2.8. Additionally, because the SA method allows breadth and depth in searching, the initial radius and angle are selected to be half of the array radius at the angle of the highest correlated sensor. Using these initial conditions, modified SA with a linear temperature decrement is applied.

From the starting position, the cost function is evaluated and stored. Next, a random, in bound perturbation of the configuration is made. The cost function is re-evaluated at the new perturbed value and compared with the previous cost. Downhill moves are always accepted, however, if the new perturbed configuration is at a higher

cost than the previous one, it is only accepted with a Boltzmann probability, as a function of the current temperature. This is evaluated by generating a random number and comparing it with the probability at the current temperature. The process is repeated until the maximum Markov chain length is exceeded, or until the inner loop valve value is reached. Under these conditions, the temperature is then reduced by a specified fraction; as the temperature approaches zero, the probability of accepting uphill moves decreases. Alternately, the temperature can also decrease if the algorithm reaches a predefined number of downhill cost function moves; in this case, the algorithm is said to have reached equilibrium at that temperature. This evaluation process continues until the number of iterations is exhausted, at which point the lowest cost configuration is determined to be the solution.

Once an acceptable solution is discovered, minimization code adopted from [16] is used to apply the PR conjugate gradient directions in (2-9). First, the function and its derivative are evaluated at the end point of SA, and then the search begins in the direction of steepest descent (opposite to the gradient). During each search, quadratic interpolations are performed to minimize the cost function along the search direction. There, the function and derivative are re-evaluated and a new search begins with the updated Polak-Ribière conjugate gradient direction. Finally, to improve precision, strict Wolfe-Powell conditions that limit the ratio of the calculated to previous slopes and the slope of the minima are applied. In the next section, the algorithm parameters used during simulation testing are defined. All code is made available in Appendix A.

3.2 Test Parameters

The following parameter values are used in simulation testing to describe the circular sensor array, source position, signal strength, and sampling scheme:

Parameter	Value	Units
M	8	sensors
I	1	sources
R	6λ	m
A	1	V
N	10000	samples
SNR	40	dB
r_i	$0.3R$	m
φ_i	$\pi/3$	rads

Table 3-1. Simulated source and sensor parameters.

For $M = 8$ sensors with a large SNR, unit signal amplitude and a radius of six wavelengths, a source position with its radius normalized to the sensor array is selected. Because the sensor array radius, r, is normalized to wavelength, the array size can be scaled and still detect a source, so long as the position satisfies $0 \le r_i \le R$. The next set of parameters describes the general conditions used in the modified SA algorithm. Recall that the initial guess is made at the angular position of the sensor with the greatest signal strength and at half of the radius.

Parameter	Value	Description
α	0.6	Temperature step size
SW	40	Step width (in iterations)
EQ	4	Energy decrements to equilibrium
ε_{in}	0.1	Inner valve value
ε_{out}	0.1	Outer valve value
T_0	1000	Initial temperature
K	800	Iterations
MCL_{max}	5	Markov Chain Length

Table 3-2. Simulated annealing parameters.

Using modified code from [16], the application of Polak-Ribière Conjugate Gradients is done with the parameters shown in Table 3-3 below.

Parameter	Value	Description
σ	1 x 10^{-8}	Convergence coefficient
ρ	5 x 10^{-9}	Convergence coefficient
MAX	250	Maximum function evaluations
MAX_RATIO	9.85	Maximum consecutive slope ratios
MAX_INT	0.1	Maximum interpolation limit
MAX_EXT	3.0	Maximum extrapolation limit

Table 3-3. Conjugate gradient parameters.

As stated at the end of Chapter 1, the tolerance used in these simulations before being considered "converged" is within a radial distance of 0.01λ and angular displacement of $1°$. Additionally, any false errors due to dual representations of the same source position, i.e. $\left(0.3R, \pi/3\right) = \left(-0.3R, 4\pi/3\right)$, are filtered out.

3.3 Single Source

In the single source case, rapid convergence to a well defined global minimum occurs. Using a source position of $(r_1, \varphi_1) = (0.3R, 60^\circ)$, the source is angularly located between the second $(\theta = 45^\circ)$ and third $(\theta = 90^\circ)$ sensors, but closer to the second sensor. Thus, the second sensor will have the highest cross correlation term and the initial guess used in the SA algorithm will be $(\hat{r}_1, \hat{\varphi}_1) = (0.5R, 45^\circ)$. From here, a series of random perturbations or configurations, bounded by the adjacent sensors will be created and accepted as the new solution based on the temperature and cost function. As the temperature gradually reduces, as shown in the following annealing schedule, the probability of accepting new solutions that are higher in cost decreases. Over a period of 800 iterations, the temperature decays to zero, and an acceptable solution is found.

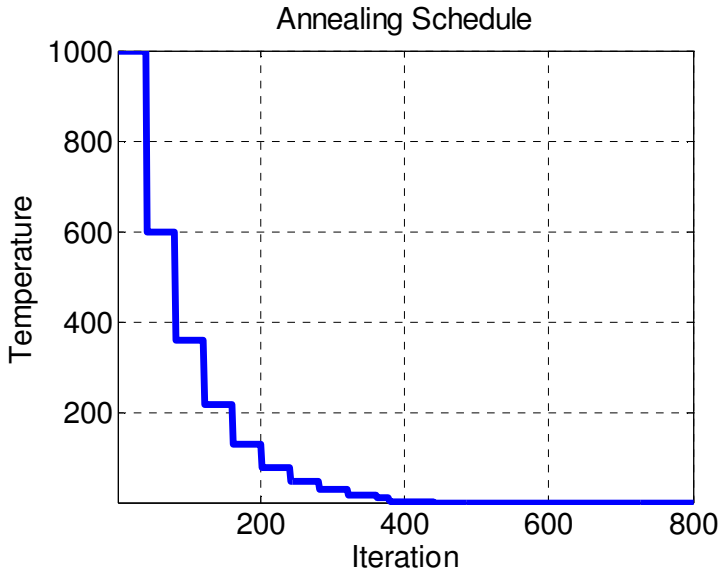


Figure 3.1. Annealing schedule used to globally minimize J(r,φ).

While this solution is not the exact position of the source, it is close enough to the true minimum to allow the conjugate gradient algorithm to more closely resolve the source position. Figure 3.2 below shows the lowest cost configurations attempted by SA.

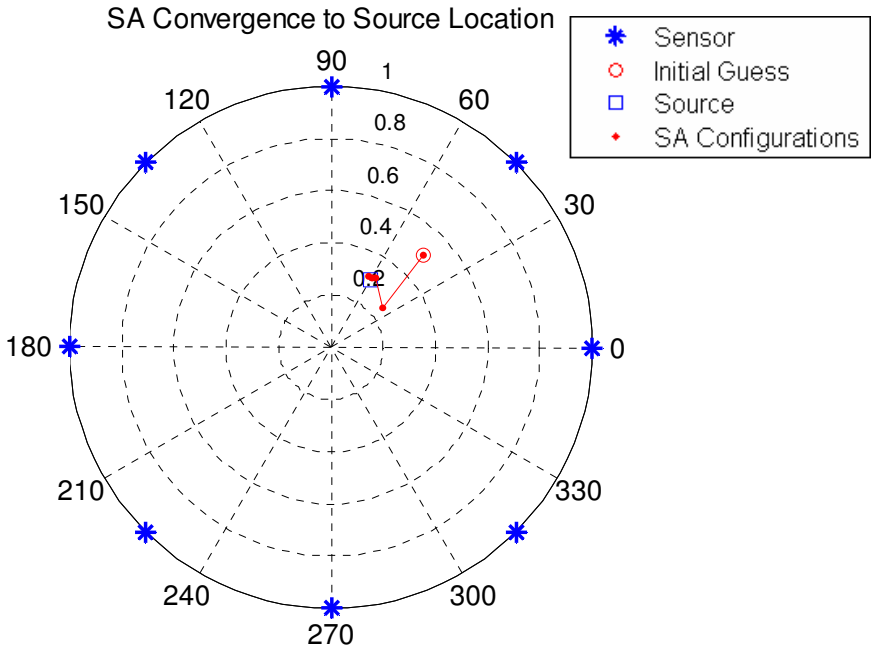


Figure 3.2. Constellation of simulated annealing solutions, normalized to radius.

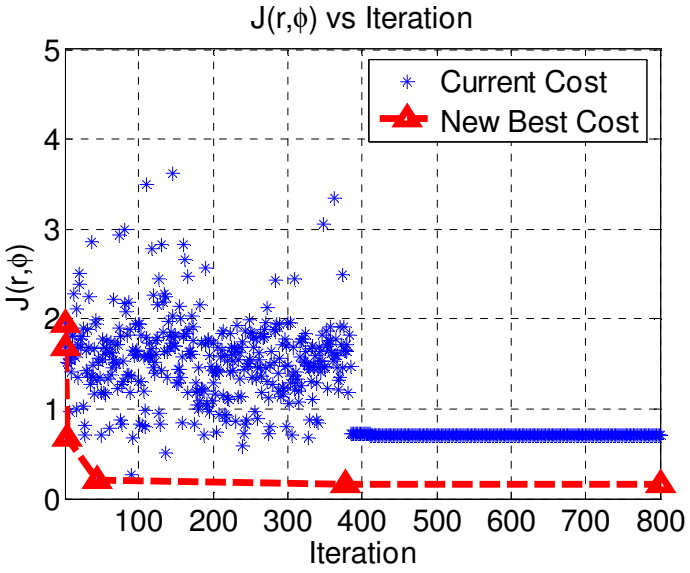


Figure 3.3. Learning curve used to minimize J(r,φ) by simulated annealing.

In Figure 3.3, each asterisk represents the cost value evaluated at each iteration. Similarly, the dotted line is the memory used to track the best overall performance of the algorithm. From this figure, it is evident that as the temperature decreases, the perturbations are gradually reduced until an acceptable, low cost solution is reached.

In the next set of figures, the performance of the conjugate gradient approach is analyzed in terms of radial and angular convergence. As mentioned in Section 2.4, because the CG approach is largely dependent on initial conditions, an acceptable solution from the SA algorithm is used as the new starting point. From Figure 3.4, it can be seen that the SA solution generated was only 3.3% away from the true source radius. Similarly, the graph of Figure 3.5 shows an initial angular displacement, determined by SA only 1.1% from the true location. Moreover, the convergence of both parameters is very rapid, concluding in fewer than six steps. The final solution is less than 0.0004% and 0.0011% away from the true location of the source.

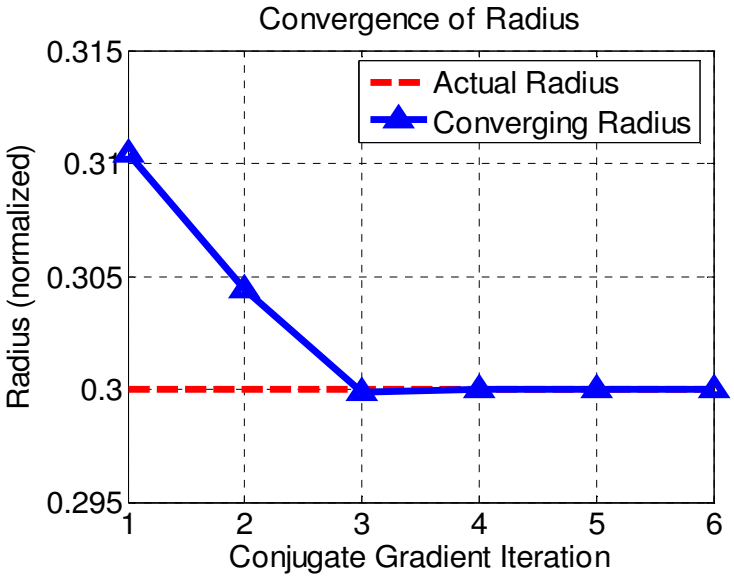


Figure 3.4. Convergence of radius by conjugate gradients using final SA solution.

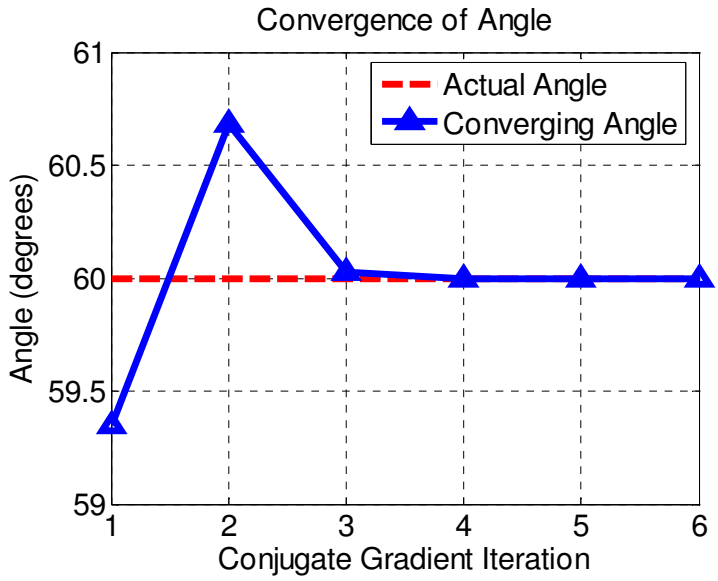


Figure 3.5. Convergence of angle by conjugate gradients using final SA solution.

3.4 Cramér Rao Lower Bound (Single Source)

The Cramer Rao Lower Bound (CRLB) is an upper bound on the variance of error for any unbiased estimator and is defined as the inverse of the Fisher Information Matrix (FIM) [35]. The closed form expression for the CRLB for a single source is related to the second derivative of the log-likelihood function, which is given by $\frac{\partial^2 \log\left(p(x_m(k)) \mid r_1, \varphi_1, \psi_1\right)}{\partial r \partial \varphi}$ where

$$p(x_m(k) \mid r_1, \varphi_1, \psi_1) = \frac{1}{2\pi\sigma^2} \frac{1}{2\pi} \int_{-\pi}^{\pi} e^{-\frac{1}{2\sigma^2}\|x_m(k)-\mu_m(k)\|^2} d\psi_1 \tag{3-1}$$

$$= \frac{1}{4\pi^2\sigma^2 M} \exp\left(-\frac{\|x_m(k)\|^2 + |\mu_m(k)|^2}{2\sigma^2}\right) \times \int_{-\pi}^{\pi} \exp\left(\frac{A_1|x_m(k)|}{\sigma^2} \cos\left(\frac{2\pi}{\lambda}\sqrt{r^2 + r_1^2 - 2rr_1\cos(\theta_m - \varphi_1)} - (w_0 k + \psi_1) - \angle x_m(k)\right)\right) d\psi_1 \tag{3-2}$$

and

$$\mu_m(k) = A_1 e^{j(w_0 k + \psi_1)} \frac{e^{-j\frac{2\pi}{\lambda}\sqrt{r^2 + r_1^2 - 2rr_1\cos(\theta_m - \varphi_1)}}}{\sqrt{r^2 + r_1^2 - 2rr_1\cos(\theta_m - \varphi_1)}} \tag{3-3}$$

However, the closed form solution for the log-likelihood of the data for this application does not exist. As an alternative, the simulated CRLB is considered. The variance of the error for both r and φ, are defined by

$$\sqrt{\frac{1}{K}\sum_{i=1}^{K}(\hat{r}_i - r)^2} \tag{3-4}$$

$$\sqrt{\frac{1}{K}\sum_{i=1}^{K}(\hat{\varphi}_i - \varphi)^2} \tag{3-5}$$

where K represents the number of runs, and results are plotted against the SNR in dB for $K = 1000$ runs. This was performed over nine values of SNR, linearly spaced from $-20\ dB$ to $20\ dB$ for a fixed source location of $(r_1, \varphi_1) = \left(0.3R, \pi/3\right)$. According to the simulated CRLB plots shown in Figure 3.6 and Figure 3.7, the proposed algorithm does not appear to be highly susceptible to noise and clutter.

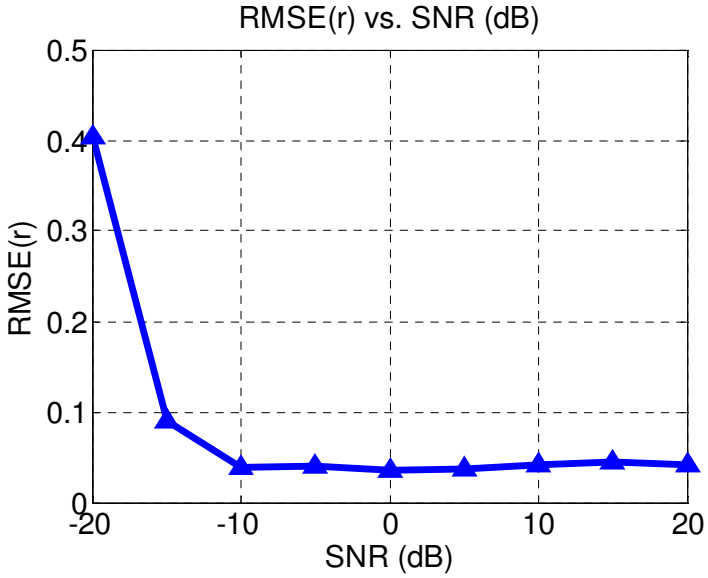


Figure 3.6. RMSE of radial estimation of one fixed source.

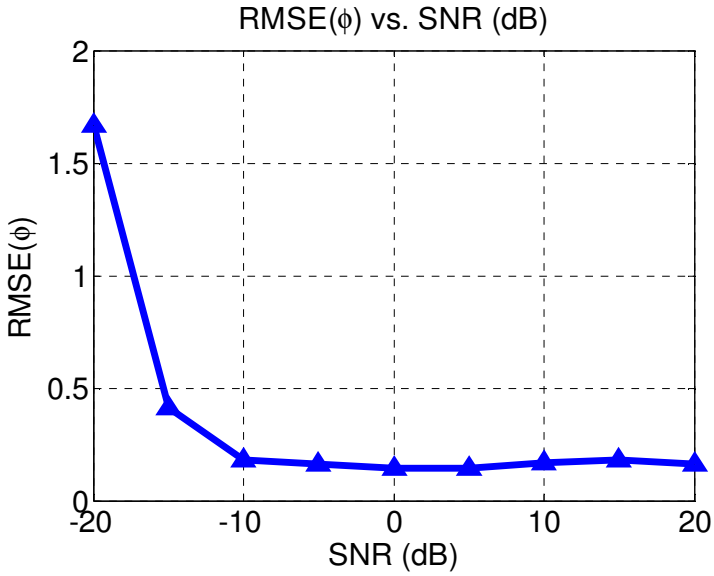


Figure 3.7. RMSE of angular estimation of one fixed source.

As expected, the error is reduced with larger SNR, suggesting that the estimator used is statistically efficient; in fact, the RMSE displays asymptotic behavior as early as $-10\ dB$. While Figure 3.6 and Figure 3.7 behave similarly, the majority of the error can be expected in the angular estimation of the source position.

3.5 Multiple Sources

The localization of single stationary sources has proven to be effective, but in practical applications, there is a high likelihood for the presence of multiple sources. When applying the proposed algorithm in an environment with multiple sources enclosed in the sensor array, the predominant minimum of the performance surface will occur at the location of the source with the largest SNR. Thus, this method will always localize the source with the strongest SNR. To localize another source, the normalized MSE performance surface must be modified to present an optimum solution at that source. Recall that the input data matrix is represented by a summation of the signals presented at each sensor by all of the sources as in (1-2). Thus, for $I > 1$ sources, the contribution of each source, including the signal of (1-3) and the attenuation factor associated at each sensor, must be decoupled from the data in order to be able to localize all sources.

However, the hybrid SA-CG localization algorithm yields only spatial information, and therefore, an estimate of the attenuation factor at each sensor, but no knowledge of the signal itself. Additionally, because all sources are assumed to operate at the same frequency, no orthogonality exists between sources, making it more difficult to eliminate each term in the summation of (1-2). Therefore, in order to cancel the contribution of the highest SNR signal from the data captured at the m-th sensor, a priori

knowledge of the signal must be available—including amplitude, frequency, and phase information. Otherwise, the contribution of the strongest signal must be removed by some other means. One method this can be accomplished is through adaptive sidelobe cancellation [6]-[8]. By temporarily identifying the high SNR signal to be an interfering signal and placing a null in its direction, only the remaining enclosed sources will be considered in the input covariance matrix and cost function. Alternately, a simpler approach which only requires the input covariance matrix will be shown.

In this method, the number of sources enclosed in the array is assumed to be known. In practice, several reliable methods for estimating the number of sources present are available [17]-[20], which generally involve eigendecomposition of the input covariance matrix. As in [1], the signal and noise subspaces are orthogonal and can be separated in this manner. Subsequently, in the signal subspace, it can be shown that a number of eigenvalues equal to the number of enclosed sources will be present in the covariance matrix. An obvious limitation to this is that the number of sensors M must be greater than the number of enclosed sources, I, however successive removal of each source can reveal all eigenvalues, as long as enclosed sources have a high SNR. Given the number of enclosed sources, a recursive algorithm to remove the contribution of the strongest source to the cost function, and hence localize multiple sources is proposed in Figure 3.8, which includes an optional adaptive beamformer with the intent of placing temporarily placing nulls in the direction of sources that have already been localized in order to generate a new estimate for the covariance matrix.

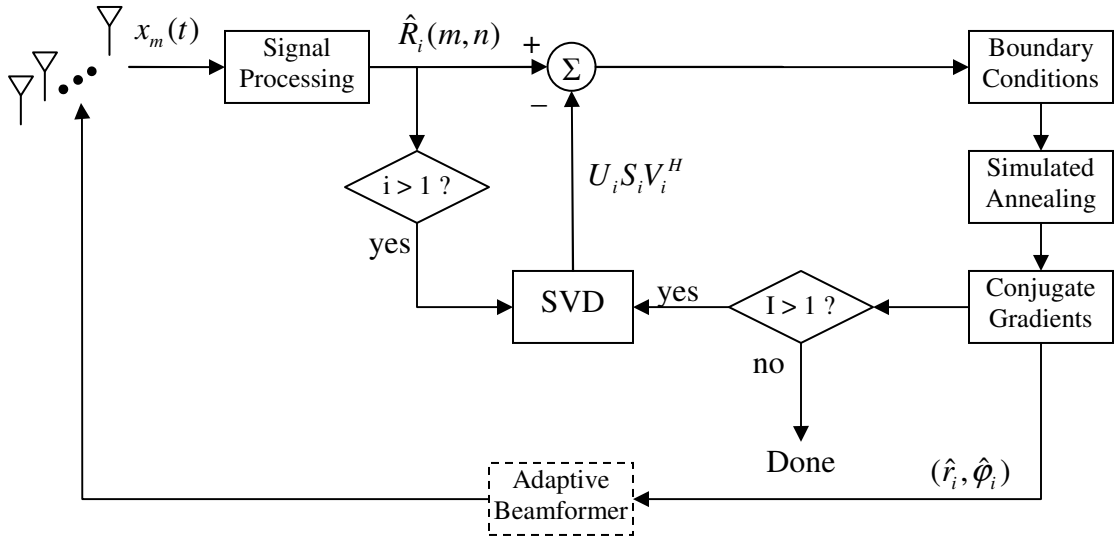


Figure 3.8. Block diagram of multiple source localization algorithm.

In trial simulations, two significant features were identified about the approach presented above. First, a sufficient eigenvalue spread of the input covariance matrix is required; that is the power between enclosed sources must be well separated for effective localization. Second, by removing the contribution of the largest eigenvalue, corresponding to the higher SNR source, the second source can be localized to a limited tolerance, which is a function of the spatial separation of the sources. Test trials indicated that while the highest SNR source was still well within 1% of a wavelength (as in the single source case), converged solutions of successive lower SNR sources were further and further away from the actual source positions. This is partly due to the proximity and signal strength of adjacent sources; from Figure 1.3, it is apparent that sources separated by a small distance may constructively form a global minimum and appear as a single source located in a position other than either source's true location.

However, to demonstrate the feasibility of the multiple localization algorithm, two sources with $40dB$ and $20dB$ SNRs were placed in opposite quadrants. The first source

was placed at $\left(0.6R, \pi/3\right)$ and the second source was placed at $\left(0.6R, 4\pi/3\right)$. All other parameters, including those for the sensor array, simulated annealing and conjugate gradients, remained unchanged. The results are shown in Figure 3.9 through Figure 3.12, followed by a tabulated summary of position errors with respect to spatial separation.

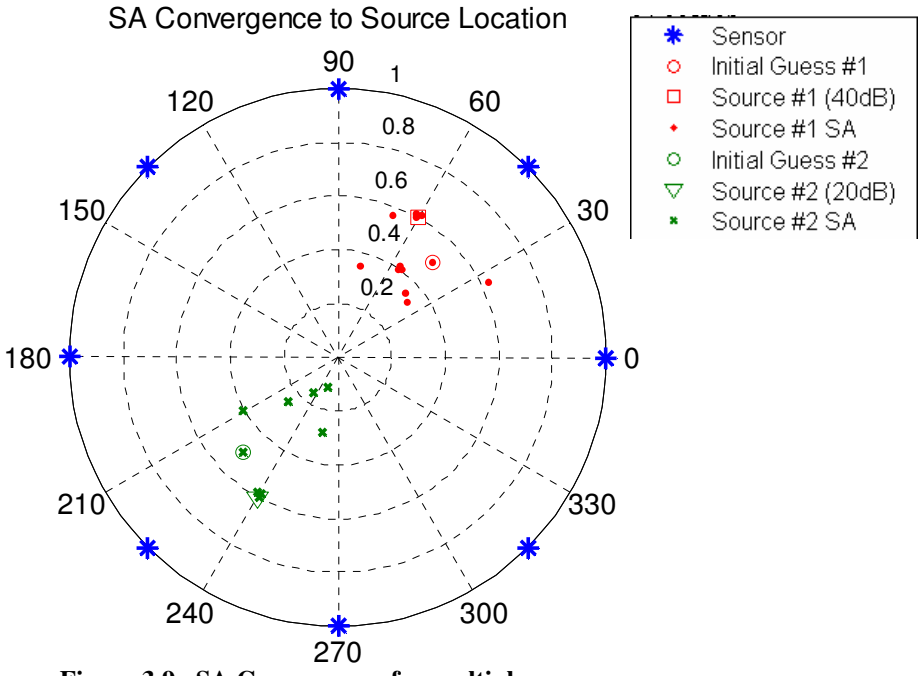


Figure 3.9. SA Convergence for multiple sources.

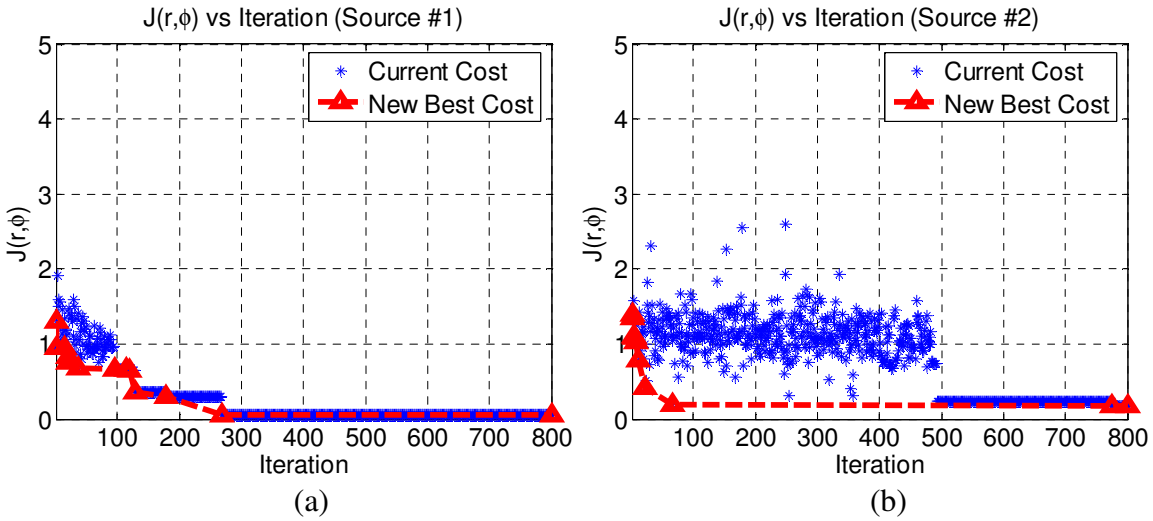


Figure 3.10. Learning curves for localization of (a) source #1 and (b) source #2.

Notice from Figure 3.10 that the source with the larger SNR converges more rapidly than the one with the smaller SNR during SA. Additionally, the lowest cost achieved by the second source is also slightly higher than the first source; this is because the large SNR of the first source corresponds to a deeper minimum on the performance surface.

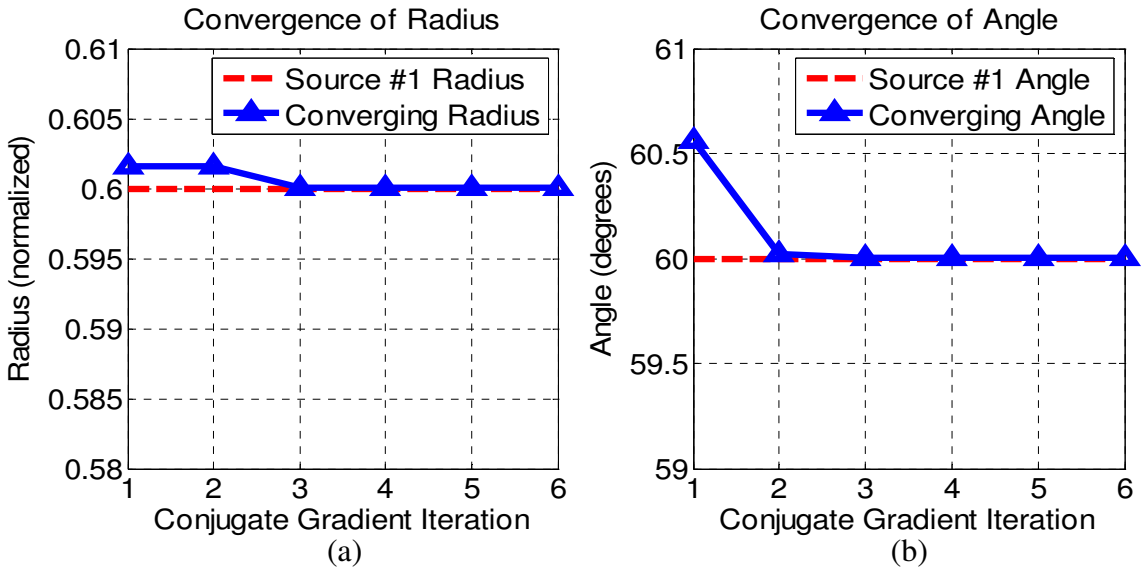


Figure 3.11. Convergences of (a) radius and (b) angle for source #1 (40 dB).

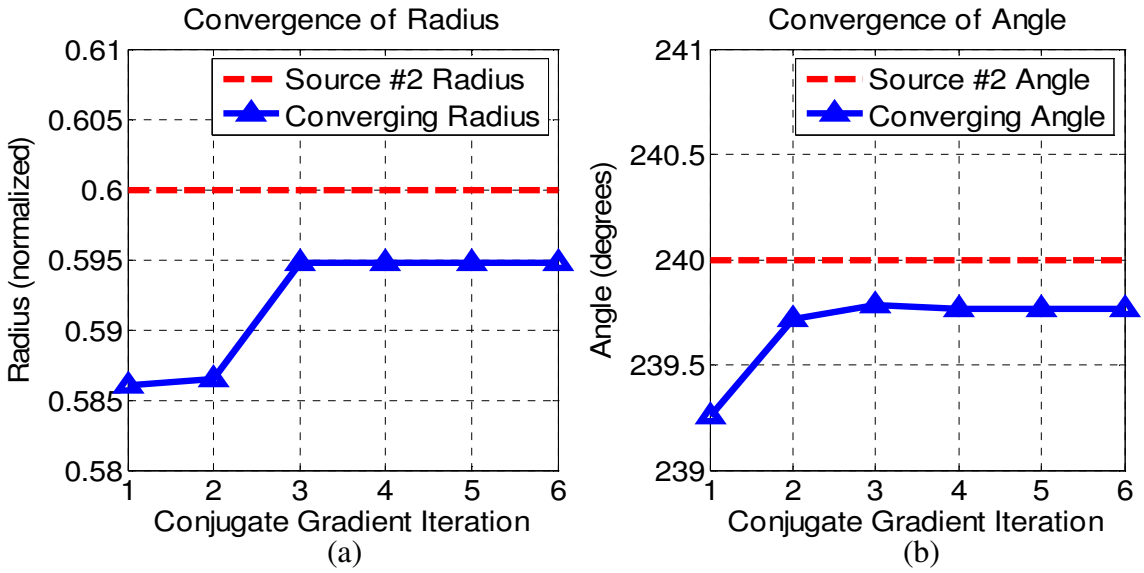


Figure 3.12. Convergences of (a) radius and (b) angle for source #2 (20 dB).

Shown in Figure 3.11 and Figure 3.12 are the convergences to the true source positions by conjugate gradients, using the final SA solutions obtained in Figure 3.10. Again, notice that the source with the larger SNR, shown in Figure 3.11, converges to a much closer value than does the source of Figure 3.12 with $20dB$ less SNR. The latter case exhibits radial and angular errors of 0.87% and 0.098% respectively, while the former only has 0.017% and 0.0045% respectively. Though both are considered converged, this pair of sources is well separated. To observe the impact on proximity in multiple source localization, consider the following table, which summarizes the average percent error over 100 trials of different source arrangements. Note that the location of source two is relative to source one, which remains fixed at $\left(0.6R, \pi/3\right)$.

Absolute Separation	**Relative Separation**		**Source 1 (40 dB) Error (%)**		**Source 2 (20 dB) Error (%)**	
	Δr	**Δφ**	**Radius**	**Angle**	**Radius**	**Angle**
0.5λ	0.5λ	0	0.24	4.95	82.9	73.3
1λ	1λ	0	0.13	2.75	19.2	31.5
3λ	3λ	0	0.81	3.76	7.48	7.50
5λ	2.2λ	π	0.18	3.84	6.51	1.14
2.76λ	0	π/4	0.46	2.84	29.7	20.7
5.09λ	0	π/2	0.506	4.08	1.84	2.98
7.2λ	0	π	0.51	4.08	4.63	1.33

Table 3-4. Percentage error with respect to spatial separation.

In general, as the sources become further separated, the spatial resolution of the algorithm improves dramatically. However, for sources close together, some improvement in spatial resolution can be made by ensuring a large eigenvalue spread in the covariance matrix. In some cases where the sources are very close to one another, as in the 0.5λ and π/4 cases, no amount of SNR separation between sources will allow them to be resolved accurately. Table 3-5 shows the impact of increased power separation on localizing

additional sources which are proximate to the first source. For two reasonably close sources, the SNR of source two was kept fixed at $20dB$, while source one was increased by $20dB$, $40dB$ and $60dB$. Again, 100 trials were performed at each separation to obtain an average error.

SNR Separation (dB)	Source 2 Error (%)	
	$\Delta r = 1\lambda, \Delta\varphi = 0$	
	Radius	Angle
20	19.2	31.5
40	12.3	20.2
80	8.64	13.7

Table 3-5. Percentage error with respect to signal power separation.

As shown above, a distinctive power separation between proximate sources can moderately improve the spatial resolution of the multiple source localization algorithm.

3.6 Tracking

Thus far, source localization has been considered for individual as well as multiple stationary sources. In practical applications, however, sources are not always stationary; in many cases, they are mobile and move in an unpredictable manner. The localization of mobile sources, also known as tracking [21]-[22], has a wide range of applications from GPS and target acquisition to mobile telephony, and is explored further in this section.

First, the source is assumed to be moving slowly with a constant velocity. Linear displacement functions are then assigned to both the radial and angular parameters, such that the position of the source would slowly vary over a range of N samples. A portion of these samples is designated as a "buffer" space at the start and end of each trajectory function to simulate the source in a halted position. The signal parameters used are the

same as shown in Table 3-1, with the exception of SNR, which is lowered to $30dB$. Next, the data observed at each sensor, as in (1-2), is generated for each sample. For every n samples, an estimate of the covariance matrix is made. Using this covariance matrix, the hybrid heuristic algorithm is applied to localize the moving source with the same parameters as outlined in Table 3-2 and Table 3-3. The resulting solutions are stored and compared alongside the actual source's trajectory, using the following experimental parameters:

Parameter	Value	Description
b	0.1	Fraction of total samples used as buffer space
n	100	Samples used to construct covariance matrix
N	1000	Total number of samples observed

Table 3-6. Simulation parameters for tracking experiment.

In the following set of figures, a source moving at 0.0013 wavelengths per sample and 0.5625 degrees per sample is tracked by the algorithm. The average error in this particular case is 2.16% for the radial motion and 2.65% for angular motion.

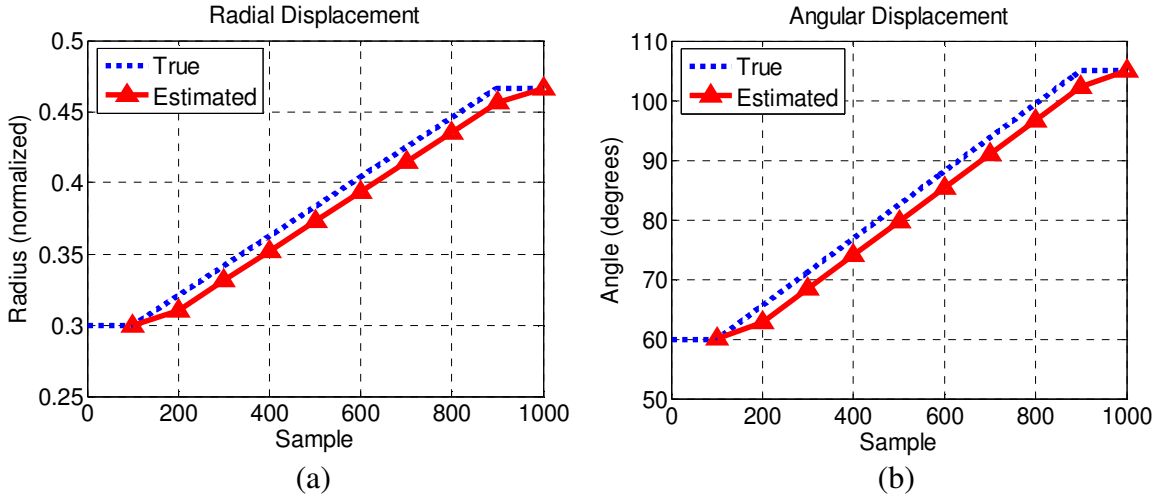


Figure 3.13. Constant, linear (a) radial and (b) angular displacements.

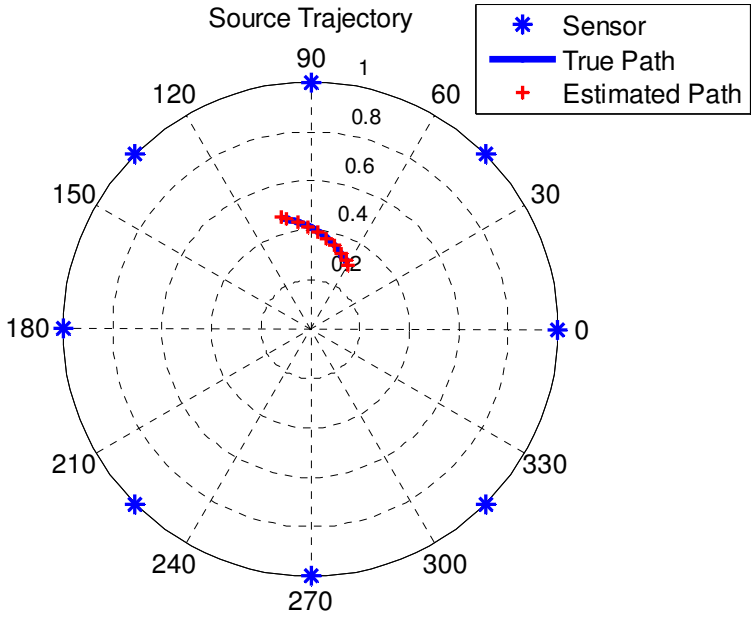


Figure 3.14. Slow, linear source trajectory from (0.3R, π/3).

Tracking slow, non-linear motion was also attempted. In Figure 3.15 below, a portion of a sinusoid represented the radial displacement, which decelerated from 0.0016 to -0.0013 wavelengths per sample, while the angular displacement was modeled by a small velocity, accelerating from -0.0115 to -0.1031 degrees per sample. Here, the average position error was 0.0084% and 2.22% for the radial and angular displacements respectively.

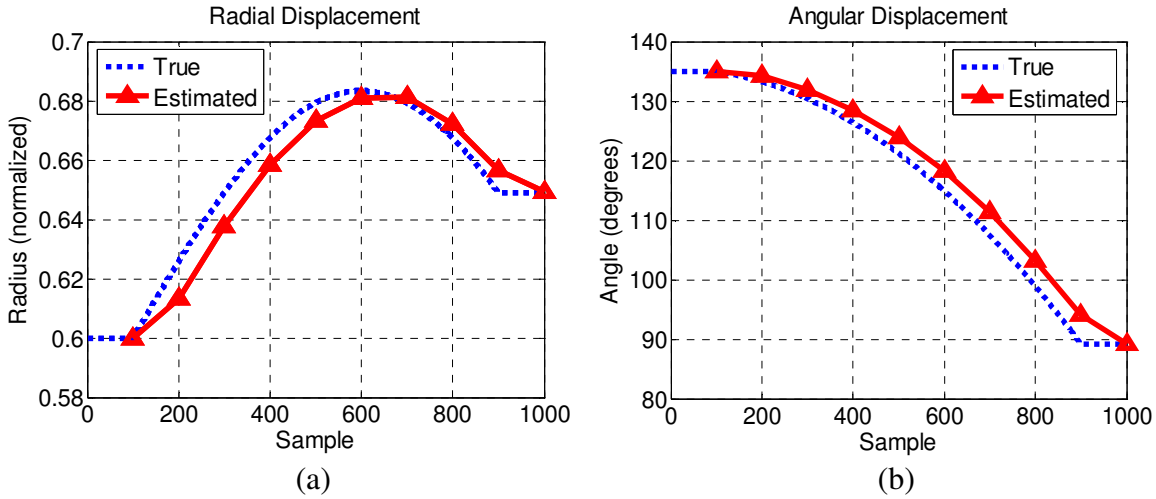


Figure 3.15. Non-linear (a) radial and (b) angular displacements.

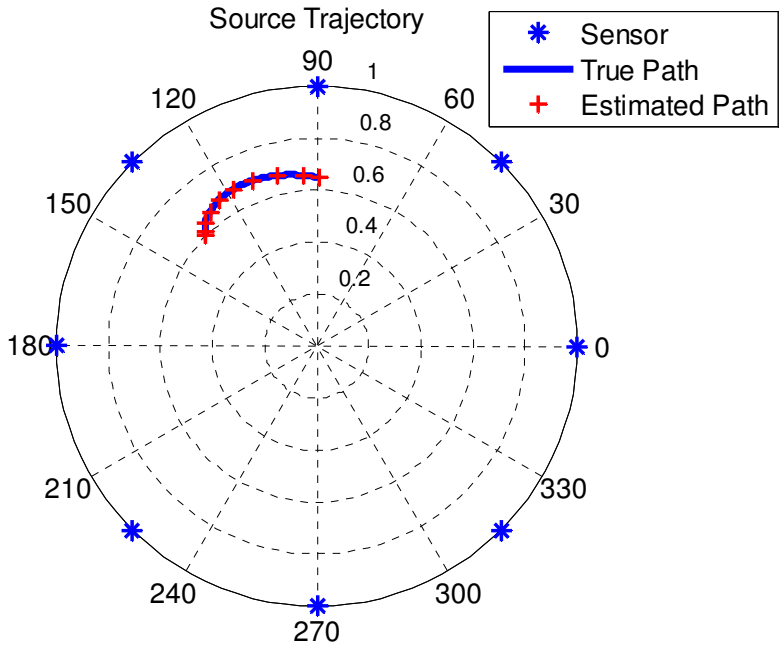


Figure 3.16. Slow, non-linear source trajectory from (0.6R, 3π/4).

So far, the algorithm has been able to effectively track slow, linear and non-linear moving sources, however, there is a limit to the effectiveness of this approach. Sources traveling faster than the rate of convergence of the algorithm are difficult to localize consistently. Therefore, the tracking is a function of the covariance matrix sampling rate.

Using the same test conditions as in the previous section, the limits of the tracking capabilities were tested. To do this, the radial position was fixed at $0.6R$, and only the angular displacement was observed. The speed of the source was taken as the arc length traversed per sample and scaled according to the number of samples used to estimate a covariance matrix. Thus, the resulting velocity would be normalized to the rate of covariance matrix sampling, which was fixed at 100 samples. Through testing, it was determined that over a range of 1000 samples, moving at a constant velocity beyond 0.5 wavelengths per covariance matrix sample caused the tracking algorithm to diverge.

Table 3-7 below summarizes the average tracking position errors seen in five separate cases where the source was simulated moving at the specified rate for 100 trials.

Velocity (λ / sample*)	Displacement Error (%)	
	Radius	Angle
0.5	0.28	1.92
0.75	15.8	7.17
1	32.2	9.08
2	47.4	14.9
5	60.3	47.2

Table 3-7. Average position error for different velocities.
*Note that the velocity is relative to each covariance matrix sample

It is clear that between estimations of the covariance matrix, if the source is displaced too far, more error will be present in tracking the source. One approach to improve the algorithm's speed is to construct the covariance matrix with fewer samples, which will have a negative impact on accuracy and require more computations. Alternately, a higher sampling frequency can also be used.

3.7 Summary

In this chapter, the simulated hybrid heuristic algorithm was described in further detail and the parameters used were defined. The behavior of the algorithm was discussed in the single source case, and its performance was analyzed in the form of the simulated CRLB. Additionally, a feasible algorithm for localizing multiple sources was introduced. In simulation, this algorithm was shown to be susceptible to multiple proximate sources, which constructively formed minima in locations other than the true source positions. It was discovered that for multiple proximate sources, convergence was more effective for a larger eigenvalue spread. On its own, the proposed multiple source localization technique appeared insufficient for many sources, however, it could

significantly benefit from the use of adaptive beamforming, which can be used to place nulls in the locations of sources already discovered in order generate a new covariance matrix estimate and better resolve sources that remain undetected. Finally, the tracking capabilities of the proposed algorithm have also been explored and it has been shown that for slow moving sources with little or no acceleration, the hybrid heuristic approach is able to track the trajectory of a mobile source.

4 Design and Characterization of Antennas

4.1 Microstrip Patch Antenna

4.1.1 Simulated Design

Two types of antennas are used in the proposed sensor array: a rectangular microstrip patch antenna [21], which acts as a receiver and is shown in Figure 4.1 below, and a wire antenna, shown in Section 4.2.

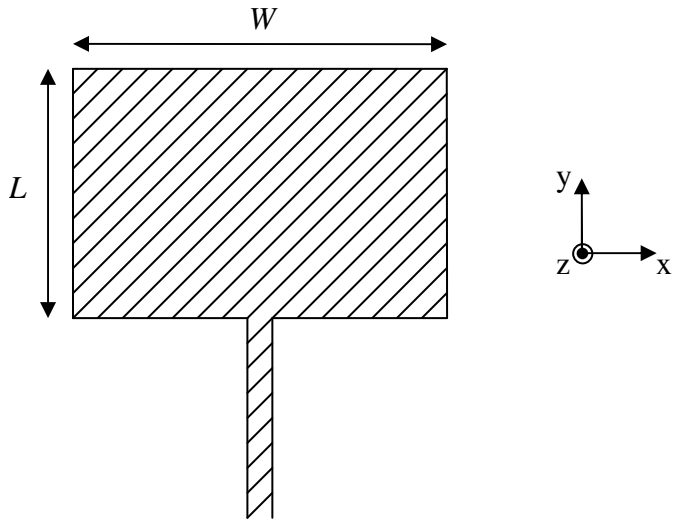


Figure 4.1. Rectangular microstrip patch antenna dimensions.

The simulated design of the microstrip patch antenna is performed in Ansoft High Frequency Structure Simulator (HFSS), using approximately half of the free space wavelength as an initial width and half of the dielectric wavelength as the initial length. The initial dimensions of the patch are defined as $W = 4.7416\ cm$ and $L = 3.8477\ cm$. Additionally, to simulate the losses of the dielectric substrate, a loss tangent of $\tan(\delta) = 0.0012$ is used. As shown in [13], the width of the patch determines the resonant frequency of the antenna. Thus, to ensure the proper frequency would propagate

through this antenna, the width of the patch has been adjusted by 0.5 mm increments. By identifying the frequency shift per width adjustment, the nominal width is adjusted to center the resonance around 2.45 GHz.

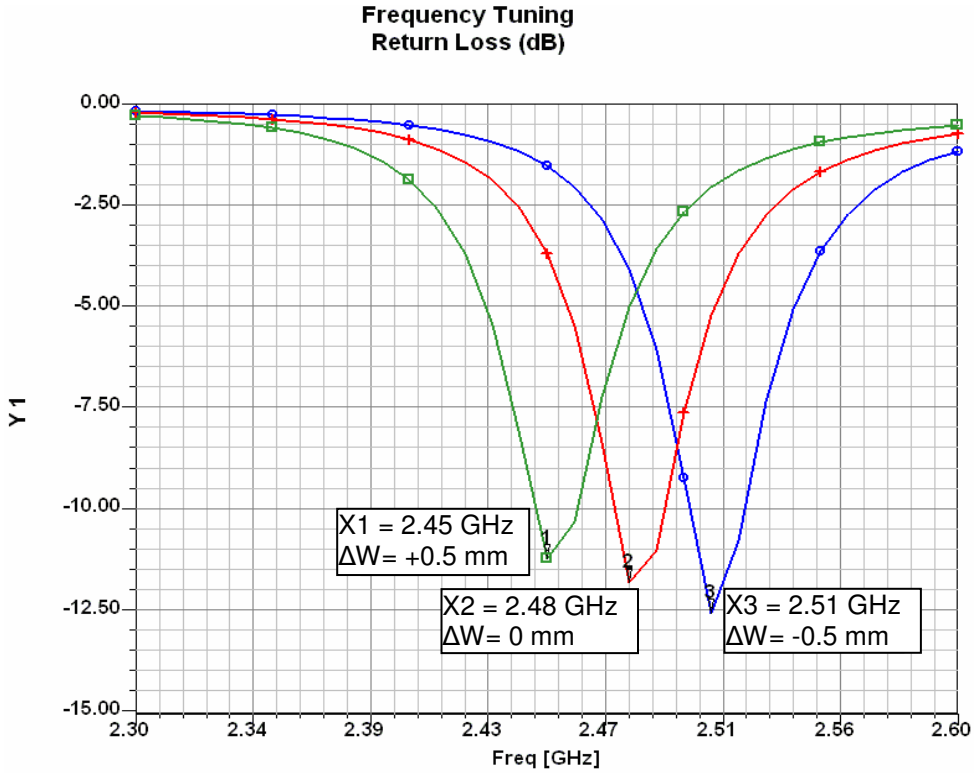


Figure 4.2. Frequency tuning of microstrip patch antenna in Ansoft HFSS.

Though resonance is achieved at the desired frequency, the corresponding return loss can still be improved by matching the impedance of the design. To do this, the feed impedance of the patch is first determined by placing a lumped port at the edge of the patch, centered along the *W* dimension, as shown in Figure 4.3. Using an adaptive frequency of 2.45 GHz, 25 passes are made with a Δs value of 0.01. With this setup, a fast sweep was performed from 2.3 to 2.6 GHz in 10 MHz step sizes. Afterward, the smith tool is used to identify the real part of the input impedance, as shown in Figure 4.4.

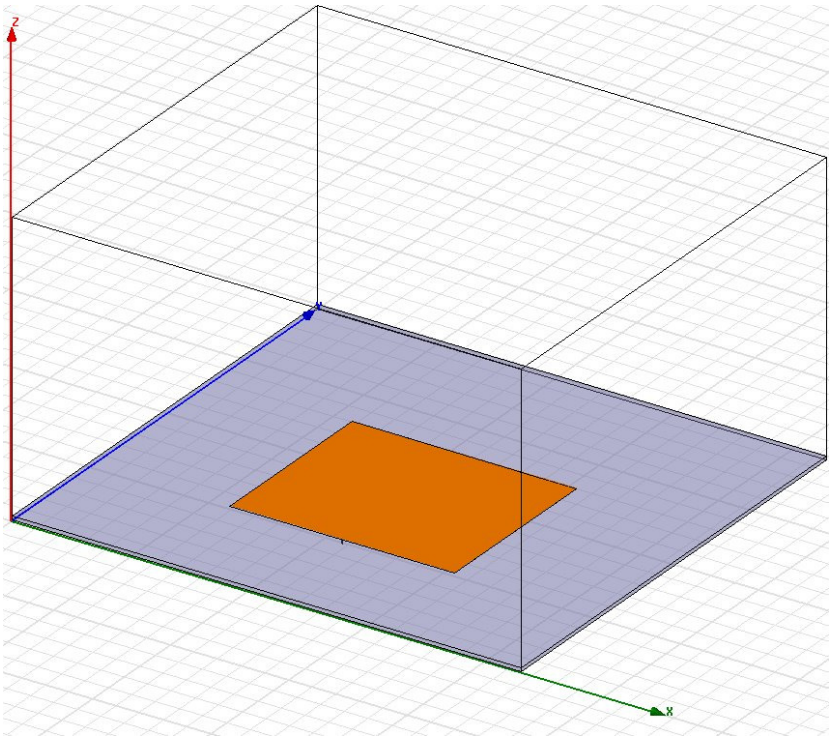


Figure 4.3. Microstrip patch antenna feed point impedance simulation.

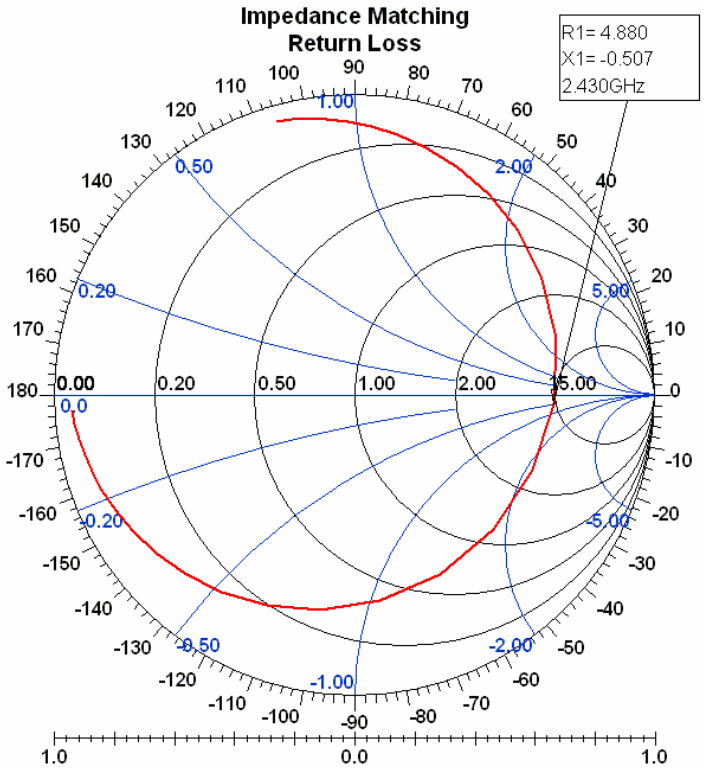


Figure 4.4. Smith chart of feed point impedance from 2.3 – 2.6 GHz.

From the smith chart tool, the de-normalized impedance is taken to be $Z_{feed} = 244 - j25.35\Omega$. From the real portion of this impedance, the quarter wave transformer is designed to be a 110Ω transmission line.

Using the same substrate and loss tangent parameters shown previously, the corresponding widths and lengths of the quarter wave transmission line are generated by the Ansoft Designer transmission line synthesis tool to be $W_{QWT} = 0.526504\ mm$ and $L_{QWT} = 22.6286\ mm$. Finally, the quarter wave transformer is terminated with a short length of 50 Ω microstrip transmission line of width $W_{50\Omega} = 2.3393\ mm$, and a new waveport is defined at this edge with dimensions large enough to encompass any fringing fields that may occur at the input. The final matched and tuned design has dimensions of $W = 4.7916\ cm$ and $L = 3.8477\ cm$; it is analyzed with a discrete sweep, 15 passes and a Δs value of 0.04. Under these conditions, the following return loss, Voltage Standing Wave Ratio (VSWR) and 2:1 VSWR bandwidth have been achieved at 2.45 GHz:

Parameter	**Simulated**
Resonant Frequency (GHz)	2.45
Return Loss (dB)	-27.15
VSWR	1.09
2:1 VSWR Bandwidth (Hz)	20 MHz

Table 4-1. Simulated patch antenna parameters.

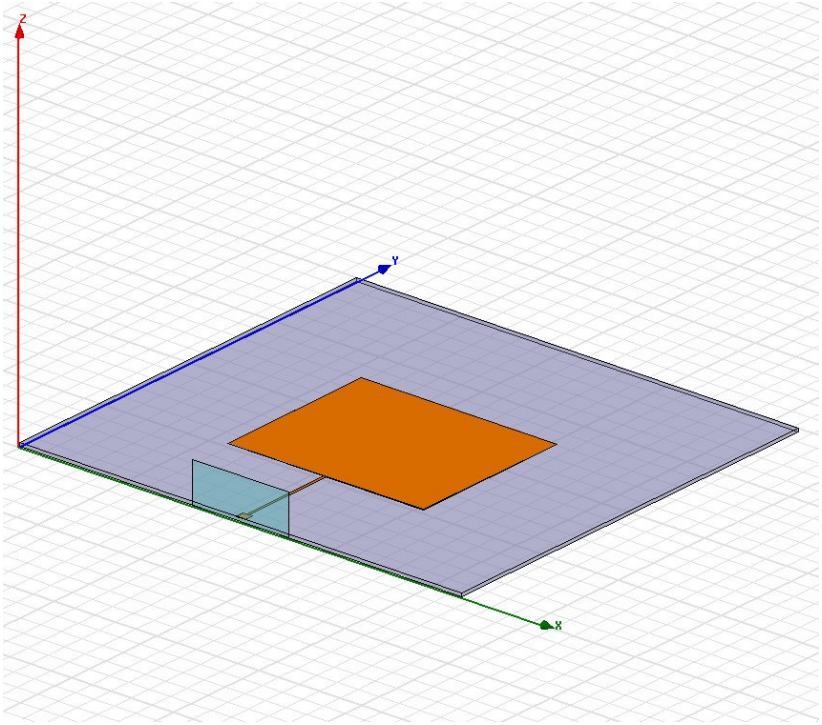


Figure 4.5. Microstrip patch antenna with quarter wave transformer.

The final tuned and matched design is shown in Figure 4.5 above; corresponding graphs of VSWR and return loss have also been plotted from 2 GHz to 3 GHz in Figure 4.6 and Figure 4.7. In addition, tthe far field radiation patterns at 2.45 GHz are shown in Figure 4.8 through Figure 4.10 for each plane. From Figure 4.9, it can be seen that the patch antenna has a maximum realized gain of 7.588 dBi at 4˚ in the Y-Z plane (φ = 90˚).

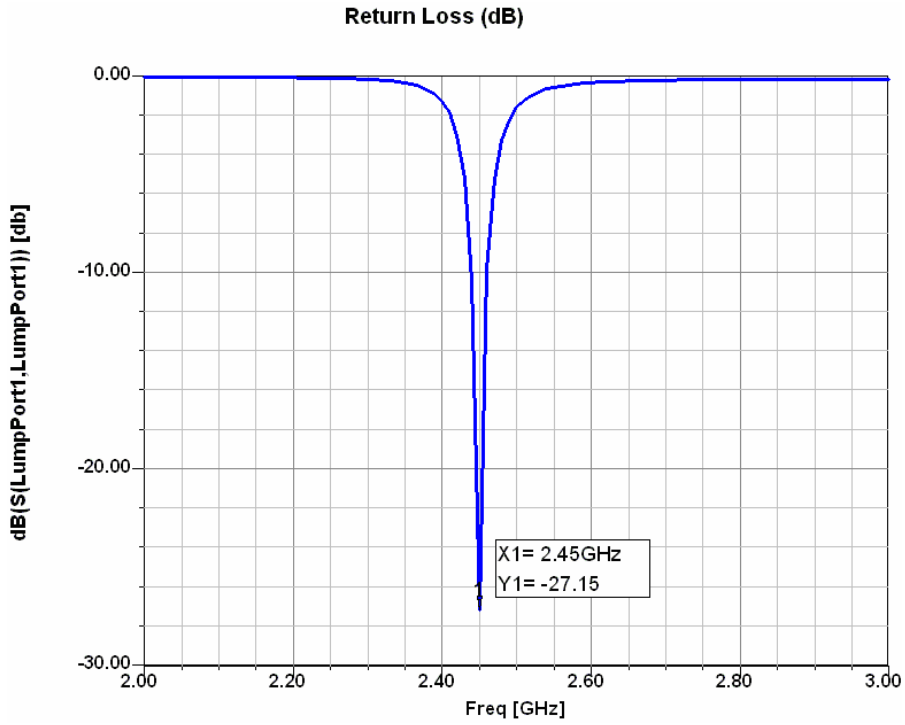


Figure 4.6. Return loss (dB) of microstrip patch antenna, tuned for f_0 = 2.45 GHz.

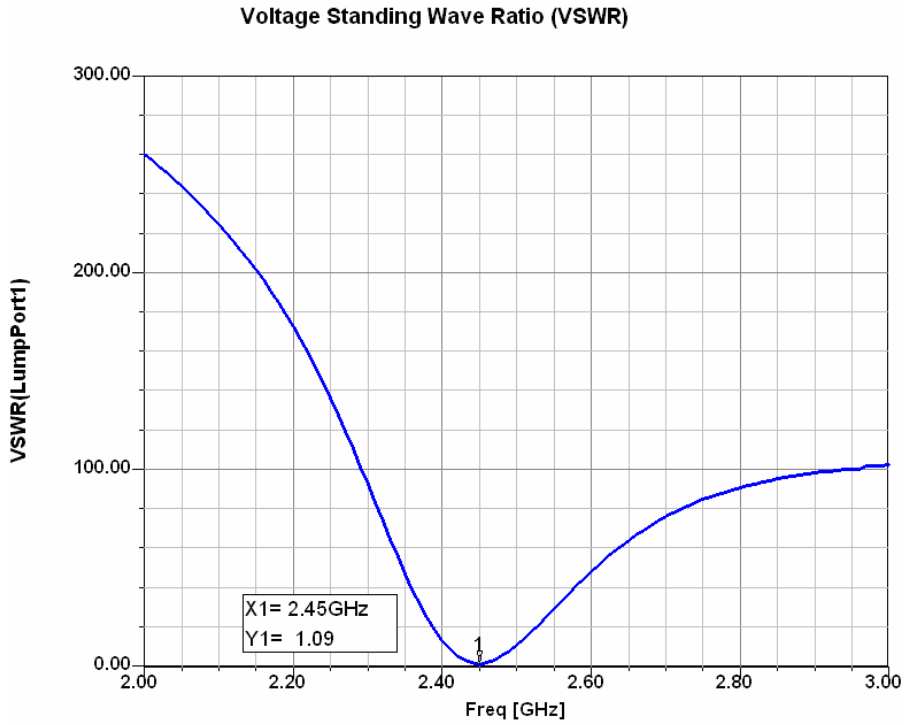


Figure 4.7. VSWR of microstrip patch antenna, tuned for f_0 = 2.45 GHz.

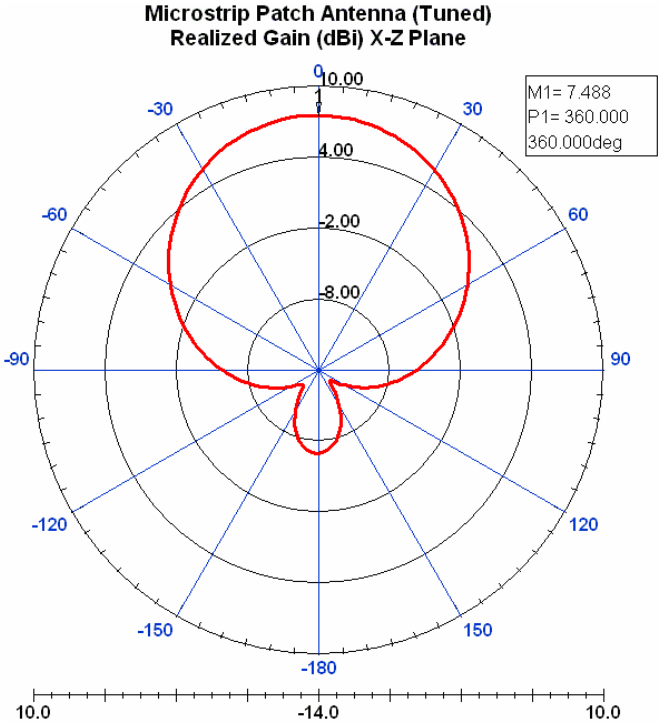


Figure 4.8. Radiation pattern of tuned patch antenna in the X-Z Plane (φ = 0˚).

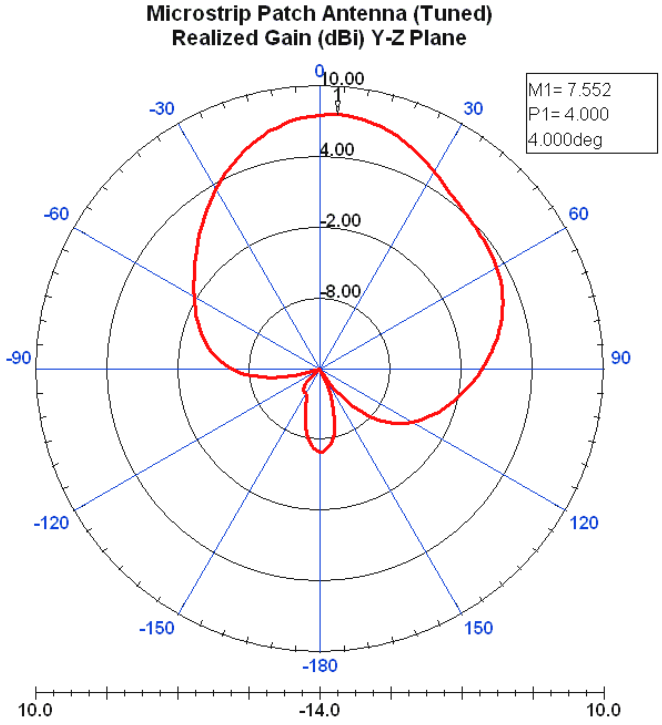


Figure 4.9. Radiation pattern of tuned patch antenna in the Y-Z Plane (φ = 90˚).

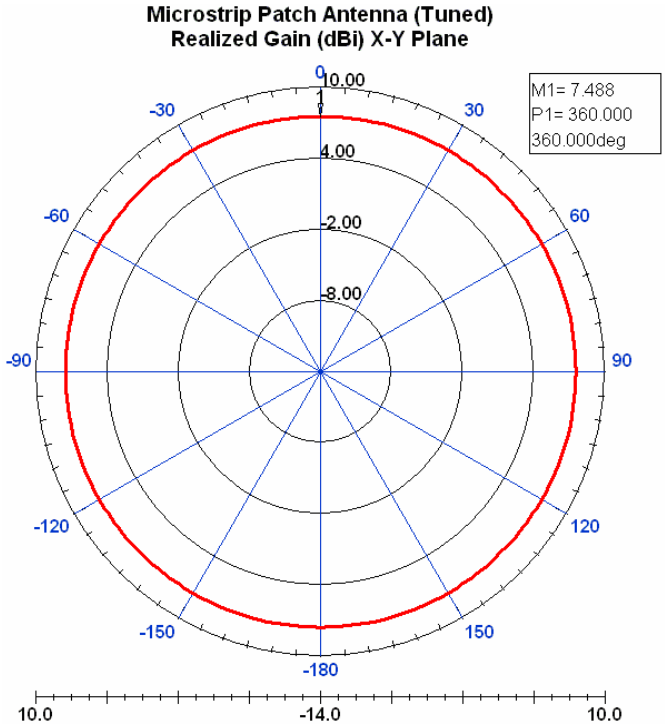


Figure 4.10. Radiation pattern of tuned patch antenna in the X-Y Plane (θ = 90˚).

4.1.2 Hardware Implementation

To allow for coaxial connectors to be soldered onto the input port of the final design, the length of the 50 Ω transmission line segment is increased before being etched. The chemical etching has been performed on Rogers RT 5870 Duroid by Nationwide Circuits, Inc. and is tested on an Agilent Technologies E8363B PNA Series Network Analyzer. Before doing so, one port calibrations are performed on all of the microstrip patch antennas using a Hewlett Packard 85052B 3.5mm shorts, opens, loads, and thrus (SOLT) calibration kit and a female to male Gore-Tex SMA coaxial cable fitted with a male to female k type (40 GHz) adapter. The loads used for calibration are a female short, female open, and female lowband matched load. The resulting return loss and VSWR for each microstrip patch are then recorded and compared to simulated results.

The following figures are taken from the first microstrip patch:

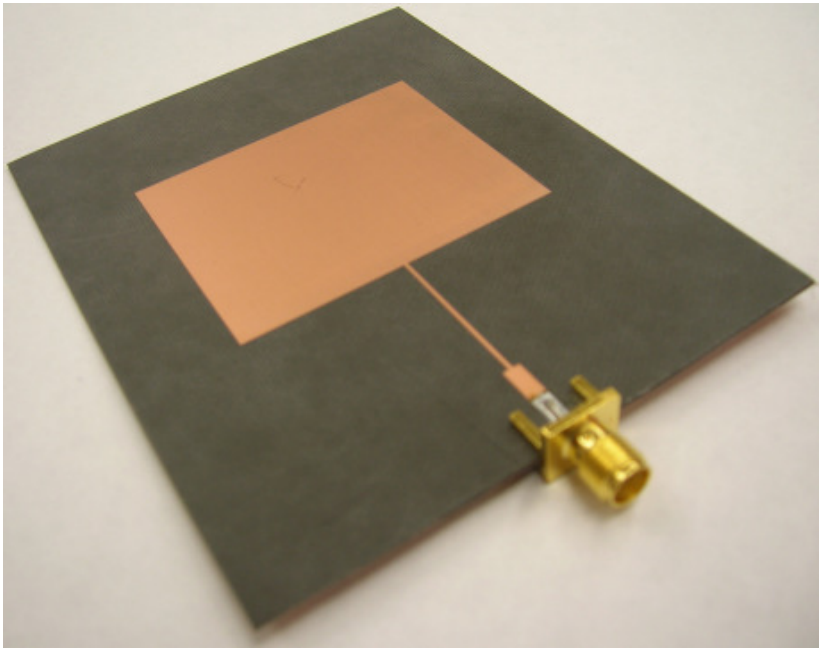


Figure 4.11. Microstrip patch antenna, chemically etched on RT 5870 Duroid.

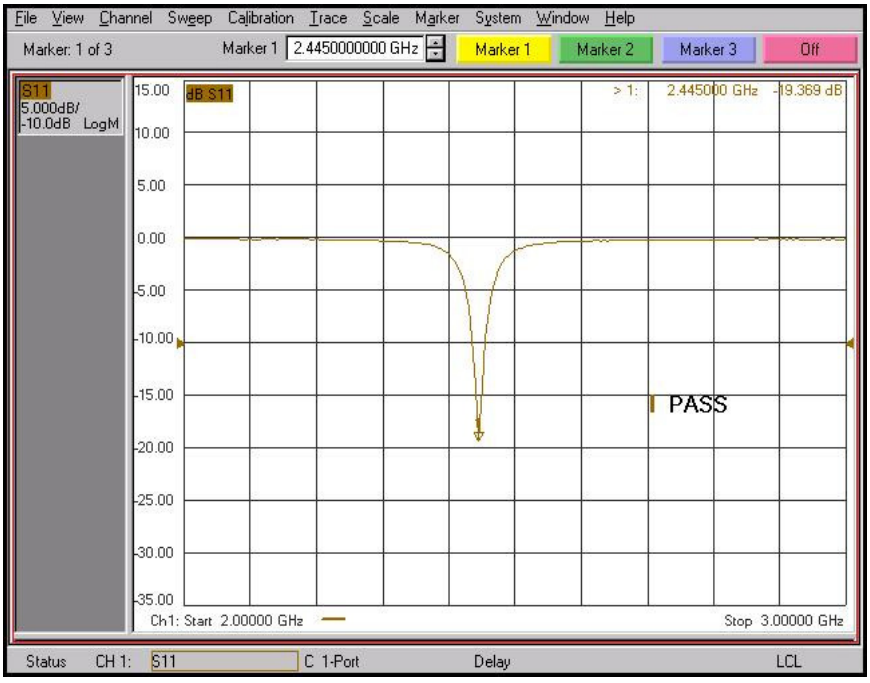


Figure 4.12. Measured return loss (dB) of microstrip patch antenna.

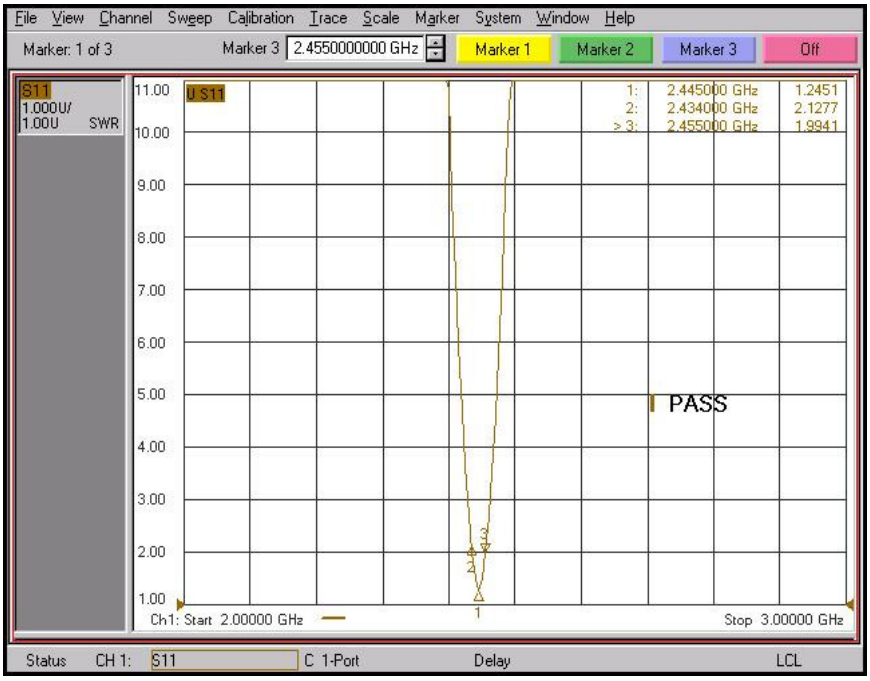


Figure 4.13. Measured VSWR and 2:1 bandwidth of microstrip patch antenna.

This particular microstrip patch antenna is shown to be resonant at 2.455 GHz, with a 2:1 VSWR bandwidth of about 20 MHz and a return loss of -19.369 dB. Though the VSWR at the resonant frequency is larger than in simulation, it only corresponds to a power loss of 1.15 % due to reflection, well below the industry standard of 4%. The following table summarizes the hardware measurements of each patch antenna and compares them with the simulated results of Table 4-1:

Patch	Resonant Frequency (GHz)		Return Loss (dB)		VSWR		2:1 VSWR BW (MHz)	
	Value	% Error	Value	% Error	Value	% Error	Value	% Error
1	2.445	0.2	-19.36	28.7	1.24	13.8	21	5
2	2.435	0.6	-18.15	33.1	1.31	20.2	21	5
3	2.435	0.6	-14.83	45.4	1.44	32.1	24	20
4	2.445	0.2	-14.59	46.3	1.45	33.0	16	20

Table 4-2. Summary of microstrip patch antenna hardware measurements.

Due to the extremely narrow bandwidth, the resonant frequencies are very susceptible to being offset, however, the offset in each case is very minimal, and still within the desired frequency range. In each case, the corresponding VSWR measured at the resonant frequency indicated that less than 4% of the power incident to each patch is lost due to reflection. Additionally, the variations in measured results can be attributed to the use of different SMA connectors, as well as uneven variations in the substrate edges from non-uniform cuts made on the main board.

In addition to the parameters measured in Table 4-2, the radiation pattern of each antenna has also been obtained, using the following apparatus:

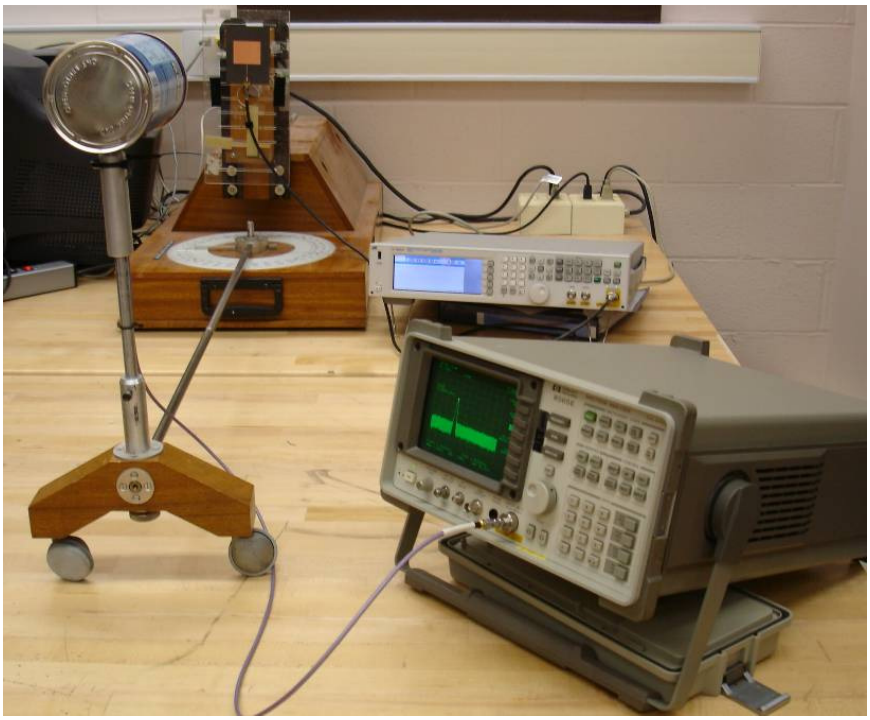


Figure 4.14. Experimental apparatus for measurement of radiation patterns.

In this arrangement, a cavity containing a 2.45 GHz wire antenna is used as the receive antenna, which is connected to a Hewlett Packard 8565E Spectrum Analyzer via a male

to male SMA cable. The microstrip patch antenna is centered on the mounting block via hook and pile tape and leveled with the cavity. The patch is then excited at its resonant frequency with a 0 dBm CW signal propagating through an RG223/U SMA cable by an Agilent N5182A MXG Vector Signal Generator. The measurement angle, relative to the transmitting antenna was read from the dial at the base of the apparatus; measurements were taken in five degree increments between -45 and +45 degrees, and recorded to the nearest 0.5 dB of power. Using this method, the relative gains have been plotted in the radiation patterns, and normalized to the maximum gain observed in each case:

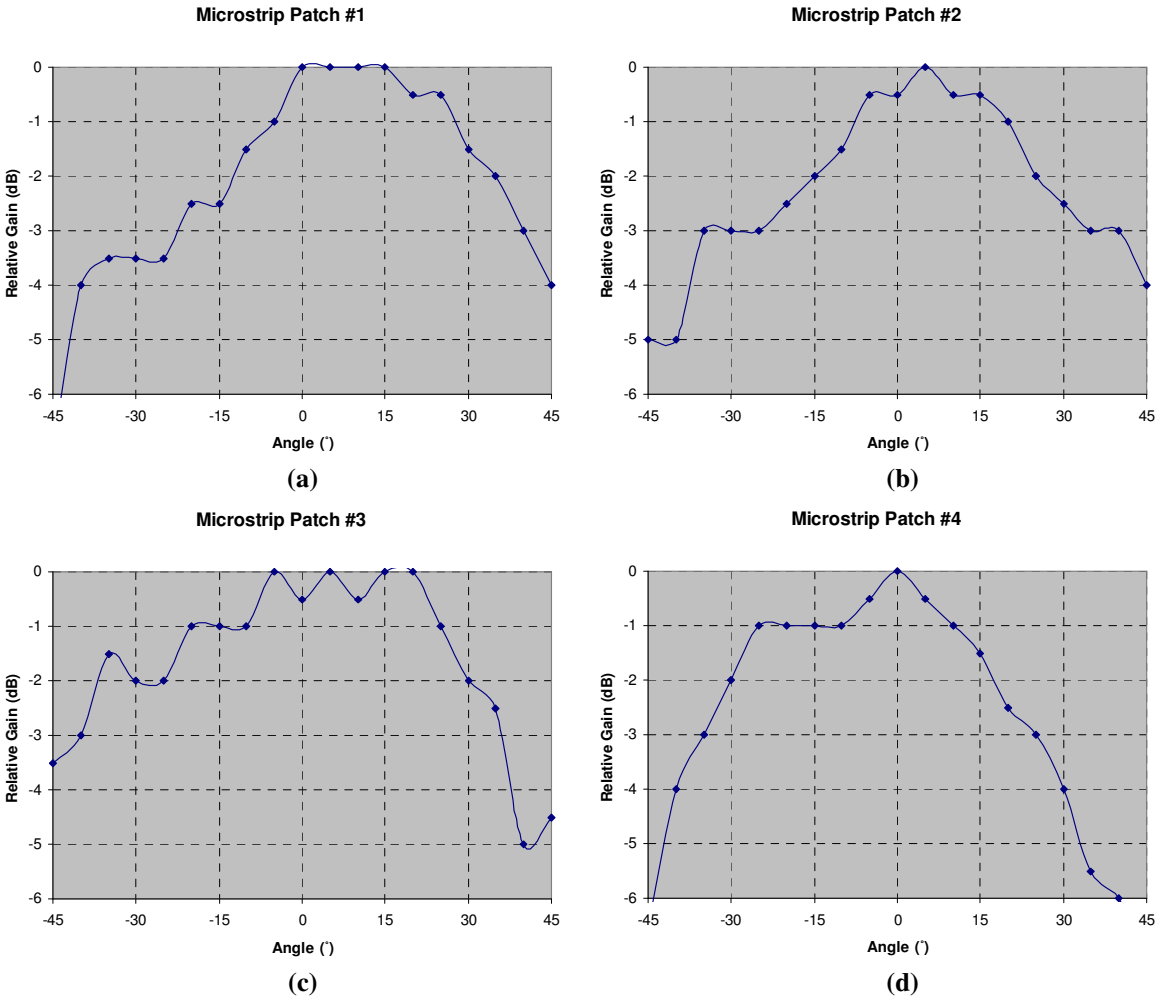


Figure 4.15. Measured azimuth radiation patterns of microstrip patch antennas.

Though not all patterns shown are exactly the same, the resulting radiation patterns are generally what is expected of a microstrip patch antenna. Some of the beams shown appear slightly lopsided or phased, which is concurrent with the simulated elevation radiation pattern shown in Figure 4.9, and would likely be noticeable in azimuth patterns if the receive and transmit antenna are not perfectly leveled with one another. The only exception to the norm is patch #3, as shown in Figure 4.15c, which appears to have some closely spaced grating lobes broadside to it. A number of factors could have contributed to this including non-uniform substrate edges, fringing fields from insufficient spacing, a feed connector mounted at an angle, and poor connectivity to the ground plane. Fortunately, the localization algorithm presented in this paper makes no assumptions about sensor radiation patterns; so long as the radiation pattern in Figure 4.15c does not cause it to receive too many interfering signals, it should still be a sufficient sensor.

4.2 Wire Antenna

With the sensors designed and tested, the next antenna to discuss is the one which represents the stationary sources. The design criteria for the emitter are that it must operate at the same frequency and ideally present an omnidirectional radiation pattern in the Z-plane. In the end, two options for the source are considered: a simple coaxial wire antenna or an omnidirectional microstrip antenna [36]. In the interests of time and simplicity, the former design was pursued with great success. In this section, the theory behind the wire radiator and its implementation will be shown.

4.2.1 Theoretical Development

The quarter wave monopole antenna excited at the base, makes use of the ground plane to act as an effective electromagnetic radiator. Analysis of this can be shown using simple image theory principles. Based on these principles, energy from a radiator that is directed towards the ground will be reflected. The reflected energy seen at a field point can be represented by a virtual sources, or image, mirrored on the opposite side of the ground plane, as shown in Figure 4.16 below:

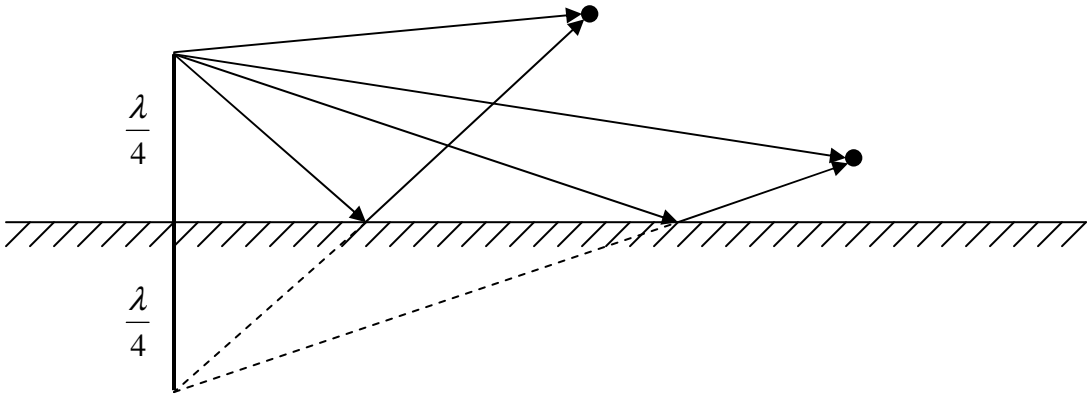


Figure 4.16. Actual and virtual λ/4 emitters due to ground plane reflections.

The radiation pattern presented by this arrangement is very similar to a center-fed half-wave dipole, but because the virtual source is imaginary, the quarter wave antenna only radiates above the ground plane with half of the radiated power.

4.2.2 Hardware Implementation

Implementation of the wire antenna is very simple; it consisted of a female N-type double bulkhead connector and a length of paperclip, trimmed such that the exposed portion is approximately $\lambda/4$ in length. Once cut to the correct length,

the paperclip is soldered into one end of the bulkhead connector. Using an N-type to SMA adapter, the return loss and VSWR are measured using the same approach as with the microstrip patch antennas.

Figure 4.17. λ/4 wire antenna, soldered into 50Ω female N-type bulkhead connector.

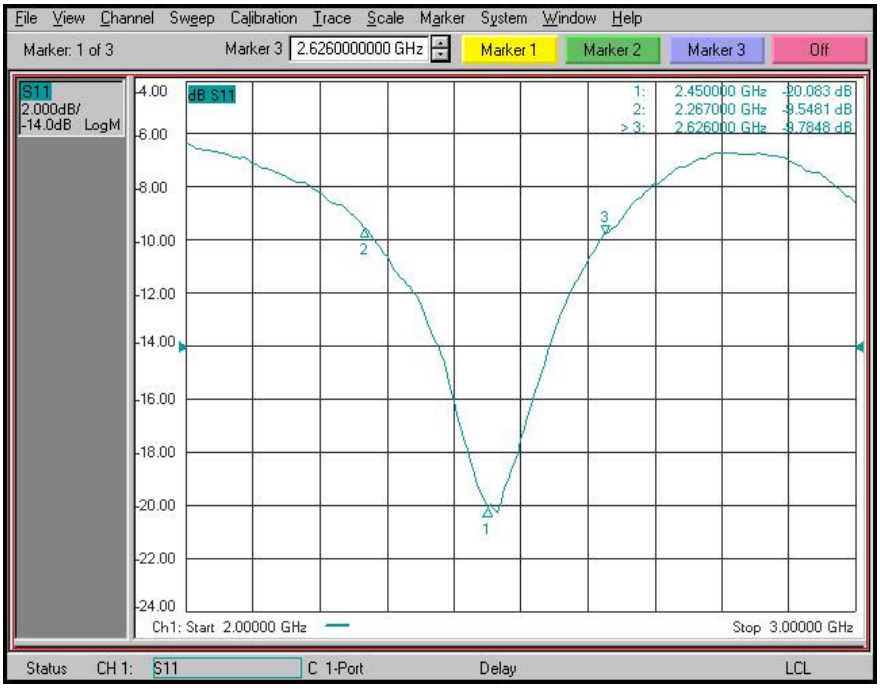


Figure 4.18. Measured return loss of quarter-wave wire antenna.

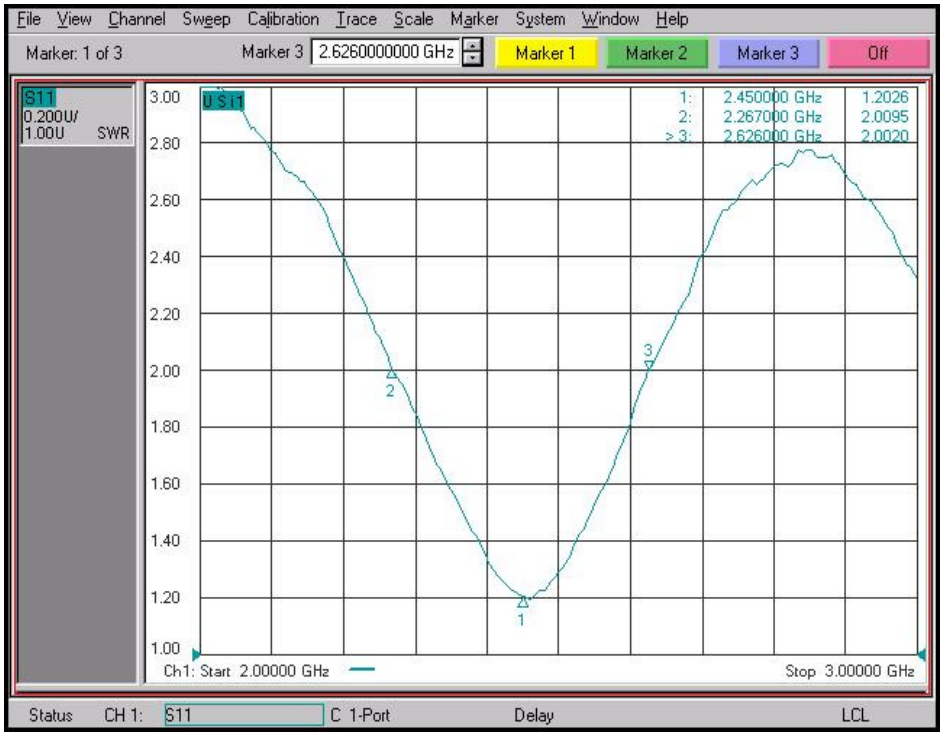


Figure 4.19. Measured VSWR and 2:1 bandwidth of quarter-wave wire antenna.

From Figure 4.19, the wire antenna exhibits a resonant frequency of 2.45 GHz, with a corresponding VSWR of 1.2, which resulted in only 0.83 % power lost due to reflection. Additionally, the wire antenna presented a -20.08 dB return loss and a 2:1 VSWR bandwidth of 441 MHz or an 18% normalized bandwidth, which is more than sufficient for the narrow bandwidth patch antennas used.

5 Proof of Concept

5.1 Materials and Apparatus

After successful attempts to localize sources in MATLAB simulations using the algorithm described in Section 2.6, the uniform circular sensor array is prototyped in hardware as a proof of concept. In this apparatus, four sensors and one source are modeled inside a box. With four sensors, instead of the eight used in simulation, a square geometry is adopted instead of a circular arrangement; this allows the patch antennas to be easily mounted, while still allowing the emulation of a circular sensor array. To minimize reflections and multipath, the box has not been made out of a conductive material. Instead, a Styrofoam box is used because it is easily available, light weight, and has a free space permittivity. The proposed assembly as viewed from above is shown in the following figure:

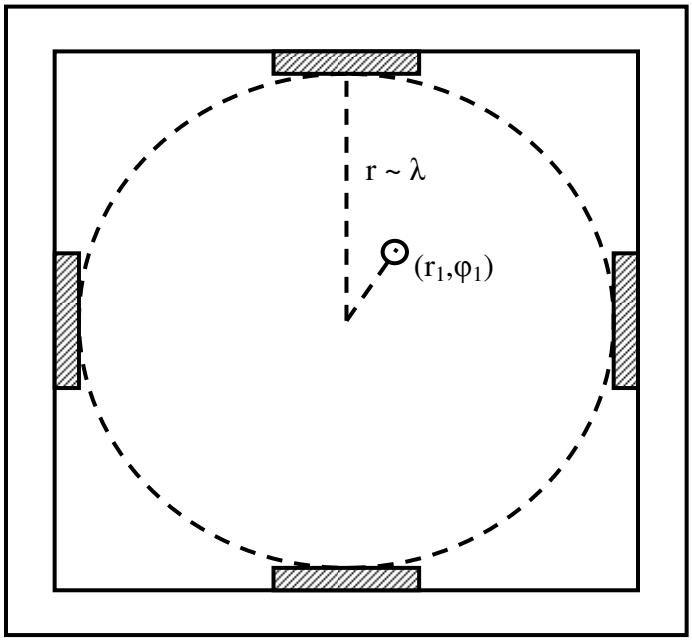


Figure 5.1. Top view of proposed experimental apparatus.

Within the box, the sensors used are modeled by the four microstrip patch antennas, simulated and designed as in Section 4.1. They are attached with hook and pile tape (Velcro) at the center of each inner side of the box, with the feed traces arranged to exit through the top of the box for ease of connection to the measuring device: an Agilent Infinium 86100C digital sampling oscilloscope.

The actual Styrofoam box acquired came in six pieces; the sides of the box have been assembled using strips of masking tape, leaving the top exposed. The complete assembly stood 8" (20.32 cm) tall and had outer dimensions of 11.875" (30.16 cm), and inner dimensions of 10.875" (27.62 cm). After taking into consideration the thickness of the substrate and the Velcro backing, the radius of the base has been marked and measured to be approximately 12.54 cm, or 1.02λ at 2.44 GHz. The actual box, with the microstrip patch antennas mounted is shown in Figure 5.2 below:

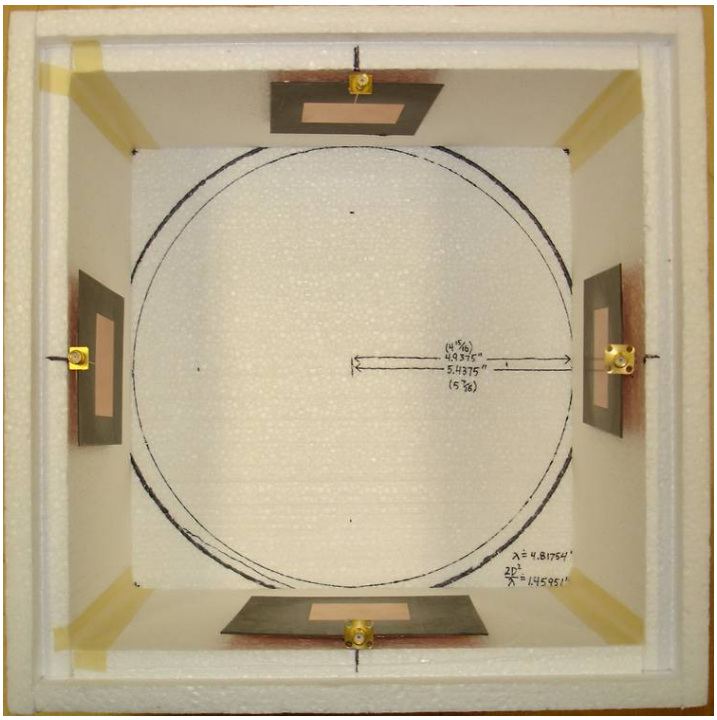


Figure 5.2. Top view of prototype experimental apparatus without source.

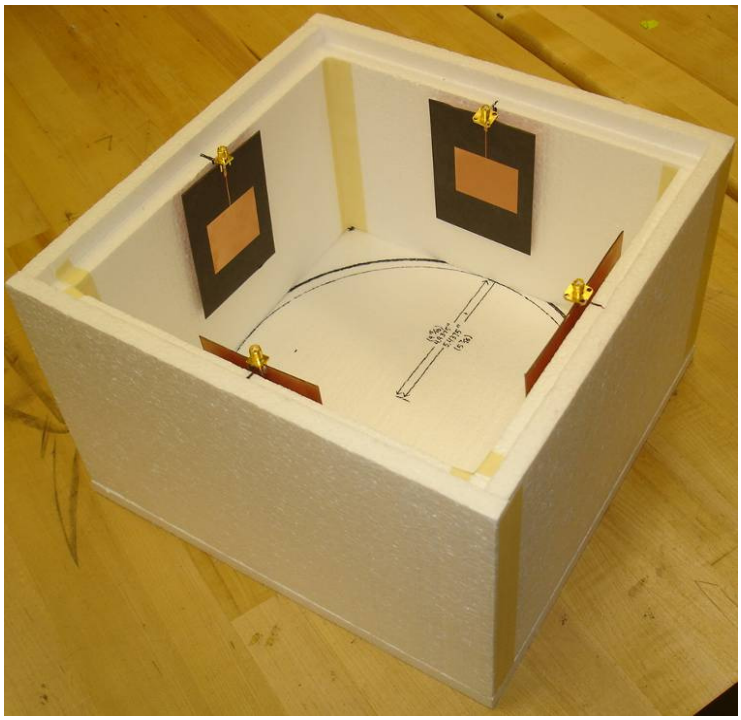


Figure 5.3. Side view of prototype experimental apparatus without source.

Similarly, to model the source, the quarter-wave wire antenna described in Section 4.2 is used. The pseudo omni-directional radiation pattern presented by this radiator ensures that all sensors receive the emitted signal and allow a covariance matrix to be generated from the data collected at each patch. However, due to the softness and flexibility of the Styrofoam box base, fixed positions can not be easily achieved consistently between experiments. Thus, a harder, clear 3/16" (0.5 cm) acrylic base is machined to fit inside the box. Because of the rigidity of the acrylic, six 5/8" holes are drilled in known locations using a precision mill to accommodate the bulkhead connector of the wire antenna. Table 5-1 shows the fixed positions of the holes drilled, where the wire antenna can be placed.

Letter Designator	Normalized Radius	Angle (°)
A	0.33	300
B	0.44	67.5
C	0.33	120
D	0.66	225
E	0.18	180
F	0.21	45

Table 5-1. Hardware source positions, drilled into acrylic base.

Finally, to hold the connector in place, a 50 Ω male N-type cable is mated to the bottom surface with a 1/16" thick washer of the same diameter as the hole. This cable is also used to excite the antenna with a CW signal from an Agilent Technologies N5182A MXG Vector Signal Generator. However, because none of the patches are resonant at precisely 2.45 GHz, and the wire antenna has a relatively large bandwidth, the average resonant frequency (2.44 GHz) is used as the operating frequency of the CW signal presented by the signal generator.

As a final consideration, the base has also been positioned such that the center of the wire antenna and the center of the patch antenna are approximately on the same plane, to ensure that the signal is being received in the main beam of the sensors with minimal polarization loss. To do this, two additional 2.5" x 1 7/8" x 1" blocks have been milled out of a durable plastic known as Delrin to elevate the acrylic platform. Additionally, the microstrip patch antenna substrates have also been evenly trimmed to approximately the same height, such that they are level when placed flush against the base. The final product and a cross sectional view of the base with leveling blocks in place is shown in Figure 5.4 and Figure 5.5.

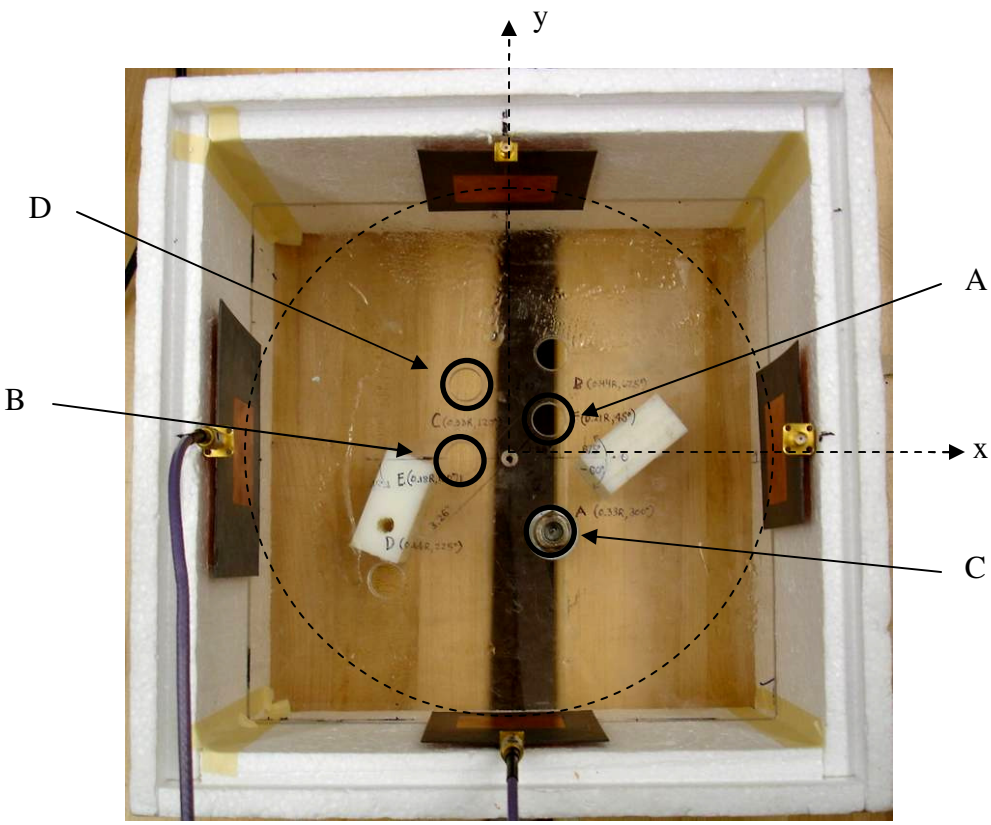


Figure 5.4. Top view of final apparatus with acrylic base and fixed source positions.

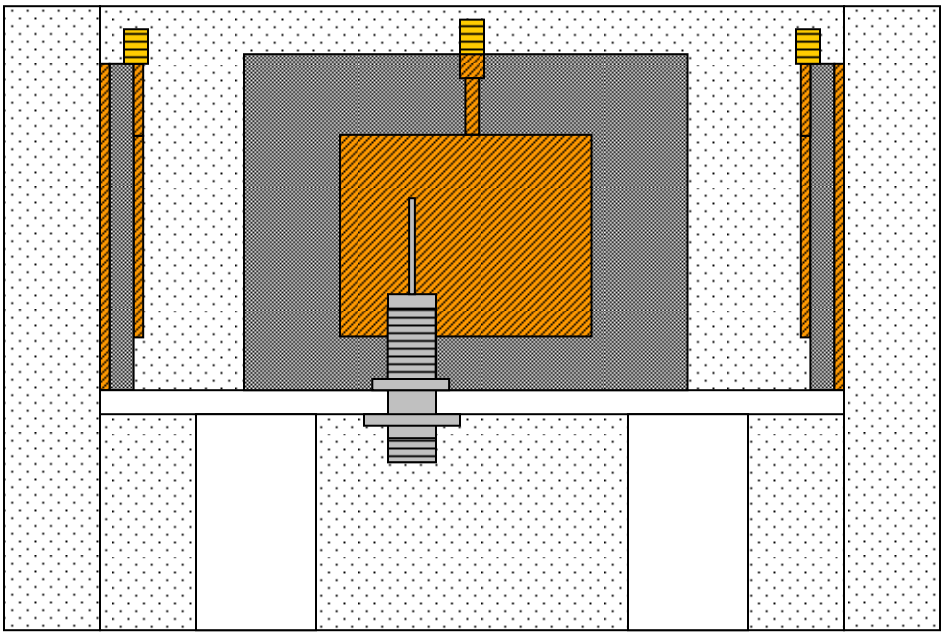


Figure 5.5. Cross sectional view of apparatus with new acrylic base.

5.2 Experimental Procedure

The following block diagram shows the configuration of each piece of test equipment with the apparatus described in the previous section:

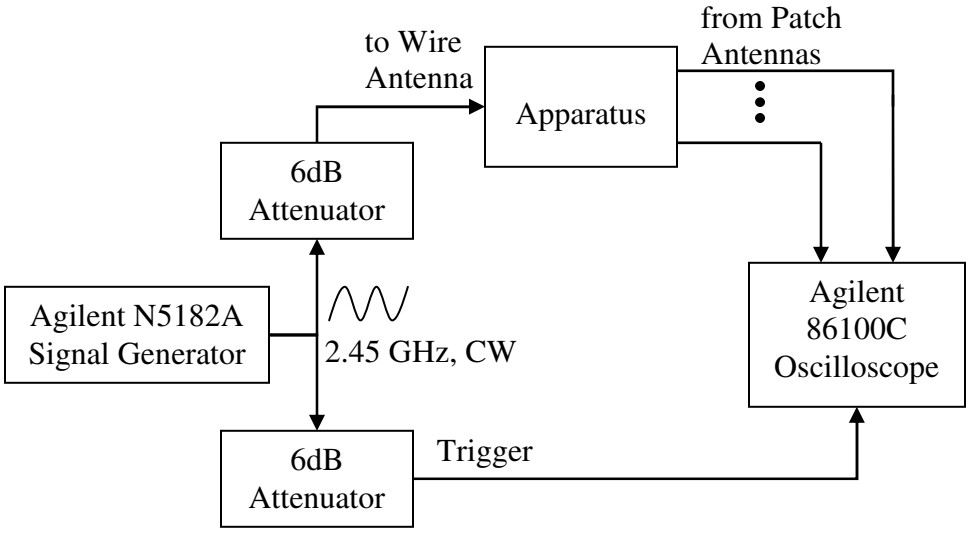


Figure 5.6. Block diagram of experiment setup.

Before connecting anything, the 20 GHz 86112A module, used to collect the data, is electronically calibrated through the Infinium 86100C. Then, the first two sensors are connected to the oscilloscope's sampling module using two Gore Tex SMA coaxial cables of the same length. To ensure the timing delay between sensors has been accurately captured, the oscilloscope is also externally triggered by the source signal. This is achieved by tapping the line from the signal generator to provide both an excitation for the wire antenna and a trigger for the oscilloscope. Using a female-male-female N-type tee in conjunction with two HP 8491A 6 dB attenuators, both lines are also electrically isolated. On the opposite end of the 50 Ω source cable, an RG223/U coaxial cable feeds the oscilloscope trigger. Before enabling the signal, the patches are aligned diametrically, horizontally, and vertically to accurately model a circular sensor array and

minimize any polarization loss. Additionally, to ensure the acrylic surface containing the source was level, the height from the base to the inside edge of the box has been consistently measured between experiments to be approximately 3.5625” (9.05 cm).

Next, the RF output is enabled and the signal generator transmits a CW signal at 2.44 GHz and 0 dBm to both the oscilloscope and the wire antenna. Using the oscilloscope, approximately six CW cycles are captured using the maximum number of data samples (4050) and a 64 tapped delay line smoothing filter. Screen captures of all the waveforms with amplitude measurements have been taken, and Comma Separated Values (CSV) text files containing the data has also been saved for each sensor. This process is repeated for the next pair of sensors using the same SMA cables to ensure measurement consistency, and then repeated for again for signal amplitudes of 5 dBm, 10 dBm and 15 dBm, after which the position of the source is changed. By the end of the experiment, data for 4 different sensors, at 4 different signal amplitudes in 6 different source positions, totaling 72 CSV files, have been recorded.

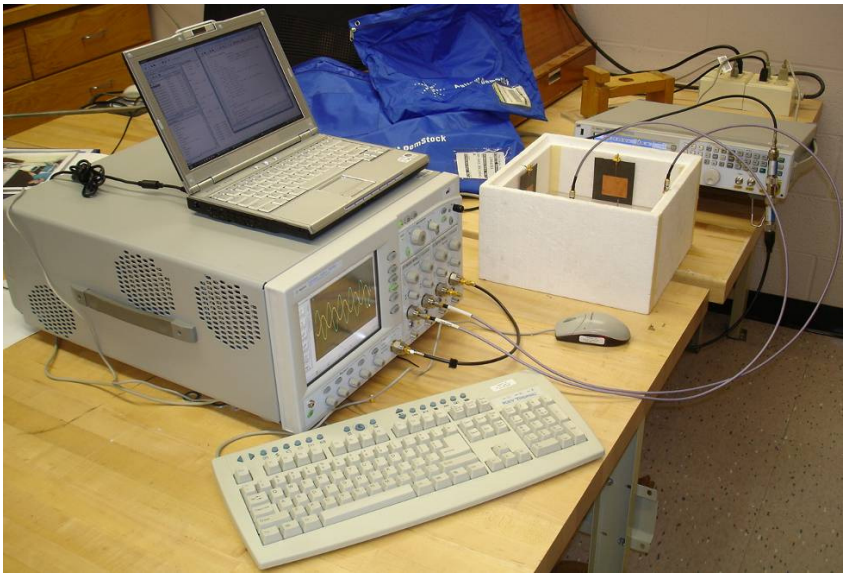


Figure 5.7. Actual experimental setup.

Finally, the results are ready for use in the localization algorithm. The CSV files contain the filtered voltage values of the CW wave observed at each sample, and are imported into MATLAB, where the data is used to estimate the covariance matrix, and localize the source with the proposed algorithm.

5.3 Hardware Results

The following section summarizes the results obtained in the hardware experimentation. The parameters used to describe the sensor array when running the MATLAB algorithm are given in Table 5-2 below, followed by sample data collected via the oscilloscope and finally, algorithm localization results.

Parameter	Value	Units
M	4	sensors
I	1	sources
R	1.02λ	m
N	4050	samples

Table 5-2. Hardware localization test parameters.

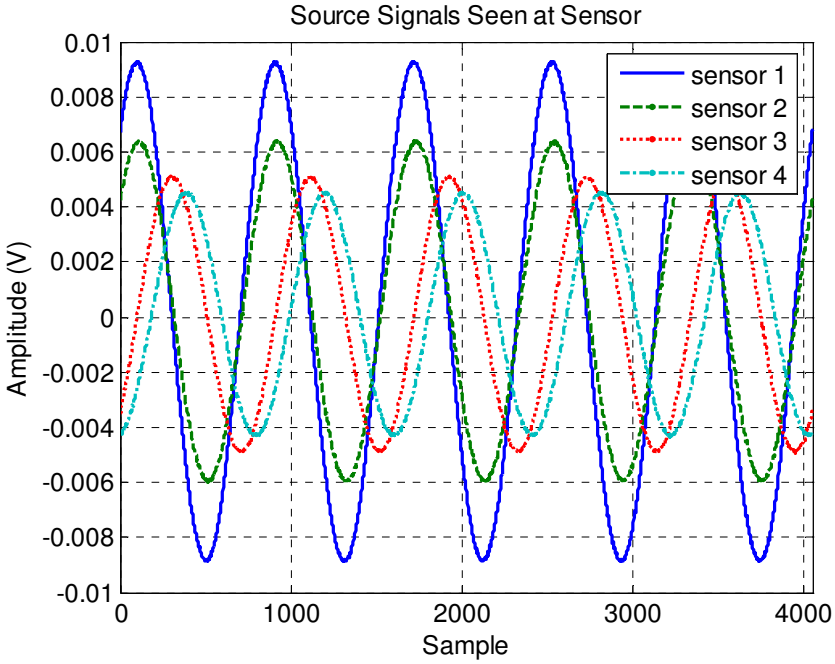


Figure 5.8. 2.44 GHz, 0 dBm CW signal seen at each sensor from source position A.

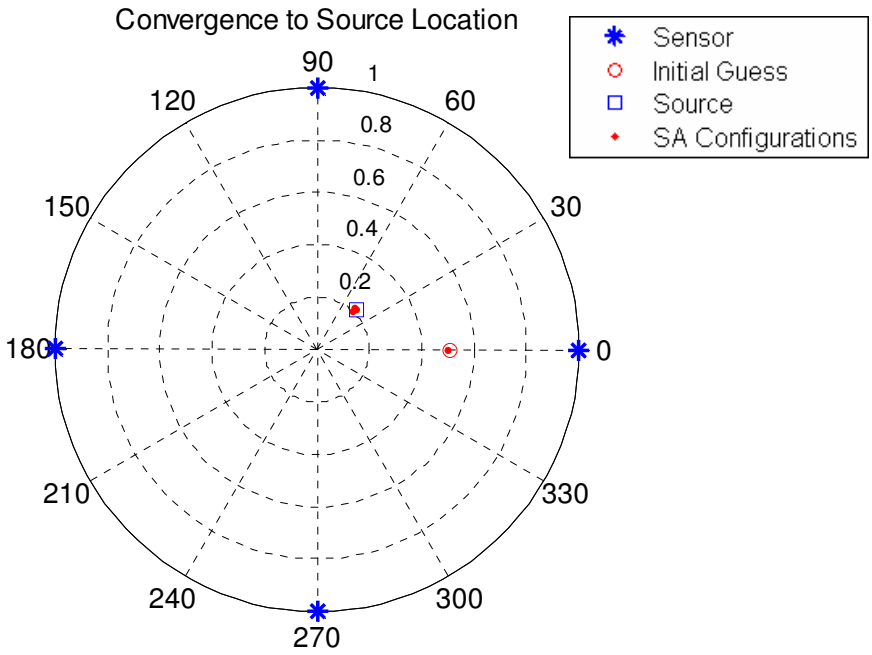


Figure 5.9. Normalized SA constellation of lowest cost configurations for position A.

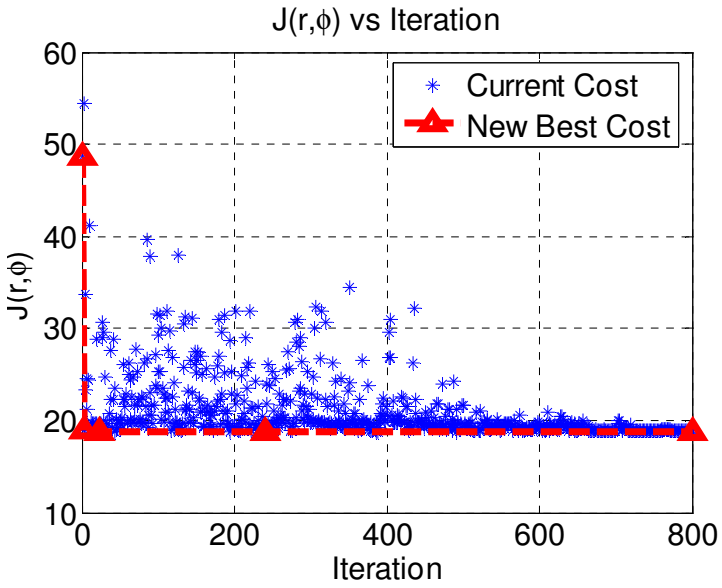


Figure 5.10. SA learning curve for localization of position A.

As shown in the learning curve above, the cost function reduces immediately and retains the lowest cost value, producing an acceptable solution very close to the true location as seen in Figure 5.9. The lowest cost configuration found by SA occurrs at $(0.201R, 46.4°)$,

only 4.19% from the actual radius and 3.09% from the actual angle. From this position, convergence to the true position is achieved by conjugate gradients, as shown below in Figure 5.11 and Figure 5.12:

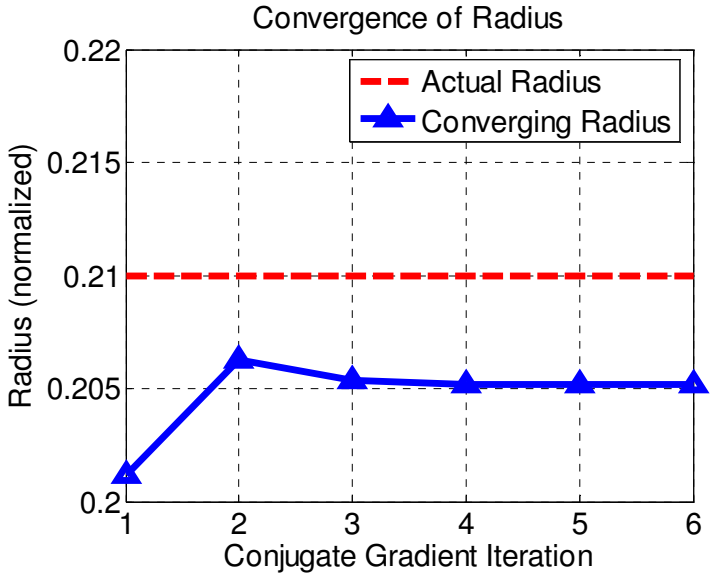


Figure 5.11. Convergence of position F radius by CG using final SA solution.

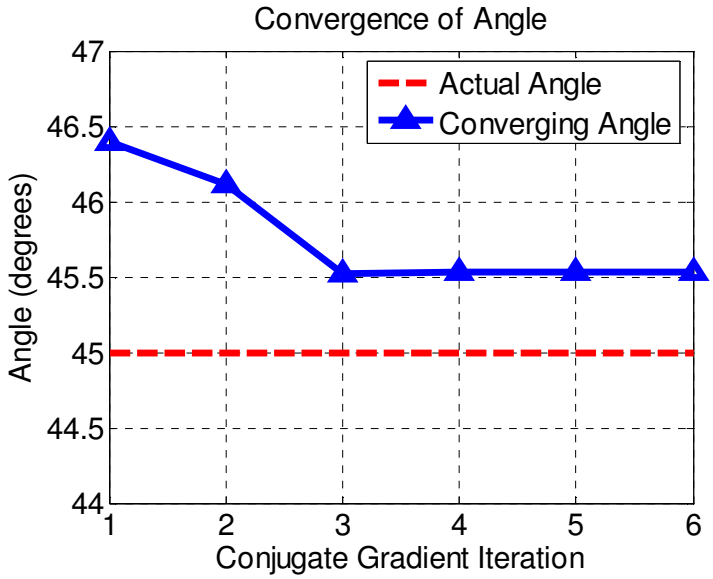


Figure 5.12. Convergence of position F angle by CG using final SA solution.

For a single trial, positions E and F could not be successfully localized with the algorithm; however, the results for four of the six positions, including position A shown above, are given in Table 5-3:

Source Location (Letter)	Source Location		Source Location Localized by Algorithm		Error (%)	
	Normalized Radius	Angle (°)	Normalized Radius	Angle (°)	Radius	Angle
A	0.21	45	0.2052	45.53	2.38	1.11
B	0.18	180	0.1732	182.1	3.79	1.16
C	0.33	300	0.3053	311.6	7.49	3.88
D	0.33	120	0.3160	137.5	4.23	14.58

Table 5-3. Hardware convergence results for positions A, C, E, and F.

Additionally, for all the positions listed above, average percent errors were obtained by taking the mean position error for four different signal amplitudes; these results are presented in Table 5-4.

Letter Designator	Position Error (%)	
	Radius	Angle
A	28.5	0.606
B	10.4	3.31
C	4.82	4.34
D	1.64	13.2

Table 5-4. Average position errors seen at each fixed source location.

In general, several positions have been effectively localized, with the exception of position E and position F. Coincidentally, these two sources have the longest radial displacements, which suggests that the source signal may not have been received in the main lobe of its closest sensors. Additionally, though position A and C are diametrically opposed to one another, position A has been resolved more closely than C. This confirms that inconsistencies between sensors exist during the experiment. The significance of these conjectures will be discussed further in the next chapter.

6 Discussion

6.1 Comparison of Results

When comparing simulated and hardware results, it has been determined that localization of single stationary sources with simulated data performs much better than localization with real data taken in experimentation, as expected. The simulated learning curve shown in Figure 3.3 has much lower cost values and settles to lower cost values slightly faster than the hardware learning curve of Figure 5.10. Though they occurred in the same number of iterations, the simulated convergence of the radial and angular parameters shown in Figure 3.4 and Figure 3.5 are also significantly closer to the true values than the hardware results of Figure 5.11 and Figure 5.12. Based on these error percentages, the simulated results appear to have over 1000 times more spatial resolution than hardware. Though the simulated source signal uses a $40dB$ SNR, which may have helped to reduce the error percentage, in hardware trials, the averaging feature of the oscilloscope averages out the presence of noise on the waveforms captured at each sensor. Therefore, it is unlikely for noise to be a significant cause of the increased error in hardware. Instead, it appears that the difference in performance can be attributed to a simplified simulation model, which is unable to account for the inconsistencies inherent in hardware experimentation.

6.2 Sources of Error

The simulation has been modeled with the purpose of 2-D spatial parameter estimation, where the sensors and the source are assumed to be uniform isotropic

radiators on the same plane. However, this has not been the case in hardware experimentation; not only did the array occupy a 3-D space and not just a single plane, inconsistent sensor characteristics, such as directivity are inherent in the hardware design as shown in Chapter 4. Because of the simplified simulation model, any inkling of non-uniformity or inconsistency among sensors influences the attenuation factor associated with the signal received to be biased. This is substantial because the desired spatial parameters are embedded in this attenuation factor, so any misrepresentation of it will impact the accuracy of the spatial estimation made by the proposed algorithm. As such, a significant portion of the errors seen in the hardware experiment are attributed to the design of the antennas, as well as the overall construction of the system.

In practice, the sensors used are not uniform as assumed in simulation, because each patch antenna has been trimmed and soldered individually by hand. Therefore, the edges are not perfectly straight, dimensions are not exact and even the connectors used are not consistent between patches. However, besides physical dimensions, inconsistencies also existed in the measured radiation patterns, as shown in Figure 4.15; the gain pattern is not uniform, and varied between sensors. Thus, if a source signal arrived in the main beam of a sensor, it is perceived to be closer than if the source signal arrives in a sidelobe, where the gain is significantly lower. This also holds true of the source emitter; if the radiation pattern of the wire antenna is not perfectly omni-directional with a uniform gain, the transmitted signal to a particular sensor may be weaker than to another. Similarly, the strength of signals received at a fixed distance and frequency also varies from sensor to sensor due to inconsistent impedance matching. In

other words, because not all sensors are resonant at the same operating frequency with the same VSWR, as seen in Table 4-2, some sensors lose more power due to reflection than others. Nonetheless, even though each component of the experimental apparatus carries its own potential for error, the arrangement of the system as a whole can significantly compound these errors.

In particular, the entire apparatus may not have been perfectly aligned. Inconsistencies in the Styrofoam box dimensions, unleveled testing platforms and crooked patch mounting are all potential sources of error in this experiment. Additionally, because the wire antenna is not perfectly straight, and the patch antennas are flexible, more measurement inconsistencies are presented. Since the algorithm assumes a 2-D sensor array, any one of these factors could cause polarization loss by not maintaining the source and sensor precisely on the same plane. Additionally, skewed results may be achieved if the patches are offset in their uniform circular arrangement. Methods to improve the accuracy of this algorithm by minimizing these inconsistencies are explored in the next section.

6.3 Algorithm Limitations and Improvements

Thus far, the practical difficulties with localizing single stationary sources have been discussed in detail. These errors can be summarized as biased attenuation factors, due to non-uniform radiation patterns, polarization loss, and sensor inconsistencies. These issues present the most critical limitation to the algorithm because they directly impact the spatial parameters. However, the most effective way to resolve this issue is through antenna calibration to determine an accurate array manifold. With this

information, the source signal can be normalized by the gain of each sensor, allowing a more accurate representation of the attenuation factor and consequently, improved spatial resolution. Additionally, to limit the effect of inconsistencies between sensors, the microstrip patch antennas used can also be designed with a larger bandwidth. Moreover, it is presumed that the resolution and speed of the algorithm can also be increased by using more sensors. This would serve to further limit the search space as well as average out any inconsistencies between sensors. However, the spacing between sensors in larger arrays must be sufficient to eliminate any mutual coupling, which has been negligible thus far for four sensors at the tested frequency.

Though multiple sources and tracking have not been tested in hardware, simulated results suggested that several limitations in these applications also exist. Multiple source localization suffers the most from proximate sources, though it was shown in Table 3-5 that a significant power difference between adjacent sources improves spatial resolution:

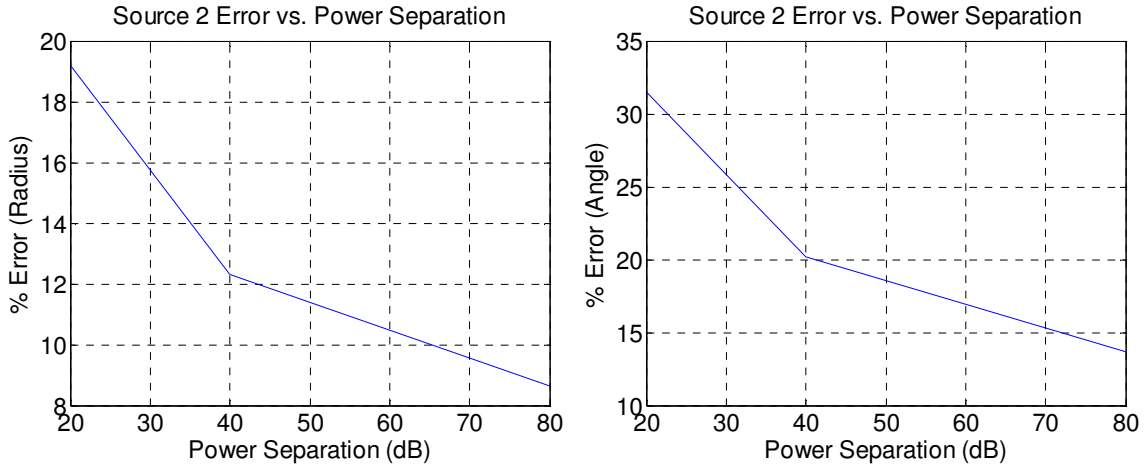


Figure 6.1. Source 2 position error with respect to power separation from source 1.

Although the algorithm presented is susceptible to the presence of multiple, proximate sources, it is also significant to note that the proposed algorithm makes no

assumptions about any beamforming techniques used by the sensors. Thus, in addition to the eigendecomposition of the input covariance matrix, sidelobe cancellation may significantly improve the multiple source localization algorithm or present a better alternative by temporarily nulling out localized sources, and allowing a new covariance matrix to be formed. However, research in beamformers has shown that a large eigenvalue spread of the input covariance matrix, recommended for the proposed approach, is detrimental to the convergence speed of adaptive arrays. Fortunately, several types of beamformers are available today to compensate for this requirement, as shown in the Automatic Gain Control (AGC) Applebaum Beamformer [37] and the Cascaded Applebaum Array [38].

Finally, simulations have also shown that tracking non-stationary sources is feasible, but difficult if the sources are moving too quickly, as seen in Figure 6.2 below:

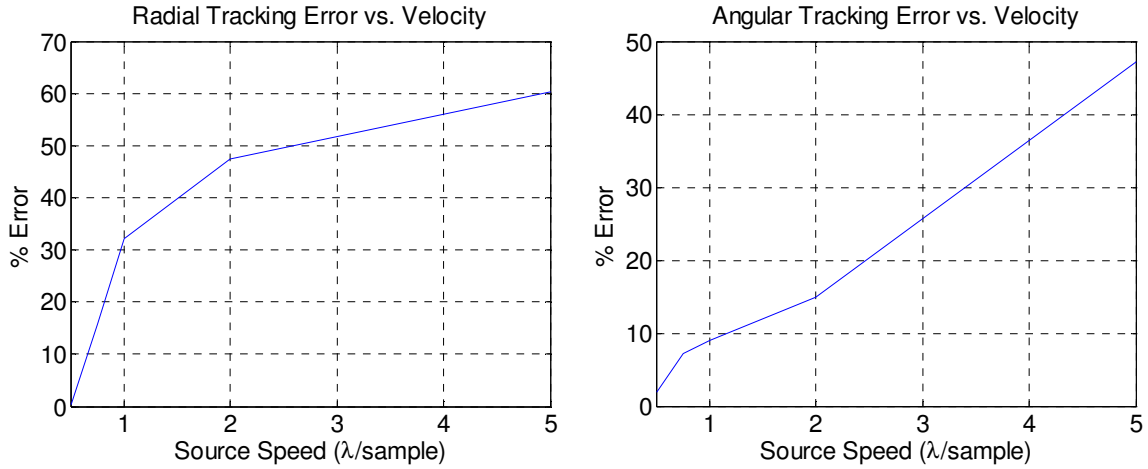


Figure 6.2. Tracking error vs. source speed per covariance matrix sample.

One way to improve the tracking resolution is to use a higher frequency sampling rate; so long as this rate remains faster than the velocity of the source, it can localize mobile sources within a sufficient tolerance. Alternately, more frequent estimates of the

covariance matrix can be made with fewer samples at the same rate. In this approach, however, the number of computations increases and the accuracy of the covariance matrix becomes a concern.

6.4 Application of This Work

The simulation and testing performed with the proposed localization algorithm in uniform circular sensor arrays has enormous potential, however several limitations in hardware modeling still exist. The research presented in this publication is meant to be a foundation for future work to explore and possibly eliminate these limitations. If this is achieved, the research completed here will provide an alternate approach to the problem of source localization for use in networking and mobile telephony applications.

7 Conclusions

7.1 Summary

A hybrid heuristic algorithm using a modified simulated annealing algorithm and conjugate gradients has been used for source localization in a 2-D uniform circular sensor array. It has been shown that a stationary source with an unknown location within the array could be localized by minimizing the normalized MSE cost function. Simulated results indicate that the convergence of the algorithm to the actual source locations occurred within 1% of a wavelength and 1° of the angular position in few iterations. Additionally, the simulated CRLB has also been presented, which suggests a low susceptibility to noise at reasonable SNRs.

The same algorithm has also been extended for detection of multiple stationary sources through eigendecomposition of the covariance matrix. In this application, it has been shown that the algorithm is sensitive to proximate sources, which could constructively form global minima on the normalized MSE performance surface in locations other than the true position of the sources. However, for proximate sources that are not too close, the spatial resolution can be improved by increasing the power separation between source signals. However, to improve this algorithm, the use of an auxiliary beamformer has been proposed to temporarily null localized sources and construct a better estimate of the covariance matrix for localizing additional sources.

Non-stationary performance of the proposed algorithm has also been explored, to include tracking capabilities for individual moving sources exhibiting slow linear and

non-linear motion. For a fixed covariance matrix sampling rate, the performance of the tracking algorithm has been evaluated for various rotational speeds. Consequently, it has been shown that if the source has traveled too far or too fast between matrix samples, the algorithm diverges.

Finally, four microstrip patch antennas have been constructed and arranged in a ring as receive sensors, with a wire antenna acting as the transmitting source inside the ring. The wire antenna has been excited with a CW signal and the data observed at each sensor has been collected by a digital sampling oscilloscope in a pairwise fashion. Several experimental trials have been completed in this manner using different signal amplitudes and positions. Results validate the algorithm, and also show that it is sensitive to sensor calibration, non-uniformity between sensors and polarization loss.

7.2 Future Work

Though a significant amount of research has been done with the proposed algorithm, there is still much that can be done to evaluate and improve its performance as well as extend its application. This section will outline some of the future work that should be considered, as well as the significance of each piece.

7.2.1 Sensor Calibration

The problem of sensor calibration, first discussed in Section 6.2, is a major factor in the accuracy of a practical sensor array. This algorithm assumes a uniform radiation pattern, however not all sensors have uniform directivity patterns. Even among those that do, fluctuations in gain at different angles of arrival are more than likely. Therefore, it is desirable to normalize the source signals seen at each sensor with their respective gains.

However, in order to accomplish this, the direction of arrival must be known first, which is not the case. Alternately, the directivity pattern of the sensor also can be incorporated as part of the normalized MSE cost function, making the amplitude at the m-th sensor relative to the direction of arrival as a function of r_i, θ_m, and φ_i. Through simple trigonometric properties, the angle of arrival, with respect to the m-th sensor, can easily be derived as:

$$\beta = \pi - \sin^{-1}\left(\frac{r_i \sin(\theta_m - \varphi_i)}{\sqrt{r^2 + r_i^2 - 2rr_i \cos(\theta_m - \varphi_i)}}\right) \tag{7-1}$$

However, implementation becomes difficult because by introducing variable amplitudes, the normalized MSE performance surface may also be minimized by choosing a set of values that minimizes the gain; therefore, the global minimum of the normalized MSE cost function may not be entirely representative of the spatial parameters in the attenuation factor alone.

7.2.2 Comparison of Optimization Algorithms

As seen in Chapter 2, many optimization algorithms exist, with comparable performance. In practical applications, speed is always a key concern; as such, the performance of the SA algorithm with different types of annealing schedules (for example: Gaussian, exponential or harmonic) can be attempted. Alternately, variants of GA introduced in Section 2.5.1 can also be paired with conjugate gradients or an alternative conventional minimization algorithm to improve the rate of convergence.

7.2.3 Sidelobe Cancellation and Beamforming

First mentioned in Section 3.5, adaptive sidelobe cancellation can be considered as a means of temporarily ignoring localized sources, given the spatial information provided by the algorithm proposed in this paper. The new received data, without the localized source, can be used to detect additional enclosed sources, without being skewed by the originally localized source. Initially, the performance of this approach can be compared to the multiple source localization algorithm proposed in Section 3.5. Then, the performance of both algorithms combined as suggested in Figure 3.8 to determine if the convergence of successive sources improves. Finally, with all enclosed sources localized, beamforming algorithms can be implemented to optimize the radiation patterns of each sensor as in [6]-[8], [37], and [38].

7.2.4 Elevation Plane

The work discussed in this publication concerned a 2-D sensor array, which provides a foundation, but is generally not very practical. Therefore, an extension of the model into an elevation plane, such that sources are able to impinge on the circular array from above and below, and not just within the sensor array may be valuable. This would also add a dimension for use in steerable smart antennas, which may be useful in locations where several elevation variations are present in the terrain features.

7.2.5 Multipath and Fading

Multipath is a real world concern that is not unique to this application, and is an issue common to all communications systems. It occurs when a transmitted signal is

reflected off of an object, such as a building or metal post while propagating. At the receiving end, the result is in the form of multiple correlated signals arriving at different times with different phases or attenuations. According to [39], in DOA estimation using MUSIC, multipath issues have a tendency to increase cross-correlation, possibly reducing the rank of the cross correlation matrix; under these conditions, the authors propose spatial smoothing [40] as a viable pre-processing technique to this potential problem. The impact of multipath effects on this algorithm and its ability to localize multiple sources should be explored, and the application of spatial smoothing should be considered as a potential solution.

References

[1] R. O. Schmidt, "Multiple Emitter Location and Signal Parameter Estimation," *IEEE Trans. On Antennas and Propagation,* vol. AP-34, no. 3, pp. 276-280, 1986.

[2] B. Ottersten and T. Kailath, "Direction-of-Arrival Estimation for Wide-Band Signals Using the ESPRIT Algorithm", *IEEE Trans. On Acoustics, Speech, and Signal Processing*, vol. 38, no. 2, pp. 317-327, Feb. 1990.

[3] H. Krim and M. Viberg, "Two Decades of Array Signal Processing Research," *IEEE Signal Processing Magazine*, pp. 67-94, July 1996.

[4] F. Gross, Smart Antennas for Wireless Communications, New York: McGraw-Hill Companies, Inc., 2005, p 210.

[5] B. D. Van Veen, and K. M. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering," *IEEE Acoustics Speech and Signal Processing Magazine*, April 1998, pp. 4-24.

[6] B. Widrow, P. E. Mantey, L. J. Griffiths, and B. B. Goode, "Adaptive Antenna Systems," *Proc. Of the IEEE*, vol. 55, no. 12, pp. 2143-2159, 1967.

[7] L. J. Griffiths, "A Simple Adaptive Algorithm for Real-Time Processing in Antenna Arrays," *Proc. Of IEEE*, vol. 57, no. 10, pp. 1696-1704, 1969.

[8] S. P. Applebaum, "Adaptive Arrays," *IEEE Trans. on Antennas and Propagation*, vol. 24, no. 9, pp. 585-598, 1976.

[9] G. C. Carter, "Time Delay Estimation," *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 29, no. 3, pp. 461-2, 1981.

[10] M. D. Farrell and R. M. Mersereau, "Comparison of Adaptive Array Processing Schemes for Land Mine Detection Using Hyperspectral Imagery," *Proc. of SPIE,* vol. 5913, 2005.

[11] J. Capon, "High resolution frequency wave number spectrum analysis," *Proc. IEEE*, vol. 57, pp. 1408-1418, 1969.

[12] V. F. Pisarenko, "The Retrieval of Harmonics from a Covariance Function," *Geophys. J. Roy. Astron. Soc.*, vol. 33, pp. 347-366, 1973.

[13] C. A. Balanis, Antenna Theory: Analysis and Design, 3rd ed., Hoboken, N.J.: John Wiley & Sons, Inc., 2005, pp. 817-20, 962-5.

[14] I. Ziskind and M. Wax, "Maximum Likelihood Localization of Multiple Sources by Alternating Projection," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. 36, no. 10, pp. 1553-1560, 1988.

[15] B. Widrow and S. D. Stearns, Adaptive Signal Processing, New Jersey: Prentice-Hall, Inc., 1985, pp. 46-63.

[16] C. E. Rasmussen, "Minimize," [Online document] 2006, Avaliable at HTTP: http://www.kyb.tuebingen.mpg.de/bs/people/carl/code/minimize/

[17] A. Di, "Multiple Source Location-a Matrix Decomposition Approach," *IEEE Trans. On Acsoustics, Speech and Signal Processing*, vol. 33, no. 5, pp. 1086-91, 1985.

[18] H. Krim, "A Data-Based Enumeration of Fully Correlated Signals," *IEEE Trans. On Acoustics, Speech, Signal Processing*, vol. 42, no. 7, pp. 1662-8, 1994.

[19] T. J. Shan, M. Wax, and T. Kailath, "On Spatial Smoothing for Directions of Arrival Estimation of Coherent Signals," *IEEE Trans. On Acoustics, Speech and Signal Processing*, vol. 33, no. 4, pp 806-11, 1985.

[20] M. Wax, T. J. Shan, and T. Kailath, "Spatio-Temporal Spectral Analysis by Eigenstructure Methods," *IEEE Trans. On Acoustics, Speech and Signal Processing*, vol. 32, no. 4, 1984.

[21] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Trans. On Automatic Control*, vol. AC-24, no. 6, pp. 843-854, 1979.

[22] C. R. Rao, C. R. Sastry, and B. Zhou, "Tracking the Direction of Arrival of Multiple Moving Targets," *IEEE Trans. On Signals Processing*, vol. 42, no. 5, pp. 1133-1144, 1994.

[23] J. R. Shewchuk, "An Introduction to the Conjugate Gradient Method Without the Agonizing Pain", Pittsburgh, Penn.: Carnegie Mellon University, 1994.

[24] E. G. Birgin and J. M. Martinez, "A Spectral Conjugate Gradient Method for Unconstrained Optimization," Applied Mathematics and Optimization, vol. 43, pp. 117-128, 1999.

[25] N. Andrei, "A Scaled Nonlinear Conjugate Gradient Algorithm for Unconstrained Optimization," Optimization, J. E. Martinez-Legaz, Ed., 2006.

[26] N. Andrei, "Conjugate Gradient Algorithms for Molecular Formation Under Pairwise Potential Minimization," Optimization, 2007.

[27] F. Busetti, "Genetic Algorithms Overview," [Online document] (2007), Available at HTTP: http://www.geocities.com/francorbusetti/gaweb.pdf

[28] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller and E. Teller, "Equation of State Calculations by Fast Computing Machines," J. Chem. Phys., vol. 21, no. 6, pp. 1087-1092, 1953.

[29] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, no. 4598, May, pp. 671-680, 1983.

[30] O. de Weck and C. Jilla, "Simulated Annealing: A Basic Introduction," [Online document] (2007), Available at HTTP: http://ocw.mit.edu/NR/rdonlyres/Aeronautics-and-Astronautics/16-888Spring-2004/5F6CFF91-524F-4792-859D-98331A73AC7C/0/l10a_sa.pdf

[31] U. W. Thonemann, "Finding Improved Simulated Annealing Schedules with Genetic Programming," *Proc. of First IEEE International Conference on Evolutionary Computation*, vol. 1, pp. 391-395, 1994.

[32] F. Busetti, "Simulated Annealing Overview," [Online document] (2007), Available at HTTP: http://www.geocities.com/francorbusetti/saweb.pdf

[33] S. Moins, "Implementation of a Simulated Annealing Algorithm for MATLAB," Linköping, Sweden: Linköping University Electronic Press, 2002.

[34] J. Q. Guo, and L. Zheng, "A modified simulated annealing algorithm for estimating solute transport parameters in streams from tracer experiment data," Environmental Modelling & Software, vol. 20, no. 6, Jun., pp. 811-815, 2004.

[35] R. Klemm, Principles of Space-Time Adaptive Processing, 2nd Ed., London: The Institution of Electrical Engineers, 2002, p 394.

[36] R. Bancroft and B. Bateman, "An Omnidirectional Planar Microstrip Antenna," *IEEE Trans. on Antennas and Propagation*, vol. 52, no. 11, pp. 3151-3153, 2004.

[37] K. Lee, D. Han, and M. Cho, "Adaptive AGC Applebaum Array," *Military Communications Conference Proceedings, MILCOM 1999*, vol. 1, pp. 661-5, 1999.

[38] M. W. Ganz, "Rapid Convergence by Cascading Applebaum Adaptive Arrays," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 30, no. 2, pp. 298-306, 1994.

[39] G. A. Mavridis, C. G. Christodoulou and M.T. Chryssomallis, "Performance Evaluation of MUSIC in Estimation of Direction of Arrival of Signals", *Antennas and Propagation Society International Symposium 2006, IEEE*, pp. 2541-2544, Jul. 2006.

[40] S. U. Pillai and B. H. Kwon, "Forward/Backward Spatial Smoothing Techniques for Coherent Signal Identification," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 37, no. 1, pp. 8-15, 1989.

Publications

D. Z. Zhu, S. A. Dianat and J. Venkataraman, "Localization of Stationary Sources within a Uniform Circular Sensor Array," *Proc. of IEEE Antennas and Propagation Society International Symposium 2007*, 11 June 2007.

D. Z. Zhu, S. A. Dianat and J. Venkataraman, "A Hybrid Heuristic Approach for Source Localization in Uniform Circular Sensor Arrays," *International Union of Radio Science 2007*, 26 July 2007.

Appendix A: MATLAB Code

```
function [X,A,sigs]=gendata(M,R,A,w0,r,phi,N,SNR);
%GENDATA(M,R,A,w0,r,phi,N,SNR)   Generates an M x N matrix of data,
where:
%   *R and r are normalized to the wavelength, lambda.
%
%   M = # of sensors
%   R = radius of the stationary circle
%   A = array of signal amplitudes
%   w0 = phase constant (Beta = 2 * pi / lambda)
%   r = array of distances to the source from the center of the circle
%   phi = array of angular displacements of the source, wrt +x axis
%   N = # of discrete time samples of data in each signal
%   SNR = array of Signal-to-Noise-Ratio of source signals in dB
I = length(r);
snr = 10.^(SNR/10);
sigs = A^2/(2*snr(1));
A(2:I) = sqrt(2*sigs*snr(2:I));
k = 1:N;
theta = (0:1:M-1)*2*pi/M;

for m=1:M
    num(m,:) = exp(-j*w0*sqrt(R^2+r.^2-2*R.*r.*cos(theta(m)-phi)));
    den(m,:) = sqrt(R^2+r.^2-2*R.*r.*cos(theta(m)-phi));
end
T = (num ./ den);
S = [];
X = zeros(M,N);
Rmn = zeros(M);
for i=1:I
    p = (rand(1,N)-0.5)*2*pi; % Uniformly distributed between -pi, pi
    S = [S; A(i)*exp(j*(0.34*pi*k + p))];
    X = X + T(:,i)*S(i,:);
    Rmn = Rmn + A(i).^2.*(num(:,i)*num(:,i)')./(den(:,i)*den(:,i)');
end
X = X + sqrt(sigs/2)*randn(M,N) + j*sqrt(sigs/2)*randn(M,N);
```

```
function [J,dJ]=Jerr(x,M,R,As,sigs,w0,Rapprox)
%JERR(x,M,R,A,w0,Rapprox)   Calculates the error function, J(r,phi):
%
%   x = [r phi]' (position to calculate J and dJ at)
%   M = # of sensors
%   R = radius of the circular antenna array
%   As = amplitude of the ith source presumed near x
%   sigs = estimated noise power
%   w0 = phase constant (Beta = 2 * pi / lambda)
%   Rapprox = Estimated cross-correlation between sensors

rad=x(1);
ang=x(2);
theta=(0:1:M-1)*2*pi/M;
for m=1:M
    num(m,:)=exp(-j*w0*sqrt(R^2+rad.^2-2*R.*rad.*cos(theta(m)-ang)));
    den(m,:)=sqrt(R^2+rad.^2-2*R.*rad.*cos(theta(m)-ang));
    dr(m,:)=2.*rad-2*R.*cos(theta(m)-ang);
    dp(m,:)=-2*R.*rad.*sin(theta(m)-ang);
end
Rmn=As^2*((num*num')./(den*den'))+sigs*eye(M); % estimated noise power
J=sum(sum(abs(Rmn-Rapprox).^2))/sum(sum(abs(Rapprox).^2));

dRrmn=(As^2*num*num'.*(den*den'.*(-j*w0/2*((dr./den)*ones(1,M)-
ones(M,1)*(dr./den)'))-
0.5*(dr./den*den'+den*(dr./den)'))./(den*den').^2;
dPrmn=(As^2*num*num'.*(den*den'.*(-j*w0/2*((dp./den)*ones(1,M)-
ones(M,1)*(dp./den)'))-
0.5*(dp./den*den'+den*(dp./den)'))./(den*den').^2;

dJ(1,1)=sum(sum(dRrmn.*conj(Rmn)+conj(dRrmn).*Rmn-dRrmn.*conj(Rapprox)-
conj(dRrmn).*Rapprox))/M^2;
dJ(2,1)=sum(sum(dPrmn.*conj(Rmn)+conj(dPrmn).*Rmn-dPrmn.*conj(Rapprox)-
conj(dPrmn).*Rapprox))/M^2;
```

```
function [con, fX, i] = minimize(X, f, length, varargin)

% Minimize a differentiable multivariate function.
% Usage: [X, fX, i] = minimize(X, f, length, P1, P2, P3, ... )
% where the starting point is given by "X" (D by 1), and the function
% named in the string "f", must return a function value and a vector of
% partial derivatives of f wrt X, the "length" gives the length of the
% run: if it is positive, it gives the maximum number of line searches,
% if negative its absolute gives the maximum allowed number of function
% evaluations. You can (optionally) give "length" a second component,
% which will indicate the reduction in function value to be expected in
% the first line-search (defaults to 1.0). The parameters P1, P2,
% P3, ... are passed on to the function f.
%
% The function returns when either its length is up, or if no further
% progress can be made (ie, we are at a (local) minimum, or so close
% that due to numerical problems, we cannot get any closer). NOTE: If
% the function terminates within a few iterations, it could be an
% indication that the function values and derivatives are not
% consistent (ie, there may be a bug in the implementation of your "f"
% function). The function returns the found solution "X", a vector of
% function values "fX" indicating the progress made and "i" the number
% of iterations (line searches or function evaluations, depending on
% the sign of "length") used.
%
% The Polack-Ribiere flavour of conjugate gradients is used to compute
% search directions, and a line search using quadratic and cubic
% polynomial approximations and the Wolfe-Powell stopping criteria is
% used together with the slope ratio method for guessing initial step
% sizes. Additionally a bunch of checks are made to make sure that
% exploration is taking place and that extrapolation will not be
% unboundedly large.
%
% See also: checkgrad
%
% Copyright (C) 2001 - 2006 by Carl Edward Rasmussen (2006-09-08).

% Modifications by Danny Zhu (2007-01-11)
%  *Adjusted convergence parameters
%  *Added storage of convergence path for plotting
%  *Included option to restart after every N iterations
%  *Included option of Birgin-Martinez Conjugate Gradient direction
%  *Formatted to fit appendix

% (C) Copyright 1999 - 2006, Carl Edward Rasmussen
%
% Permission is granted for anyone to copy, use, or modify these
% programs and accompanying documents for purposes of research or
% education, provided this copyright notice is retained, and note is
% made of any changes that have been made.
%
% These programs and documents are distributed without any warranty,
% express or implied.  As the programs were written for research
```

```
% purposes only, they have not been tested to the degree that would be
% advisable in any important application.  All use of these programs is
% entirely at the user's own risk.

INT = 0.1;% don't reevaluate within 0.1 of the limit of current bracket
EXT = 3.0;          % extrapolate maximum 3 times the current step-size
MAX = 250;                % max 20 function evaluations per line search
RATIO = 10;                                % maximum allowed slope ratio
N = 0;                          % restart every N>0 iterations (for small N)
SIG = 1e-8; RHO = SIG/2; % SIG and RHO control the Wolfe-Powell
% conditions. SIG is the maximum allowed absolute ratio between
% previous and new slopes (derivatives in the search direction), thus
% setting SIG to low (positive) values forces higher precision in the
% line-searches. RHO is the minimum allowed fraction of the expected
% (from the slope at the initial point in the linesearch). Constants
% must satisfy 0 < RHO < SIG < 1. Tuning of SIG (depending on the
% nature of the function to be optimized) may speed up the minimization;
% it is probably not worth playing much with RHO.

% The code falls naturally into 3 parts, after the initial line search
% is started in the direction of steepest descent. 1) we first enter a
% while loop which uses point 1 (p1) and (p2) to compute an
% extrapolation (p3), until we have extrapolated far enough (Wolfe-
% Powell conditions). 2) if necessary, we enter the second loop which
% takes p2, p3 and p4 chooses the subinterval containing a (local)
% minimum, and interpolates it, unil an acceptable point is found
% (Wolfe-Powell conditions). Note, that points are always maintained in
% order p0 <= p1 <= p2 < p3 < p4. 3) compute a new search direction
% using conjugate gradients (Polack-Ribiere flavour), or revert to
% steepest if there was a problem in the previous line-search. Return
% the best value so far, if two consecutive line-searches fail, or
% whenever we run out of function evaluations or line-searches. During
% extrapolation, the "f" function may fail either with an error or
% returning Nan or Inf, and minimize should handle this gracefully.

if max(size(length)) == 2, red=length(2); length=length(1); else red=1;
end
if length>0, S='Linesearch'; else S='Function evaluation'; end

con = [];
i = 0;                                     % zero the run length counter
ls_failed = 0;                       % no previous line search has failed
[f0 df0] = feval(f, X, varargin{:});  % get function value and gradient
fX = f0;
i = i + (length<0);                                          % count epochs?!
s = -df0; d0 = -s'*s;   % initial search direction (steepest) and slope
x3 = red/(1-d0);                           % initial step is red/(|s|+1)

while i < abs(length)                                  % while not finished
  i = i + (length>0);                                  % count iterations?!
  con(i,:)=X;
```

```
  X0 = X; F0 = f0; dF0 = df0;           % make a copy of current values
  if length>0, M = MAX; else M = min(MAX, -length-i); end

  while 1                      % keep extrapolating as long as necessary
    x2 = 0; f2 = f0; d2 = d0; f3 = f0; df3 = df0;
    success = 0;
    while ~success && M > 0
      try
        M = M - 1; i = i + (length<0);                 % count epochs?!
        [f3 df3] = feval(f, X+x3*s, varargin{:});
        if isnan(f3) || isinf(f3) || any(isnan(df3)+isinf(df3)),
error(''), end
        success = 1;
      catch                        % catch any error which occurred in f
        x3 = (x2+x3)/2;                            % bisect and try again
      end
    end
    if f3 < F0, X0 = X+x3*s; F0 = f3; dF0 = df3; end % keep best values
    d3 = df3'*s;                                                % new slope
    if d3 > SIG*d0 || f3 > f0+x3*RHO*d0 || M == 0 % done extrapolating?
      break
    end
    x1 = x2; f1 = f2; d1 = d2;                % move point 2 to point 1
    x2 = x3; f2 = f3; d2 = d3;                % move point 3 to point 2
    A = 6*(f1-f2)+3*(d2+d1)*(x2-x1);         % make cubic extrapolation
    B = 3*(f2-f1)-(2*d1+d2)*(x2-x1);
    x3 = x1-d1*(x2-x1)^2/(B+sqrt(B*B-A*d1*(x2-x1)));
    if ~isreal(x3) || isnan(x3) || isinf(x3) || x3 < 0
      x3 = x2*EXT;                           % extrapolate maximum amount
    elseif x3 > x2*EXT          % new point beyond extrapolation limit?
      x3 = x2*EXT;                           % extrapolate maximum amount
    elseif x3 < x2+INT*(x2-x1) % new point too close to previous point?
      x3 = x2+INT*(x2-x1);
    end
  end                                                % end extrapolation

% keep interpolating
  while (abs(d3) > -SIG*d0 || f3 > f0+x3*RHO*d0) && M > 0
    if d3 > 0 || f3 > f0+x3*RHO*d0                  % choose subinterval
      x4 = x3; f4 = f3; d4 = d3;              % move point 3 to point 4
    else
      x2 = x3; f2 = f3; d2 = d3;              % move point 3 to point 2
    end
    if f4 > f0
      x3 = x2-(0.5*d2*(x4-x2)^2)/(f4-f2-d2*(x4-x2));  % quad. interp.
    else
      A = 6*(f2-f4)/(x4-x2)+3*(d4+d2);            % cubic interpolation
      B = 3*(f4-f2)-(2*d2+d4)*(x4-x2);
      x3 = x2+(sqrt(B*B-A*d2*(x4-x2)^2)-B)/A;% num. error possible, ok!
    end
    if isnan(x3) || isinf(x3)
      x3 = (x2+x4)/2;       % if we had a numerical problem then bisect
    end
```

```
% don't accept too close
    x3 = max(min(x3, x4-INT*(x4-x2)),x2+INT*(x4-x2));
    [f3 df3] = feval(f, X+x3*s, varargin{:});
    if f3 < F0, X0 = X+x3*s; F0 = f3; dF0 = df3; end % keep best values
    M = M - 1; i = i + (length<0);                      % count epochs?!
    d3 = df3'*s;                                             % new slope
  end                                                % end interpolation

  if abs(d3) < -SIG*d0 && f3 < f0+x3*RHO*d0  % if line search succeeded
%     x0 = X; g0 = df0; % Bergin-Martinez Parameters
    X = X+x3*s; f0 = f3; fX = [fX' f0]';              % update variables
%     fprintf('%s %6i;  Value %4.6e\r', S, i, f0);

    % Bergin-Martinez parameters and CG direction
%     t = ((X-x0)'*(X-x0))/((df3-g0)'*(X-x0));
%     s = ((t*(df3-g0)-(X-x0))'*df3)/((df3-g0)'*(X-x0))*s - t*df3;

% Polack-Ribiere CG direction
    s = (df3'*df3-df3'*df0)/(df0'*df0)*s - df3;
    df0 = df3;                                          % swap derivatives
    d3 = d0; d0 = df0'*s;
    if (d0 > 0 || mod(i,N)==0)            % new slope must be negative
      s = -df0; d0 = -s'*s;          % otherwise use steepest direction
    end
    x3 = x3 * min(RATIO, d3/(d0-realmin)); % slope ratio but max RATIO
    ls_failed = 0;                      % this line search did not fail
  else
    X = X0; f0 = F0; df0 = dF0;            % restore best point so far
    if ls_failed || i > abs(length)% line search failed twice in a row
        break;                   % or we ran out of time, so we give up
    end
    s = -df0; d0 = -s'*s;                                % try steepest
    x3 = 1/(1-d0);
    ls_failed = 1;                             % this line search failed
  end
end
fprintf('\n');
```

```
clear
% SIMULATION PARAMETERS
M = 8;                  % # Sensors
N = 10000;              % # Data samples
SNR = 40;               % SNR in dB
A = 1;                  % Amplitude of source 1
R = 6;                  % Radius of sensor array, normalized to lambda
w0 = 2*pi;              % Lossless propagation constant
tol = .01;              % Radial convergence tolerance in wavelengths
K = 800;                % Total number of iterations

% GENERATE RANDOM SOURCE LOCATION (FIXED)
s = 1; % # of sources to test individually
% rand('seed',sum(100*clock));
% src = [rand(s,1)*R rand(s,1)*2*pi];
src = [0.3*R, pi/3];

bad=[];

for t = 1:s
  theta = (0:1:M-1)*2*pi/M;
  r = src(t,1); phi = src(t,2);
  [X,A,sigs] = gendata(M,R,A,w0,r,phi,N,SNR);
  Rapprox = (X*X') / N;
  [m,n] = find(abs(Rapprox)==max(max(abs(Rapprox))));
  Rmax = [m n];

  prev = Rmax(1)-1; next = Rmax(1)+1;
  switch(Rmax(1))
    case 1; prev = M;
    case M; next = 1;
  end
  dp = rerr(Rapprox(prev,prev),max(max(Rapprox)));
  dn = rerr(Rapprox(next,next),max(max(Rapprox)));
  switch(dn<dp)
    case 1; Rmax(2)=next;
    case 0; Rmax(2)=prev;
  end
  if(abs(diff(Rmax))>1), theta(1)=2*pi; end  % use appropriate theta(1)
  x = [R/2 theta(Rmax(1))];
  bcs = x(1,2) + pi/M * [-2 2];

  % SIMULATED ANNEALING
  [sa,E,cost_index]=Anneal(x,bcs,'Jerr',K,M,R,A,sigs,w0,Rapprox);
  % CONJUGATE GRADIENT HYBRIDIZATION
  cg = minimize(sa(end,:)','Jerr',K,M,R,A,sigs,w0,Rapprox);

% Remove false errors
  cg(:,2) = (mod(cg(:,2)-(cg(:,1)<0)*pi+2*pi,2*pi));
  cg(:,1) = ((cg(:,1)>0).*cg(:,1)-(cg(:,1)<0).*cg(:,1));

  if(abs(src(t,1)-cg(end,1)) > tol || abs(src(t,2)-cg(end,2) > pi/180))
```

```
      bad = [bad' t]'; % Track results outside of converging tolerance
  end
end

disp(['Success Rate: ' num2str((1-length(bad)/s)*100) '%']);

% POLAR PLOT OF ANNEALING CONSTELLATION, CG CONVERGENCE
figure(1)
plotsensors(M,1,x(t,2),x(t,1)/R,'or'); hold on;
polar(src(t,2),src(t,1)/R,'sb');
polar(sa(:,2),sa(:,1)/R,'.r');
legend('Sensor','Initial Guess','Source','SA Configurations',...
    'Location','NorthEastOutside');
title('SA Convergence to Source Location','FontSize',12);
set(gca,'FontSize',12);

figure(2)
plot(1:length(E),E,'*b','MarkerSize',8); hold on;
plot([cost_index' length(E)],[E(cost_index) E(cost_index(end))],'--
^r','LineWidth',4,'MarkerSize',9);
legend('Current Cost','New Best Cost')
xlabel('Iteration','FontSize',16);
ylabel('J(r,\phi)','FontSize',16);
title('J(r,\phi) vs Iteration','FontSize',16);
axis([1 length(E) 0 5]);
set(gca,'FontSize',16);
grid on;

% RADIUS & ANGLE CONVERGENCE PLOTS
figure(3)
plot(ones(1,length(cg))*src(t,1)/R,'--r','LineWidth',4); hold on;
plot(cg(:,1)/R,'-b^','LineWidth',4,'MarkerSize',9);
legend('Actual Radius','Converging Radius');
xlabel('Conjugate Gradient Iteration','FontSize',16);
ylabel('Radius (normalized)','FontSize',16);
title('Convergence of Radius','FontSize',16);
set(gca,'FontSize',16);
grid on;

figure(4)
plot(ones(1,length(cg))*src(t,2)*180/pi,'--r','LineWidth',4); hold on;
plot(cg(:,2)*180/pi,'-b^','LineWidth',4,'MarkerSize',9);
legend('Actual Angle','Converging Angle');
xlabel('Conjugate Gradient Iteration','FontSize',16);
ylabel('Angle (degrees)','FontSize',16);
title('Convergence of Angle','FontSize',16);
set(gca,'FontSize',16);
grid on;
```

```
function [err]=SAMscript(src)
% SIMULATION PARAMETERS
M = 8;                  % # Sensors
N = 10000;              % # Data samples
SNR = [80 20];          % SNRs in dB
sort(SNR,'descend');    % Store SNRs in descending value
A = 1;                  % Amplitude of source with highest SNR
R = 6;                  % Radius of circular sensor array
w0 = 2*pi;              % Lossless propagation constant
tol = .05;              % Radial convergence tolerance in wavelengths
K = 800;                % Total number of iterations

bad=[];
I = length(src);
r = src(:,1); phi = src(:,2);
[X,A,sigs,Rmn] = gendata(M,R,A,w0,r,phi,N,SNR);
Rapprox = (X*X') / N;

% figure(1);
% plotsensors(M,1);
% key{1} = 'Sensor';

for t=1:I;
  theta = (0:1:M-1)*2*pi/M;
  if(t>1)
      [U,S,V] = svd(Rapprox);
      Rapprox = Rapprox - U(:,1)*S(1)*V(:,1)';
  end

  [m,n] = find(abs(Rapprox) == max(max(abs(Rapprox))));
  Rmax = [m n];
  prev = Rmax(1)-1; next = Rmax(1)+1;
  switch(Rmax(1))
    case 1; prev = M;
    case M; next = 1;
  end
  dp = rerr(Rapprox(prev,prev),max(max(Rapprox)));
  dn = rerr(Rapprox(next,next),max(max(Rapprox)));
  switch(dn<dp) % identify next closest sensor
    case 1; Rmax(2)=next;
    case 0; Rmax(2)=prev;
  end
  if(abs(diff(Rmax))>1), theta(1) = 2*pi; end % use appropriate theta(1)
  x = [R/2 theta(Rmax(1))];
  bcs = x(1,2) + pi/M * [-2 2];

  % SIMULATED ANNEALING
  [sa,E,cost_index]=Anneal(x,bcs,'Jerr',K,M,R,A(t),sigs,w0,Rapprox);

  % CONJUGATE GRADIENT HYBRIDIZATION
  cg = minimize(sa(end,:)','Jerr',K,M,R,A(t),sigs,w0,Rapprox);
%   [sa; cg]
```

```
% Remove false errors
  cg(:,2) = (mod(cg(:,2)-(cg(:,1)<0)*pi+2*pi,2*pi));
  cg(:,1) = ((cg(:,1)>0).*cg(:,1)-(cg(:,1)<0).*cg(:,1));
%   disp(['Solution ' num2str(t) ': ' num2str([cg(end,1) cg(end,2)])]);

%   if(abs(src(t,1)-cg(end,1))>tol || abs(src(t,2)-cg(end,2))>pi/180)
%     bad = [bad; t];  % track results outside of converging tolerance
%     disp('FAILED!')
%   else
%     disp('OK!')
%   end
  err=100*[abs(src(t,1)-cg(end,1))/src(t,1) abs(src(t,2)-
cg(end,2))/src(t,2)];
%   disp(['dr = ' num2str(err(1)), '%; dp = ', num2str(err(2)),'%'])
% POLAR PLOT OF ANNEALING CONSTELLATION, CG CONVERGENCE
%   figure(1); hold all;
%   polar(x(1,2),x(1,1)/R,'or')
%   polar(src(t,2),src(t,1)/R,'sb');
%   polar(sa(:,2),sa(:,1)/R,'.r');
%   key = {key{:}, ['Initial Guess #' num2str(t)], ['Source #'
num2str(t) ' (' num2str(SNR(t)) 'dB)'], ['Source #' num2str(t) ' SA
Configs']};
%
%   figure(3*t-1)
%   plot(1:length(E),E,'*b','MarkerSize',8); hold on;
%   plot([cost_index' length(E)],[E(cost_index) E(cost_index(end))],'--
^r','LineWidth',4,'MarkerSize',9);
%   legend('Current Cost','New Best Cost')
%   xlabel('Iteration','FontSize',16);
%   ylabel('J(r,\phi)','FontSize',16);
%   title('J(r,\phi) vs Iteration','FontSize',16);
%   axis([1 length(E) 0 5]);
%   set(gca,'FontSize',16);
%   grid on;
%
%   % RADIUS & ANGLE CONVERGENCE PLOTS
%   figure(3*t)
%   subplot(1,2,1)
%   plot(ones(1,length(cg))*src(t,1)/R,'--r','LineWidth',4); hold on;
%   plot(cg(:,1)/R,'-b^','LineWidth',4,'MarkerSize',9);
%   legend(['Source #',num2str(t),' Radius'],'Converging Radius');
%   xlabel('Conjugate Gradient Iteration','FontSize',16);
%   ylabel('Radius (normalized)','FontSize',16);
%   title(['Convergence of Radius'],'FontSize',16);
%   set(gca,'FontSize',16);
%   grid on;
%
%   subplot(1,2,2)
%   plot(ones(1,length(cg))*src(t,2)*180/pi,'--r','LineWidth',4);
%   hold on;
%   plot(cg(:,2)*180/pi,'-b^','LineWidth',4,'MarkerSize',9);
%   legend(['Source #',num2str(t),' Angle'],'Converging Angle');
%   xlabel('Conjugate Gradient Iteration','FontSize',16);
```

```
%   ylabel('Angle (degrees)','FontSize',16);
%   title(['Convergence of Angle'],'FontSize',16);
%   set(gca,'FontSize',16);
%   grid on;
end

% figure(1);
% legend(key,'Location','NorthEastOutside');
% title('SA Convergence to Source Location','FontSize',12);
% set(gca,'FontSize',12);

% disp(['Success Rate: ' num2str((1-length(bad)/length(src))*100) '%']);
```

```
function [sa,E,cost_index]=Anneal(x,bcs,f,K,M,R,varargin);
%ANNEAL(x,bcs,f,K,M,R,varargin) performs simulated annealing based on
%   the following parameters:
%
%   x = [r,phi] Initial configuration of system
%   bcs = [min max] Boundary conditions that govern phi
%   f = Energy/Objective function to anneal
%   K = Total number of iterations to anneal for
%   M = # of sensors, used to define angular perturbation limits
%   R = Radius of sensor array, used to define radial perturbation
%       limits
%
%   Additional parameters to evaluate f should be included in order
% after R Anneal also makes use of an inner loop valve value, memory
% and imposes logical constraints on the search space.

T0 = 1e3;          % Initial melting temperature
dT = 0.6;          % Scaling coefficient for linear stepped T
Lb = eps;          % Temperature will not decrease if below this lower
bound
Ss = 0;            % Steady state temperature value at Kth iteration
stype = 0;         % Types of annealing schedule to implement
                   %    [0]: Conditional linear step (default)
                   %           T = T*dT + Ss
                   %    [1]: Damped harmonic decay
                   %           T = (T0-Ss)*exp(-alpha*dT*k)*cos(k/Eq)+Ss
                   %    [2]: "Gaussian" exponential decay (Ss = 0)
                   %           T = T*(K/k)*sin(k/K)
                   %    [3]: Exponential decay of order Eq
                   %           T = T0*(1-k/K)^Eq + Ss
Eq = 4;            % Number of energy decrements before equilibrium
                   % (or damped sinusoidal period for stype = 1)
Sw = 40;           % Step width, where T lowers after every Sw>0 steps.
                   % If Sw <= 0, T will only lower if equilibrium is
reached
RF = 10.0;         % Restore best point after T reduces by factor of RF

MAX_MCL = 5;       % Maximum Markov chain length
valve = 0.01;      % Inner and outer loop valve value

plotsched = 0;                                % plot annealing schedule?
sa = []; cfg = x;                % initialize SA solution constellation
cost = Inf;                                  % initialize lowest energy cost
cost_index = [];     % track indices of concurrent lowest energy states
Tf = T0; temps = T0;      % initialize temperatures and schedule vector

for k=1:K
  E(k) = feval(f,cfg,M,R,varargin{:});
  if(E(k) < cost)
      cost = E(k);
      sa = [sa; cfg];
      cost_index = [cost_index; k];
  end
```

```
  if(E(k) < valve), K = k; break; end;
  [T,temps] = Tsched(stype,temps,k,K,dT,Lb,Ss,(length(sa)-1),Eq,Sw);
  p = 1; x(p,:) = cfg; best = Inf;
  while(p < MAX_MCL && cost > valve)
    while(1) % randomly perturb until an in-bound solution is generated
      x(p+1,1) = x(p,1) + (2*rand(1) - 1) * R/2;
      x(p+1,2) = x(p,2) + (2*rand(1) - 1) * (pi/M);
      if(x(p+1,1)>0 && x(p+1,1)<R && x(p+1,2)<bcs(2) && x(p+1,2)>bcs(1))
        break;
      end
    end
    E1 = feval(f,x(p,:),M,R,varargin{:}); % cost of current Markov link
    E2 = feval(f,x(p+1,:),M,R,varargin{:});  % cost of next Markov link
    dE = E2 - E1;
    if(dE > 0 && (exp(-(dE)/T)<rand(1)))% Boltzmann prob to move uphill
        x(p+1,:) = x(p,:);
    end
    if(E2 < best)    % keep track of best energy states in Markov chain
      best = E2;
      cfg = x(p+1,:);
    end
    p = p + 1;
  end
  if (Tf/T > RF), Tf = T; x(k+1,:)=sa(end,:); end;    % restore best pt
end

if(plotsched)
  % PLOT ANNEALING SCHEDULE
  figure(5)
  plot(1:length(E),temps(1:length(E)),'LineWidth',4)
  title('Annealing Schedule','FontSize',16)
  ylabel('Temperature','FontSize',16)
  xlabel('Iteration','FontSize',16)
  set(gca,'FontSize',16);
  axis tight;
  grid on;
end
```

```
function [T,temps]=Tsched(stype,temps,k,K,dT,Lb,Ss,success,Eq,Sw)
%TSCHED(stype,temps,k,K,dT,Lb,Ss,success,Eq,Sw) varies the next
temperature
%   step and appends the annealing schedule, where:
%
%   stype = type of annealing schedule to pursue
%          0 : Linear step (default)
%          1 : Damped harmonic decay
%          2 : "Gaussian" exponential decay
%          3 : Exponential decay of order Eq
%   temps = annealing schedule vector
%   k = current annealing iteration
%   K = total iterations to anneal
%   dT = proportionality constant
%   Lb = lower bound of temperature
%   Ss = steady state temperature goal at Kth iteration
%   success = number of decrements taken so far
%   Eq = # of successful solutions to reach equilibrium
%   Sw = step width (number of iterations before reducing temperature)

if (stype == 1)
    alfa = -log(0.03*((Ss==0)+Ss)/(temps(1)-Ss))/K;   % 3% steady state
    T = (temps(1)-Ss)*(exp(-k*alfa*dT)*cos(k/Eq))+Ss; % Damped harmonic
else if(temps(end) > Lb)
       switch(stype)
         case 2
           T = temps(end)*(K/k)*sin(k/K);              % Gaussian decay
         case 3
           T = temps(1)*(1-k/K)^Eq + Ss;            % Exponential decay
         otherwise
           if(mod(success,Eq)==0 || mod(k,Sw)==0 && temps(end) > Lb)
             T = temps(end)*(dT) + Ss;
           else
             T = temps(end);
           end
      end
    else
      T = temps(end);
    end
end
temps = [temps; T];
```

```
function [X]=trackdata(M,R,A,psi,w0,r,phi,k,sigs);
%TRACKDATA(M,R,A,w0,r,phi,N,SNR)   Generates an M x N matrix of data,
% where:
%   *R and r are normalized to the wavelength, lambda.
%
%   M = # of sensors
%   R = radius of the circular sensor array
%   A = signal amplitude
%   psi = signal phase (R.V. uniformly distributed from -pi to pi)
%   w0 = phase constant (Beta = 2 * pi / lambda)
%   r = distance to the source from the center of the array
%   phi = angular displacement of the source, wrt +x axis
%   k = current discrete time sample
%   sigs = noise power

theta = (0:1:M-1)*2*pi/M;
for m=1:M
    num(m,:) = exp(-j*w0*sqrt(R^2+r.^2-2*R.*r.*cos(theta(m)-phi)));
    den(m,:) = sqrt(R^2+r.^2-2*R.*r.*cos(theta(m)-phi));
end
T = (num ./ den);
X = zeros(M,1);

S = A*exp(j*(0.34*pi*k + psi));
X = X + T(:,1)*S(1,:) + sqrt(sigs/2)*randn(M,1) +
j*sqrt(sigs/2)*randn(M,1);
```

```
function [err]=tracking(wps)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SIMULATION PARAMETERS
% i. Array Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
M = 8;                    % # sensors
R = 6;                    % radius of sensor array, normalized to lambda

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% ii. Source Signal Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
src = [0.6*R, 3*pi/4];    % source location
A = 1;                    % amplitude of source
rand('seed',sum(100*clock));
psi = (2*rand(1,1)-1)*pi; % phase of source, uniform RV between -pi, pi
SNR = 30;                 % SNR in dB
snr = 10.^(SNR/10);
sigs = A^2/(2*snr);
N = 1000;                 % total # of data samples
w0 = 2*pi;                % propagation constant

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% iii. Algorithm Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
K = 800;                % Max number of SA iterations
n = 100;                % # of samples used to create covariance matrix

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% iv. Trajectory Parameters
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
b = 0.1; % percentage of buffer space before start and after halt
% Linear Trajectory Functions {SIZE: 1 x (1-2*b)*N samples}
dr = 0;  % radial displacement (in wavelengths)
rad_traj = linspace(0,dr,round((1-2*b)*N));
dp = 127.324*wps; % angular displacement (in degrees)
ang_traj = linspace(0,dp*pi/180,round((1-2*b)*N));

% Non-Linear Trajectory Functions
% k = (1:(1-2*b)*N);
% rad_acc = -0.000001; % wavelengths per sample^2
% rad_vel = -0.0007;  % wavelengths per sample
% rad_traj = rad_acc*k.^2 + rad_vel*k;
% rad_traj = dr*sin(pi*k/N);

% ang_acc = -0.000001;
% ang_vel = -0.0002;
% ang_traj = ang_acc*k.^2 + ang_vel*k;;

rad = src(1) + padarray(padarray(rad_traj,[0 round(b*N)],'pre'),[0
round(b*N)],rad_traj(end),'post');
ang = src(2) + padarray(padarray(ang_traj,[0 round(b*N)],'pre'),[0
round(b*N)],ang_traj(end),'post');
```

```
track = [];
track_index = [];

for k = 1:N
    X(:,k) = trackdata(M,R,A,psi,w0,rad(k),ang(k),k,sigs);
    if(mod(k,n) == 0)
        Rapprox = X(:,k-n+1:k)*X(:,k-n+1:k)'/n;
        [cg,sa] = hybrid(K,M,R,A,sigs,w0,Rapprox);
        track = [track; cg(end,:)];
        track_index = [track_index k];
    end
end
% figure(1)
% plotsensors(M,1,ang,rad/R,'-b',track(:,2),track(:,1)/R,'+r');
% legend('Sensor','True Path','Estimated Path')
% title('Source Trajectory','FontSize',12)
% set(gca,'FontSize',12);
%
% figure(2)
% plot(1:N,mod(ang*180/pi,360),':b','LineWidth',4)
% hold on;
%  plot(track_index,track(:,2)*180/pi,'^-r','LineWidth',4,'MarkerSize',9)
% legend('True','Estimated','Location','NorthEast')
% title('Angular Displacement','FontSize',16)
% xlabel('Sample','FontSize',16)
% ylabel('Angle (degrees)','FontSize',16)
% grid on
% set(gca,'FontSize',16);
%
% figure(3)
% plot(1:N,rad/R,':b','LineWidth',4)
% hold on;
% plot(track_index,track(:,1)/R,'^-r','LineWidth',4,'MarkerSize',9)
% legend('True','Estimated','Location','NorthWest')
% title('Radial Displacement','FontSize',16)
% xlabel('Sample','FontSize',16)
% ylabel('Radius (normalized)','FontSize',16)
% grid on
% set(gca,'FontSize',16);
%
% disp('Travel speed: ')
% wps = (rad((1-b)*N)-rad(b*N))/((1-2*b)*N);
% dps = (ang((1-b)*N)-ang(b*N))*(180/pi)/((1-2*b)*N);
% disp([num2str(wps) ' wavelengths / sample'])
% disp([num2str(dps) ' degs / sample'])
%
% fprintf('\nTotal distance traveled:\n')
% disp([num2str(wps*(1-2*b)*N) ' wavelengths'])
% fprintf([num2str(dps*(1-2*b)*N) ' degrees\n\n'])

err = [mean(abs(rad(track_index)'-track(:,1))./rad(track_index)')
mean(abs(ang(track_index)'-track(:,2))./ang(track_index)')]*100;
% disp(['Avg Error (%r,%phi): ', num2str(err)])
```

```
function [sa,cg,err]=hardware(src,filename)
% SIMULATION PARAMETERS
M = 4;                  % # Sensors
R = 1.02;               % Radius of sensor array, normalized to lambda
w0 = 2*pi;              % Lossless propagation constant
K = 800;                % Total number of iterations

for m=1:M
    X(m,:)=csvread([filename,'wfm',num2str(m),'.txt']);
end
Rapprox=X*X'/length(X);
As = (max(max(X,[],2))-min(min(X,[],2)))/2;
theta = (0:1:M-1)*2*pi/M;
[m,n] = find(abs(Rapprox)==max(max(abs(Rapprox))));
Rmax = [m n];

prev = Rmax(1)-1; next = Rmax(1)+1;
switch(Rmax(1))
    case 1; prev = M;
    case M; next = 1;
end
dp = rerr(Rapprox(prev,prev),max(max(Rapprox)));
dn = rerr(Rapprox(next,next),max(max(Rapprox)));
switch(dn<dp)
    case 1; Rmax(2)=next;
    case 0; Rmax(2)=prev;
end
if(abs(diff(Rmax))>1), theta(1)=2*pi; end      % use appropriate
theta(1)
x = [R/2 theta(Rmax(1))];

bcs = x(1,2) + pi/M * [-2 2];

% SIMULATED ANNEALING
[sa,E,cost_index]=Anneal(x,bcs,'Jerr',K,M,R,As,w0,Rapprox);
% CONJUGATE GRADIENT HYBRIDIZATION
cg = minimize(sa(end,:)','Jerr',K,M,R,As,w0,Rapprox);

cg(:,2) = (mod(cg(:,2)-(cg(:,1)<0)*pi+2*pi,2*pi)); % Removes false
errors
cg(:,1) = ((cg(:,1)>0).*cg(:,1)-(cg(:,1)<0).*cg(:,1));

srcr = [src(1) src(2)*pi/180];
err = (srcr-cg(end,:))./srcr;

% POLAR PLOT OF ANNEALING CONSTELLATION, CG CONVERGENCE
% figure(1)
% plotsensors(M,1,x(1,2),x(1,1),'or'); hold on;
% polar(src(1,2)*pi/180,src(1,1),'sb');
% polar(sa(:,2),sa(:,1)/R,'.r');
% legend('Sensor','Initial Guess','Source','SA Configurations',...
%     'Location','NorthEastOutside');
```

```
% title('Convergence to Source Location','FontSize',16);
% set(gca,'FontSize',16);
% grid on;

% figure(2)
% plot(1:length(E),E,'*b','MarkerSize',8); hold on;
% plot([cost_index' length(E)],[E(cost_index) E(cost_index(end))],'--
^r','LineWidth',4,'MarkerSize',9);
% legend('Current Cost','New Best Cost')
% xlabel('Iteration','FontSize',16);
% ylabel('J(r,\phi)','FontSize',16);
% title('J(r,\phi) vs Iteration','FontSize',16);
% set(gca,'FontSize',16);
% grid on;

% RADIUS & ANGLE CONVERGENCE PLOTS
% figure(3)
% plot(ones(1,length(cg))*src(1,1),'--r','LineWidth',4); hold on;
% plot(cg(:,1)/R,'-b^','LineWidth',4,'MarkerSize',9);
% legend('Actual Radius','Converging Radius');
% xlabel('Conjugate Gradient Iteration','FontSize',16);
% ylabel('Radius (normalized)','FontSize',16);
% title('Convergence of Radius','FontSize',16);
% set(gca,'FontSize',16);
% grid on;

% figure(4)
% plot(ones(1,length(cg))*src(1,2),'--r','LineWidth',4); hold on;
% plot(cg(:,2)*180/pi,'-b^','LineWidth',4,'MarkerSize',9);
% legend('Actual Angle','Converging Angle');
% xlabel('Conjugate Gradient Iteration','FontSize',16);
% ylabel('Angle (degrees)','FontSize',16);
% title('Convergence of Angle','FontSize',16);
% set(gca,'FontSize',16);
% grid on;
```

```
function plotsensors(M,R,varargin);
%PLOTSENSORS(M,phi,r) creates a polar plot of M sensors and I sources
% of distance r(1), r(2),... r(I) and angular displacement phi(1),
% phi(2),... phi(I).

if( M > 0 )
    theta=[0:1:M-1]*2*pi/M;
    polar(theta,R*ones(1,M),'b*');
end
if(nargin > 2)
    hold on;
    for k=3:3:length(varargin)
        polar(varargin{k-2:k});
    end
end
```

```
function err=rerr(a,b)
%RERR(a,b) returns the relative distance of a from b

err=(abs(a-b))./abs(b);
```