AN OPTICALLY-SWITCHED TRANSMIT/RECEIVE LENS ARRAY FOR BEAM-SPACE ADAPTIVE COMMUNICATION SYSTEMS

by

JAMES EDWIN VIAN

B.S., University of Colorado at Boulder, 1994

M.S., University of Colorado at Boulder, 1996

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

2000

This thesis entitled:

An Optically-Switched Transmit/Receive Lens Array for Beam-Space Adaptive Communication Systems

written by James Edwin Vian

has been approved for the

Department of Electrical and Computer Engineering

Zoya Popović

Louis Scharf

Date

The final copy of this thesis has been examined by the signatories;

and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline

HRC Protocol

Vian, James Edwin (Ph.D., Electrical Engineering)

An Optically-Switched Transmit/Receive Lens Array for Beam-Space Adaptive Communication Systems

Thesis directed by Professor Zoya Popović

With the growth in todays modern communication world, multiple user and multi-path environments are becoming an ever increasing problem. These environments reduce the quality of the communication link through multiple users interfering with each other or self interference in multi-path environments. A common technique used to combat these environments, is to use a planar antenna array that steer its receptivity pattern in the direction of the desired user. The receptivity patterns for the arrays are control through adaptive algorithms that adjust the magnitude and phase of each received signal at each element in the planar array before coherently combining the signals. The magnitude and phase adjustments (complex weights) reverse the phase shifts induced in the signals as they propagate across the surface of the planar array. The disadvantage of this technique is that it requires significant amounts of computational processing power for large arrays containing many antenna elements. By using a microwave lens array instead of a planar array in the communication system, the received signals are transformed from a phase-space representation to a beam-space representation, which can reduce the processing load for the algorithm and may increase the overall signal to noise ratio (SNR) through partial beam-forming before the noise is added to the system. For these reasons, an optically controlled transmit/receive lens array is developed. The optical control of the lens array allows the array to switch between transmit and receive modes rapidly with negligible interference to the microwave signals. A low optical power single pole double throw switch is developed for routing the transmit and receive signals in the array that has insertion loss of 0.3 dB and isolation of 36 dB. To model the performance of lens arrays for wireless communication systems, code is developed that calculates the imaging properties of lens arrays under different design conditions. The lens array modeling code is used in conjunction with a modified Least Mean Square (LMS) algorithm that turns off small valued complex weights, to improve the overall SNR and adaptation rate for the communication system.

DEDICATION

Dedication . . . This thesis is dedicated to my family. To parents, Wayne and Sharon Vian, who are my first teachers that made me excited about the world of science. To my sister, Carol Vian, for the healthy competition while growing up. And finally to my loving wife, Trina Vian, who helped me keep everything in perspective.

ACKNOWLEDGMENTS

Acknowledgments ... This work would not be possible without the mentorship of my advisor Prof. Zoya Popović. Besides her day-to-day support and encouragement, it is her unwavering dedication to developing and maintaining a top quality research laboratory that made this work possible. Note that this work was supported ONR and the Office of the Secretary of Defense through the MURI program grant N00014-97-1006.

I would also like to thank the past and present members of Zoya Popović's research group. To Stein Hollung, whose work on transmit/receive lens arrays is the bases for my work research. To Eric Bryerton, Joe Tustin, Michael Forman, Todd Marshall, Manoja Weiss and Jan PeetersWeem, whose conservations on on electromagnetic theory (among other things) help me to better understand this difficult topic. A special thanks to Shawn Stone, Pete Kirkpatrick and Paul Smith, who help with the buying, building and testing of many of the items in this thesis. I would also like to thank Michael Forman, Todd Marshall and Jan PeetersWeem for their time and patience in teaching me the ways of Linux and LaTex which made the writing of this thesis possible.

In addition to the members of Zoya Popović's group, I would like to thank Prof. Dana Anderson, Prof. Kelvin Wagner and their research groups (Edeline Fotheringham, Leslie Czaia, Greg Kriehn, Ken Anderson, Friso Schlottau, Andrew Kiruluta, Paulo Silveria) for there help and guidance in the world of optics. I am also grateful to Prof. Lloyd Griffiths for his help on adaptive arrays and the LMS algorithm. I want to thank our administrative assistants, Helen Frey and Rachel Tearle, who often went out of their way to make sure that our day-to-day operations

ran smoothly.

CONTENTS

TABLES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 Adaptive Communication Systems

In recent years there has been an explosion in growth in civilian wireless communications. With the increased use of wireless communications systems, such as cellular phones, and data communications systems, such as wireless local area networks (WLAN), the wireless environment is getting increasingly more complex. With the exponential increase in users and faster data rates, the once plentiful resources of bandwidth (BW) and transmission power are now scarce. In an effort to increase the capacity of communication systems where multiple users and multiple propagation path interference becomes a problem, advanced modulation, coding and power control techniques have been developed, [3, 4, 5, 6, 7]. These techniques have gone a long way towards increasing the systems capacity without sacrificing system performance.

In wireless communication systems, multiple antennas may be used to improve the system performance through diversity transmission and reception. Diversity is effectively transmission and/or reception of multiple copies of the same information that are slightly different. When the multiple copies are combined coherently or non-coherently, the resulting effective signal to noise ratio (SNR) of the system is much greater than that of a single copy. The more copies of the signal the larger the performance improvement of the system (usually measured in bit error rate, BER, decrease).

The concept of diversity dates back to the 1940's when it was used to improve radar performance. Much of the early published work focuses on diversity reception where multi-path fading is a problem [8, 9, 10, 11, 12]. For example, transmission over long distances (i.e., part-way around the world) results in multiple reflections of the waves off the ionosphere, and the different reflected waves interfere with each other at the antenna. When this interference is destructive, it is referred to as "multi-path fading", since the signal fades in that case. Price and Green [12], showed that if delayed copies of the received signals where combined together in what they called a "Rake" receiver, much of the effects of the multi-path fading can be removed. Multi-path fading also occurs in communication systems where the transmission distances are shorter. However, if multiple antennas are used, the fading at each antenna will be different and depends on the distance between antennas. The farther the antennas can be separated, the more uncorrelated the fading at each antenna is. When two antennas are about ten free-space wavelengths (10λ) apart, the multi-path fading at each antenna can be considered independent and the signals may be combined based on their average signal to noise ratio (SNR) [8, 9, 10]. Turin [11] showed that if the fading at each antenna is correlated, then the optimal combination of the received signals is a coherent weighted sum using the mean of the signal fading for the weights, and a non-coherent weighted sum based on the covariance of the the signal fading for the weights. A completely coherent summation is preferred for these statistical diversity systems, since in this case the SNR grows as the square of the order of diversity, in contrast to linearly for the non-coherent case.

With the advent of the microprocessor it becomes feasible to build antenna diversity systems that use adaptive algorithms to control the combination of the signals received at the different antenna elements. Since the algorithm is able to perform real time tracking of the received signals, the antenna signals may be coherently combined depending on the instantaneous relationship between the signals, and not the average off all possible realizations of the correlation between them. As a result, the SNR for an adaptive antenna diversity system is usually better than the SNR for a statistical antenna diversity system. Since the instantaneous relationship between the signals is important, the antennas may be be placed at a smaller spacing (on the order of $\lambda/2$), making the system more compact. As mentioned earlier, modern communication systems have to maintain quality of service with multiple users interfering with each other, as well as the same user interfering with itself due to multi-path fading. Since adaptive arrays can learn about their environment, they can adapt to the environment without any prior knowledge of the signal sources, assuming that the environment changes slowly when compared to the convergence rate of the algorithm.

One of the first adaptive antenna diversity systems dates back to the 1950's when Howells and Applebuam used a dual-antenna system to remove the effects of an unwanted user, [13]. The magnitude and the phase of one antenna signal is adjusted so that when it is combined with the second antenna signal, the unwanted user is canceled out. This is an example of an adaptive antenna diversity system that isolates users in a multi-user environment. If the second received signal is another copy of the "desired" user's signal due to multi-path, then the magnitude and phase of the antenna signals would be adjusted such that the received signals combine coherently.

The magnitude and phase changes in the received signals are often referred to as weights, in reference to a complex weighted sum. The weights in the algorithm have an interpretation in microwave antenna array theory. The weights are the relative magnitudes and phases of the generators at each antenna element for a transmit array. Because of reciprocity, this is equivalent to the amplitude and phases that would need to be imposed on the receiver voltages at each element in reception for equal transmit and receive far-field radiation patterns. In an adaptive system, the complex weights vary based on a feedback signal. This in turn changes the far-field radiation pattern of the array in some pre-determined fashion. For example, it places the main beam in the direction of the "desired" user, and a null in the interference direction. If the user is in a multi-path environment, then the weights are adjusted so that there is a lobe pointed in each direction from which the "desired" signal is received. For a multi-user environment, a null is placed in the direction of the "other" user, and very little of its signal is received.

Military applications are also showing an increased need for adaptive arrays both for multi-user communication systems and radar systems. As with the civilian communication environment, military environments are prone to interference from other users and multi-path fading. The military environments also have an additional interferer, "The Enemy", who is jamming "desired" user signals with a signal that can be many times greater in magnitude than that of the "desired" user. If the complex weights are implemented with relative phases, the system is narrowband and results in changes of radiation pattern as the frequency changes. This is often referred to as "beam squint" and is yet an unsolved problem in practical broadband arrays. For broadband signals, a complex weighted sum of time delayed versions of the signals are needed to prevent beam squint. This processing technique is referred to as true-time delay processing.

One of the disadvantages of adaptive array systems is that they require significant computational power to calculate the adaptive weights. The number of computations increases geometrically with the number of array elements. In this thesis, a new type of front end (antenna array and receiver) is explored in the context of adaptive systems. In a "standard" planar array, each element receives a phase-shifted version of the same input plane wave. In the lens array presented in this thesis, the antenna array performs a Fourier transform and images the signal sources onto a focal surface, and the signal image is then sampled with receivers. Most applications require both transmit and receive functions, and a part of this thesis is concerned with fast efficient routing of signals through transmit and receive paths. To that end, the transmit and receive array presented in this thesis uses optical control which does not interfere with microwave signals and can be very fast and energy efficient.

1.2 Adaptive Planar Array Communication Systems

The basic architecture of a narrowband adaptive array is shown in Figure 7.3. It consists of an array whose signals are adjusted both in magnitude and phase (i.e. each element has a complex weight) before being combined. The weights are controlled by an algorithm which uses information about the structure of the "desired" user's signal (i.e. modulation, coding, pulse shape, etc.) to estimate the instantaneous relationships between the received signals. The information about the "desired" signal is contained in the training signal. There are many different algorithms used in adaptive antennas depending on the needs of the system [13, 14, 15, 16, 17]. One of the earliest algorithms is the LMS (Least Mean Square) algorithm develop by Widrow [18]. It is a gradient search algorithm that minimizes the error between the output signal, y_k, and the training signal d_k (Figure 7.3). The result is a set of weights that converges to the optimal set of weights known as the Wiener solution [19]. Once they converge, the weights stay fixed until the communication environment changes and the algorithm proceeds to readjust the weights.

The adaptive system in Figure 7.3 is intended for a narrowband application of bandwidth BW, where the delay of the received signals across the array or the difference in delay between multi-path signals is less than $\frac{1}{BW}$. For very broadband systems, such as some used by the military, or for large delay multi-path environments, true-time delay processing is needed. The length of the delay line is at most

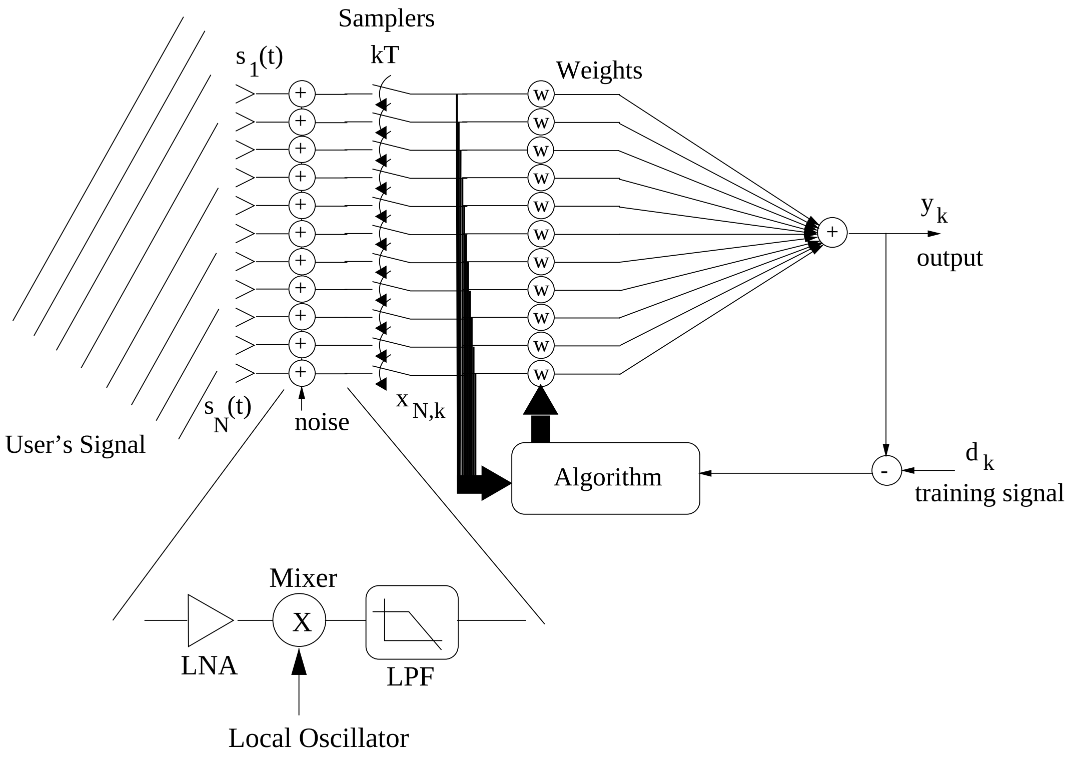


Figure 1.1. An example of a narrowband planar array adaptive receiver shown for the single user case. The received signals, $s_i(t)$, are amplified with LNAs and down-converted (mixer, local oscillator and LPF) before being sampled (represented by switches). For this model, the effect of the LNAs and down-converters is the addition of noise. The received signals plus noise are sampled with a sampling period T, giving signals, $x_{i,k}$, used by the algorithm. The algorithm uses the training signal, d_k, to adjust the weights, W, such that the SNR at the output, y_k, is maximized.

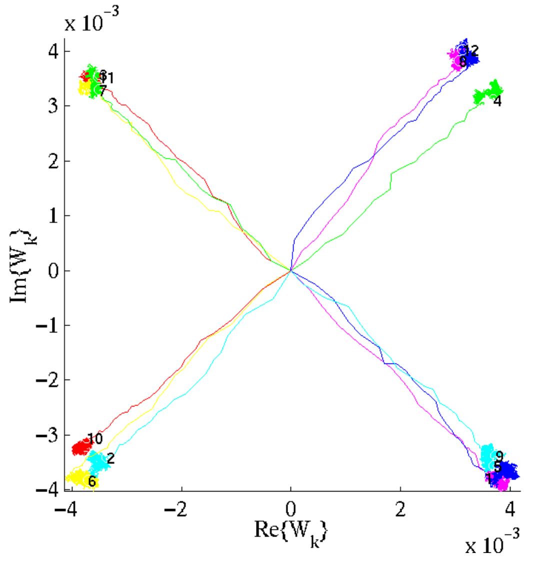


Figure 1.2. The weight trajectories in the complex plane for a planar array adaptive receiver converging to the optimal weights.

twice the maximum delay expected to be seen between signals, and the time interval are spaced at a period of $\frac{1}{BW}$. An entire area of research focuses on this type of adaptive system, referred to as Space-Time Adaptive Processing, STAP [6].

The Wiener solution is the optimal solution for the weights that minimizes the error between the training signal and the output. The Wiener solution

$$W_{opt} = R^{-1}P \tag{1.1}$$

where $(R)_{ij} = E[x_i x_j^*]$ is the correlation of the received signals, and $(P)_i = E[x_i d]$ is the correlation of the training signal and the received signals. For a standard narrowband planar array, each of the array elements receives approximately the same amplitude (for far-field sources), but with a relative phase shift that depends on the array period and angle of incidence. Using these assumptions in Equation (1.1), the solution results in complex weights that lie on a circle, Figure 7.4. The weight trajectories in Figure 7.4 show the convergence of the LMS algorithm for a plane wave incident on a planar array with ten elements in a row. If an interfering signal is present, the weights rotate so that the power of the interfering signal is reduced below the noise floor of the system. This weight distribution in the complex plane implies that each of the received signals has equal importance. For adaptive algorithms, the adaptation rate is inversely proportional to the number of weights and the computational load is proportional to the number of weights.

For systems that require small beamwidths (i.e., radar), have large numbers of interfering users and need many nulls, or require multiple beams due to multi-path, a large array with many elements is needed to provide the degrees of freedom for a good solution. In these situations, adaptive planar arrays are slow require large and require significant amounts of computational power [20].

1.3 Adaptive Lens Array Communication Systems

The multi-user, multi-path and jammer environments have one thing in common: they all have signals arriving from different directions. For a planar array, the angle of arrival is encoded in the phase variations across the surface of the array. If a microwave lens is used, the signals are instead imaged onto a focal surface at different locations "behind" the lens. By sampling the focal surface with detectors (antennas and receivers), the algorithm only has to process those detectors illuminated by the "desired" user's signal. The architecture of the lens naturally separates spatially the received signals much like a pre-formed beam space array. If an interfering signal is present, then the weights for the detectors illuminated by the "desired" user's signal adjust to cancel out any of the interfering signal they receive. For multi-path environments, there are several spots illuminated by the "desired" user's signals and the corresponding detectors are used. One obvious advantage of lens arrays is that the number of weights depends on the number of paths the "desired“ user's signal takes to the array and not the number of elements in the array. By using fewer weights, the adaptive system should adapt faster and require significantly less computational power.

There are many architectures for microwave lenses. For example, dielectric lenses look and act like optical lenses. Network lenses use beam-forming networks that mimic the beams produced by sampling the image surface of a lens (e.g., Butler matrices). Such networks can be made efficient [21, 22, 23] and are lossless since the beams are orthogonal [24, 25, 26]. However, for large arrays, these networks can have significant loss due to the power combiners and phase shifters contained in them [27]. Finally, there are "Boot lace" lens arrays which are two arrays connected together in element pairs with transmission lines. The transmission lines simulate the delay that a microwave signal undergoes through a dielectric lens. The transmission lines are longer in the center than on the edge, in analogy to an optical lens being thicker in the center. Lens arrays are also prone to loss just as network lenses, but the loss in a array lens grows slower than the loss in a network lens as the number of elements increases, [27].

Lens arrays may have an advantage over planar arrays by performing some beam-forming before noise in added by receivers. A discussion on the effects of lens arrays on communication systems, which points to interesting future work in given in the Section 7.3.2.

1.4 Fast Optical Control of Lens Arrays

Bi-directional lens arrays are are needed for radar and communications systems. However, it is possible to design bi-directional lenses with amplifiers that add gain to the system. The amplifiers are connected between each pair of antenna elements. One of the major advantages of active lens arrays is that they can produce very powerful transmitting signals through spatial power combining, which in turn means that lower-power, less expensive, more efficient power amplifiers can be used in transmission [27].

In a bi-directional array, the transmit signals are routed through power

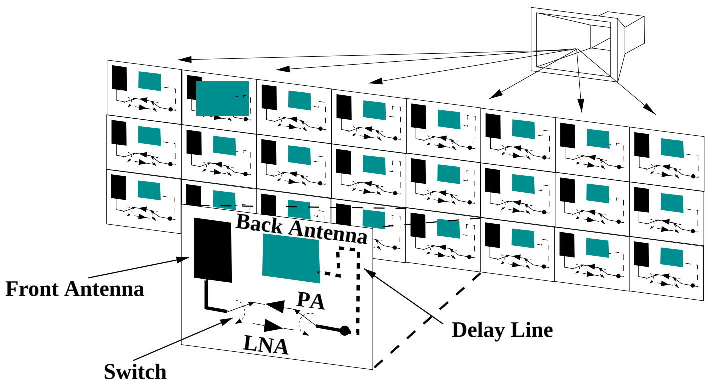


Figure 1.3. A cylindrical bi-directional lens array which uses SPDT switches to route through either the transmit path (PA) and receive path (LNA) respectively.

amplifiers (PAs) and the receive signals are routed through LNAs. This can be accomplished in two ways. The first way is to use circulators that direct signals depending on their direction of propagation. Unfortunately, these devices are too large for use in lens arrays. The second way is to direct the signals using single pole double throw (SPDT) switches, as in Figure 1.3. By using two SPDT switches in each element, the lens array is limited to half- duplex operation, which can be limiting for some communication systems, but not for radar. The rise time and fall time for the switches should be on the order of a few nanoseconds to accommodate wide bandwidth signals. For example, a 2 ns rise time, τ_{rise}, switch can handle a $BW = \frac{1.8}{2\pi\,\tau_{rise}} = 140\,\text{MHz}$ [28] signal at its fastest switching rate.

In the past, half-duplex bi-directional lens arrays have been electrically controlled [1]. The control lines for the switches weave in and out between the unit cells, as can be seen in the photograph in Figure 1.4. To minimize the number of control lines in the array, the switches for the unit cells are connected in parallel. By connecting the switches in parallel, the RC time constants for the switches add, and this slows down the rise time of the array. An array with hundreds of unit cells can have a switching rise time of a microsecond, even though nanosecond switches are used in each element.

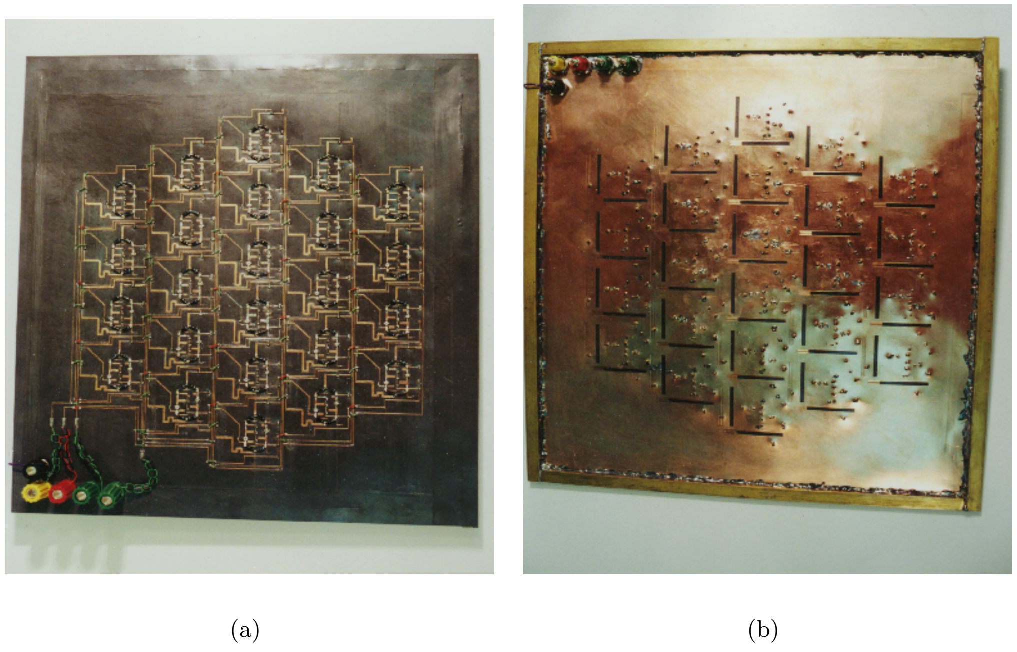


Figure 1.4. Example of an electrically controlled bi-direction lens array that uses orthogonally polarized slot antennas (b) for the array elements [1].

Instead of electrically controlling the switches in parallel, it is possible to optically address each individual unit cell. By using optical fibers, the control optical signals are naturally isolated from the microwave signals and the fibers do not interfere with the microwave radiation. Since each switch is individually controlled, the rise time for the array is the same as the rise time for the unit cell, independent of array size.

1.5 Organization

This thesis presents work on applying lens arrays to communication and radar systems. It covers the development of a fast, low-power optically-controlled transmit/receive lens array and simulations of a lens array adaptive communication system. Chapter 2 reviews microwave switch design, the current work done in optical switches and the development of an optically controlled resonant microwave switch. Chapter 3 discusses the development and characterization of an improved optical switch. Chapter 4 discusses the design and implementation of an optically-controlled transmit/receive lens array. Chapter 5 discusses a numerical model for lens arrays designed to explore the tradeoffs in lens array design. Chapter 6 reviews the LMS algorithm and simulations of an adaptive lens array systems with tradeoffs in hardware and required computational processing. Chapter 7 discusses some topics for future directions that are motivated and validated by the work presented in this thesis.

CHAPTER 2

8 GHZ RESONANT SWITCH AND UNIT CELL DESIGN

2.1 Microwave Switches Background

The principle goal of the research described in this chapter is to design a fast optically controlled, half-duplex transmit/receive active antenna array. One basic component of the array is an optically controlled single pole double throw switch (SPDT) used to route the transmit and receive signals. The design of the SPDT switch consists of microwave circuit, optical circuit and design .

Switches at microwave frequencies in principle work the same way as switches at low frequencies (few hundred MHz and below): the signal chooses the path that has the lower impedance. There are three measures of performance for a switch: insertion loss (IL); isolation (ISO); and return loss (RL). IL is the ratio of power delivered to a load for an ideal switch in its "on" state to the actual switch in it's "on" state. Therefore IL is a measure of the power lost from the load due to the imperfections in the switch. It is usually expressed in positive decibels. Using scattering parameters, IL=$1/|s_{21}|$ for the switch in the "on" state. ISO is the ratio of amount of power delivered to the load for an ideal switch in the "on" state to the amount of power delivered to the load for actual switch in the "off" state. ISO is a measure of how well the switch turns "off" the load and is expressed as $1/|s_{21}|$ for the switch in the "off" state. Return loss appears only at microwave frequencies and is the amount of power reflected at the switch because the load is no longer ideally matched to the source. It is expressed as $1/|s_{11}|$ for the switch.

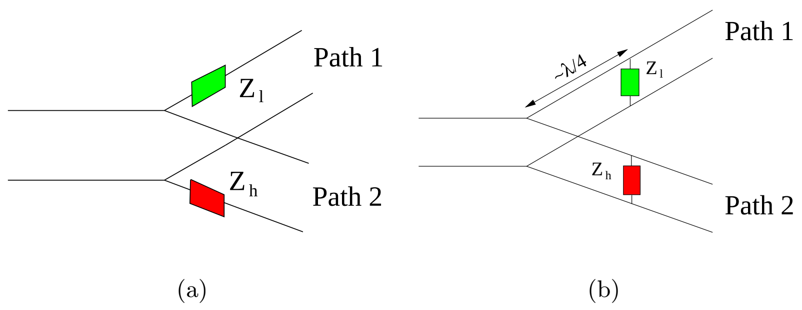


Figure 2.1. Transmission line schematics for a SPDT switch with microwave devices,such as PIN diodes, in series (a) and shunt (b).

The main difference between microwave and low frequency switches is that effective impedance of a device, e.q. a PIN diode, depends on its position in the switch. This leads to two main configurations of microwave switches, the series and shunt configuration, Figure 2.1.

The series configuration operates in the same intuitive way as a low frequency switch with the low impedance device in the "on" path and the high impedance device in the "off" path. The equations for IL, ISO and RL as

$$IL = \left| \frac{(2+Z_h)(1+Z_l)+Z_0(1+Z_h)}{2Z_0(1+Z_h)} \right|^2 \tag{2.1}$$

$$ISO = \left| \frac{(2+Z_l)(1+Z_h)+Z_0(1+Z_l)}{2Z_0(1+Z_h)} \right|^2 \tag{2.2}$$

$$RL = \left| \frac{Z_{load}-Z_0}{Z_{load}+Z_0} \right| \tag{2.3}$$

$$Z_{load} = \frac{(Z_l+Z_0)(Z_h+Z_0)}{Z_l+Z_h+2Z_0} \tag{2.4}$$

As seen from 2.1-2.3, good switch design requires a device that has a very small impedance in the "on" path and a very large impedance in its "off" path.

As mentioned earlier, in microwave circuits, the effective impedance of a device depends on its position in the circuit. The shunt configuration for SPDT microwave switch exploits this property to improve the IL or ISO of the switch in some cases. For the "on" path, the high impedance state of the device is in parallel with the intrinsic transmission line impedance (Z_0), resulting in an combined impedance approximately equal to Z_0. Thus, there is a small reflection at the device for sufficiently large impedance devices. For the "off" path, the low impedance of the device shorts the transmission line, causing a large reflection at the device. To prevent the short in the "off" path from shorting out the "on" path, a $\lambda/4$ section of transmission line is used, to transform the short to high impedance (open) at the Y-junction. The shunt configured switch has inherently smaller bandwidth than the series configured switch due to this $\lambda/4$ section of transmission line.

To solve for the IL, ISO and RL for shunt configured switch, $\left[\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right]$ calculated at the Y-junction and used to find IL, ISO, and RL as:

$$IL \text{ or } ISO = \frac{1}{4}|A + \frac{B}{Z_0} + CZ_0 + D|^2 \tag{2.5}$$

$$RL = \frac{A + B - C - D}{A + B + C + D}, \tag{2.6}$$

where

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Y_1 & 1 \end{bmatrix} \begin{bmatrix} 0 & \frac{j}{Z_0} \\ jZ_0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ Y_2 & 1 \end{bmatrix}, \tag{2.7}$$

$$\tag{2.8}$$

and Y_1, Y_2 are give for the IL and ISO, respectively, as

$$\text{IL:} \qquad Y_1 = \frac{Z_l \| Z_0}{Z_0^2} \qquad Y_2 = \frac{1}{Z_h}, \tag{2.9}$$

$$\text{ISO:} \qquad Y_1 = \frac{Z_h \| Z_0}{Z_0^2} \qquad Y_2 = \frac{1}{Z_l}. \tag{2.10}$$

There are two basic categories of devices use in microwave switches. The first categories is active devices like diodes and transistors. The second categories is mechanical structures that use moving metal contacts.

Active microwave switches have been around for decades and are available as microwave monolithic integrated circuits (MMICs). A typical MMIC SPDT

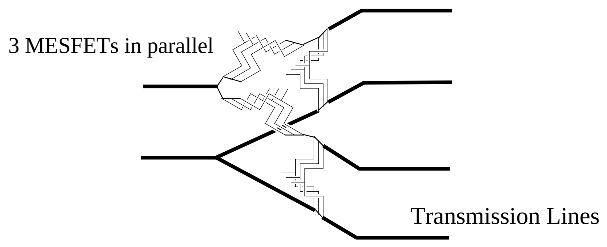


Figure 2.2. An example of a MMIC SPDT switch with multiple devices (in this case MESFETs) used to improve IL and ISO.

switch, figure 2.2, uses multiple devices to improve the IL and ISO of the switch over a larger BW. Due to the non-linear nature of the devices, they generally generate large inter-modulation products if two signals are present in the switch at the same time. Table 2.1 shows some typical values of IL, ISO, RL and third-order inter-modulation product (IP3) for commercially available switches.

The recent development of mircoelectromechanical systems (MEMS) has allowed the development of mechanical microwave switches. An excellent tutorial on microwave MEMS switches is given by Elliott Brown, [29], which discusses the advantages, disadvantages and areas of development for MEMS switches. MEMS are mechanical structures built on the micron scale using semiconductor etching techniques developed for integrated circuit. Cantilever and air bridge are the two basic styles for MEMS switch.

The cantilever switch has a conductive arm connected to one side of a gap and suspended over a metal contact on the other side of the gap. To close the switch,

Table 2.1: Examples of commercially available MMIC switches

manufacture	M/A-Com	M/A-Com	Alpha
part number	2954-2004	MASW20000	AS018R2-00
IL (dB)	1.8	2.1	2.2
ISO (dB)	50	50	42
VSWR	2	1.8	1.85
BW (GHz)	2-18	DC-18	DC-18
Switching speed	20 ns	2ns	3ns
Inter-modulation Intercept Point	N/A	43dBm	46dBm

an electric field is applied between the metal arm and the pull down electrode in the gap. The electrostatic forces generated pulls the arm down making contact with the metal contact on the other side of the switch. The pull-down electrode usually has an insulator deposited upon it to prevent the shorting of the control voltage. By reducing the electric field to a value typically less than the threshold voltage needed to close the switch, the switch will open. This switch is commonly used with microstrip transmission line architectures.

The air bridge switch lends itself to co-planar waveguide (CPW) architectures, where the air bridge is connected between the two ground strip of the CWP and is suspended over the center line. Again, by applying an electric field between the air bridge and an electrode on the center line (not shown), the air bridge bends down to short out the CPW transmission line. In this case the switch path is "off" when the electric field is applied. Then by reducing the electric field, the air bridge opens and turns the switch "on".

The IL, ISO and RL for cantilever and air bridge MEMS switches depends on the difference between the "close" and "open" capacitance between the arm (or bridge) and the transmission line. For good IL in cantilever switch and ISO in the air bridge switch, the capacitance should be as large as possible or preferably an ohmic contact. For a good ohmic contact, strong electrostatic forces are needed to compress the contacts together. For good ISO in a cantilever switch and IL in the air bridge switch, the capacitance should be as small as possible, leading to a large gap between arm (or bridge) and transmission line. MEMS switches can exhibit IL of 0.4 dB below 1 GHz to just below 1 dB up to 40 GHz and ISO of 40 dB below 1 GHz which degrades about 20 dB at 40 GHz.

There is a natural tradeoff between switch rate, performance and control voltage. In order to make a fast switch, the movement of the pieces should be minimized, which reduces the IL or ISO of the switch. By increasing the electric field strength and the stiffness of the arm (or bridge), the arm is forced to the contact quicker and relaxes faster. It appears that the MEMS switches are fundamentally limited to a "close" time of about $1\,\mu$s and a "open" time of about $10\,\mu$s due to material properties.

Since electrostatic forces are used to control the MEMS switch, the required energy needed to control the switch equals the difference in the stored energy in the switch capacitance between the "open" and "close" states. The total control power is the stored energy times the switch rate, $\frac{1}{2}(C_{close} - C_{open})V^2 f_s$. For example, with "close" capacitance ("open" capacitance is negligible) of 13 pF, a close state voltage of 4 V and a switch rate of 10 kHz, the total power dissipated is approximately $1\,\mu$W. Besides low control power, MEMS switches also display superior linearity from the lack of non-linear devices like diodes and MESFETs.

2.2 Photonic Switches Background

The earliest photonic switches were developed at much lower RF frequencies [30, 31]. The switches consist of a gap in a conductive path filled with photo-conductive materials. The switch is "closed" when the photo-conductive material is illuminated with watt-level optical power. Much of the practical work focuses on RF and microwave photonic switches that are fast [32, 33, 34]. These switches are used in high-power inductive energy storage pulse power systems [35]. One reported switch is used to control a 100 kW power source with a PIN diode and 2D laser diode array [36]. Only in the past five to ten years has work been done on microwave photonic switches. Most of the effort has been focused on developing a PIN diode that acts as good switch diode as well as a good photodiode. This is one technique for developing a microwave photonic switch, two more techniques are presented in section 2.3.1.

By using a PIN diode as the microwave and the optical device, the optical signal provides the power for control; no additional power from supply bias line is needed. The optical signal generates the carriers in the PIN diode, reducing the diode impedance. The more light illuminates the device, the lower the devices impedance. Therefore, the device is in a high impedance state with no light, and in the low impedance state with high optical power density.

A device that has good optical performance and microwave performance is not easy o obtain. From the microwave point of view, the PIN diode should have a small junction capacitance for a high “off” impedance and low “on” resistance. From the optical point of view, the PIN diode should have a large active area for more light gathering ability and use materials that have good responsivity (generation of carriers from photons). These conditions oppose each other in the optimization of the PIN diode design. For example, a large active area causes a large junction capacitance. Low “on” resistance means more optical power, or large active area and good responsivity materials. Materials used in PIN photodiodes that have good responsivities typically have larger loss than the materials used for high quality microwave PIN diodes.

In [37], demonstrated a switch is presented with IL=2.2 dB for 1 mW of optical power and with rise and fall time of .01 μs to 10 μs depending on optical power. The IL is limited by the mount of optical power delivered to the PIN diode, while the ISO is limited by the diode capacitance. [38] demonstrated a switch with IL of 1.2 dB for 40 mW of optical power and ISO of 30 dB. Both switches need on the order of milliwatts of optical power to achieve good IL with good ISO.

2.3 Developing Photonic Switch

As presented in the previous section 2.2, most of the work done in photonic switches has been done in order to develop devices that could sever both as a microwave and optical device. In this work, we use off-the-shelf devices for both the microwave and optical circuits. Most of the fast optical components are develop for the optical communication industry, limiting the wavelengths to 880 nm, 1310 nm and 1500 nm, which are all in the infrared (IR).

Besides being limited to off-the-shelf components, the ability to machine parts is limited to what can be manufactured on a printed circuit board (PCB) prototyping milling machine model 93s made by LPKF. The milling machine can cut, drill and gouge substrate materials, FR4 (PCB material), brass and aluminum of up to 1.6 mm (64 mils). It has an accuracy of 8 μm (0.3 mils) with a minimum drill hole size of 300 μm in diameter. The minimum cutting round is 250 μm (10 mils) for substrates, 1 mm for FR4 and 2 mm for metal.

One of our goals was also to minimize the size ofthe switch and the required optical power. A small switch size is needed it insure proper array element spacing to prevent grating lobes. Low optical power is needed not only for overall efficiency, but also in order to be able to control large arrays with a single standard laser-diode.

2.3.1 Techniques for Optical Control The previous work shown in section 2.2 demonstrated only one means of optical control of a microwave switch. In principle there are three techniques for optical control. The first is to have a optoelectronic device like a photodiode or photo-transistor act as the microwave device that switches between the high and low impedance state, Figure 2.3(a). This is the technique used in section 2.2. The second is to use a photo-voltaic device to provide the bias power for a microwave device, which changes impedance states Figure, 2.3(b). For example, a PIN photodiode which generates carries when illuminated and therefore biases a microwave PIN diode. The third technique uses a optoelectronic device to control the bias current for a microwave device, Figure 2.3(c). Each technique has its own advantages and disadvantages, and has been investigate to some degree in this project.

The first technique uses a optoelectronic device in a microwave switch. This

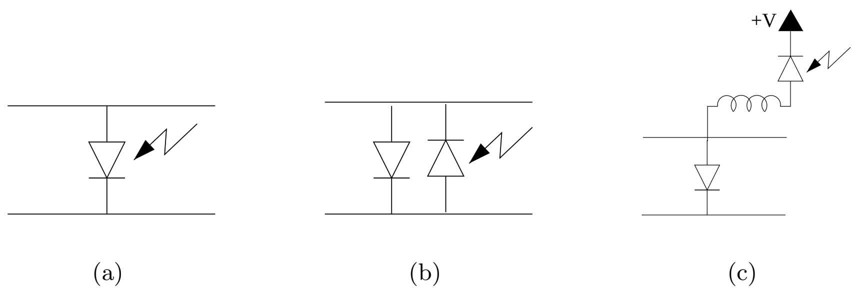


Figure 2.3. Three principle ways to optically control a microwave switch: use a photo-device as a microwave devices in the switch (a); use the photo-device to power the microwave device in a switch (b); and use the photo-device to control the bias power to a microwave device (c).

technique requires the fewest number of devices and supply bias lines, however it is difficult to produce an optoelectronic device that functions well as both a microwave and optical device(see section 2.2). Another disadvantage of the this technique is the IL and ISO are a function of optical power.

The second technique uses the photo-voltaic device to bias a microwave device, which in turns provides the high and low impedance for the microwave switch. Again, this type of control does not require external bias for the microwave switch, but still requires substantial optical power (10's of mW). By using a microwave PIN diode in the switch, the IL and ISO equal that of a traditional microwave switch. Since the microwave PIN diode "on" impedance depends on bias current, the IL or ISO for this technique depend on the optical control power.

The third technique uses an optoelectronic device to control the bias voltage for the microwave device. This technique also has the advantage of good IL and ISO, as well as the potential of requiring much less optical power, however it requires an external supply bias line for the switch. If the switch is integrated with microwave amplifiers or other active components that require bias power, then the photonic switch does not necessarily add extra bias lines.

2.3.2 Investigating Optical Devices Used as Microwave Switch Components The first technique for optical control uses an optoelectronic device as the microwave device in a microwave switch, which is ideal due to the small numbers of optoelectronic and space required. Several PIN photodiodes with active areas ranging from 300 μm to 80 μm are tested in series and shunt configurations in hopes of being able to use the first type of optical control. Unfortunately, there is little difference (few dB) between the IL and ISO for the "on" and "off" states of the PIN photodiodes. In section 3.1.1 there is evidence presented that points to the bond wires used in the packaging of the photodiode leading to the poor IL and ISO performance. There is also significant loss (between 3 dB and 6 dB) due to the resistive nature of the photodiodes.

2.3.3 Investigating Optical Devices That Bias Microwave Switch Component The second technique of optical control using a photo-voltaic device to power a microwave device has the advantage in not needing an external power source [39, 40]. This technique is very difficult to test in the laboratory because of the difficulty in coupling large amounts of optical power onto the active area of the device. Most microwave PIN diodes require about 1 mA to 3 mA of current at 1 V to 1.5 V and it is not easy to generate enough current at the required voltage from a photodiode. Several photodiodes can be connected in series to produces the required voltage. A typical responsivity for infrared InGaAs photodiodes is 0.95 A/W, implying that 3 mW of optical power is needed at each photodiode to generate enough current to bias the microwave PIN diode "on".

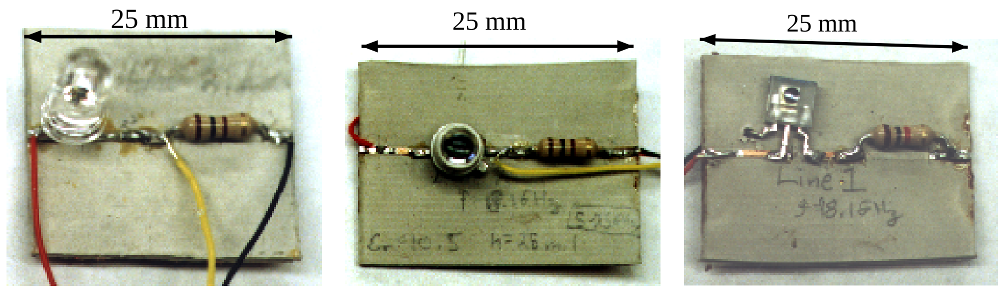
Eight PIN photodiodes connected in series to control a microwave varactor diode with 450 μW of optical power has been demonstrated [41, 42]. The success of this design relies heavily on the fact that reverse bias current for a varactor diode is only a few μA. The complexity of delivering large amounts of optical power to multiple photodiodes per switch makes this technique impractical for array applications.

Table 2.2: Performance data for devices in Figure 2.4

manufacturer	Siemens	Texas Instruments	QT Optoelectronics
type	Photo-transistor	Photo-amplifier	Photo-Darlington
part number	SHF 300	TSL250	LI4F1
wavelength	880 nm	840 nm	850 nm
Irradiance Responsivity	2-4 (mA/mW/cm^2)	80 (mV/μ W/cm^2)	3 (mA/mW/cm^2)
Rise/Fall Time	10 μs	90 μs	$250 - 300\,\mu$s
Collector Current	50 mA	1.6 mA	

2.3.4 Investigating Optical Devices That Control the Bias of Microwave Switch Component The third technique uses an optoelectronic device to control the bias current to microwave device. If a photodiode is used for the optoelectronic device, then 3 mW of optical power is still required to generate enough carriers for the 3 mA current to flow through the photodiode. This is a lot of optical power for control, especially when bias power is available. Photo-transistors and photo-amplifiers have much larger effective responsivity through the use of devices withgain, such as transistors or operational amplifiers. Two photo-transistorx and one photo-amp, Table 2.2, are tested with a microwave PIN diode. All three of the devices are able to control the PIN diode with very little optical power (μW), however the response time for the devices ranges from 10 μs to 250 μs. This is on the order of what is possible for electrical control for large arrays, which does nott warrant the extra cost and complexity of optical control. The physical size of the devices also prohibits their use in microwave arrays, Figure 2.4.

2.3.5 Photo-MESFET The photo-amps in sections 2.3.4 use MOS-FETs devices for gain which are inherently slow compared to microwave standards. Using the photo-amp concept, but a MESFET instaed as the gain device with a faster response time, the resulting photo-MESFET should be able to control the



(a) (b) (c)

Figure 2.4. Three optoelectronic device used for control of microwave PIN diode: SFH300 photo-transistor (a), LI4F1 photo-Darlington (b) and TSL251 photo-amp (c). Note that the total circuit dimension is roughly one free-space wavelength at 10 GHz.

PIN diode with very little optical power.

A true photo-MESFET would use the optical signal to generate carriers in its gate region. Without the ability to produce specialized devices such as photo-MESFETs, a circuit version is developed in this work, Figure 2.5. The photodiode controls the gate bias voltage on the MESFET, thus turning it "on" or pinching it "off". The circuit form of a photo-MESFET does require an additional negative voltage bias line to pinch off the MESFET.

Two photo-MESFETs are developed using the components given in Table 2.3. The first design uses the EG&G photodiode, but we later switched to the Fermionics photodiode for its smaller package with negligible change in performance. As mentioned in section 2.1, MESFETs can be used as the "high" and "low"

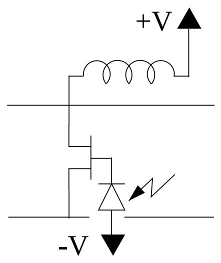


Figure 2.5: Schematic for the circuit implementation of a photo-MESFET

Table 2.3. The performance data of the MESFET and two photodiodes used to implement the photo-MESFET

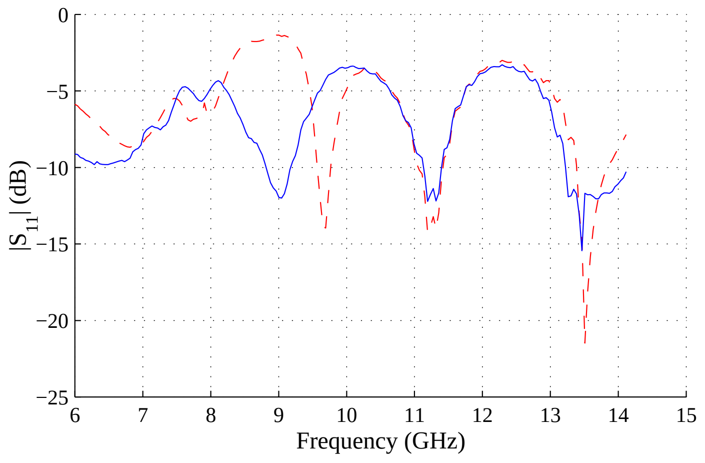
	HP AFT-26836		EG&G C30637ECER	Fermionics FD80S3
Type	General Purpose	Active Area (dia)	75 μm	80 μm
Gain	9 dB	Responsivity	0.86 A/W	0.85 A/W
Saturated Drain Current	30 mA	Capacitance	0.4 pF	0.4 pF
Bandwidth (GHz)	16 16	Bandwwidth (GHz)	3.5 3.5	2 2

impedance devices in microwave switches resulting in a simpler switch design than having the photo-MESFET control another microwave device. The shunt configured photo-MESFET switch performs the best at 9.1 GHz with an IL of -2.3 dB, ISO of -19.2 dB and bias current of 30 mA, Figure 2.6. A RL of -1.37 dB when the channel is “off” and -11.7 dB when the channel is “on” shows that there is 1 dB to 2 dB of power loss in the MESFET itself.

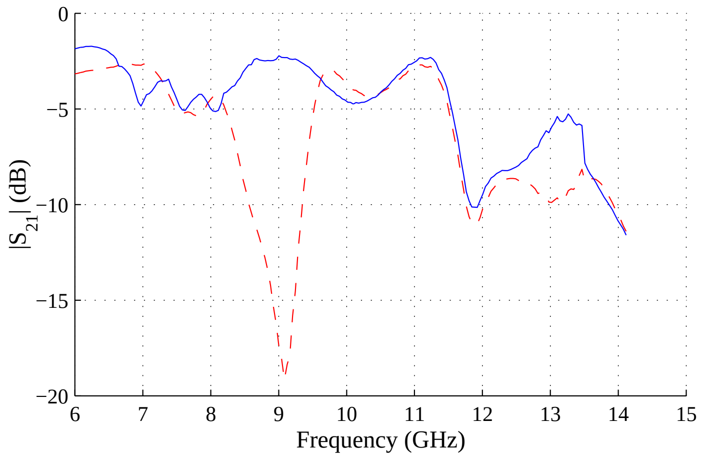
The 30 mA of current required by the photo-MESFET is undesirable for use in the switch, so the photo-MESFET is tested with microwave diodes. Two of the diodes used with the photo-MESFET are given in Table 2.4 and have surprisingly similar results compared to the photo-MESFET. Like the MESFET, the diodes give the best IL and ISO in a shunt configuration. The HSMP3892, Figure 2.7, has IL of -4.4 dB and ISO of -17.4 dB at 9.08 GHz for a bias current of 30 mA. The MA4E2054, Figure 2.8, has an insertion loss of 4.2 dB and ISO of -20.2 dB at 9.12 GHz for a bias current of 3.3 mA. The MA4E2054 diode has the same performance as the other

Table 2.4. Performance characteristics for two microwave diodes tested with the photo-MESFET.

Manufacture	Hewlett Packard	M/A-Com
Type	PIN Diode	Schottky Diode
Part number	HSMP-3892	MA4E2054
Capacitance	0.30 pF	0.35 pF
“on” Resistance	2.5 Ω	10 Ω

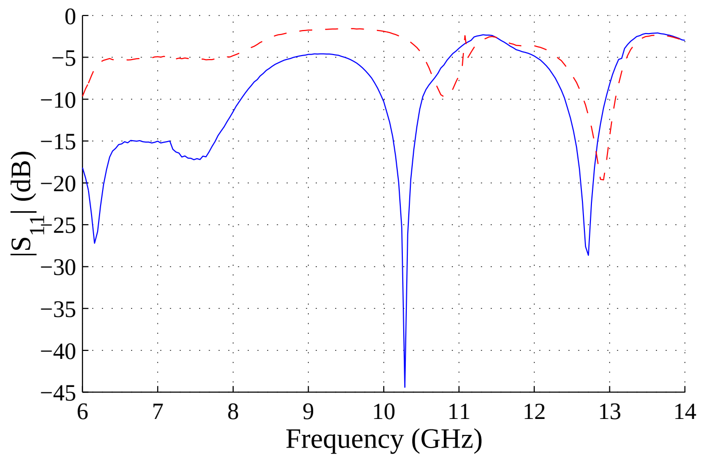


(a)

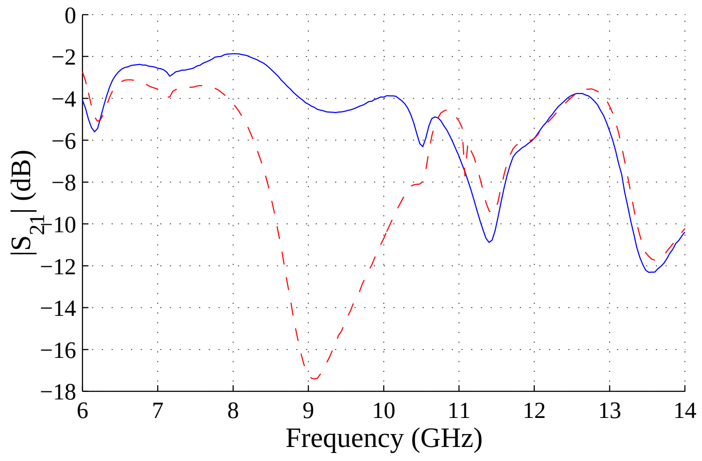


(b)

Figure 2.6. The measured RL (a), IL (blue –) and ISO (red --) (b) for a photo-MESFET in a shunt configuration

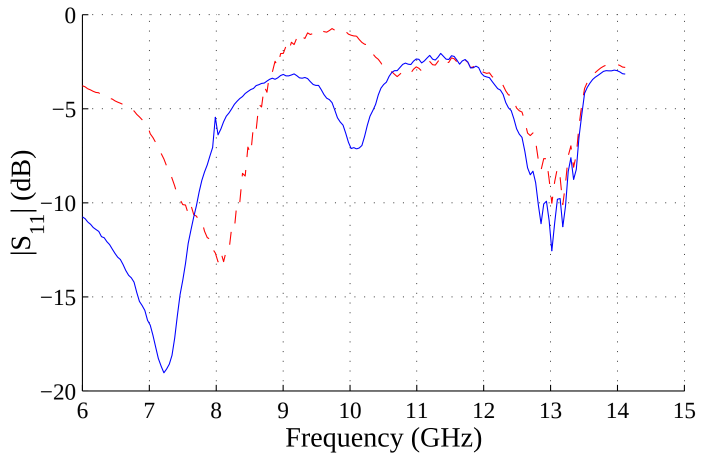


(a)

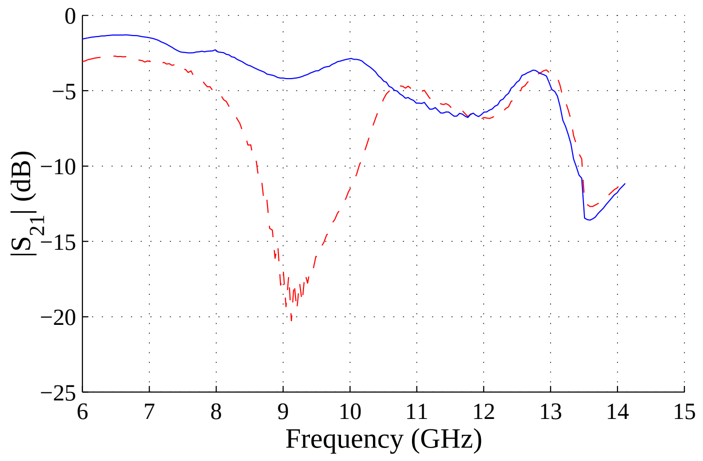


(b)

Figure 2.7. The measured RL (a), IL (blue –) and ISO (red - -) (b) for a HSMP-3892 in a shunt configuration. All of the s-parameter measurements were maded using HP 8510C Network Analyzer.



(a)



(b)

Figure 2.8. The measured RL (a), IL (blue –) and ISO (red - -) (b) for a MA4E2054 in a shunt configuration

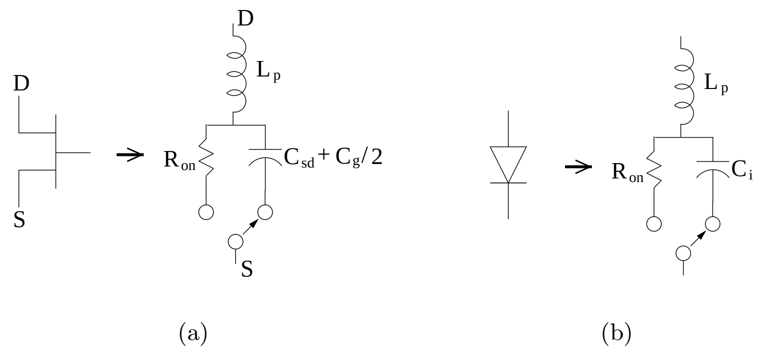


Figure 2.9. Lumped element models for the "on" on "off" characteristics for a MESFET (a) and a diode (b).

devices with only 3.3 mA of current, and is the device chosen for the SPDT switch to be used in the array.

2.3.6 Resonant Switch Operation The similar performance between three very different devices (MESFET, PIN diode and Schottky diode) hints towards an unusual mode of operation for the devices in the switch. All three devices have similar IL and ISO of about -20 dB in a very narrow bandwidth around 9.1 GHz. Another interesting anomaly is the fact that the switch is "off" when the device is "off" and vise versa for a shunt configuration, which is opposite to the theory presented in section 2.1.

The narrow bandwidth of the ISO suggests a resonant operation of the device. The small circuit models for the devices and their packages, Figure 2.9, show that a resonance can occur between the junction capacitance of the device and the bond wire inductance in the package when the device is "off". The series resonance of the inductor and capacitor forms a very low impedance that shorts out the transmission line turning the switch "off".

When the device is "on", the bond wire inductance is in series with the "on" resistance of the device. Estimating that the bond wire inductance of about 0.5 nH for a 0.5 pF junction capacitance resonates at 9 GHz, the resulting IL is about

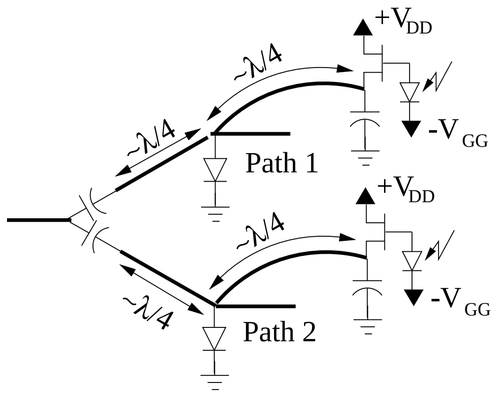


Figure 2.10. Schematic for an optically controlled SPDT switch with PIN diodes in a shunt configuration. Thick lines represent transmission lines.

1.2 dB. This is about 1 to 2 dB better than measured, suggesting some additional "ohmic" loss. The RL for this model is about 3.5 dB which is consistent with measurements for the three devices. Obviously, the package bond wire inductance limits the performance of the switch and a chip PIN diode would perform better. M/A-Com MA4GP032 chip PIN diodes were purchased, but would take 10 weeks to deliver, so work is done on a resonant switch design in the interim as a learning and backup process.

2.4 SPDT Switch Design

The design of a SPDT switch is more elaborate than presented in section 2.1. Care must be taken to isolate the switching bias currents for the two diodes and to suppress loading of the microwave signal by the diode bias circuitry (i.e. photo-MESFET). Series capacitors are used to separate the switching bias currents for the two diodes Figure 2.10. A 70 Ω microstrip line (the highest impedance line that we can mill) connects the photo-MESFET to the diode. A 100 pF capacitor is in shunt with the microstrip line $\lambda/4$ away from the diode, which presents a high impedance to the diode for its microwave short. The initial development of the SPDT switch is done with the HSMP3892 diode, but latter was changed to the MA4E2054 diode

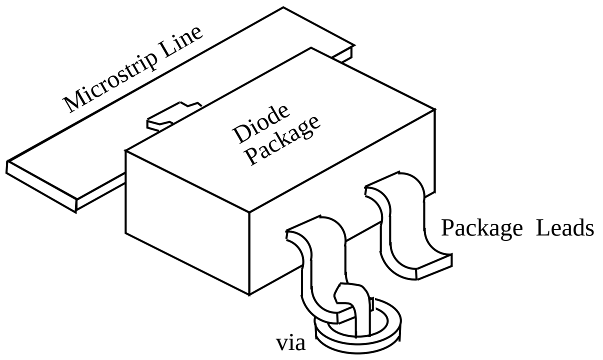


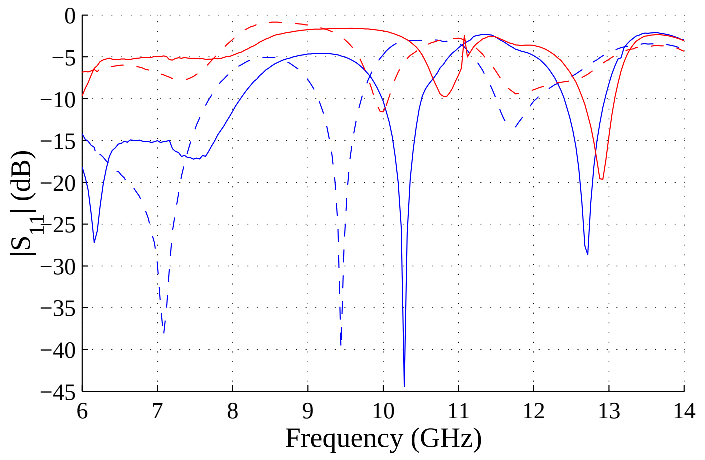
Figure 2.11. Illustration of diode connected to microstrip line with annular ring around the ground via

for its lower current level performance.

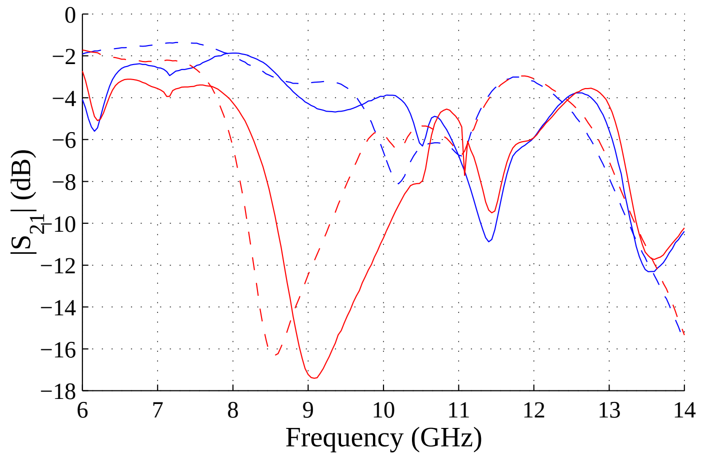
2.4.1 Manufacturing Tollerances A robust repeatable switch design is necessary for success in arrays that contain hundreds of these switches. The first SPDT switch constructed with the HSMP3892 PIN diode demonstrated the sensitivity of the switch to manufacturing variations. The resonant frequency of the diode is very sensitive to how they are connected to the microstrip line. For example, two diodes from the same lot are used in a switch. Both are shorted to ground using a wire, but one has an annular ring, Figure 2.12. The resonant frequency of the switch with the annular ring moved from 9.08 GHz to 8.56 GHz, Figure 2.12(b).

Another experiment was performed to investigate diode placement, Figure 2.13. Since the diode impedance is reflected to the Y-junction, its placement helps determine the RL for the switch as well as the operation frequency. By moving the diode 0.3 mm away from the Y-junction the resonant frequency moves from 8.9 GHz to 8.7 GHz, Figure 2.14(b). The ISO and IL stays relatively unchanged at approximately 23 dB and 5.4 dB respectively.

In order to insure good placement of the diodes, notches are cut into the microstrip lines for the diodes Figure 2.15. To test the effectiveness of the notches a set of diodes are measured in the circuit, remove and then replace back into the



(a)



(b)

Figure 2.12. Measured RL (a), IL (blue) and ISO (red) (b) for a HSMP-3892 PIN diode in a switch with shunt configuration without (–) and with an annular ring around the ground via (- -).

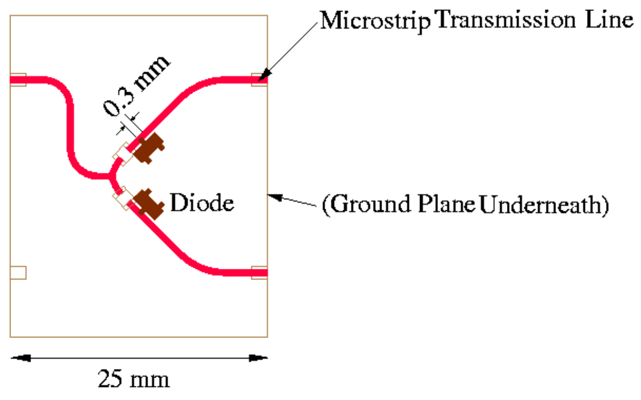
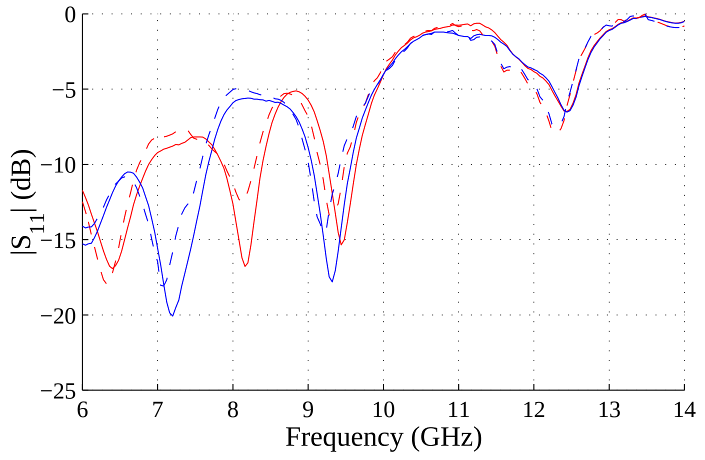


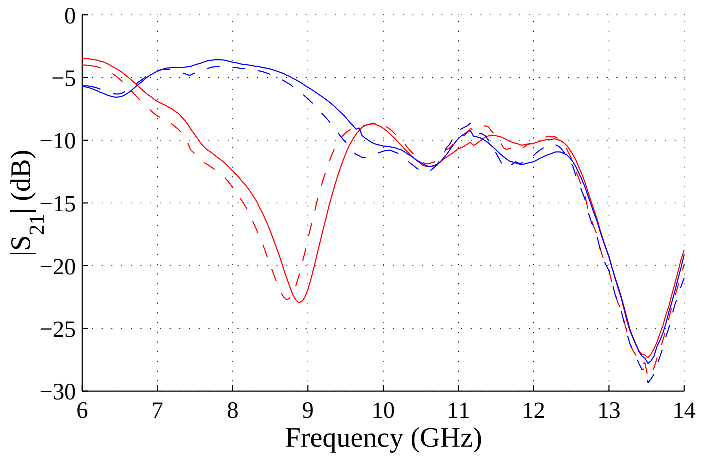
Figure 2.13. Layout of the SPDT switch illustrating the change in placement for the diode along the microstrip transmission line.

circuit. The overall resonant frequency of the switch did not change, however the ISO was reduced by 2 dB from 19.3 dB to 17.4 dB and the IL changed from 7.4 dB to 3.2 dB, Figure 2.16(b). There are two possible sources for the change. First, the soldering, de-soldering, and re-soldering may have done some thermal damage to the devices, effecting their performance. Second, the solder bubble connecting the diode to the microstrip line has a different shape each time and probably filled the notch gap better in the second set of measurements.

Another experiment examines the switch sensitivity to different sets of diodes. The measurements resulting fro placing a new set of diodes shows a 0.16 GHz change in resonant frequency between the old and new sets of diodes from 8.72 GHz to 8.56 GHz, Figure 2.17(b). The ISO improved from -19.3 dB to -21.0 dB, and the IL improved from -7.4 dB to -4.8 dB. These simple experiments show that even with meticulous care in fabricating the SPDT switch, it will be difficult to construct several switches with similar enough performance. This becomes a problem when the resonant frequency of the switch needs to be the same as the resonant frequency of the patch antenna in each element of the array.



(a)



(b)

Figure 2.14. The measured RL (a), IL (blue) and ISO (red) (b) for the SPDT in Figure 2.13 before (–) and after (--) the new diode placement.

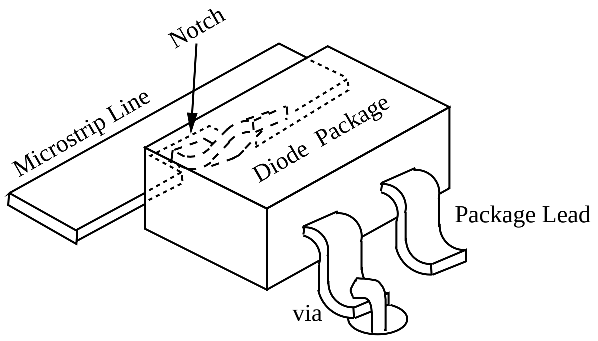
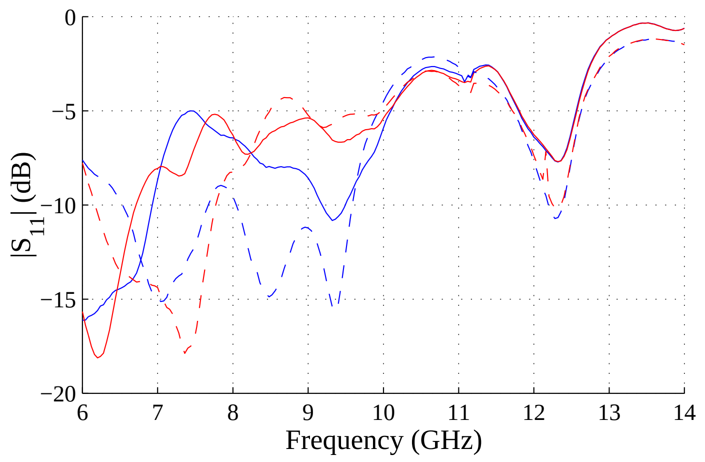


Figure 2.15. An illustration of a diode connected to a microstrip line using a notch for percise placement.

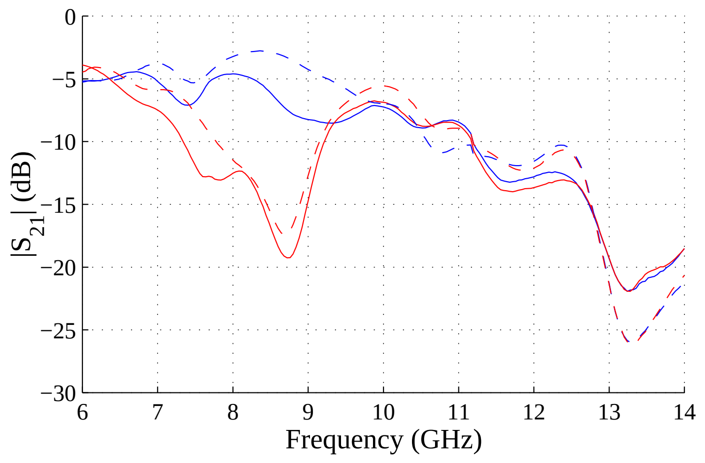
2.4.2 Four Different SPDT Switch Designs Using a resonant diode in a SPDT switch proves to be a difficult to simulate and predict with any accuracy, therefore a shot gun approach to the problem is adopted. Four different SPDT switches are made with different lengths of transmission line between the MA4E2054 Schottky diodes and the Y-junction Figure 2.18.

From the experiments in section 2.4.1, variations in the diodes, placement and soldering resulted in switches that worked at frequencies lower than the 9.1 GHz response predicted in section 2.3.6. The four SPDT switch designs have incremental changes in the layout based on a switch design centered at 8 GHz. Theoretically, the distance between the diode and the Y-junction should be $\lambda/4$. The four switches designated "a","b","c" and "d" use transmission line lengths of 0.8 $\lambda/4$, 0.89 $\lambda/4$, $\lambda/4$ and 1.07 $\lambda/4$. From the IL and ISO measurements, circuit "a" has the best performance with a resonant frequency at 8.15 GHz, IL=3.6 dB, ISO=24.6 dB and RL=6.3 dB, as shown in Figure 2.19.

2.4.3 Effects of Optical Control Until now, most of the effort focuses on the RF performance of the switch and how it relates to the diodes, setting the optical control aside. The optical control part of the SPDT switch requires a photo-MESFET to control the diode current and optical mounts to align fibers to the photodiode, as shown in photograph Figure 2.20. To understand the effects of the

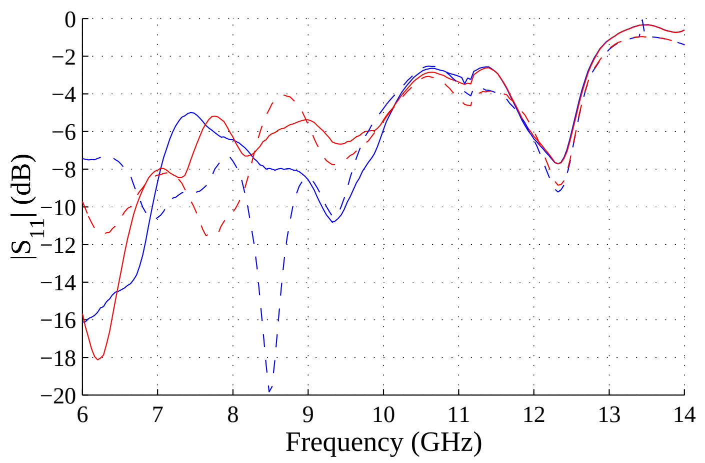


(a)

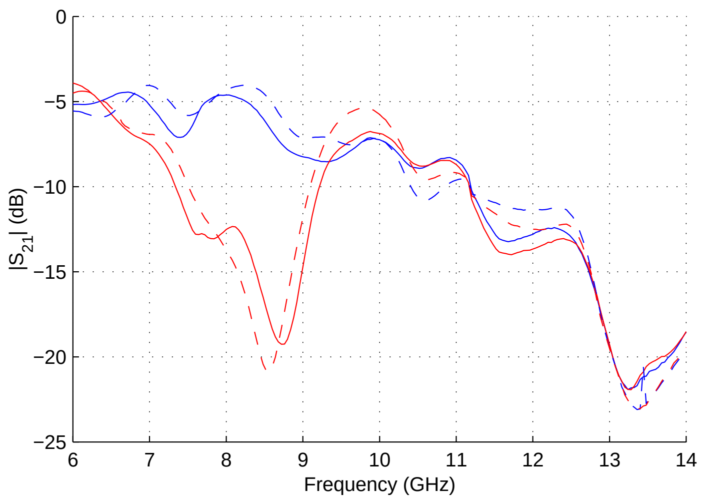


(b)

Figure 2.16. Measured RL (a), IL (blue) and ISO (red) (b) for diodes placed using notches as shown in Figure 2.15. The same set of diodes are measured in the switch (–), then removed and replaced (- -)



(a)



(b)

Figure 2.17. Measured RL (a), IL (blue) and ISO (red) (b) for two sets of diodes (– and - -) in a SPDT switch that uses notches for placement.

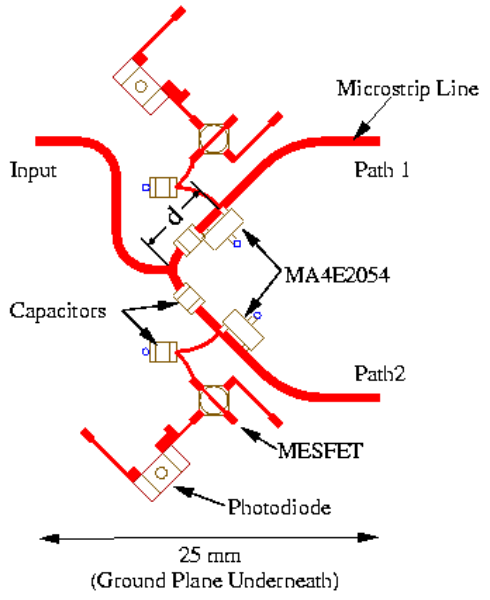
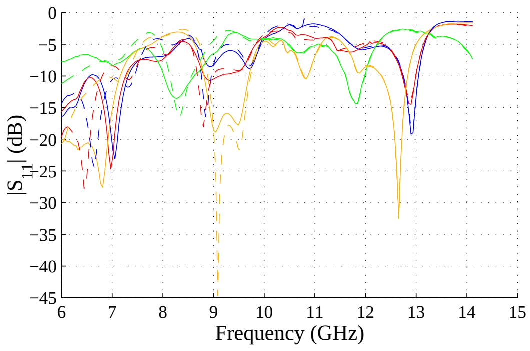


Figure 2.18. The general layout of optically controlled SPDT switch for the circuit in Figure 2.10. The placement of the diode (distance d) is adjusted to optimize the SPDT switch performance.

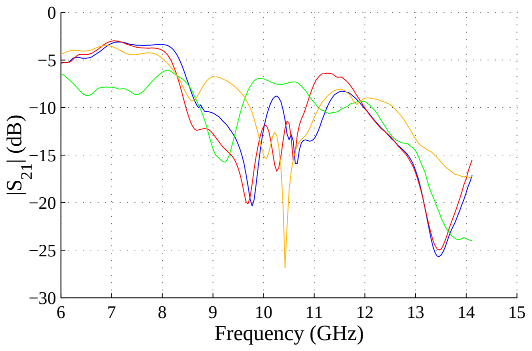
optical control on the SPDT switch, the “a” SPDT switch is tested under different stages of construction. The first stage has just the Schottky diodes and capacitors, with the bias control provided through a bias tee. The second stage has MESFETs added with their gate bias being controlled. The third stage adds the optical mount. The optical mount is made out of FR4 material with a 3 mm minimum spacing between the FR4 and the microstrip lines. Finally, the photodiode is added and the switch is controlled using a laser-didoe. At each stage of the experiment, IL and ISO are affected with a net result of 2-3 dB increase in IL and ISO, as shown in Figure 2.21.

2.5 Unit Cell Array Element Design

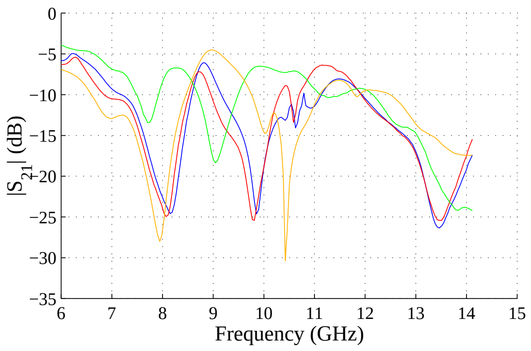
The unit cell design, Figure 3.1, is essentially two SPDT switches connected



(a)



(b)



(c)

Figure 2.19. The measured RL (a), IL (b) and ISO (c) for four different SPDT switch designs illustrated in Figure 2.18: d=0.8 $\lambda/4$ (blue), d=0.89 $\lambda/4$ (red), d=$\lambda/4$ (green) and d=1.07 $\lambda/4$ (yellow). All four circuits are measured with the same calibration.

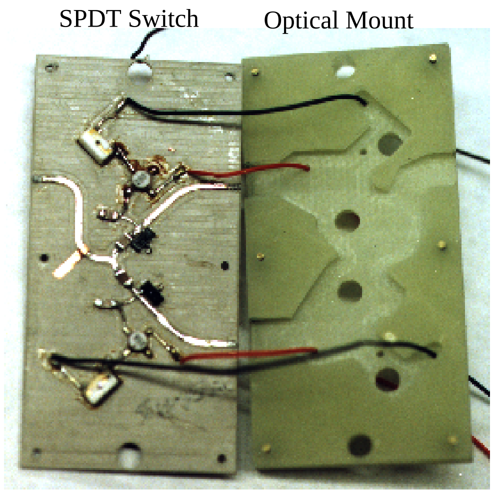
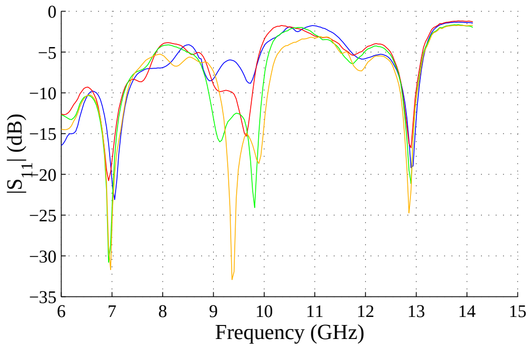


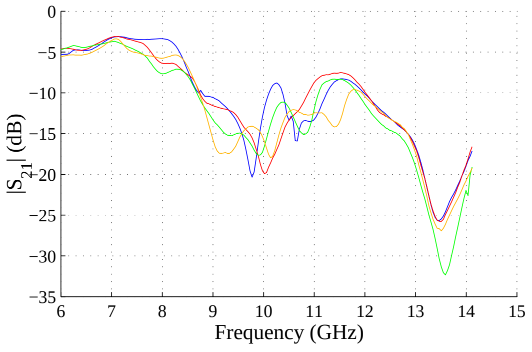
Figure 2.20. Photograph of Optically-controlled 8 GHz SPDT switch with optical mount

back-to-back with the power amplifier (PA) and low noise amplifier (LNA) connecting them. The photo-MESFET circuit is modified to contain a push-pull photodiode pair for rapid turning “on” an “off” of the MESFET [43]. Each MESFET in turn provides the bias control for the appropriate pair of PIN diodes in the two SPDT switches. The switches and MMICs are powered through a 5 V supply line with a 1 kΩ current-limiting resistor for the switches. The two antennas are connected to the Y-junction ports of the SPDT. Due to the amount of circuitry in the unit cell, a substrate with a high dielectric constant is needed to shrink the overall circuit layout. Rogers’ Duriod 6010.5 with a thickness of 0.508 mm (25 mils) is used.

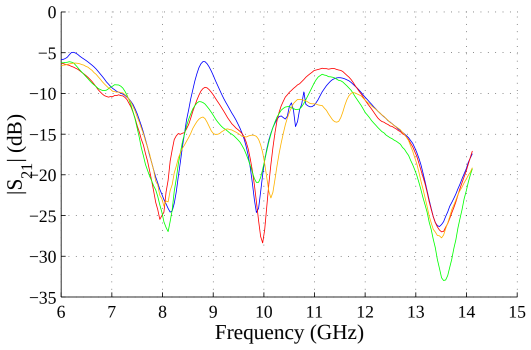
2.5.1 Low Noise Amplifier and Power Amplifier The MMIC amplifiers, Table 2.5, are chosen for three practical reasons. They are low cost, available with short delivery times, and require only a single supply voltage. Both MMICs have input and output matching filters that reject the lower frequency switching current, Figure 2.23. The matching filters are designed assuming gold bond wires that are 0.254 mm (10 mil) long and have a diameter of (0.7 mils). The MMICs are mounted on a brass pedestal to minimize total bond wire lenght, Figure 2.24.



(a)



(b)



(c)

Figure 2.21. The measured RL (a), IL (b) and ISO (c) for d=0.8 $\lambda/4$ SPDT switch design (blue), with MESFET (red), with MESFET and optical mount (green), and with optical mount and photodiode (yellow).

Table 2.5: Performance data for the LNA and PA

Manufacture	Hewlett Packard	United Monolithic Semiconductor
Part number	HMMC-5618	CHA2036
Type	Power Amplifier	Low Noise Amplifier
Gain (dB)	14	16
BW (GHz)	6-20	7-13
1 dB Compression (dBm)	18	10
Supply Voltage (V)	5	5
Noise Figure (dB)	5.5	2

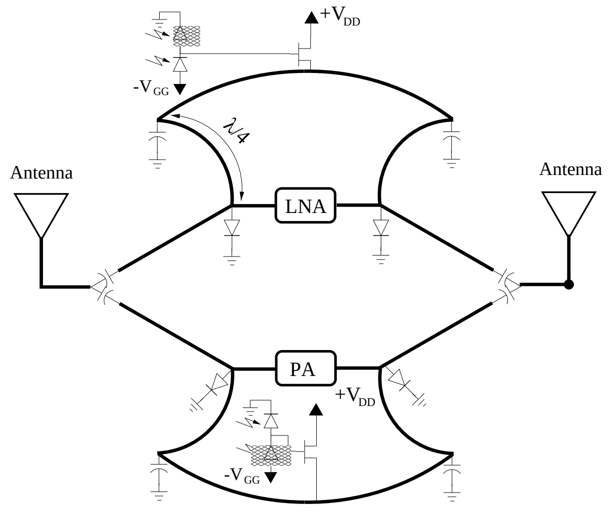


Figure 2.22. Unit cell schematic with push-pull photodiodes for faster control. Thick lines represent transmission lines.

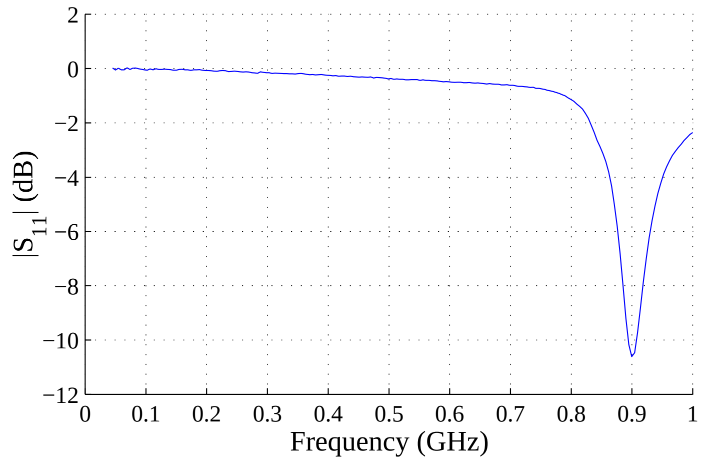


Figure 2.23. Measurement of RL for LNA at lower frequencies ($< 1\,\text{Ghz}$) demonstrating the isolation of the LNA from the diode circuitry.

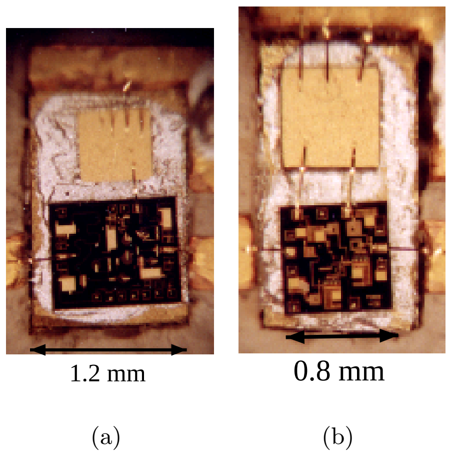


Figure 2.24. Photographs of LNA (a) and PA (b) resting on brass platforms and biased through 100pF chip capacitors.

2.5.2 Bias Line Design Supplying power to all the MMICs and switches in an array proves to be a difficult problem and has led to array instabilities [1, 44]. The instabilities arise from free-space coupling of the microwave signal onto the power bias lines. To minimize the coupling to the bias lines, a filter bias line is used. A filter bias line has a cascade of high and low impedance transmission lines that are $\lambda/4$ long at the design frequency, resulting in a notch filter, Figure 2.25. A problem with filter bias lines is their small cross-sectional area for the high impedance sections (assuming microstrip lines), which increases the resistance of the

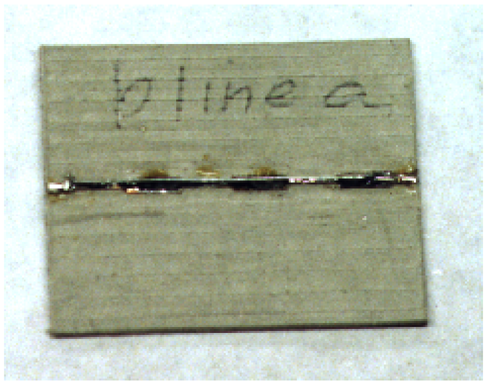


Figure 2.25. Photograph of filter bias line with a 30-gauge wire for reduced resistance.

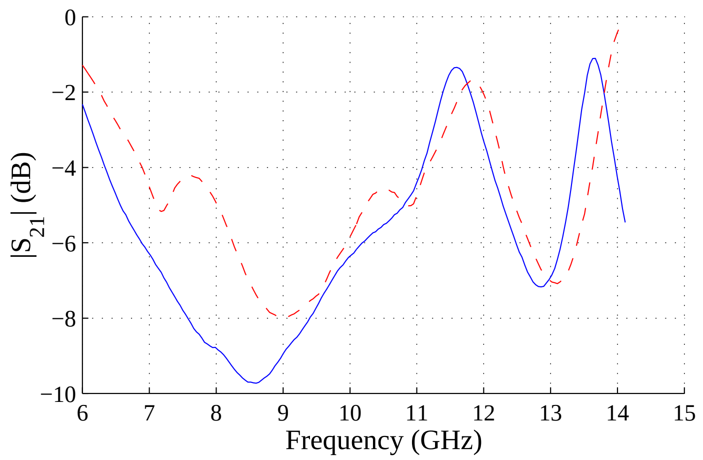


Figure 2.26. Measured IL for a 25 mm long filter bias line without 30 gauge wire (blue –) and with the wire (red - -).

bias line. To increase the cross sectional area of the bias line, a 30-gauge wire is soldered down the center of the line. Since the high frequency current density is largest at the edges of the line, the wire should have minimal effect on RF filter performance. The measurements of a 25 mm long filter bias line designed at 8 GHz shows about 10 dB suppression of 8.5 GHz signals without the wire, and about 8 dB with the wire, Figure 2.26.

2.5.3 Antenna Design Two types of printed antennas are used most often in planar lens array designs: slot antennas and microstrip patch antennas. Slot antennas consist of a slot cut into a ground plane and fed with a microstrip line(Figure 2.27) or a CPW. Slot antennas radiate on both sides of the ground plane in a hemispherical pattern with nulls in the direction of the ground plane [45]. For this reason, when slot antennas are used in planar lens arrays, polarizer are needed to isolate the two sides of the array [44, 1]. For lens arrays used in an angle diversity communication system, the addition of the polarizers would add unnecessary aberrations to the lens, reducing the communication systems performance.

Microstrip patch antennas consist usually of a rectangular piece of metal above a ground plane. They are fed using microstrip lines (Figure 2.28), coaxial

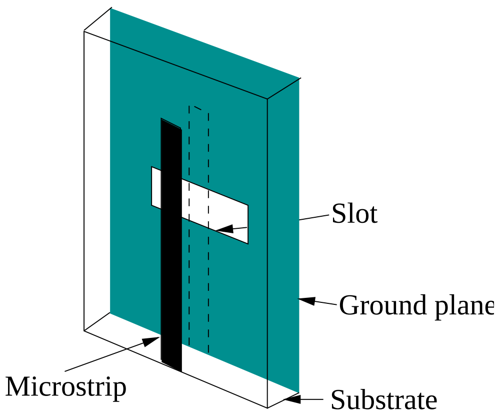


Figure 2.27: Illustration of a microstrip fed slot antenna.

lines or aperture feeds. The microstrip patch antenna radiates due to fringing electric fields at the edges of the patch, in a hemispherical pattern on one side of the ground plane with nulls in the ground plane direction [45]. Using microstrip patch antennas in a planar lens array eliminates the need for polarizers. Unfortunately, microstrip patch antennas are resonant and have bandwidths on the order of 2-4%. The bandwidth can be increased using electrically thick substrates at the cost of radiation efficiency due to substrate modes [21].

A 8 GHz non-radiating edge fed patch antenna is designed using a method of moments simulator *Ensemble* by Ansoft. The patch is 5.47 mm by 7.82 mm with

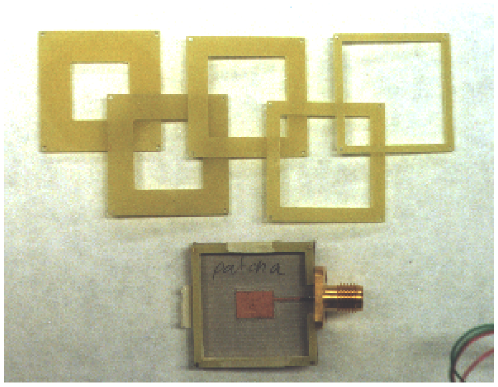


Figure 2.28. Photograph of 8 GHz patch antenna on 10.5 dielectric substrate. Accompanying the patch antenna are five FR4 apertures used to measure the effects of FR4 mount on the patch radiation

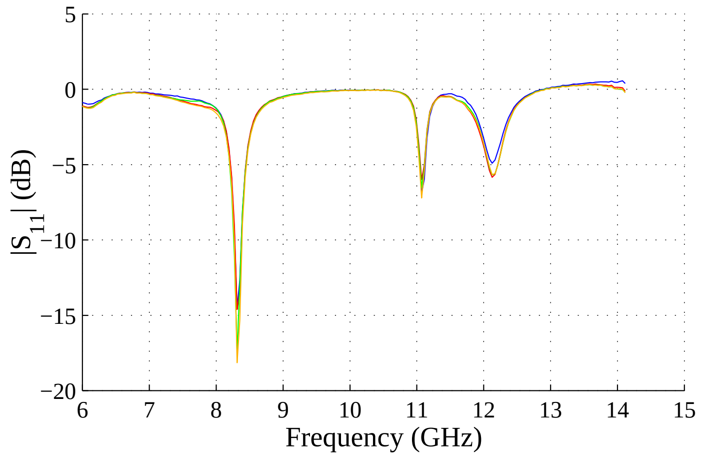


Figure 2.29. The measured RL for the patch antenna as it radiates through different size mounts: no mount (blue), 20 mm by 20 mm (red), 17.5 mm by 17.5 mm (green) and 15 mm by 15 mm (yellow). The FR4 mounts with apertures 20 mm by 20 mm or larger have negligible effect on the antenna radiation.

the microstrip feed 0.76 mm from the center of the non-radiating edge. *Ensemble* is used to estimate the coupling between the patch antenna and a microstrip line in its vicinity. In order to maintain 30 dB isolation between the patch antenna and nearby microstrip transmission lines to prevent oscillations, it was found that all lines should be 3 mm from the radiating edge and 2 mm from the non-radiating edge of the patch antenna.

Besides the patch antenna coupling to other parts in the circuit, the patch antenna needs to radiate through a aperture in the optical mount. Since the FR4 material used in the optical mount is lossy at 8 GHz, by measuring the return loss of the patch antenna as the antenna radiates through different apertures, we can determine the effect of the mount. As the FR4 material begins to load the near field of the patch antenna, the return loss should decrease. Five different apertures were built: 15 mm by 15 mm, 17.5 mm by 17.5 mm, 20 mm by 20 mm, 22.5 mm by 22.5 mm and 25 mm by 25 mm, (Figure 2.28). The two apertures smaller that 20 mm by 20 mm reduce the reflected power, as seen in Figure 2.29. A 20 mm by 20 mm aperture is the smallest aperture that the patch antenna can radiate through unchanged. Radiation

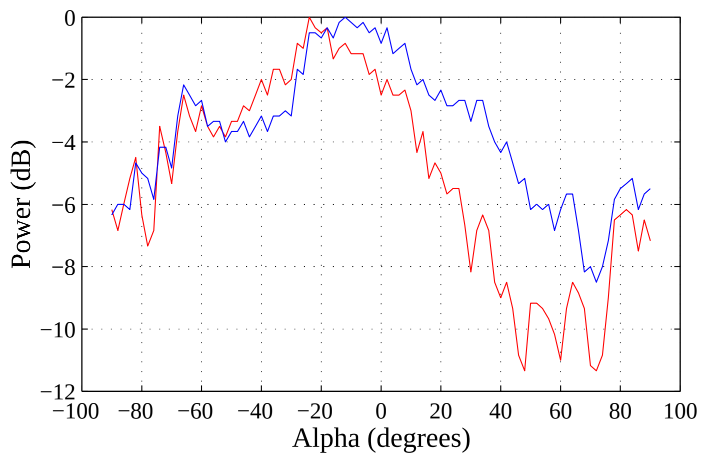


Figure 2.30. Measured radiation patterns for the 8 GHz patch antenna alone (blue) and radiating through 20 mm by 20 mm aperture

patterns were taken with and without the FR4 aperture and are shown in Figure 2.30. The aperture attenuates the patch radiation pattern on the sides and the ripples are due to multi-path effects in the test setup (desicribed later in section 4.3.2).

2.5.4 Slot Coupler vs. Vias Feed With a patch antenna on each side of the unit cell and a ground plane separating them, it is necessary to devise a means of coupling signals from one side of the unit cell to the other. One technique uses a slot coupler, Figure 2.31, which has been shown to have coupling losses of 2 dB and operating up to 25 GHz [46]. Early design work in slot couplers at

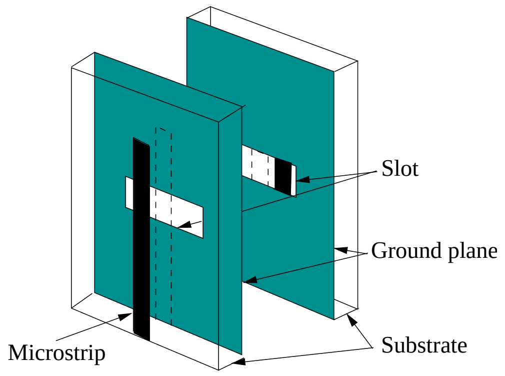


Figure 2.31: Illustration of microstrip coupler made from two pieces of substrate.

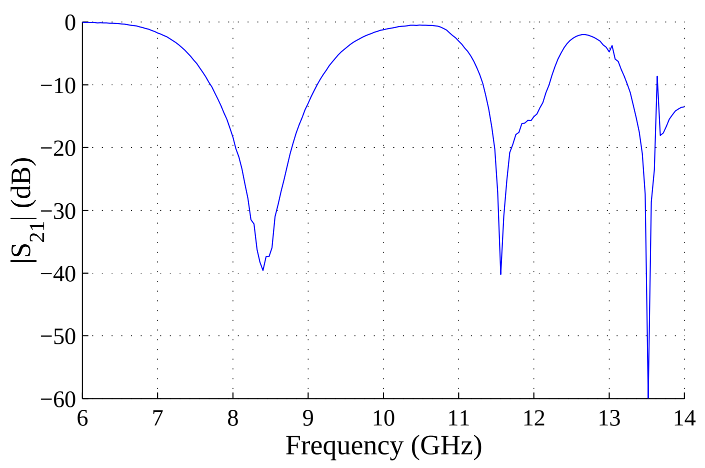


Figure 2.32: Measured IL for 10 GHz slot coupler on 10.5 dielectric substrate.

10 GHz produced simulated results of coupling loss of 2.76 dB with a RL of 34 dB, showing that slot couplers are prone to substrate modes, Figure 2.32. The measured results for the slot coupler has 0.5 dB of coupling loss and 22 dB of RL at 10.5 GHz. Unfortunately, in order to produce this good performance, meticulous care is taken to align the slots and to ensure good electric contact between the two substrate ground planes, using silver epoxy, without any air gaps. By applying pressure to the slot coupler to squeeze the air gaps the coupling loss changes from about -3 dB to the 0.5 dB. In addition, to the fabrication problems, the currents in slot couplers generally extend several substrate thicknesses beyond the slot, reducing available real-estate. For these reasons, the slot coupler is impractical for the design of the unit cell. Instead, a 0.5 mm diameter brass rod is used as a via to connect the two sides of the array with about 1 dB coupling loss, Figure 2.33.

2.5.5 Unit Cell Layout The layout for the unit cell follows from the layout of the SPDT switch. The MMICs are placed between the two back-to-back SPDT switches with 0.38 mm (15 mil) brass platforms to raise them up from the center ground plane, Figure 2.34(a). The push-pull photodiodes are connected to the MESFET and the -2.3 V bias line running horizontally across the top and bottom

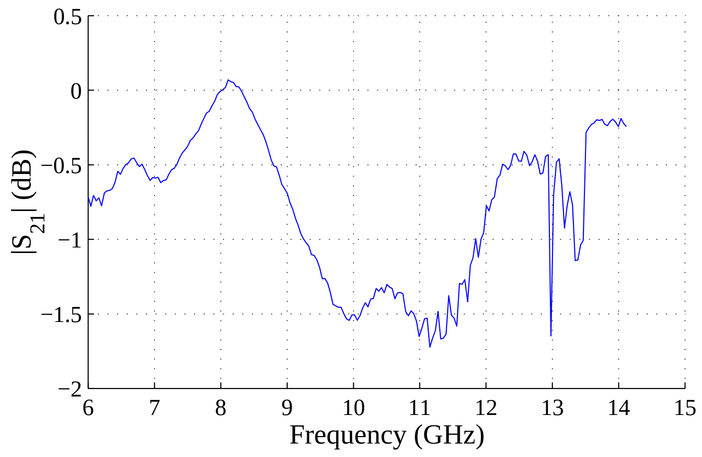


Figure 2.33. The measured loss for a 0.5 mm diameter via connecting two microstrip lines on 0.64 mm, $\epsilon_r = 10.5$ substrate material.

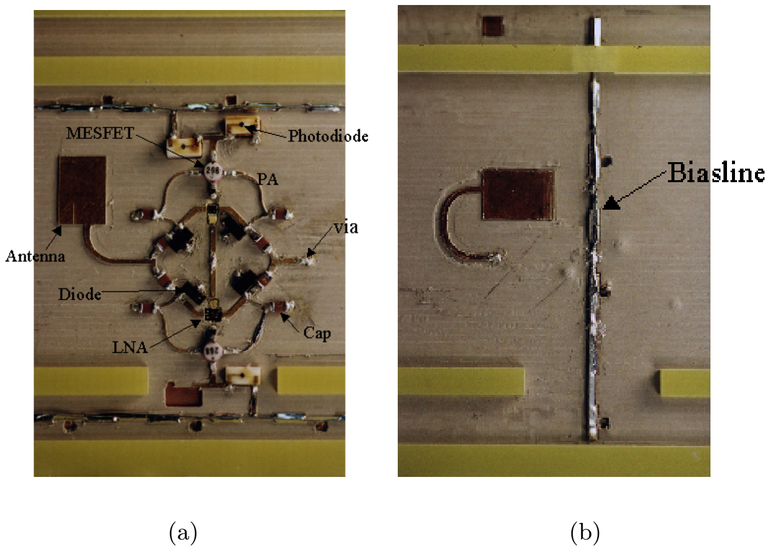


Figure 2.34. Photograph showing the active (a) and passive (b) sides of the unit cell

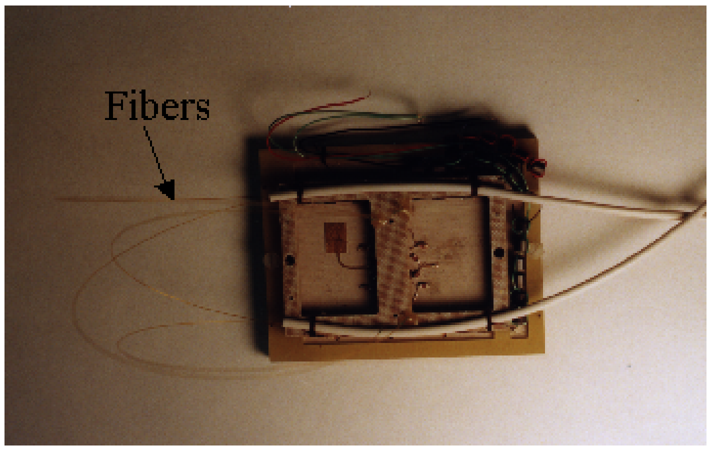


Figure 2.35. Photograph showing the optical mount placed on the unit cell with the optical fibers.

of the unit cell. The 5 Volt bias line runs vertically on the passive side of the array which also contains one of the antennas, Figure 2.34(b). The optical mount contains the optical fibers and the 20 mm by 20 mm aperture through which the other patch antennas radiates through, Figure 2.35.

2.5.6 Testing Unit Cell The gain of the unit cell is tested in the setup shown in Figure 2.36 for both transmit and receive modes. In both cases, no measurable power was transmitted through the unit cell. As mentioned before, there

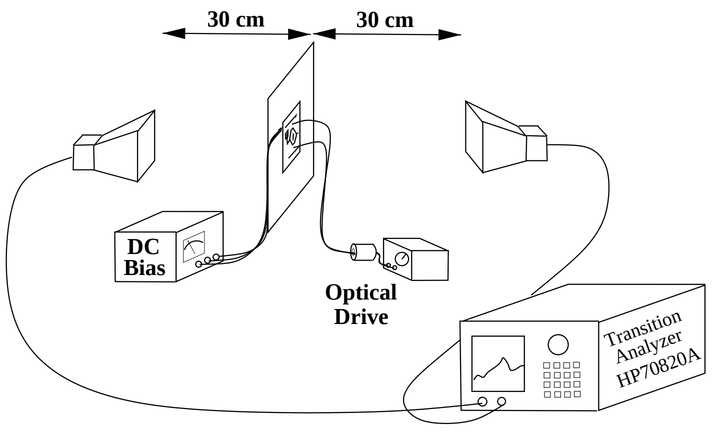


Figure 2.36. Illustration of the test setup used to measure the unit cell. The test setup is calibrated to an aperture the size of the unit cell.

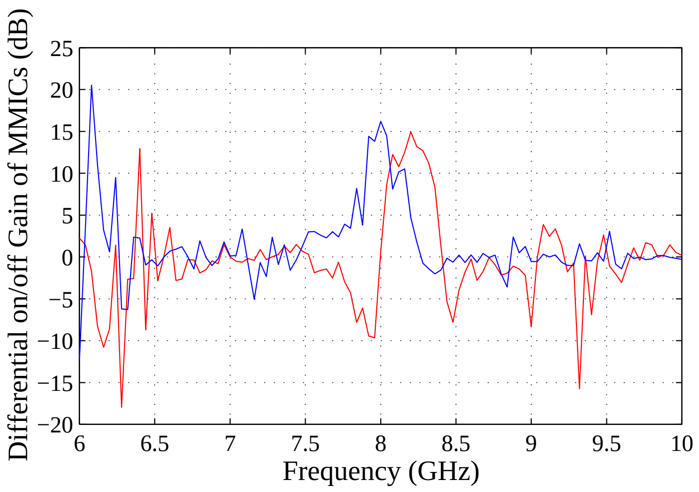


Figure 2.37. Measured differential on/off gain for the two unit cells containing only LNA (blue) and PA (red). The noise in the measurement is due the dynamic range of the HP 70820A.

are many difficulties in developing a unit cell with a resonant switch and microstrip patch antenna such that their resonant frequencies overlap. Instead, two unit cells are made, one for transmit mode and one for receive mode, with the Schottky diodes removed. The differential on/off gain of each unit cell agrees with the predicted gains of the MMIC amplifiers at 16.2 dB for the LNA and 14.9 dB for the PA, Figure 2.37. It is also interesting to point out that the patch antennas radiated at different frequencies even though they were made identically and at the same time. This demonstrates the sensitivity of the patch design to manufacturing deviations. A new wide bandwidth switch design is to needed overcome the problem of patch antenna variations.

CHAPTER 3

10 GHZ CHIP-PIN DIODE SPDT SWITCH

3.1 Improved Switch Design

In the previous chapter, a resonant switch was described. This architecture is narrow bandwidth and sensitive to parameter variations. Since the ultimate goal is to integrate this circuit into an array, a design is required which can be manufactured to behave the same many times. Figure 3.1 shows an improved unit cell design. The series and shunt capacitors used to isolate the diodes and provide a microwave short for the bias line have been replaced with high pass filters (HPF) and low pass filters (LPF), respectively. In addition to the push-pull photodiodes, there are push-pull MESFETs for rapid turning on/off of the PIN diodes. Complementary optical pulses control each set of photodiodes. For example, when the transmit photodiodes are illuminated (and the receive photodiodes are not), MESFETs labeled 2 and

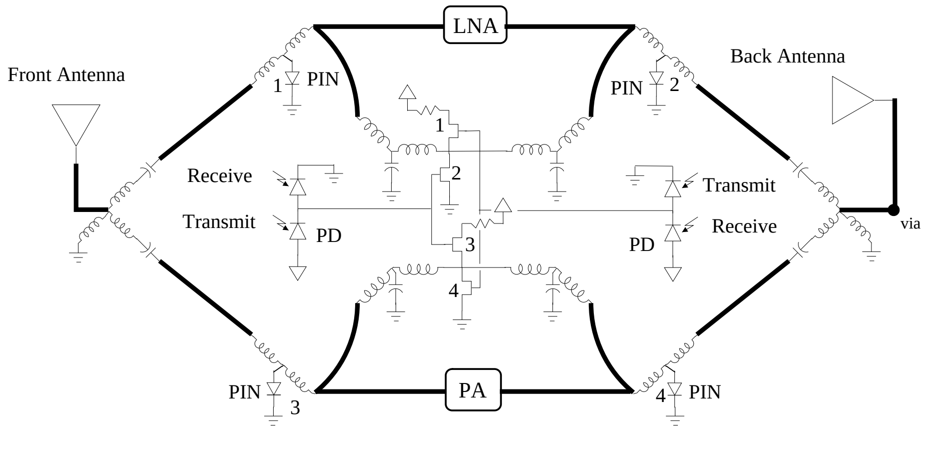


Figure 3.1. Improved unit cell design with push-pull photodiodes and MESFETs, chip PIN diodes, HPFs and LPFs.

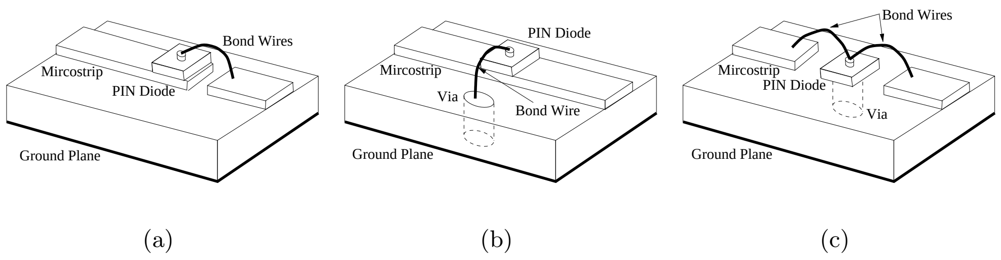


Figure 3.2. Three different configurations for connecting a chip diode into a switch: series (a), shunt (b) and T-network (c).

3 in Figure 3.1 are quickly pinched "off" while MESFETs 1 and 4 are turn "on". MESFET one sources current for PIN diodes 1 and 2, while MESFET four discharges the junction capacitance of PIN diodes three and four. With PIN diodes 1 and 2 biased "on", the LNA channel is switched "off", and when the PIN diodes, 3 and 4 biased "off", the PA channel is switched "on".

3.1.1 MA4GP032 Chip PIN diode As demonstrated in Chapter 2 the package inductance of the diodes and MESFETs limits their switching performance at X-band. The M/A-Com MA4GP032 chip PIN diodes which have an "on" resistance of $1.5\,\Omega$ and junction capacitance of 0.12 pF, are used in the new switch design. There are three configurations for the PIN diode in a switch, (series, shunt and T-network), Figure 3.2. Using lumped element models for the bond wires (inductors) and PIN diode (resistor for "on" and capacitor for "off"), *MatLab* calculations predict the IL and ISO for the series configuration to be 1.8 dB and 1.7 dB, and for the shunt configuration to be 1.2 dB and 0.95 dB at 10 GHz. The measured IL and ISO are consistent with the lumped element model, showing 0.59 dB IL and 2.1 dB ISO for the series configuration and 0.3 dB IL and 2.3 dB for the shunt configuration, Figure 3.3. However, the T-network configuration performance is greatly improved through reactance cancellation with the bond wires inductance.

There are three design parameters that affect the performance of a PIN

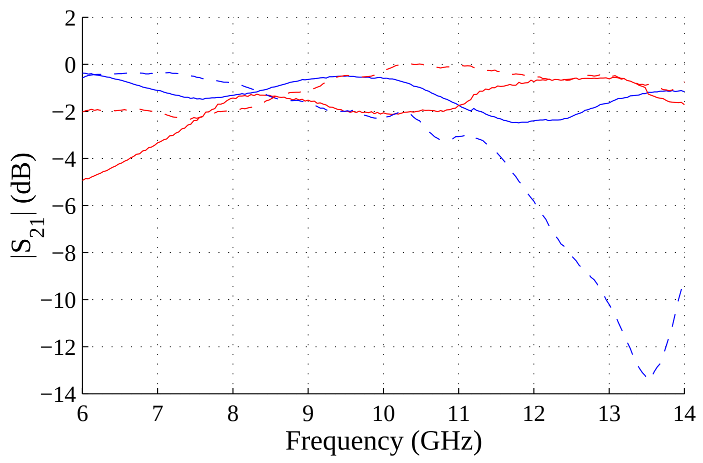


Figure 3.3. Measured IL (blue) and ISO (red) for the series (–) and shunt (- -) switch configurations.

diode SPDT switch in: bond wire inductance (L), distance between diode and Y-junction (d) and transmission line impedance (Z_0). *MatLab* code based on transmission line equations and lumped element models examines the design tradeoffs for these parameters. Previous measurements of the series and shunt configured PIN diode switches support the use of a lumped element model for predicting IL, ISO and RL. The bond wire inductance is probably the most fabrication sensitive parameter in the switch circuit at 10 GHz. Using a thin microstrip line on a dielectric substrate with $\epsilon_r = 10.5$ as the model for the bond wire above a ground plane, the inductance is estimated to be 2 nH/mm. Depending on the arc of the bond wire and where it connects to the microstrip line, this number can vary by as much as 50%.

Two *MatLab* simulations of SPDT switch are performed varying the transmission line impedance and length with the bond wire inductance held at 1 nH and 1.5 nH. The IL, ISO and RL is calculated for the switches paying close attention to the sensitivity of switch performance as the parameters change. The lower limit for the inductance at 1 nH is set by the size of the PIN diode. The PIN diode sits on a 0.5 mm diameter brass rod via, limiting the gap in the microstrip line to 1 mm.

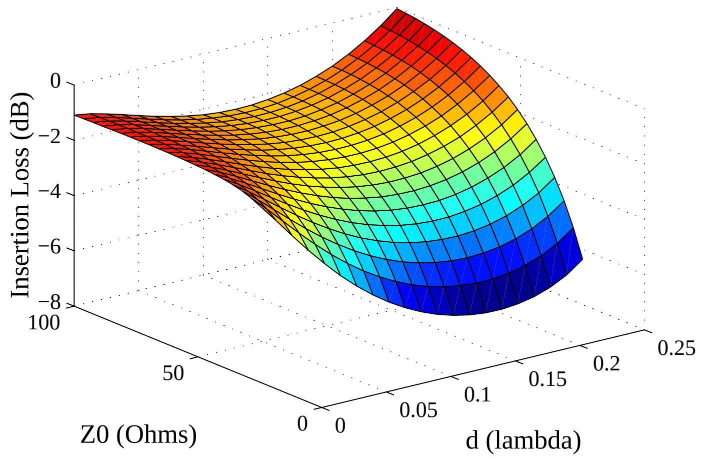
From the simulations, the IL and ISO are relatively insensitive to changes transmission line impedance and length, as for the 1 nH case in Figures 3.4. In general, the IL and ISO improves with the increase of transmission line impedance. The RL is the only parameter that significantly sensitive to design parameters, Figure 3.5. Overall, 1 nH has better performance than the 1.5 nH, however caution is warranted since the transmission line equations only give general trends of performance and neglect many high order effects.

To verify the tradeoff analysis, five different circuits are built to test the effects of bond wire inductance and transmission line impedance, Figure 3.6. The first two circuits ("a" and "b") have the PIN diode in a T-network with 1.5 nH and 1 nH bond wire respectively. The second set of circuits ("c" and "d") were built to examine the loss of exponential of $\lambda/4$ and $\lambda/2$ tapers from 50 Ω microstrip lines to 70 Ω microstrip line. The final circuit ("e") combines the $\lambda/4$ taper with 1 nH bond wire to measure the total effect of a high transmission line impedance with PIN diode.

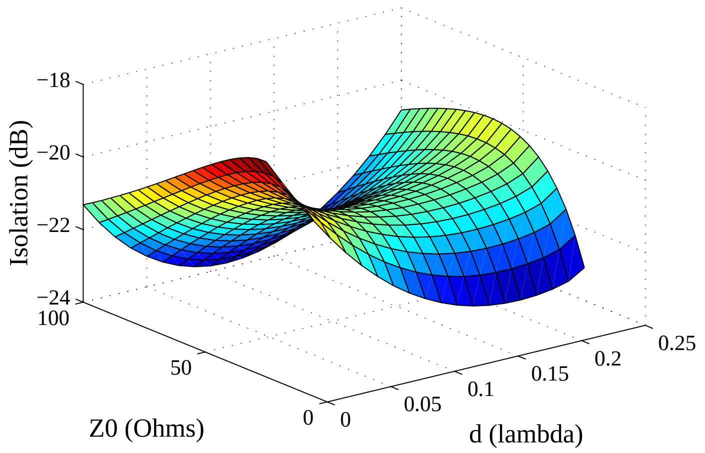
The ISO of the switch needs to be sufficient to prevent oscillations in the ring formed by two SPDT switches in the final array element design. In other words the sum of the sum of the loop gain and losses around the loop must be less than the twice the ISO:

$$G_{LNA,dB} + G_{PA,dB} - 2\,IL_{dB} - 2\,ISO_{dB} = 0, \tag{3.1}$$

where $G_{LNA,dB}$ and $G_{PA,dB}$ are gains of the LNA and PA respectively. Equation 3.1 that implies a ISO of 15 dB is needed for a switch with 30 dB loop gain (i.e. LNA = 16 dB and PA = 14 dB)to prevent oscillations assuming IL=0. Circuit "b" with 1 nH bond wire inductors has the best performance with IL=0.75 dB and ISO=20.3 dB, which is close to or better than the simulated values of 2 dB IL and 20.2 dB ISO 3.7. Circuit "b" also exhibits low loss as seen in Figure 3.8. The loss from the exponential impedance tapers does dominates over any benefit the higher impedance transmission



(a)



(b)

Figure 3.4. Simulated IL (a) and ISO (b)for T-network SPDT switch using MA4GP032 PIN diode with 1 nH bond wires

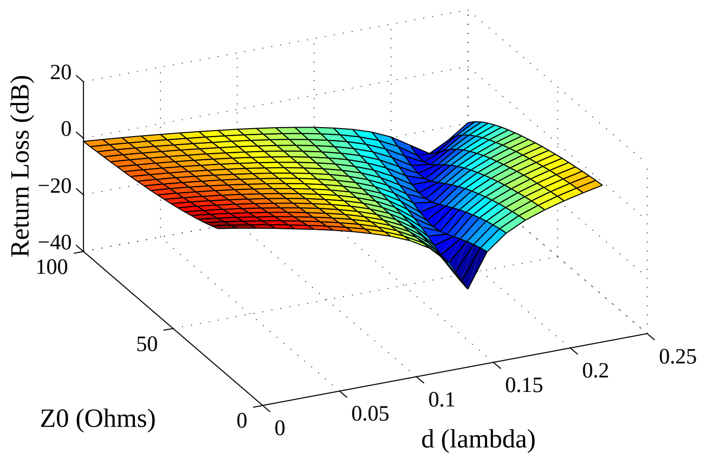


Figure 3.5. RL for T-network SPDT switch using MA4GP032 PIN diode with 1 nH bond wires

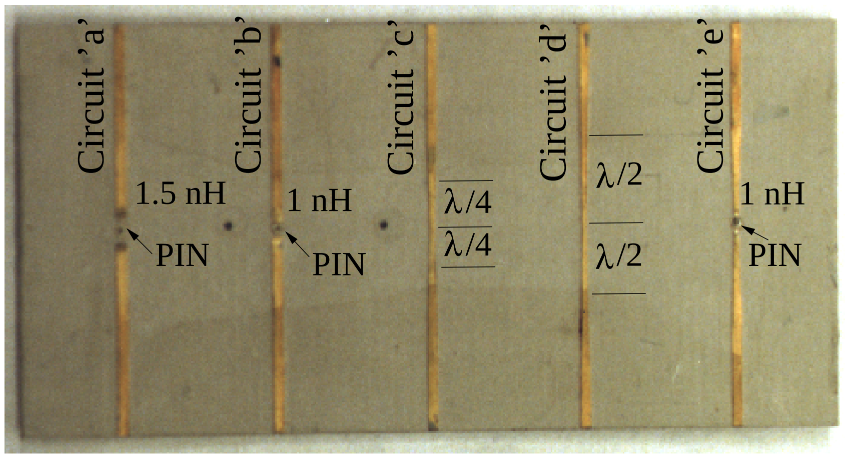


Figure 3.6. Five test circuits use to examine the performance of PIN diode in different configurations. Circuit "a" is T-network PIN diode with 1.5 nH bond wires and 50 Ω transmission line. Circuit "b" is T-network PIN diode with 1 nH bond wires and 50 Ω transmission line. Circuit "c" is $\lambda/4$ impedance taper from 50 Ω to 70 Ω transmission line. Circuit "d" is $\lambda/2$ impedance taper from 50 Ω to 70 Ω transmission line. Circuit "e" is T-network PIN diode with 1 nH bond wires and 70 Ω transmission line.

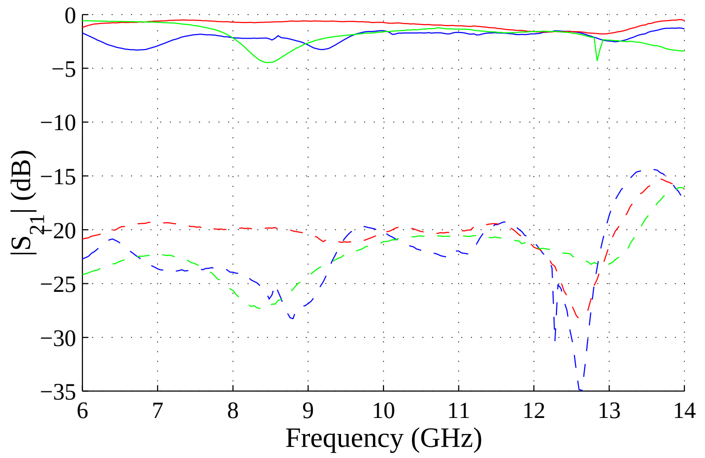


Figure 3.7. The measured IL (–) and ISO (- -) for circuit “a” (blue), circuit “b” (red) and circuit “e” (green).

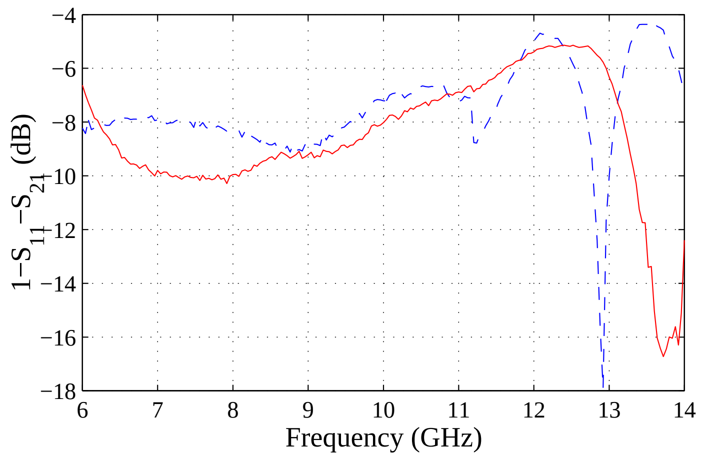


Figure 3.8. Measured loss in circuit “b” for the switch “on” (red) and the switch “off” (blue).

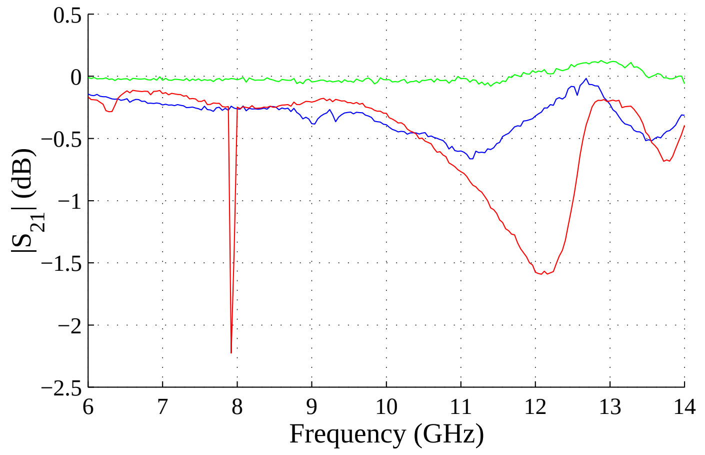


Figure 3.9. Measured IL for circuit "c" (blue), circuit "d" (red) and circuit "e" (green).

lines gives in IL and ISO, Figure 3.9.

3.1.2 High Pass Filter In designing the switch circuit, it is important to make all components very compact in order to have a unit element of the array smaller than a free-space squared. We therefore use lumped elements to design the components. A HPF, Figure 3.10, made from a chip capacitor and bond wire inductors replaces the series capacitor used in the previous SPDT switch design, Figure 2.10. The addition of the series inductor is necessary to bridge the gap in the microstrip line. To minimize the effects of this inductor, the gap distance

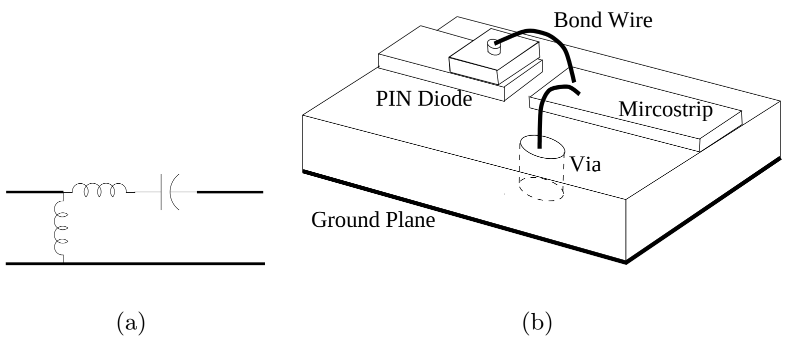


Figure 3.10. Circuit schematic (a) for HPF constructed from two bond wires and chip capacitor (b).

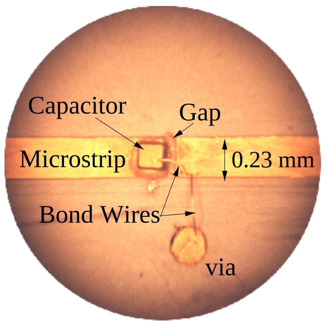


Figure 3.11. Picture of HPF with 2 nH shunt bond wire, 0.25 nH series bond wire and 1 pF chip capacitor.

is adjusted such that the inductor is resonant with the capacitor at 10 GHz. The smallest gap that can be milled on the available PC-board prototyping machine of 5 mils (.0127 mm) creates a 0.25 nH inductor which is resonant with a 1 pF chip capacitor at 10 GHz, Figure 3.11. An addition of a 2 nH shunt inductor increase the attenuation of lower frequency components (below 1 GHz) that are generated by a nanosecond switching currents. The HPF has good performance of 0.1 dB loss at 10 GHz and 20 dB rejection for frequencies below 1 GHz, Figure 3.12.

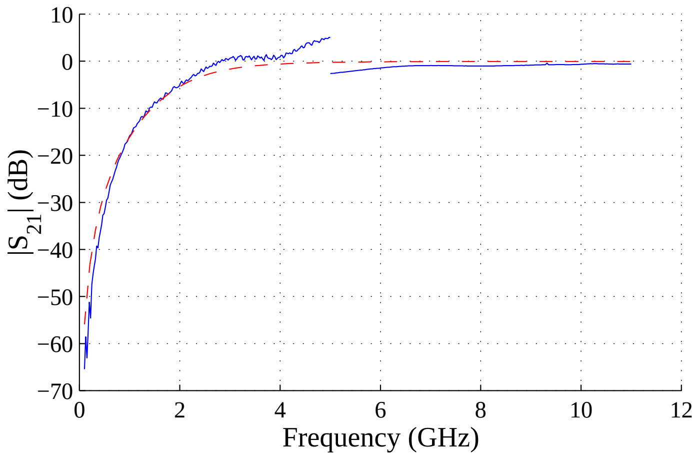


Figure 3.12. Measured (blue –) and simulated lumped element model (red - -) performance of HPF. Measurements above 5 GHz are calibrated and uncalibrated below 5 GHz.

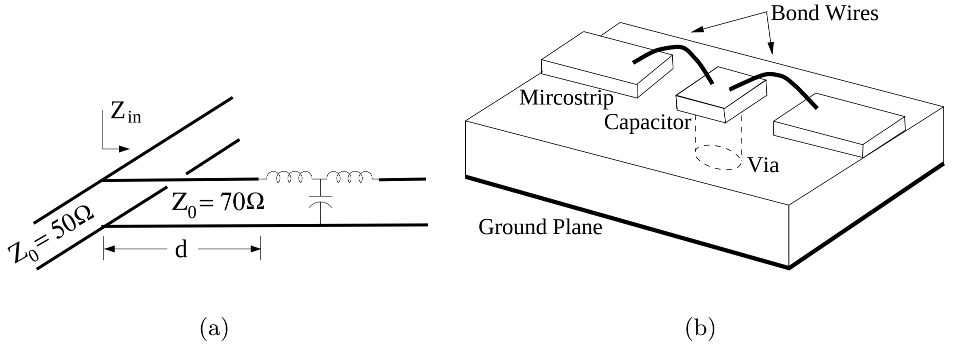


Figure 3.13. Lumped element schematic for LPF constructed from chip capacitor and two bond wires used in biasing the chip PIN diodes (a) and implementation of circuit (b). The distance ‘d’ that LPF is placed from the 50 Ω is adjusted it minimize reflections at the T-junction.

3.1.3 Low Pass Filter An inductor capacitor LPF, Figure 3.13, replaces the shunt capacitor that suppresses the microwave signal from the diode bias circuits. The 3rd order LPF provides suppression of the 10 GHz carrier while allowing the sub-gigahertz switching currents to pass.

Lumped element models are used to explore effectiveness of three different LPF designs based on three commercially available capacitor values of 1 pF, 3 pF and 5 pF. The LPF with inductance of 0.85 nH and capacitance of 3 pF provides the best compromise between rejection of RF carrier and filter bandwidth. The 3 pF capacitor is also insensitivity to reflection distance as shown in Figure 3.14 with and optimal distance of 58^o. A mistake in recording the optimal values for simulations done with *Puff* and *MatLab* led to designs with reflection line distances of 90^o and 58^o. This mistake turns out to be advantages with the SPDT switch is latter optimized. Testing both circuits, Figure 3.15, shows that the 90^o reflection line has the best performance, Figure 3.16. The 90^o bias line has a loss of 0.107 dB on the 50 Ω microstrip line at the T-junction and 34 dB suppression of a 10 GHz signal through the LPF.

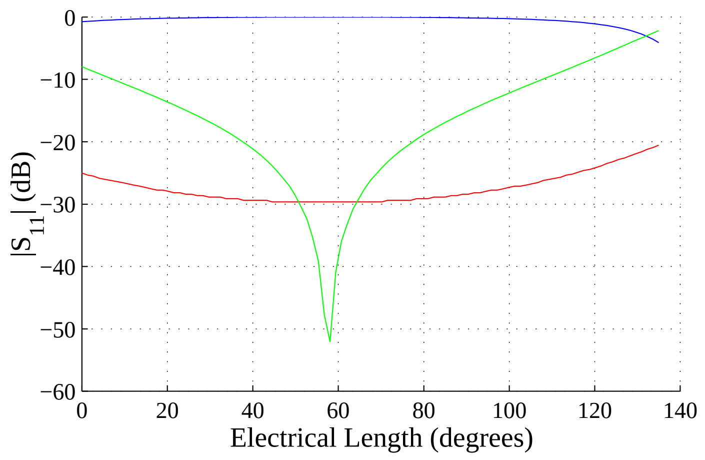


Figure 3.14. Simulated IL (blue), RL (green) for 50 Ω microstrip line and LPF isolation (green) as a function of the distance between the LPF and microstrip.

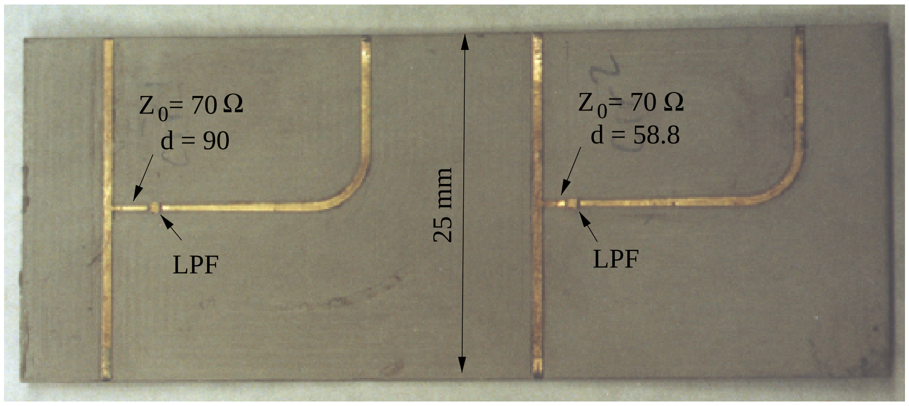


Figure 3.15. Picture showing the two different LPF designs, one with a reflection distance of 90^o and the other with a reflection distance of 58.8^o.

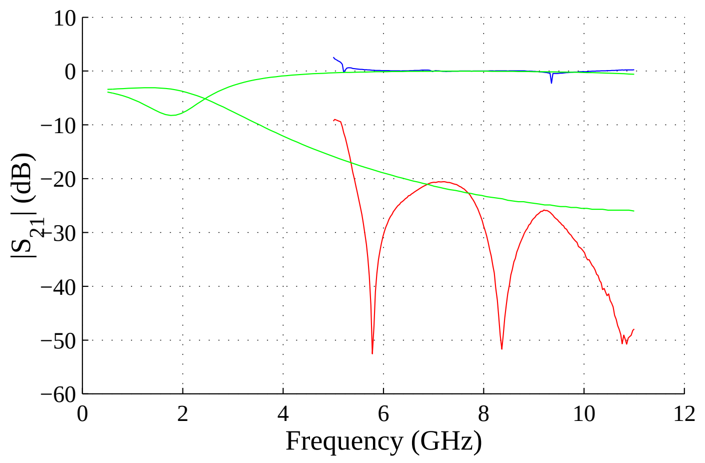


Figure 3.16. Measured IL (blue) and LPF isolation (red) for the LPF design with reflection distances of 90^o. The loss in the $50\,\Omega$ microstrip line is in (blue) and the LPF rejection is in (red). The simulated performance is in (green).

3.1.4 Simulation of Transient Response of SPDT Switch

A circuit model developed in *SPICE* by Orcad, based on physical measurements of the various components, and combinations of the components is used to model the transient response of the switch. Measured DC voltages and currents for various interconnections of the PIN diodes and MESFETs, Figure 3.17, are used to develop the *SPICE* component models. The photodiode model is based on manufacture specifications. The photodiode model itself uses a voltage-controlled current source to simulate the generation of carriers by optical signals, Figure 3.18.

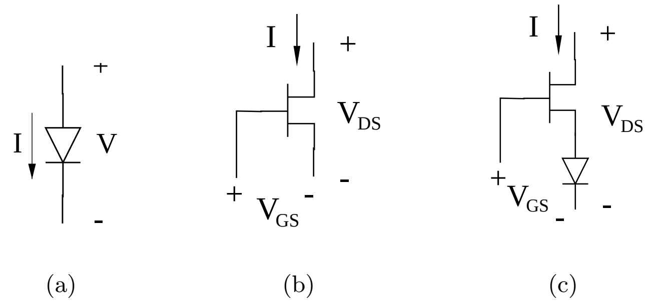


Figure 3.17. Three different test configurations used to measured DC voltages and currents for developing a *SPICE* model.

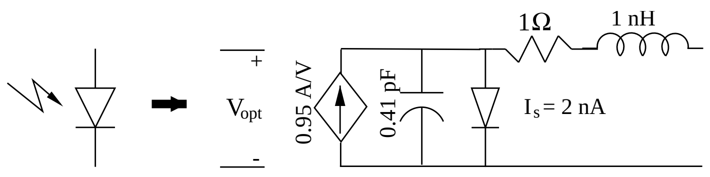


Figure 3.18. A lumped element model for photodiode using a voltage controlled currents source to simulate the generation of carriers as a result of an optical signal.

The rise and fall times for the back-to-back SPDT switch are dependent on the optical power as shown in Figure 3.19. The fastest rise time is 2.4 ns with 9 mW of optical power, limited by the RC time constant for the current limiting resistor and the sum of the PIN diode's capacitances. For optical power smaller than 9 mW, the RC time constant of the MESFET gate capacitance and the "on" resistance of the photodiode dominates. At 1 μW of optical power, the rise time for the switch is 1800 ns. The current limiting resistor increases the rise time as compared to the fall time. It turns out that the photodiode's capacitance effect on the rise and fall times is minimal, and a photodiode with four times the active area and twice the capacitance for a would be acceptable and would on the other hand greatly improve coupling of optical power.

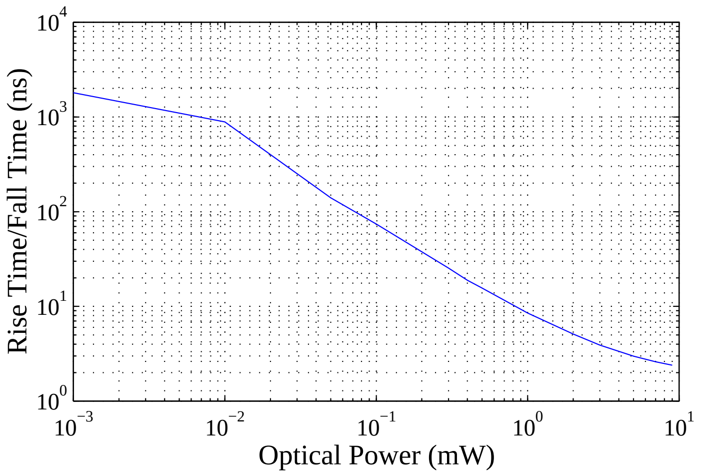


Figure 3.19. *SPICE* simulation of Rise time and fall time for two back-to-back SPDT switches used in the unit cell.

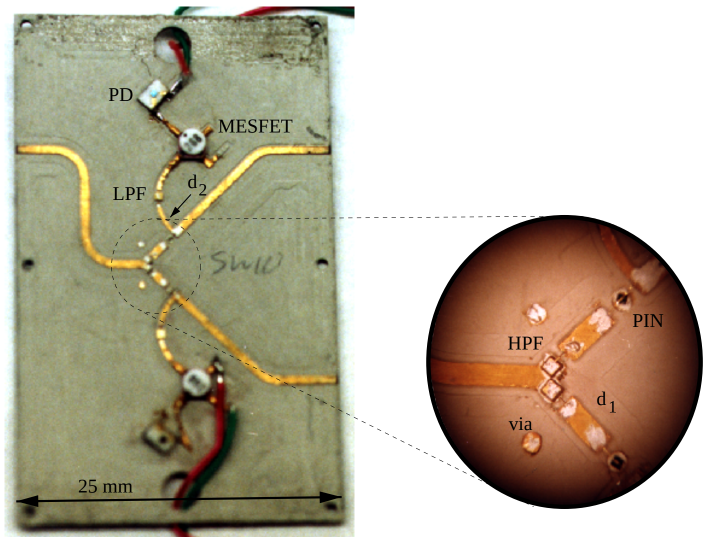
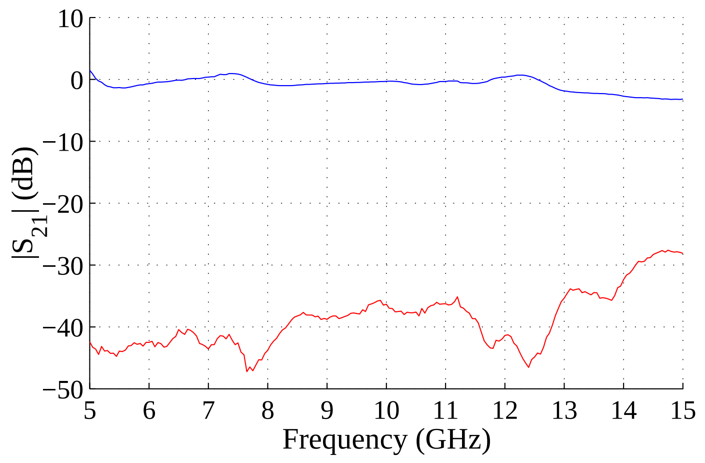


Figure 3.20. Picture of SPDT switch using MA4GP032 chip PIN diodes with HPFs and LPFs.

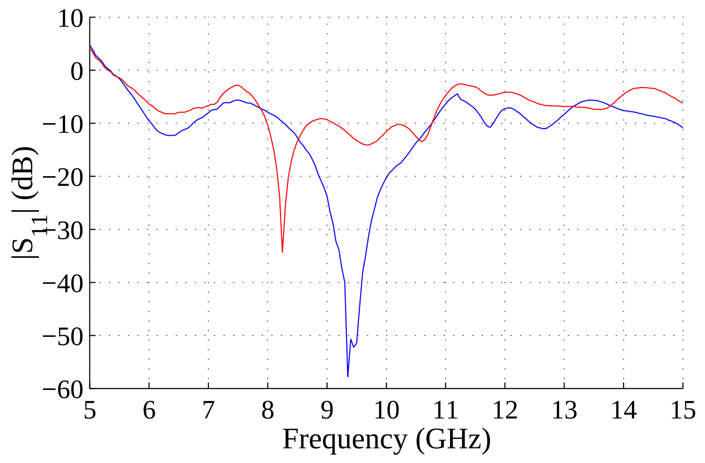
3.2 Optimized SPDT Switch Design

Two different approaches are taken in optimizing the design of the SPDT switch from its optimized components. The first design uses *PUFF* to optimize the PIN diode, LPF and HPF designs together, while the second design fixes the filter designs and optimizes only the PIN distance from the Y-junctions, Figure 3.20. The first design optimizes the PIN diode distance (d1) 54^o and LPF distance (d2) to 67^o. The second design optimizes the PIN diode distance (d1) to 52^o. The first circuit has a IL of 1.05 dB (1.55 dB simulated), ISO of 28 dB (-21.8 dB simulated) and 2:1 VSWR BW of 4.64 GHz from 6.48 GHz to 11.12 GHz. The second circuit out performed the previous with IL of 0.315 (1.81 simulated), ISO of 36.33 dB (20.6 dB) and 2:1 VSWR BW of 2.49 GHz from 8.27 dB to 10.76 dB, Figure 3.21.

When measuring the SPDT switches, the HPFs were placed in backwards and shorted the PIN diodes with their shunt inductors. These inductors were removed for the measurements shown in Figure 3.21. It ensure that HPF orientation does not affect the SPDT switch performance two Y-junction circuit are built with the HPFs place forwards and backwards as shown in Figure 3.22. The IL and ISO

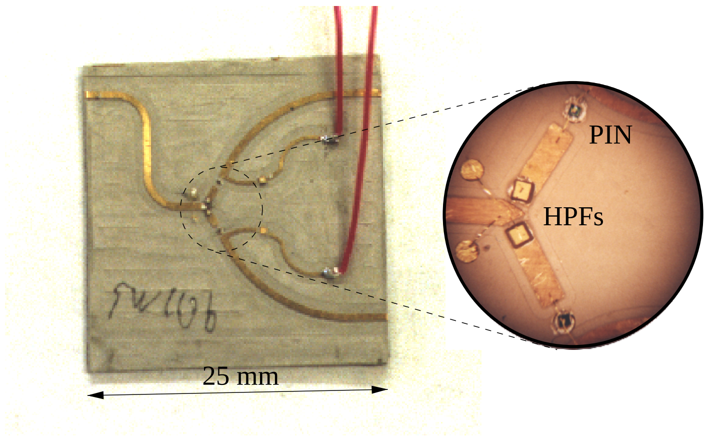


(a)

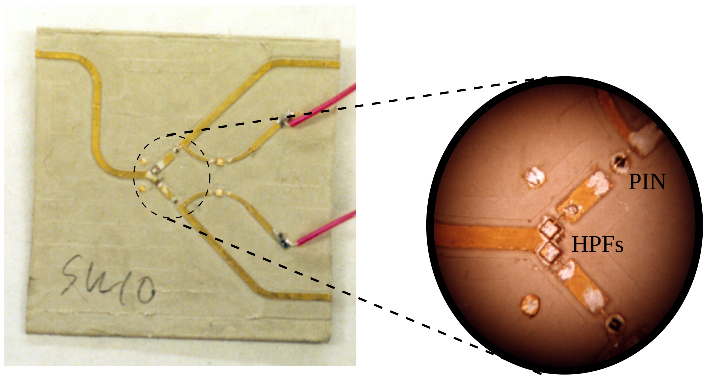


(b)

Figure 3.21. The measure RL (a), IL (blue b) and ISO (red b) for the second optimized SPDT switch



(a)



(b)

Figure 3.22. The Y-junction circuits used to measure the effect of the HPFs placed forwards (a) and backwards (b).

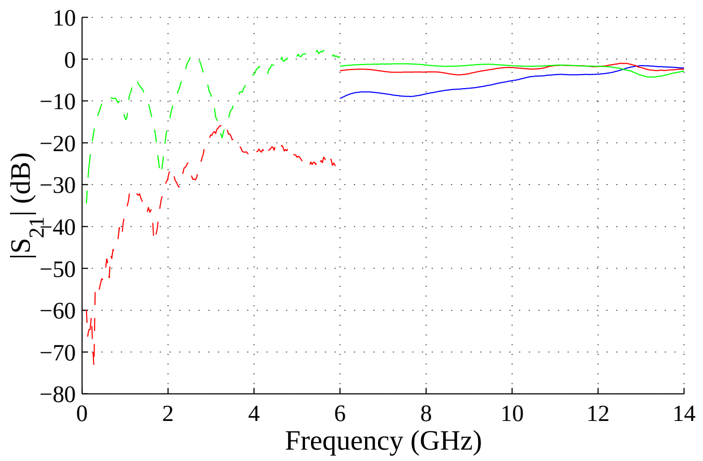
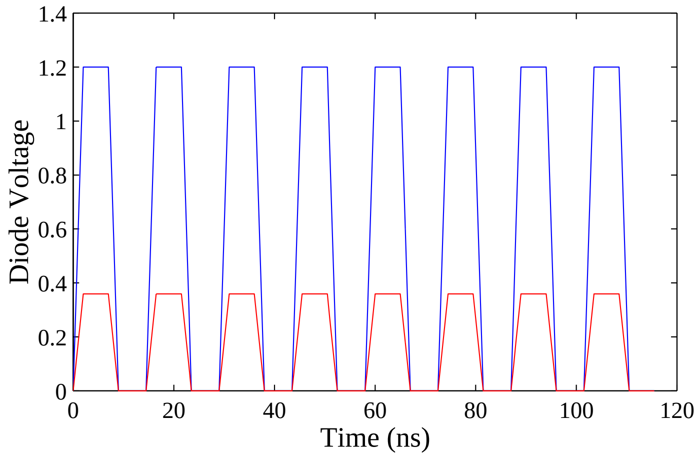


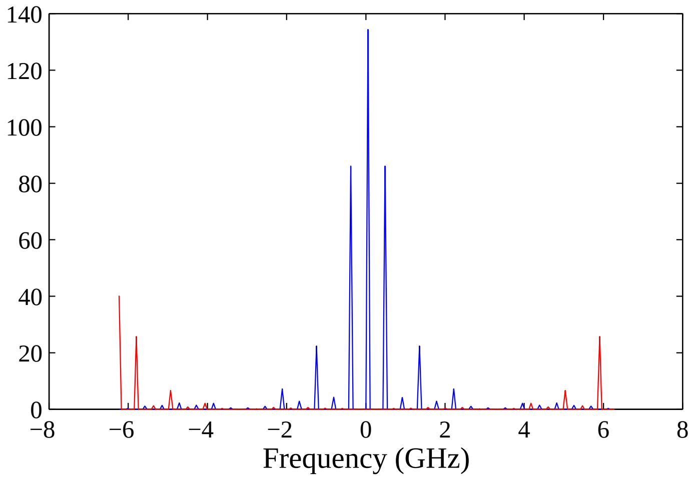
Figure 3.23. Measured calibrated (–) and uncalibrated (- -) IL for the Y-junction with the HPFs placed correctly and the shunt inductors remove one by one. The measurements with two inductors are in (blue), one inductor are in (red) and no inductors are in (green).

measurements for the two circuits show that both circuits perform the same within a few tenths of dB in IL and few dB in ISO.

3.2.1 Estimation of Switching Current Isolation The SPDT switch has its best performance when the shunt inductors in the HPFs are removed. However, by removing the shunt inductors, will the HPFs affectivily isolate the to sides of the SPDT switch at fast switch rates? From the *SPICE* model, the fastest rise time for the switch was found to be 2.4 ns. A voltage waveform for the PIN diodes with 2 ns rise and fall times, magnitude of 1.2 V and pulse period of 5 ns is constructed in *MatLab*i as shown in Figure 3.24(a). This waveform represents the fastest reasonable switch rate for the SPDT switch. The voltage waveform is filtered through the measured HPF's frequency response shown in Figure 3.23. The Fourier transform of the voltage waveform shows that most of its spectral energy is below 500 MHz (Figure 3.24(b)), where the HPF (without shunt inductors) attenuates about 10 dB. The voltage waveform seen by the other PIN diode after passing through the two HPFs is about a tenth of its orginal magnitude. Since the PIN diodes turn "on" at 1 V, the HPFs with shunt inductors do isolate the two sides of



(a)



(b)

Figure 3.24. Simulation of the time (a) and frequency (b) suppression of the switching currents generated by a 2 ns rise time/fall time 5 ns pulse period optical pulse by the HPFs. The bias switching currents on one side of the SPDT switch (blue) are filtered through HPFs using the data from Figure 3.23 with no inductors to simulates the induced switching currents on the other side of the SPDT switch (red).

Table 3.1. Measured input and output power for back-to-back SPDT switch showing maximum input power for the SPDT switch is at least 17 dBm.

Input Power (dBm)	"on" Path Output Power (dBm)	"off" Path Output Power (dBm)	Difference Difference
-5	-30.3	-65.3	-34.9
0	-25.6	-61.6	36.0
5	-20.7	-57.1	36.4
10	-15.6	-51.8	36.2
16.7	-9.1	-44.4	35.31

the SPDT switch at its fastest switch rate.

3.2.2 Maximum Input Power for SPDT Switch

The maximum microwave input power for the SPDT switch is when the PIN diodes rectify enough of the microwave signal and self-bias. When this occurs, the IL and RL will increases and the difference between the IL and ISO will decrease. The SPDT switch is tested with RF input power ranging from -5 dBm to 17 dBm while measuring the outputwave forms for the "on" and "off" paths. The output power for the two paths increases linearly with input power for the full range, while maintaining a 36 dB difference between the IL and ISO, Table 3.1.

3.3 Switch Rate Measurements

For testing the response time of the SPDT switches in the unit cell, a special back-to-back SPDT switch circuit is made, Figure 3.25, in which the antenna and MMICs are replaced with microstrip ports for connection to a microwave source and oscilloscope. The oscilloscope connected to one of the ports will record the switch turning "on" and "off". A bias tee connected between the oscilloscope and SPDT prevents the oscilloscope from loading the PIN diode bias circuitry. The optical fibers connect the SPDT switch to the optical test setup which provides complementary optical pulses with variable width.

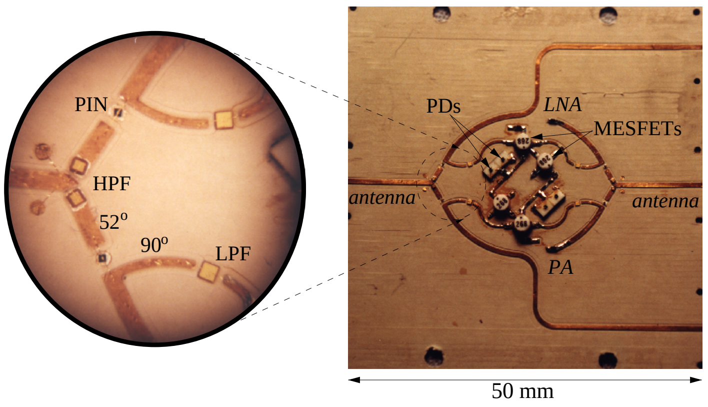


Figure 3.25. Picture of back-to-back SPDT switch layout used to test the SPDT switches transient response. The LNA, PA and antennas have been replaced with microstrip ports for testing.

3.3.1 Optical Test Setup The optical control test setup is responsible for providing variable width optical pulses with fast rise times and fall times to the photodiodes in the back-to-back SPDT switch. The power in the optical pulses also needs to vary from less than microWatt to several milliWatts. The pulses are created by splitting the power from an Ortel 10 mW 1300 nm laser diode into two electro-optical (EO) intensity modulators. A polarization-maintaining evanescent mode coupler is used to split the optical signal from the laser while still maintaining the correct polarization orientation for the EO modulators. The Ortel laser diode is fiber pigtailed with single mode fiber iwhich scrambles the output polarization of the light from the laser. By stressing the single mode fiber, the light can be forced into one polarization mode. The EO modulators are chosen to have a 3 GHz bandwidth allowing them to generate pulses with of 0.1 ns rise and fall times.

$$\tau_{rise} = \frac{1.8}{2\pi\, BW} \tag{3.2}$$

To achieve complete extinction of the optical pulse, the EO modulators are driven at V_π (4.5 V) by Veritech VMCM2.5MD-422 EO modulator drivers that provide the necessary gain needed to drive the 50 Ω input port of the EO modulator.

Two different setups are developed to stimulate the SPDT switch with

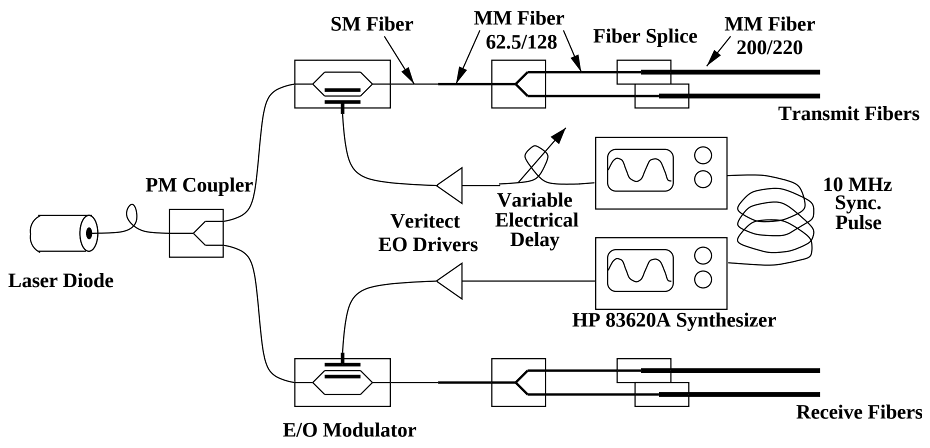


Figure 3.26. Illustration of the fast optical test setup able to generate 25 ns wide complementary optical pulses with 2 ns rise time/fall time.

optical pulses of different rise and fall times, Figure 3.26,3.27. The first setup is for high optical power, where the SPDT is expected to react in tens of nanoseconds, Figure 3.26. Two HP 83620A synthesizers are synchronized together through their 10 MHz synchronization ports. The cable connecting the two HP 83620A sweeper is long enough to delay the pulses such that the 20 MHz sine waves produced by the synthesizer are 180^o out of phase. The output sine waves drive the Veritech EO modulators creating optical pulses with 25 ns pulse width and 1.5 ns rise and fall times, Figure 3.28. An additional 2 ns variable electrical delay line is used to tune

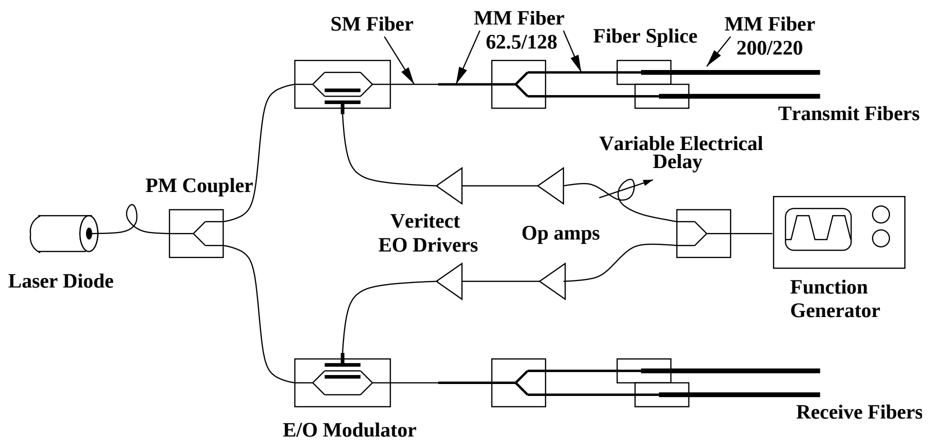
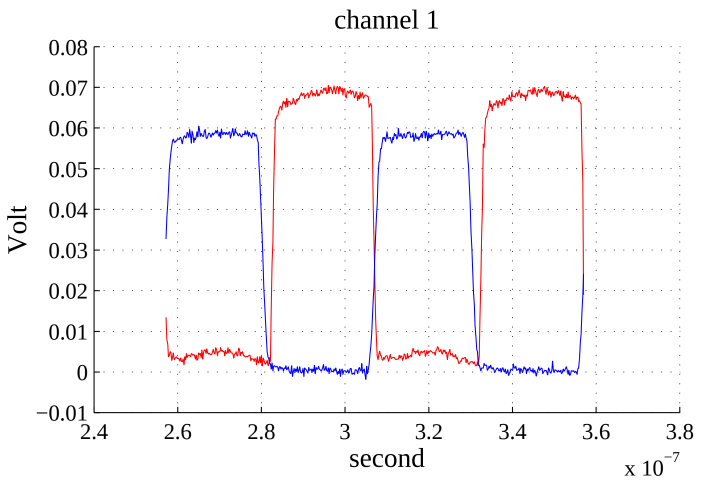
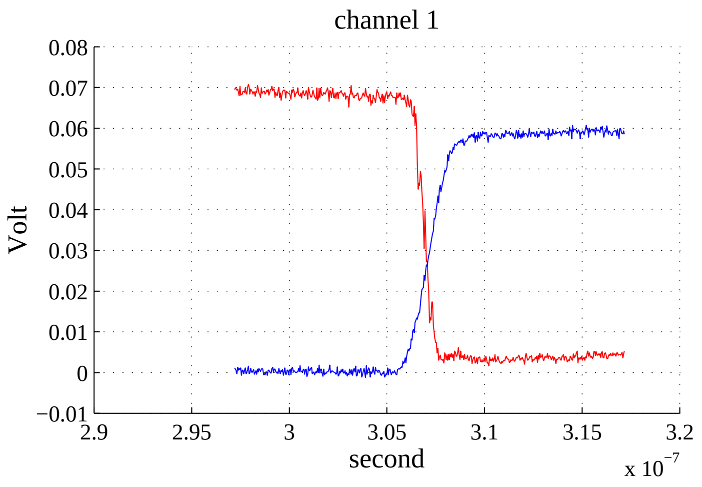


Figure 3.27. Illustration of the slow optical test setup able to generate 625 ns to 6250 ns wide complementary optical pulses with 25 ns rise time/fall time.



(a)



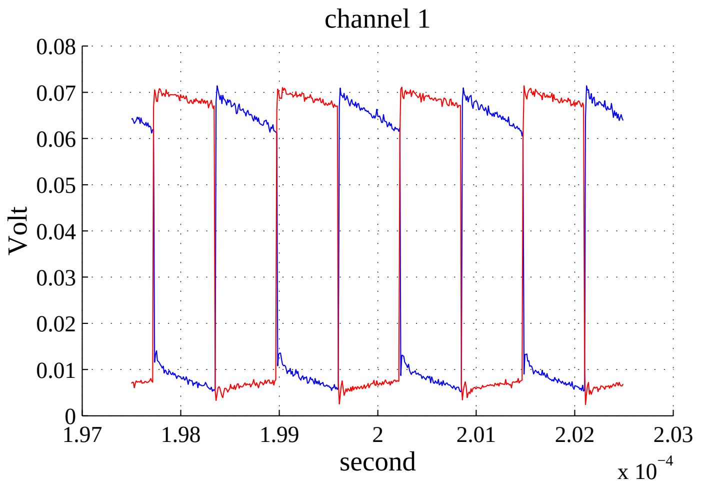
(b)

Figure 3.28. The complementary optical pulses (blue and red) for the fast optical setup (a). The rise time and fall times are shown in (b).

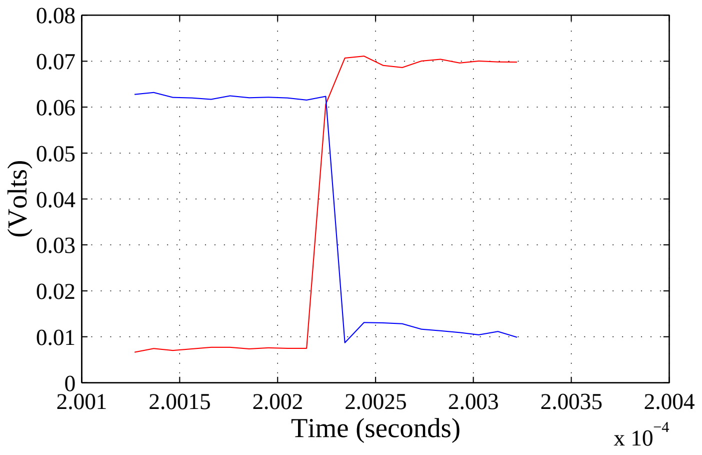
the setup for variations in cables and fiber lengths.

The second setup is for low optical power pulses, which requires long pulse widths, Figure 3.27. This setup uses a HP 8116A function generator to generate a square wave that in turn drives two operational-amplifier in inverting and non-inverting unity gain configurations. Again the Veritech EO drivers are used to amplifier the operational-amplifier outputs for the EOs. This system is able to generate pulses varying in width from 625 ns to 6250 ns with a constant rise time and fall time of 25 ns, Figure 3.29.

The output of the EO modulators is split and fed to the SPDT switch. Ideally, the output fibers of the multi-mode splitter would be glued directly to the photodiodes to minimize the coupling loss as illustrated, Figures 3.30. Optical fibers have a protective jacket that prevents water from penetrating the micro-fractures in the glass which would cause the fiber to break. To glue the multi-mode fiber to the photodiodes, the outer jacket is stripped off, making the fiber extremely delicate. It proved impossible to develop a support mount for the fibers that allows access to the microwave circuits and prevents the fibers from shearing. Instead, the output fibers from the multi-mode splitter is glued to 200 μm fibers that as suspended over the photodiodes, as shown in Figure 3.31. Due to the uncertainties in the relative position of the photodiode with respect to the fiber, the fiber is move away from the photo diode so that the natural diffraction from the end of the fiber produces a 470 μm spot on the photodiode. To prevent cross-talk between fibers, baffles made from aluminum are positioned on the PDs. A 5 mil piece of FR4 is glued to the bottom surface of the baffle to prevent it from shorting out the PDs. Assuming the multi-mode fiber uniformly illuminates the spot, the loss in the coupling is 15.7 dB. Ths is too high for a practical application, and in Chapter 7 are given suggestions as to how this loss can be rduced in future designs.



(a)



(b)

Figure 3.29. The complementary optical pulses (blue and red) for the slow optical setup (a). The rise time and fall times are shown in (b).

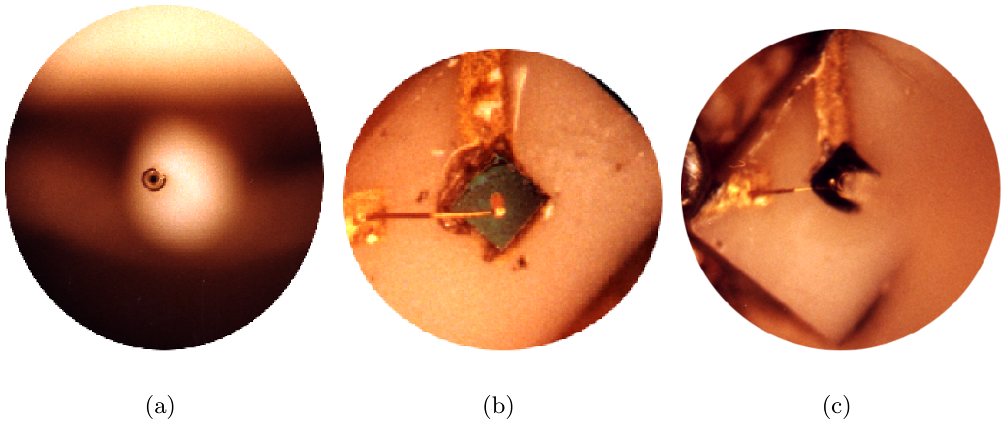


Figure 3.30. Photographs showing the 62.5 μm core to multimode fiber (a), the 80 μm active area of the photodiode (b) and the fiber and photodiode glued together (c).

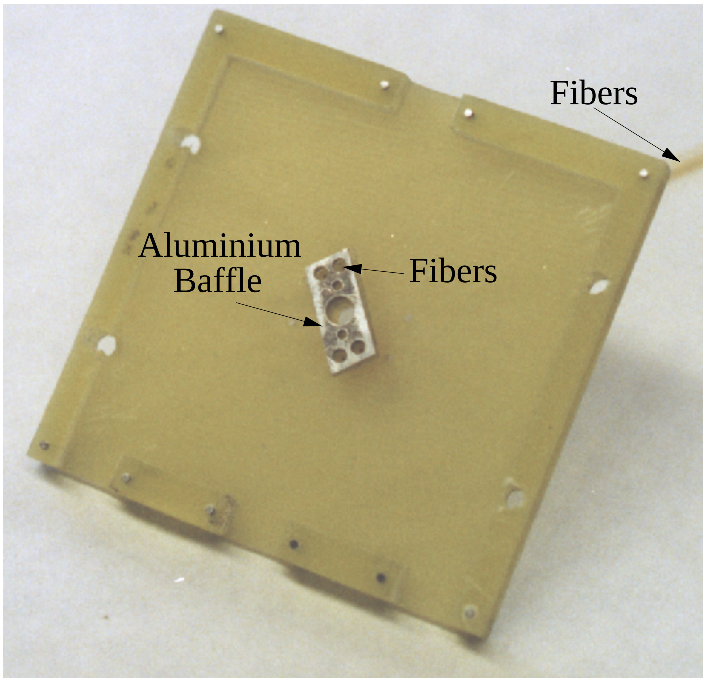


Figure 3.31. Photograph of the optical mount for the back-to-back SPDT switch . The mount contains an aluminum baffle to prevent cross-talk between photodiodes.

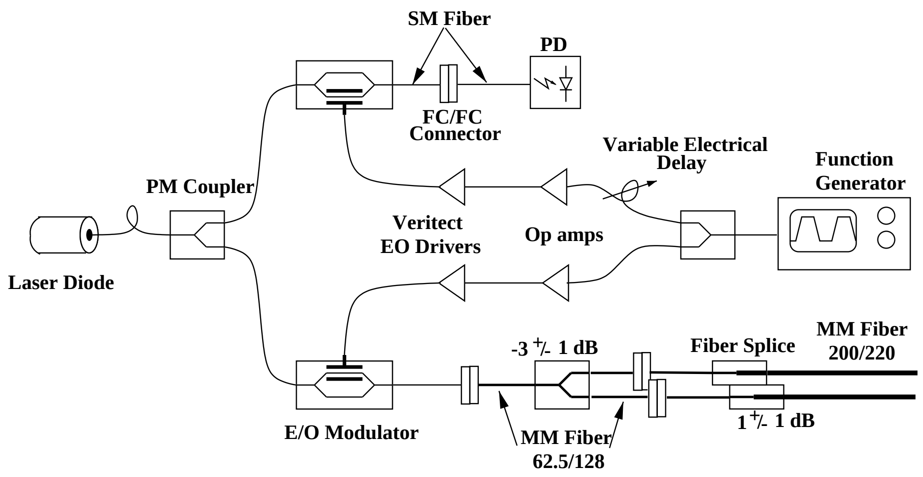


Figure 3.32. Illustration of the calibration configuration (upper arm) and the testing configuration (lower arm) of the slow optical test setup

3.3.2 Calibration of the Optical Setup It is important to estimate the uncertainties in the amount of optical power delivered to the photodiode as a result of changes in the optical setup and mechanical variations. The two optical setups are calibrated at the output of the EO modulator with fiber pigtailed Fermionics photodiodes (upper arm in Figure 3.32). After calibration, the FC/FC optical connector is broken and the multimode fiber setup is connected (lower arm in Figure 3.32). By adding the multimode fiber setup, the calibration has an additional 4 dB of loss and 2 dB of uncertainty.

Most of the loss and uncertainty is due to the free-space coupling from the fiber to the photodiode active area. To calculate the uncertainty in the coupling, each mechanical connection between the optical mount, RF mount and substrate is described as a stochastic process. The mechanical structures are discussed in more detail in section 4.1. For example, the holes used to align the optical mount to the RF mount have an uncertainty in their position of $\pm 5.7\,\mu$m. Table 3.2 describes the probability density functions (PDFs) for the various components in the system.

The uncertainty of the photodiode placement is based on measurements of 21 photodiodes. Figure 3.33 shows the PDF for alignment of various components

Table 3.2. Properties of probability density functions used to calculated the misalignment of the photodiodes with the optical fibers.

Description	pdf	mean	variance
Hole Location Optical Mount	Uniform [-10:10]	0	33
Hole Location RF Mount	Uniform [-10:10]	0	33
Alignment Pin Bending	Uniform [-100:100]	0	3333
Fiber Location in Optical Mount Hole	Uniform [-30:30]	0	300
Photodiode Placement on Mount	Uniform [-160:160]	0	8533

in the mechanical system. The resulting uncertainty in the mechanical alignment is $\pm 93\,\mu$m, mostly due to the uncertainty in the placement of the chip photodiode on its mount. The alignment between the photodiode mount and the optical fiber has an uncertainty of $\pm 71\,\mu$m. The overall loss in the free space coupling of fiber to photodiode is 15.7 dB ± 1 dB assuming that the spot created by the optical fiber has uniform power density. The total loss and uncertainty in the optical test setup is 19.7 dB $^{+1.7\,dB}_{-2.7\,dB}$.

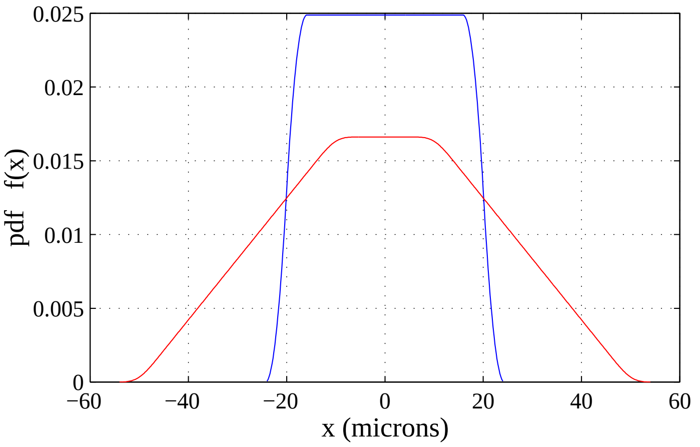


Figure 3.33. The probability density functions for the alignment for the optical fibers to the photodiode mount (blue) and the photodiodes (red).

Table 3.3: The measured rise time and fall time of the back-to-back SPDT switch.

Optical Power (μW)	Rise Time (ns)	Fall Time (ns)
15	555	469
9.3	772	785
4.95	1180	1180
2.75	2290	1600

3.3.3 Measured Switch Rate Performance The second optical setup is used to measure the SPDT switch, since the 20 dB of optical loss limits the fastest response time for the SPDT to about 500 ns. At these slow rise times and fall times, the bias tee filters the output response, and the rise and fall times cannot be measured with the oscilloscope. The bias tee is removed and the *SPICE* model is adjusted to account for the loading of the oscilloscope. The SPDT switch is measured at four different optical powers with rise and fall times shown in Table 3.3. Figure 3.34 plots the measured rise time and fall time for the different optical powers with simulated rise times and fall times and uncertainty bounds. Interestingly enough, the SPDT switch operates better than expected. These suggests that expected loss for the optical control is less than stated earlier, probably due to the assumption that

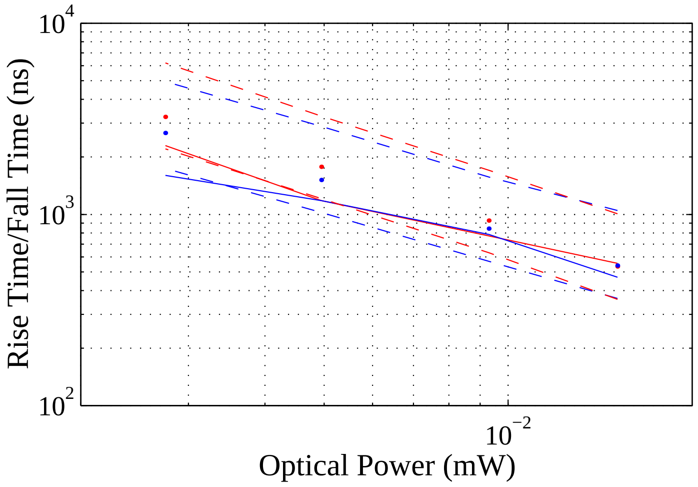


Figure 3.34. The measured rise time (red ·) and fall time (blue ·) for the back-to-back SPDT switch. The measured performance is consistent with the simulated performance (–) and within the simulated error bounds (- -).

the spot made by the optical fiber has uniform power density and not Gaussian. The minimum amount of optical power needed maintain the state of the switch is $0.25\,\mu$W.

CHAPTER 4

10 GHZ UNIT CELL AND LENS ARRAY

4.1 Unit Cell Construction

In the previous chapter the design and implementation of an optically controlled SPDT switch was presented. When this switch is inserted into an element of an active array, a number of manufacturing issues need to be solved, including electrical and mechanical ones.

The mechanical designs needs to fulfill certain requirements, while being limited to tools and materials available in the laboratory. The soft low-loss Roger's Duroid substrates used to make the microwave circuits need to be reinforced with a stiffer material to maintain their shape while being mounted into different test setups. The material used to reinforce the substrate should be a dielectric to avoid interference with the microwave circuits. If at all possible, the support structure should also protect the delicate gold bond wires on the surface of the array. The mechanical system needs to align the optical fibers to the photodiodes while allowing access to the microwave components. Precise placement of the microwave components is also critical. For convenience, all the pieces are milled on the PCB prototyping milling machine mentioned in section 2.3. The milling machine can mill FR4 (PCB material), aluminum and brass, as well as the softer Teflon-based substrates. Soft metals, such as copper gum, the milling tools, while low temperature plastics like Nylon, Delrin and polycarbonate melt from the high-speed milling tool friction.

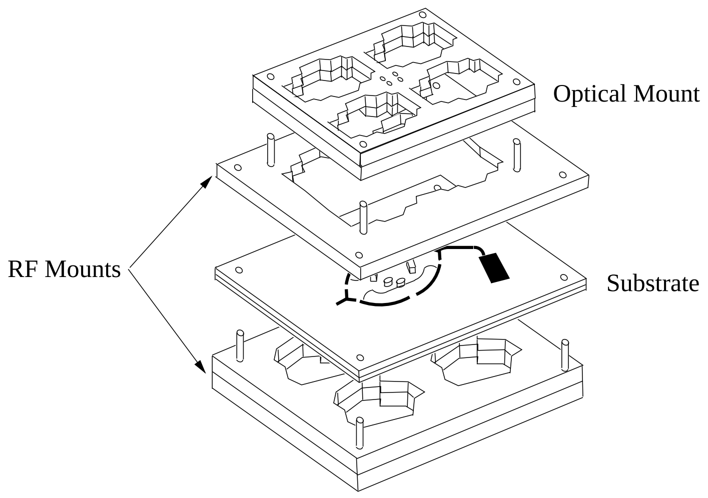


Figure 4.1. Illustration of the assembly of the FR4 mounting structures with the substrates.

Since the milling machine is limited to cutting, drilling and gouging (milling only part way through the material) on materials up to 128 mils thick, the overall unit cell is composed of a set of laminate pieces pined together, Figure 4.1. It is composed of four layers of FR4 and two layers of Duroid. The two layers of Duroid substrate contain the front and back side microwave circuits (i.e. antennas, SPDT switch, ...). The bottom three layers of the FR4 RF mount, support the soft Duroid material by sandwiching it between them. This mount also provides two sets of alignment pins used to align the substrate with the optical mount. The top two layers of FR4, referred to as the optical mount, align the optical fibers to the photodiodes while protecting the gold bond wires. Since the optical fibers are free space coupled to the photodiodes, the optical mount may be removed from the RF mount for access to the microwave circuits.

4.1.1 Unit Cell Layout The layout of the unit cell is based on the layout of the array, such that the performance and stability of the unit cell is as close as possible to that of the array. There are two sides to the unit cell, the active component side and passive component side.

On the active component side, effort is made to have about 30 dB of isolation between any two paths that can carry the microwave signal. Since the maximum gain of the MMICs is about 16 dB, this should prevent any free space oscillations. The electrical and mechanical components are restricted to be outside an area of 20 mm by 20 mm, (section 2.5.3), around the each patch antenna. The bias lines are placed on the passive side of the unit cell to prevent oscillations due to coupling through the MMICs power connections. A tilt of 6.3^o to the patch antennas allows for tighter packing of the unit cells into the array.

For good optical fiber coupling the photodiodes need to be accurately position on the substrate. The holes are cut into the active side substrate the exactly fit the ceramic mounts for a pair of photodiodes. The pressure fit of the photodiodes into these holes allows for the precise positioning. The ground plane for the active side substrate is milled away just around the holes to prevent the photodiodes from shorting. A via is provided to make the ground connection for the photodiode.

The PIN diodes and LPF capacitors are mounted with silver epoxy on 0.5 mm brass via posts in the substrate providing a good ground connection and accurate placement of the devices. In an effort to minimize the length of the bond wires to the MMICs and capacitors, they are placed on 0.38 mm (15 mil) brass platforms. Longer pass-through vias are made to connect the active and passive sides of the unit cell. The ground planes are removed from around the pass-through via holes on the both the active and passive substrates to prevent the vias from shorting to the ground plane. One set of pass through vias connects the -2.3 V bias line to the photodiodes while another set connects the 5 V bias line to the current limiting resistor and MMICs. The passive side patch antenna is also connected the active side with a pass-through via. The brass components are pictured on a penny to demonstrate their size, Figure 4.2. All of the brass pieces are made to a accuracy of 1-2 mils from bras rods or brass strips cut on the milling machine. Then the brass

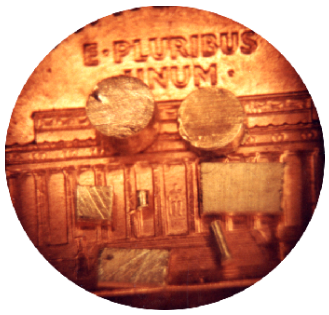


Figure 4.2. Photograph showing the size of the milled brass pieces on a penny used in the construction of the unit cell.

pieces are filed to size as illustrates in Figure 4.3.

The passive side of the unit cell contains the bias line, antenna, and the lensing delay lines. Even though the MMICs have 100 pF capacitors filtering the 5 V supply voltage, additional capacitor pads are placed approximately every $\lambda/2$ along the bias-line encase bias-line oscillations occur. Since the MMICs and SPDT switch are supplied from the same bias line and their vias are placed quite close together, additional capacitor pads are place next to the switch via to suppress the switching currents from polluting the 5 V supply to the MMICs.

Besides the FR4 and substrate pieces, special aluminum plates are made to allow gold bonding on both sides of the unit cell. The hole pattern cut into them allows from maximum heating the material for gold bonding while not damaging the circuits. Figure 4.4 pictures the aluminum plate for the lens array.

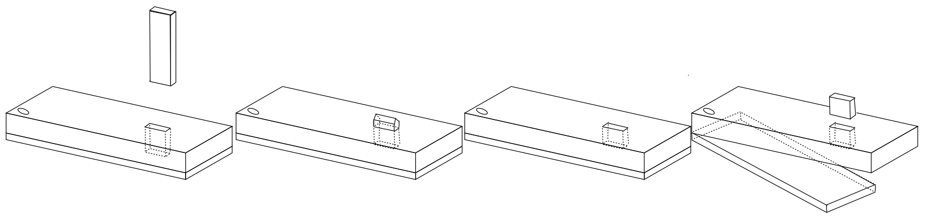


Figure 4.3. An illustration showing how to fabricate the small brass pieces used in the unit cell.

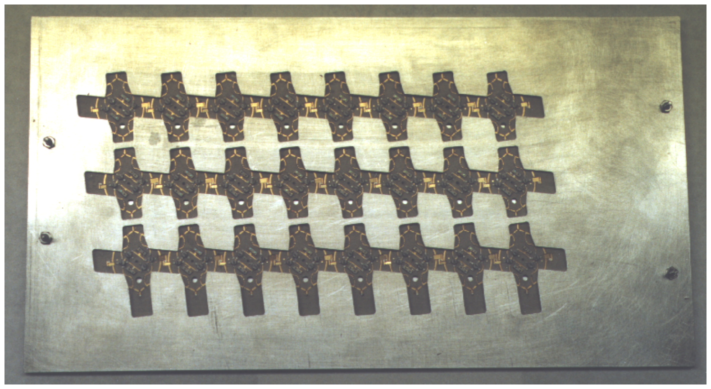
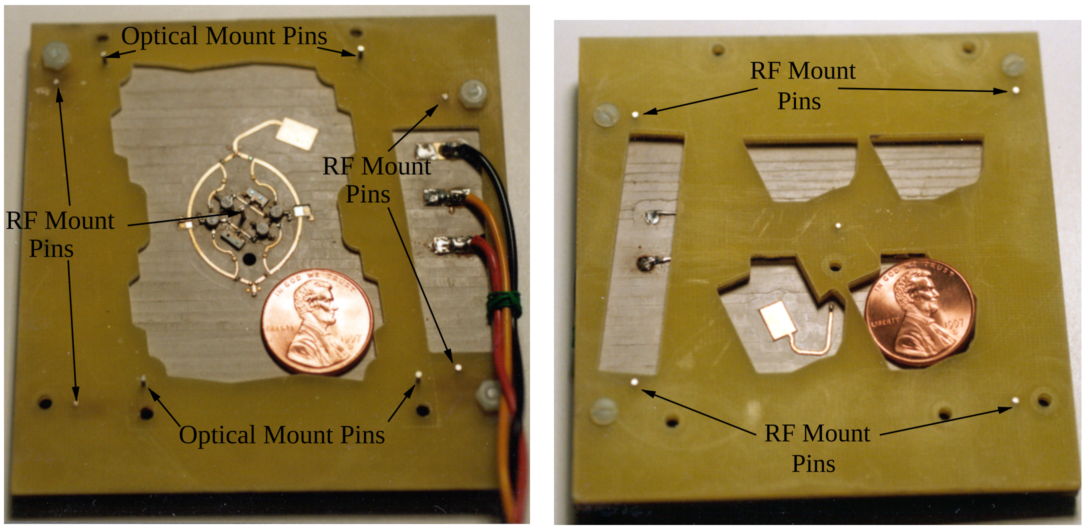


Figure 4.4. Photograph showing an aluminum plate that allows wire bonding on both side of the lens array.

4.1.2 RF Mount Design The main purpose for the RF mount is to provide support for the Duroid substrate and align the substrate to the optical mount. 0.8 mm brass pins in the lower FR4 RF mount pieces match 0.8 mm holes in the substrate and 1 mm holes in the upper FR4 RF mount pieces. The 1 mm holes have tapered ends allowing easier mating of the two pieces. An additional four pins set in the second RF mount piece match four 1 mm holes in the optical mount, Figure 4.5(a). One pin fixed in the bottom RF mount piece passed through all FR4 and substrate layers to align with the optical mount. It is in the center of the photodiode cluster allowing maximum absolute alignment between the fibers and photodiodes. 20 mm by 200 mm apertures cut into the lower RF mount pieces allow the passive side antennas radiate through the RF mount, Figure 4.5(b).

4.1.3 Optical Mount Design The optical mount is the most complicated piece made with the highest tolerances. It provides protection to the microwave circuits while maximizing the aperture (20 mm by 20 mm minimum) through which the active side patch antenna radiates. It also contains optical fibers and baffling to prevent cross-talk between the photodiodes.

The optical mount consists of three laminated layers of FR4. The top two layers contain the windows for the antennas as well as the 300 μm holes that hold



(a) (b)

Figure 4.5. Photograph showing the active side (a) and passive side (b) of the RF mount for the unit cell with their alignment pins.

the 240 μm optical fibers, Figure 4.6(a). The third layer contains the baffles which are 1.5 mm in diameter and 1.27 mm deep that fit over each photodiode, Figure 4.6(b). A post that rests on the substrate surface maintains the baffle height above the photodiodes. The fibers are fitted into the optical mount such that they rest 1.12 mm above the photodiodes and protrude 0.15 mm into the baffle. The optical fibers are secured in the 0.3 mm holes with quartz. The optical fibers protrude into the baffles to prevent the quartz wax from flowing over their ends in the gluing process.

4.2 Unit Cell Testing

The test setup for the unit cell is the same as the one used for testing the 8 GHz unit cell, Figure 2.36. There is 13.5 dB of loss in the unit cell due to aperture loss (4.7 dB x 2), IL (0.32 dB x 2), via loss (1 dB) and radiation efficiency (2.4 dB). The gain of the MMICs overcome the loss to give a positive gain for the unit

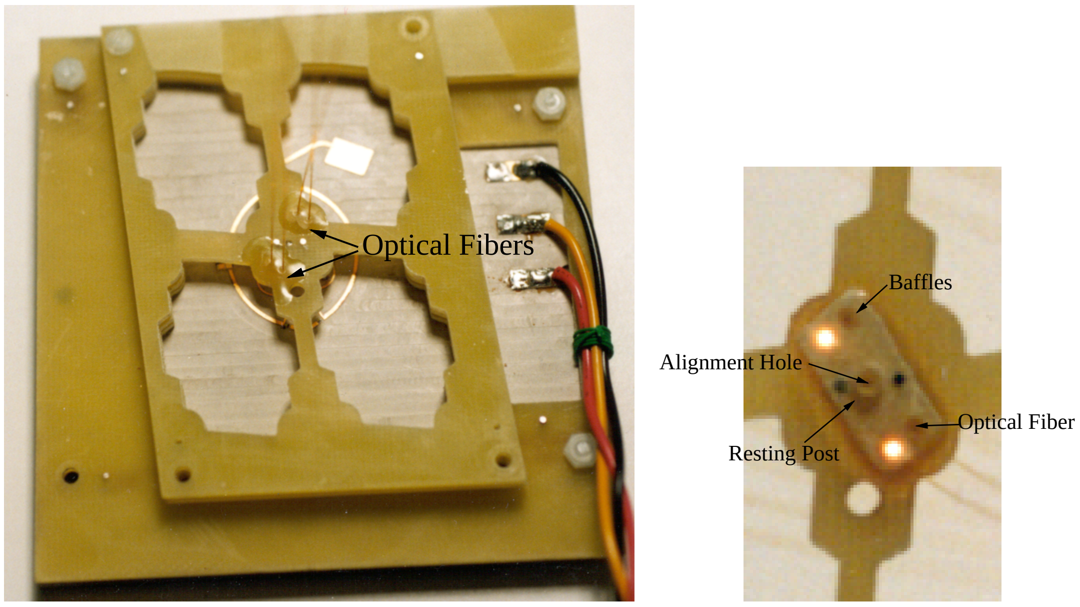


Figure 4.6. Optical mount for the unit cell placed on the FR mount (a) showing the optical fibers and antenna apertures. The back side of the mount (b) shows the FR4 baffle and substrate resting post.

cell in a “through” measurement, Figure 4.7. The unit cell exhibits at least 30 dB of isolation in anomalous modes of operation such as: transmitting in receive mode and receiving in transmit mode. Measurement performed with the unit cell powered “off” all exhibit 30 dB of isolation demonstrating that isolation measurement is limited to edge diffraction around the unit cell.

4.3 Array Design and Testing

The lens array, Figure 4.8, is designed as a cylindrical lens with 3 row of 8 unit cells. The slant to the rows allows for tighter packing of the unit cells with $3/4\,\lambda$ spacing in the focusing dimension and $1\,\lambda$ in the non-focusing dimension. The focal distance to diameter ratio is 1 and the points of perfect focus, section 5.1, are at $\pm 45^o$. Figure 4.9 shows the simulated radiation patterns for the lens with 0^o and

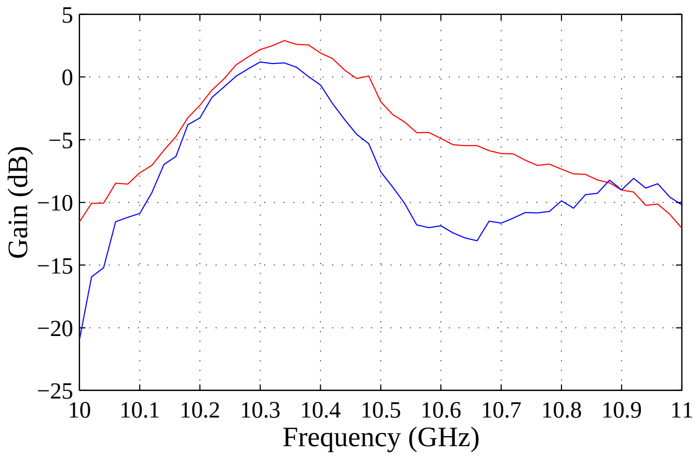
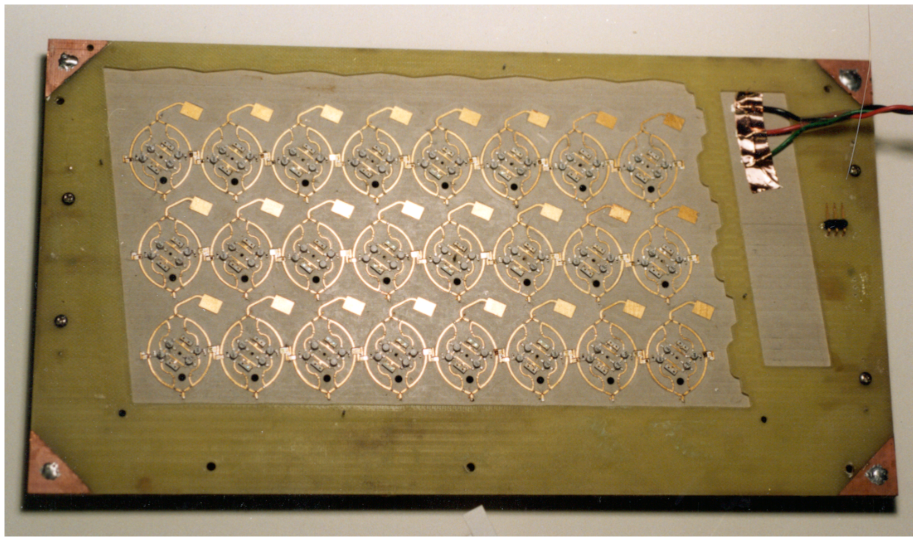


Figure 4.7. Measured gain of the unit cell in the transmit (blue) and receive mode (red) calibrated to a "through" measurement of an aperture of $3/4\,\lambda$ by $1\,\lambda$.

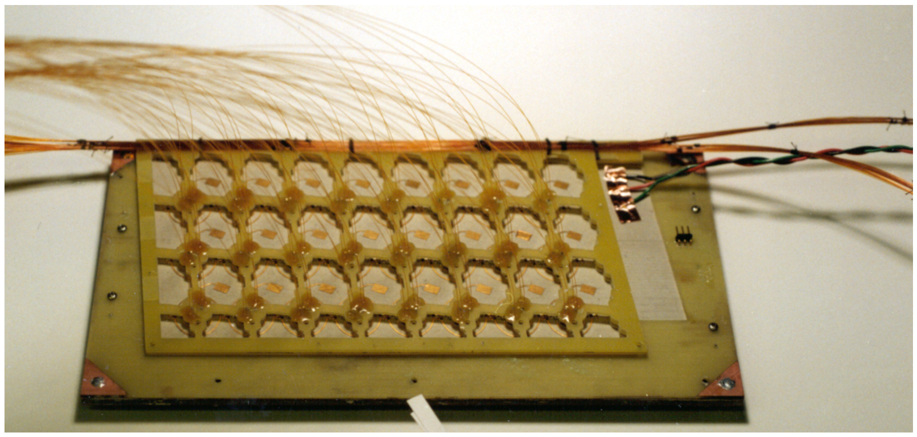
30^o off optical axis feeds. The half power beam widths (HPBW) for the 0^o feed is 8.5^o in focusing dimension and 17^o in the non-focusing dimension. The HPBW for the 30^o feed increases to 12.5^o in the focusing dimension and 16^o in the non-focusing dimension. The simulations are done using code describe in section 5.2. Due to the $1\,\lambda$ spacing, grating lobes appear at $\pm 90^o$ in the non-focusing dimension. The $3/4\,\lambda$ spacing causes grating lobes to appear when the main beam is steered off axis, Figure 4.9(b). Figure 4.10 shows the radiation pattern for a cut in the focusing dimension.

4.3.1 Optical Control Problems with the Array Measurements

The transmit and receive fibers from each unit cell are bundled together using quartz wax. Each bundle is polished with $10\,\mu$m to $0.5\,\mu$m polishing paper. particles size. The bundles are aligned to a 70 mW 1480 nm laser diode that is collimated using a 8 mm focal distance lens, Figure 4.11. Since the beam from the laser diode is elliptical in shape, it is impossible to get a perfectly collimated beam using a single lens. It is important to collimate the laser beam since the optical fibers are limited to an acceptance angle of 25.4^o or less, determined by their numerical aperture of 0.22. The optical fiber bundles were polished by hand using a specially made jig to hold them perpendicular to the polishing surface. Unfortunately, due to human



(a)



(b)

Figure 4.8: The active side for the lens array (a) with optical mount (b).

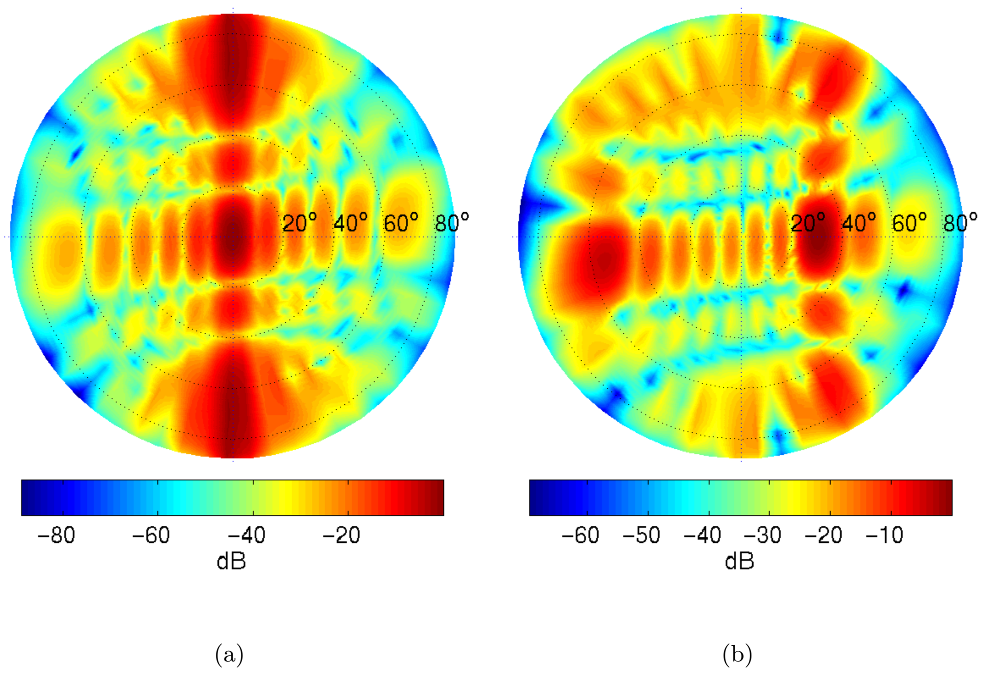


Figure 4.9. Simulated normalized radiation patterns for the lens array with 0^o off axis feed (a) and 30^o off axis feed (b). The focusing dimension is the horizontal axis and the non-focusing dimension is the vertical axis.

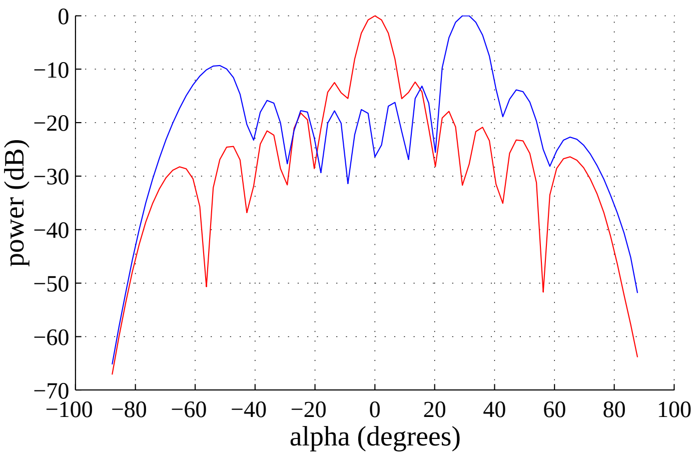


Figure 4.10. Simulated normalize radiation patterns for the 0^o off axis feed (red) and the 30^o off axis feed (blue) in the focusing dimension.

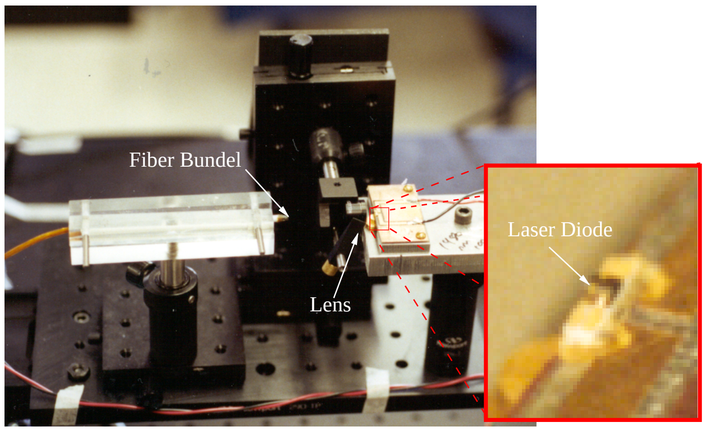


Figure 4.11. Photograph showing the coupling of the 1480 nm laser diode to the optical fiber bundle.

imperfection, the outside fibers in the bundle have rounded edges, from the wobbling of the jig.

The rounded edges and the reduced optical power at the edge of the bundle reduced the optical power coupled into the outside fibers preventing proper biasing for their unit cells. Figures 4.12 and 4.13 show the best achieved bias voltages for the PIN diodes for the array in transmit and receive modes respectively. The PIN diodes are "on" for voltages above 1 V. Voltages just below the threshold voltage present a low impedance to the switch which may not be enough to prevent oscillation.

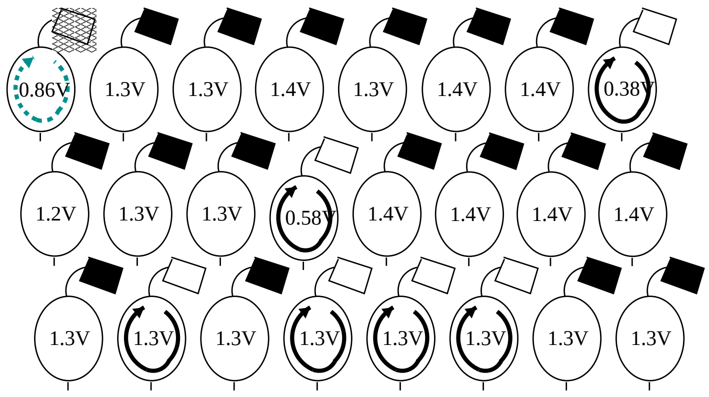


Figure 4.12. Picture showing the oscillating and questionable unit cells for the lens in transmit mode.

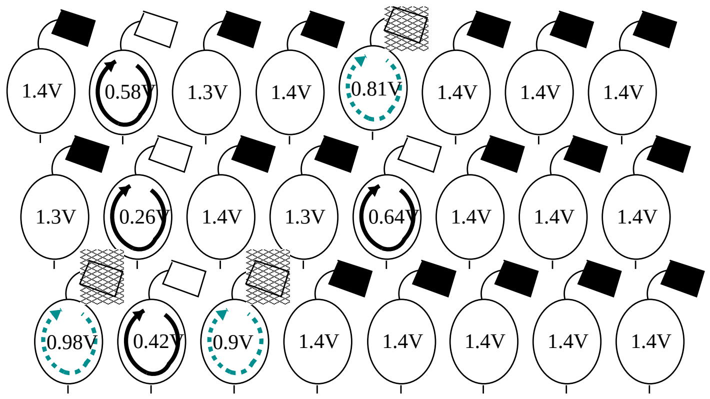


Figure 4.13. Picture showing the oscillating and questionable unit cells for the lens in receive mode.

When neither set of PIN diodes are properly biased "on", then both paths in the SPDT switch are "on" forming a ring oscillator with the LNA and PA providing the gain. With the 30 dB isolation between the unit cells, each unit cell oscillated at its own frequency, around 8 GHz and can be seen on the spectrum analyzer. By counting the number of oscillating peaks, the number of non-functioning unit cell can be determined. By moving the laser spot across the fiber bundle, it is possible to properly bias once oscillating unit cells at the expense of another unit cell oscillating. It is this test methodology that proved each unit cell is built properly and the optical control is the reason for the array oscillation.

4.3.2 Array Pattern Measurements At the time of the development of the optically controlled lens array, an anechoic chamber was not available with a rotating stage that could accommodate the optical feed setup. Therefore preliminary radiation pattern measurements are taken with a rotating stage mounted to an optical table, Figure 4.14. A horn antenna mounted on a rotating arm behind the lens array provides the feed for the array. Patterns are taken for the feed at 0^o and 30^o off optical axis. The 30^o feed radiation pattern does not steer the main beam through the fiber bundles shown in, Figure 4.14. A spectrum analyzer is used to measure the transmit and receive power while filtering out the radiated power

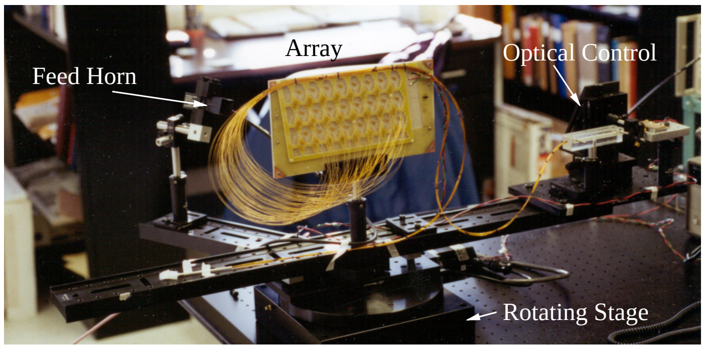
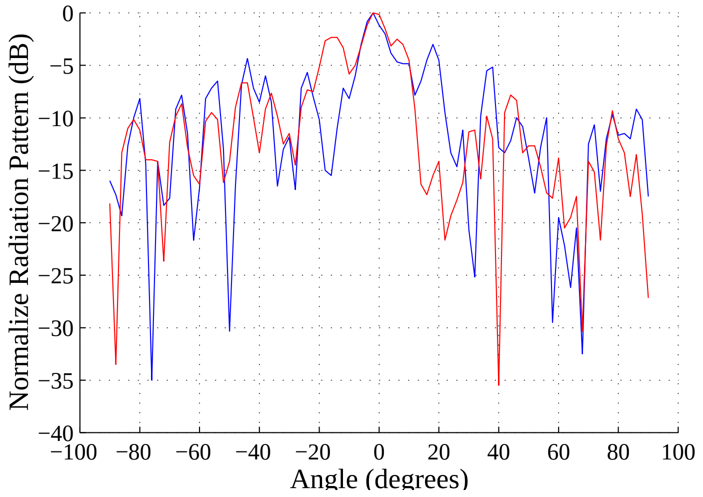


Figure 4.14. Photograph showing the setup used to measure the radiation patterns for the lens array.

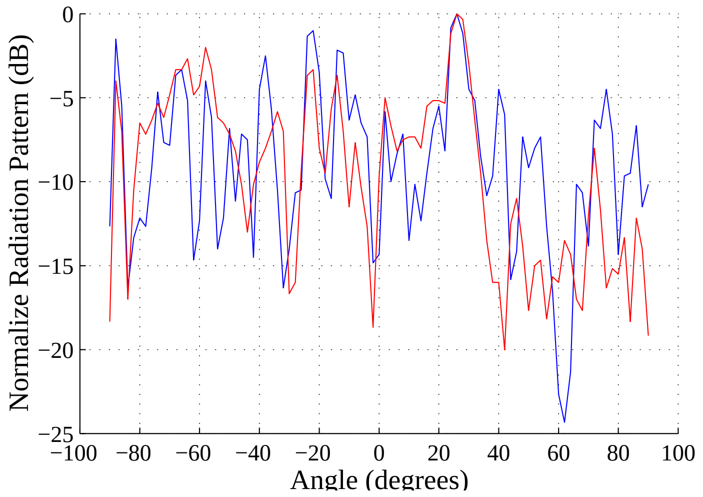
from the oscillating unit cells.

Unfortunately, in this environment, multi-path reflections and power leaking around the lens array to the feed limits the accuracy of the pattern measurements. The effects of multi-path are estimated by monitoring the spectrum analyzer power levels as objects and people move about in the room. After looking at the pattern measurements of the active and passive arrays, the best estimate for the accuracy in the radiation patterns is about 10 dB to 15 dB down from the main lobe for the active array. No discernible array pattern can be deduced from the passive arrays, implying that the errors due to multi-path and power leakage are on the order of the gain of the lens. Radiation patterns can be discern for the active lens array due to the gain of the LNA and PA.

Radiation patterns taken in the focusing dimension, show an increase in side lobe power and an asymmetry in the main beam, Figures 4.15. The characteristic grating lobes do appear in the pattern for the 30^o scan angle, Figure 4.15(b). Patterns simulations for the transmit and receive cases with the appropriate oscillating unit cells turned off and a patch antenna feed, show an increase in side lobe and some asymmetry, Figure 4.16. Most of the asymmetry in the radiation patterns must be due to multi-path errors. Three-dimensional pattern simulations show a reduction in the radiation pattern structure due to the “off” unit cells, Figures 4.17, 4.18.

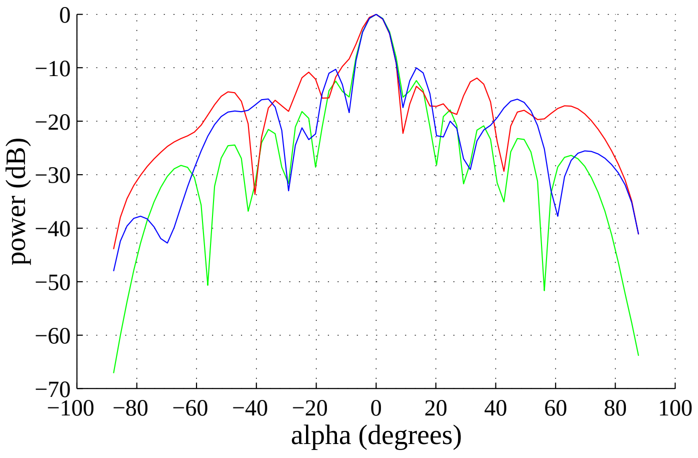


(a)

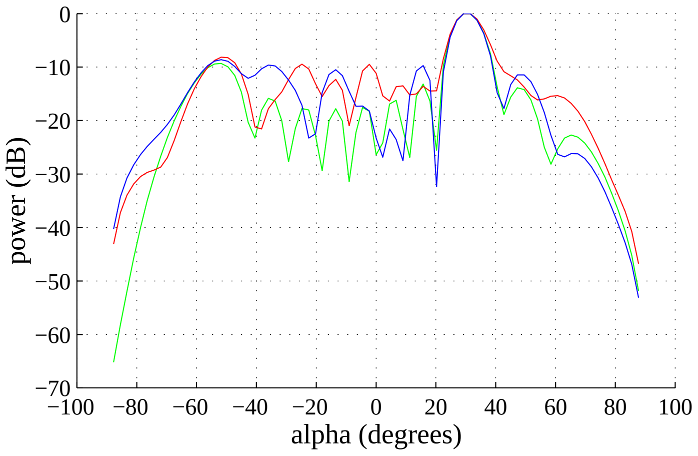


(b)

Figure 4.15. The measure normalized radiation patterns in the focusing dimension for the 0^o off axis (a) and the 30^o off axis (b) feeds. The transmit patterns are in (blue) and the receive patterns are in (red).

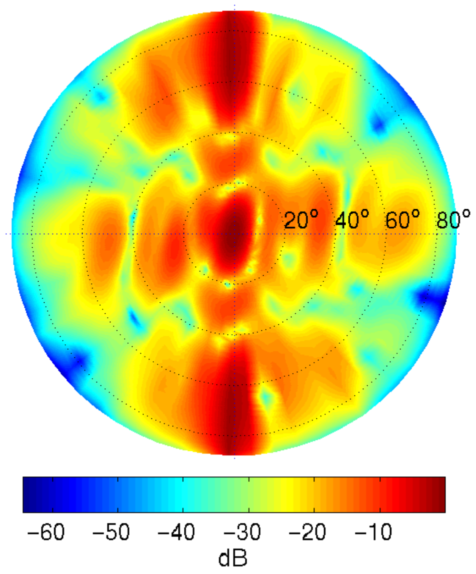


(a)

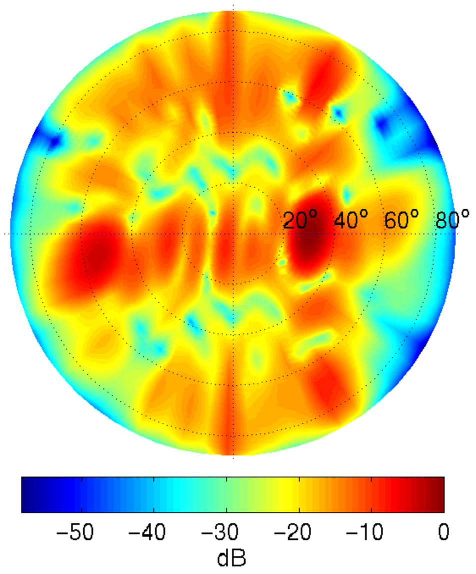


(b)

Figure 4.16. The simulated normalized radiation patterns for the 0^o off axis (a) and the 30^o off axis (b) feeds in the focusing dimension. The transmit patterns are in (blue) and the receive patterns are in (red) with the oscillating unit cells turned off. The radiation patterns for a fully function lens array are in (green).



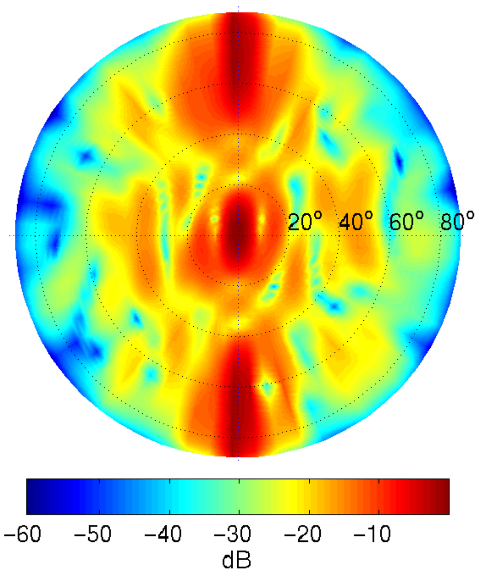
(a)



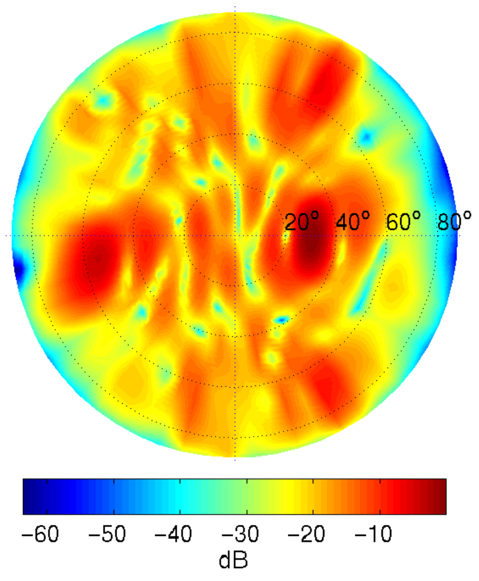
(b)

Figure 4.17. The simulated normalized radiation patterns for the 0^o off axis (a) and the 30^o off axis (b) feeds for the lens array in receive mode.

To estimate the effects of the fiber bundles on the radiation pattern, measurements are done with the passive array by itself, with the FR4 mount and with the FR4 mount and optical fibers. A different test setup is used for the passive array where the array held fix and the source horn moves. This setup seems less prone to multi-path errors and the patterns are taken in the evening when the laboratory is empty. A major source of error in these measurements is maintaining the horn pointing direction to the array. The pattern measurements for the array with the main beam steered through the fiber bundle shows about 2 dB reduction in power for the FR4 mount and the FR4 mount with optical fibers, Figure 4.19. From section 2.5.3 the FR4 mount does attenuate the patch radiation pattern off axis. There is not a noticeable difference between the FR4 and the FR4 mount with optical fibers, suggesting that the FR4 mount dominates the reduction in the radiation pattern.



(a)



(b)

Figure 4.18. The simulated normalized radiation patterns for the 0^o off axis (a) and the 30^o off axis (b) feeds for the lens array in transmit mode.

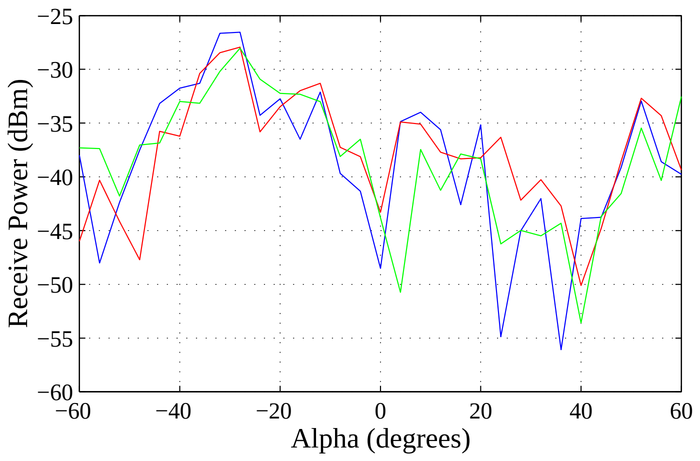


Figure 4.19. Measured passive array radiation patterns for the array (blue), array with FR4 mount (red), and array with FR4 mount and optical fibers. The 30^o off axis feed steers the main beam through the fiber bundles.

4.4 Shaped Pulse Control of SPDT Switch

For controlling very large antenna arrays, the optical control pulses maybe further optimized in shape to provide more efficient use of the optical power. As mentioned in section 3.1.4, the controlling factor on how fast the SPDT switch changes states is the RC time constant formed from the gate capacitance of the MESFET and the “on” resistance of the photodiode. The “on” resistance of the photodiode is a model for the generated carriers in the photodiode discharging the stored charge on the gate capacitance of the MESFET. The more carriers there are, the faster the capacitor discharges. Once the capacitor is discharged, there is not need for the large numbers of carriers and the optical power can be reduced to a level where the gate of the MESFET is held at the proper voltage potential. From the measurements in section 3.3.3, this requires only $0.25\,\mu$W of optical power.

The low “background level” power level suggests a novel control methodology for the SPDT switch, where the photodiodes are pulsed at a high optical power long enough to allow the switch to change state and then reduced to the minimum power for the rest of the period. For the switch rates measured in section 3.3.3, the required optical energy needed to change the state of the switch stays reasonably constant with switch rate at about 10 pJ to 20 pJ, which is far less than the reported energy required to switch a MEMS switch [29]. Figure 4.20 shows a *SPICE* simulation for this control scheme where the complementary optical pulses (blue and red) are switched from 7 mW of optical power to $0.25\,\mu$W. The bias voltage across the two sets of PIN diodes (purple and yellow) switch for “on” to “off” and vise versa, showing the switch changes state.

The real motivation for this type of control comes into play when large arrays are being controlled. For example, assuming a commercially available 100 mW 1480 nm laser diode used in yttrium doped fiber amplifiers is used to control an array at its fastest rate of 2.4 ns rise time for periods of 10 ns. How large of an array can

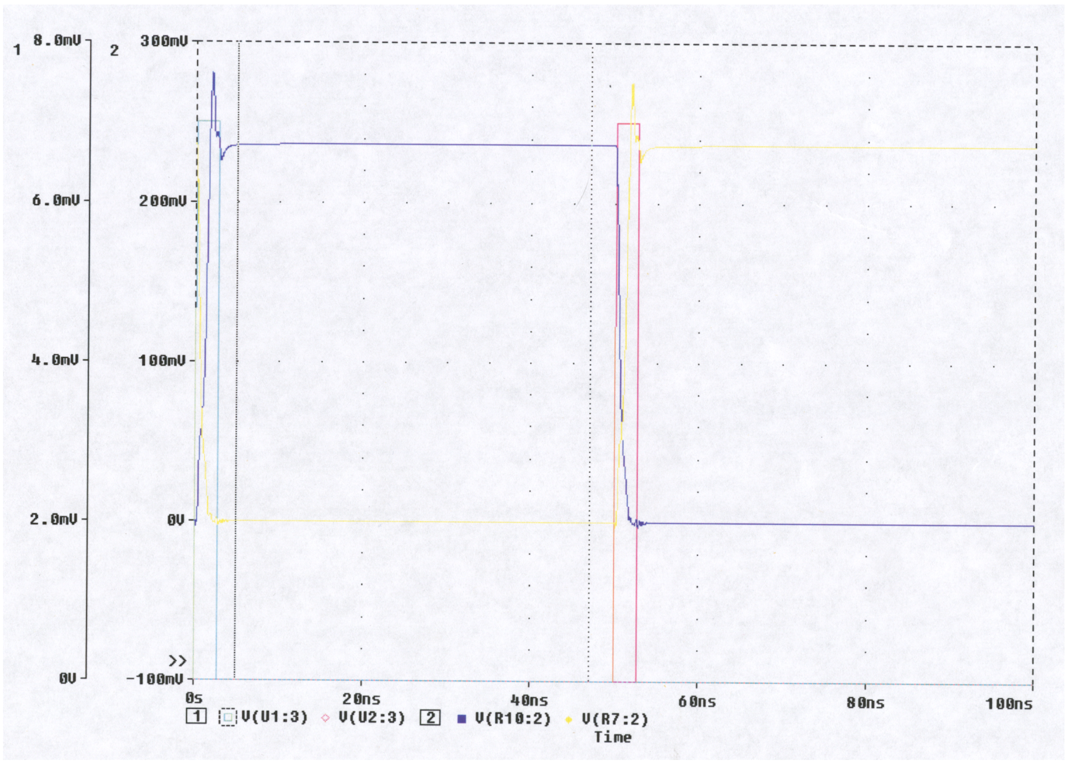


Figure 4.20. *SPICE* simulated performance for the unit cell using shaped optical pulses to reduces the required optical power needed for fast switching. The voltage sources representing the optical signals are (blue and red, far left y-axis scale) and the bias voltages for the two sets of PIN diodes are in (purple and yellow, right y-axis scale)

be controlled with and without the shaped optical pulses? The 100 mW limit on the laser diode is the average power limit dictated by the laser ability to dissipate heat. For a 2.4 ns rise time, 9 mW of optical power is needed per diode or 18 mW per unit cell. For the non-shape optical pulse that allows a

$$\frac{100\,\text{mW}}{18\,\text{mW}} = 5.5 \tag{4.1}$$

unit cells to be controlled. For the shape optical pulse the average optical power needed per unit cell is

$$2\frac{2.4\,\text{ns}}{10\,\text{ns}}\,9\,\text{mW} + 2\frac{7.6\,\text{ns}}{10\,\text{ns}}\,0.25\,\mu\text{W} = 4.3\,\text{mW} \tag{4.2}$$

allowing

$$\frac{100\,\mathrm{mW}}{4.3\,\mathrm{mW}} = 23.2 \tag{4.3}$$

unit cells to be controlled. As the pulse period increases, the power savings for the shaped pulse become more significant.

CHAPTER 5

LENS ARRAYS DESIGN

5.1 Constrained Lens Array Theory

As presented in chapter 1, adaptive array communications systems use a complex weighted sum of the array element outputs to steer a beam in the direction of the desired user and creating nulls in the directions of any unwanted users. In "standard" planar arrays, the information about the direction of the desired user is contained in the phase variations across the array, thus the adaptive algorithm uses all the elements in the array to steer the beam towards the user. However, lens arrays focus signals from different directions onto a focal surface, where the resulting image spatial decorrelates the different signal allowing for more efficient processing of the information. This is equivalent to transforming the problem from a phase-space representation (planar array) to a beam-space representation (lens array). The focusing quality of the lens array affects the overall performance of the adaptive lens array system. In order to understand how lens arrays affect the performance of an adaptive array system, it is important to understand how the design of a lens array affects its ability to focus.

There are several different types of lens arrays that date back to the 1940's, in which stacked metal plates form a waveguide lens used to enhance early RADAR performance [47]. Gent introduced "the Boot Lace Aerial" in 1957 in which transmission lines connect two antenna arrays to form a lens [48]. The two antenna arrays

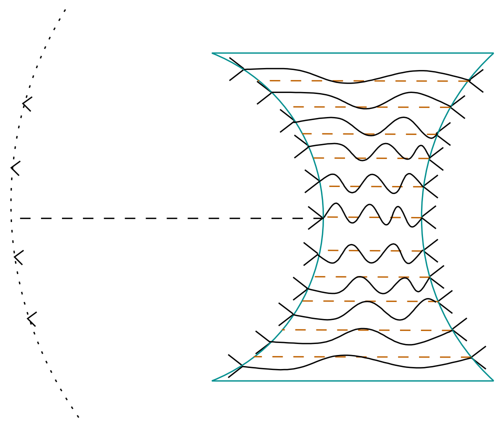


Figure 5.1. “The Boot Lace” lens array showing the four degrees of freedom: 1. Curved non-feed surface; 2. Curved feed surface; 3. Non-feed and feed elements relative offset; and 4. Delay.

function as the front and back surfaces of the lens, and the transmission lines introduce the delays through the lens. The surface of the lens on the side of the focal surface will be referred to as the feed side, and the other side the non-feed side. The transmission lines are longer in the center, in analogy to an optical lens being thicker in the center. By using transmission lines instead of waveguides structures, the lens is non-dispersive over a very large bandwidth.

In principle, the lens arrays can be categorized by the numbers of degrees of freedom in their design. There are up to four degrees of freedom in lens arrays, as indicated Figure 5.1:

1. Non-feed side curved surface;
2. Feed side curved surface;
3. The relative positions between radiating pairs of elements in the two arrays; and
4. The electrical delay between elements.

The number of degrees of freedom in the design determines the number of perfect focal points on the focal surface, one for each degree of freedom. Roa [49] and Rappaport and Zaghloul [50] have reported designs with four perfect focal points. Roa placed the focal points on a straight line where Rappaport and Zaghloul placed them

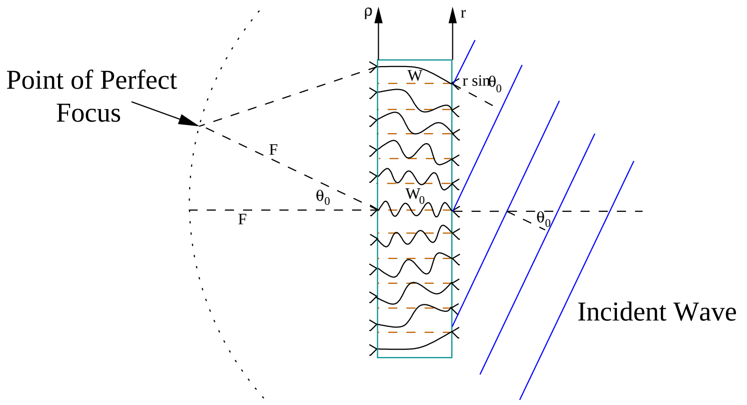


Figure 5.2. A McGrath planar constrained lens with two degrees of freedom: 1. Non-feed and feed elements relative offset and 2. Delay.

on a circle, symmetric about the optical axis. Rotman and Turner [51] presented a three-focal point array lens where the front non-feed side is constrained to be flat. Two of the three focal points lie symmetric about the optical axis, and the third is on the optical axis.

The lens array presented first by McGrath [2] is of most interest to this project since it constrains both the non-feed side and the feed side to be flat, Figure 5.2. With only two degrees of freedom, the two focal points lie symmetrically around the optical axis at angles $\pm\theta_0$. The equations that determine the location and the delay for each element in the array are derived from Figure 5.2, such that the path delays for a plane wave incident at an angle θ_0 are the same for any element pair in the array. Equation (5.1) relates the location of a feed-side element (ρ) in terms of the non-feed side elements (r), focal distance (F) and angle of perfect focus (θ_0).

$$\rho = \sqrt{\frac{F^2 - r^2 \sin^2 \theta_0}{F^2 - r^2}} \tag{5.1}$$

The delay for each element (W) is determined by the non-feed side element position, focal distance and the unit delay for the center element (W_0):

$$W = F + W_0 - \frac{1}{2}\sqrt{F^2 + \rho^2 - 2\rho F \sin\theta_0} - \frac{1}{2}\sqrt{F^2 + \rho^2 + 2\rho F \sin\theta_0} \tag{5.2}$$

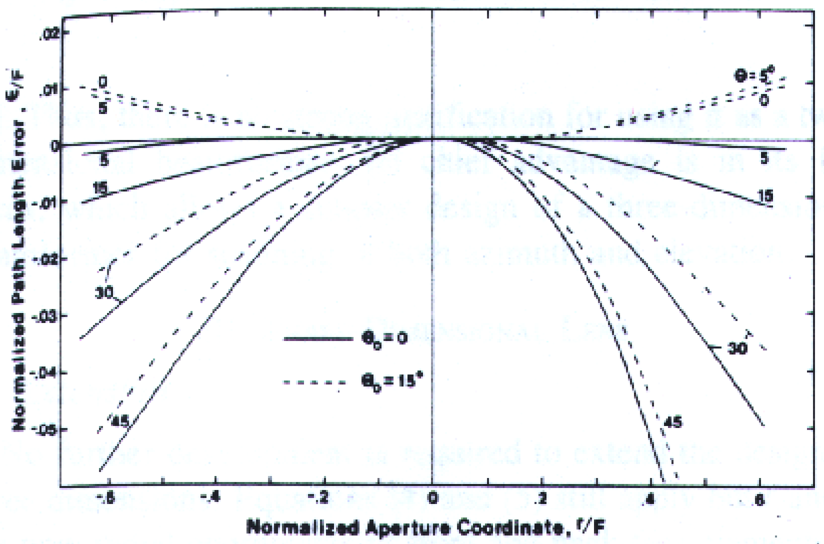
At all other points on the focal surface, the path delay error follows closely a cubic function, Figure 5.3(a), suggesting "coma" aberrations [24] which can be reduced by re-focusing. McGrath found that a new focal surface ($G(\theta)$) given by equation (5.3) minimizes the RMS error in path length integrated over the aperture of the lens, Figure 5.3(b). The $G(\theta)$ is given by:

$$G(\theta) = F\sec(\theta_0)\left[1 + \frac{1}{2}\frac{\sin^2\alpha\sin^2\theta}{(1-\sec\alpha)(1+\sin\alpha\sin\theta)}\right] \tag{5.3}$$

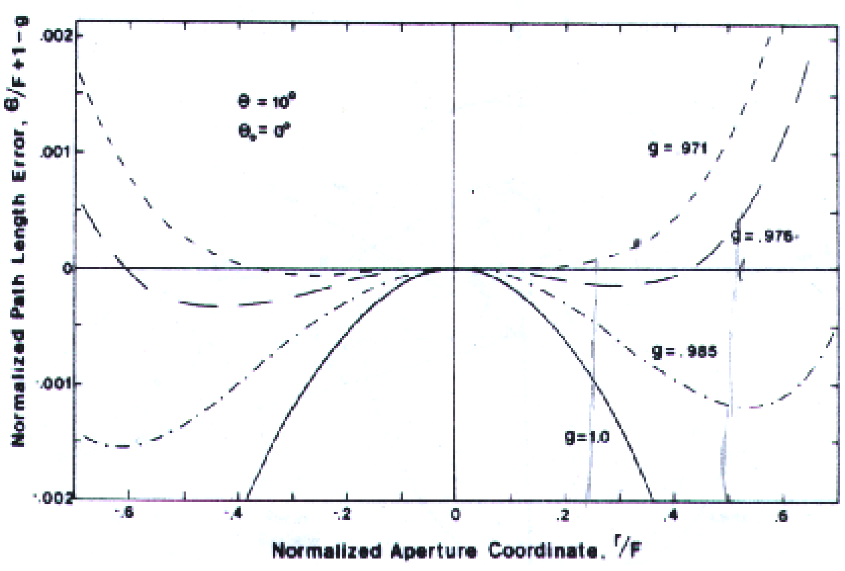
$$\alpha = \sin^{-1}\left(\frac{r_{max}}{F}\right) \tag{5.4}$$

where r_{max} is the radius of the lens. Equations (5.1, 5.2, and 5.3) are given in terms of radial coordinates of the elements, therefore a three-dimensional array can be designed by imposing symmetry about the optical axis. Roa states that in practical situations, this axial symmetry is not possible, and a lens has a cone of best focus instead of a cone of perfect focus at θ_0. Figure 5.3 also shows how the cone of best focus and the re-focused focal surface try to minimize the average error by letting the phase be both positive and negative.

Since the planar constrained lens only has two degrees of freedom, it can be uniquely specified by two design parameters (F/D and θ_0) and the feed side element spacing, where D is the diameter of the aperture. Figure 5.4 shows the layout of two different lens arrays that are $4\,\lambda$ in diameter and have inter-element spacings of $\frac{\lambda}{2}$. The first lens design has F/D=0.6 and $\theta_0 = 0$, and the second design has F/D=0.6 and $\theta_0 = 45^o$. The red x's and yellow o's represent the centers of the non-feed side and feed side radiating elements, respectively, with the focal surface for the lenses shown in blue. The first thing to note is that for small enough F/D and θ_0, it is possible to have a lens design that is unrealistic since the feed side elements lie outside the array aperture. By increasing θ_0, the elements are brought in but at the cost of deforming the focal surface away from the lens which results in increased loss. As in planar array design, the inter-element space in the lens array needs to be $\frac{\lambda}{2}$

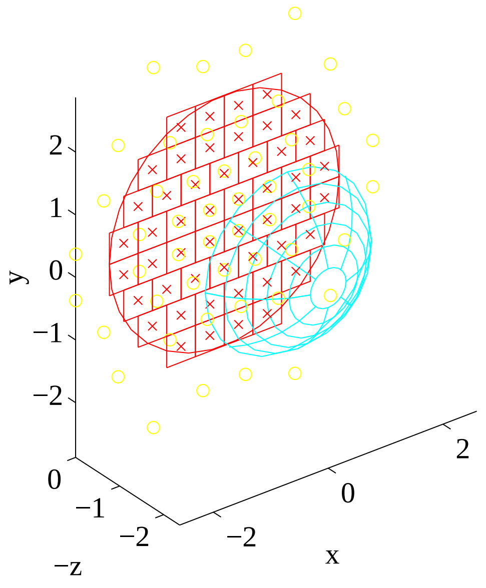


(a)

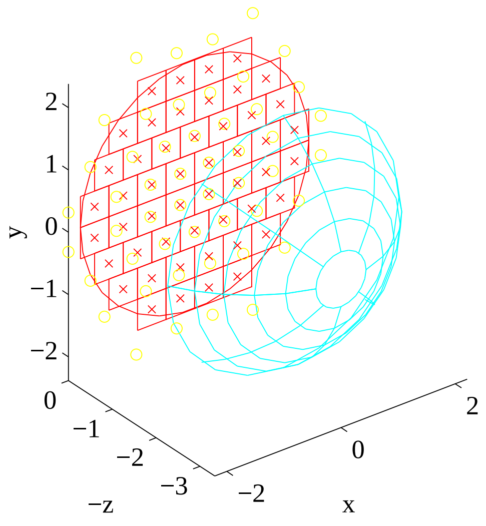


(b)

Figure 5.3. Normalized phase error versus normalized aperture coordinates for a spherical focal surface (a), and a refocused focal surface (b). (Scanned for [2]



(a)



(b)

Figure 5.4. Example of two lens designs with F/D=0.6 and cone of best focus, $\theta_0 = 0^o$ (a) and $\theta_0 = 45^o$ (b). Red x's depict the centers of the non-feed side antennas and yellow o's those of the feed side antennas. The focal surface of the lenses are shown in blue.

by $\frac{\lambda}{2}$ to prevent grating lobes when steering off axis, [21]. However, the best way to pack the elements (unit cells) into the lens aperture is yet to be determined.

5.2 Lens Array Modeling

a lens arrays, only approximates a real microwave lens. The finite sampling of the aperture and the approximation of the refraction through a lens with delay lines introduce loss and aberrations in the image of the lens. As from phase errors, there are several other factors in the physics of lens arrays that also lend to the loss and aberrations in images. For example, the polarization and radiation patterns of the antenna elements in the lens affect how the re-radiated signals are focused on the image for different scan angles. The path loss difference between the different elements in the lens and a point on the focal surface affects how the different signals interfere.

An analytical solution for the aberrations and loss in a lens array system is, however the physics of the problem makes a analytical solution intractable. Instead, a numerical model for the lens array is developed to characterize the aberration and loss. The model uses radiation coupling equations based on Franhoffer diffraction theory, since the physical size of the lens system is beyond the resources of most full-wave simulators. Using radiation coupling equations, the model will not be able to predict high order interactions between elements in the array and elements on the focal surface. The model should be flexible enough to allow changes in lens design, different types of sources excitations and focal surface detectors. It also needs to be able to simulate various microwave environments (i.e. multi-path, multi-user, moving sources, changing polarization, etc.) so the effects of lens arrays in adaptive antenna systems may be explored.

5.2.1 Physical Bases for the Numerical Model In the model for the lens array, only far-field (Franhoffer) radiation coupling is taken into account [45, 52]. This coupling is the result of spherical electromagnetic waves propagating from one antenna to another. The effective length is used to describe the coupling between antennas since it preserves both the gain and polarization of the antenna. The electric field at a point in the far field induced by an antenna with input current I_{in} is given by

$$\overrightarrow{E} = -j\eta \frac{kI_{in}}{4\pi R} \overrightarrow{l_{eff}}(\theta, \phi) e^{-jkR} \tag{5.5}$$

where $\overrightarrow{l_eff}$ is a vector describing the polarization and gain of the electric field for an antenna in the direction (θ, ϕ). k is the propagation constant for a plane wave, and R is the distance between the center of the antenna and the point of observation. The induced open circuit voltage, V_{oc}, at the antenna port with an incident electric

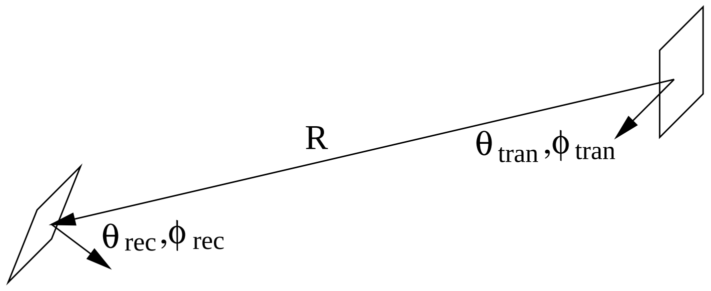


Figure 5.5: Coordinate relationships for calculating the coupling of two antennas.

field $\overrightarrow{E}$ is given by:

$$V_{oc} = \overrightarrow{E}^* \cdot \overrightarrow{l_{eff}}(\theta, \phi) \tag{5.6}$$

A^* denotes conjugate transpose of the electric field vector. The coupling between a transmit and receive antenna that are separated by, R, and in the directions (θ_{tran}, ϕ_{tran}) and (θ_{rec}, ϕ_{rec}) for transmit and receive angles, respectively, as shown in Figure 5.5, is given by:

$$V_{oc} = -j\eta \frac{kI_{in}}{4\pi\, R} \overrightarrow{l_{eff}}^*(\theta_{rec}, \phi_{rec}) \cdot \overrightarrow{l_{eff}}(\theta_{tran}, \phi_{tran}) e^{-jkR} \tag{5.7}$$

$\overrightarrow{l_{eff}}$ for both antennas are in a common coordinate system for the dot product of the effective lengths to accurately describe the polarization interaction. This can be rewritten as:

$$V_{oc} = j\eta \frac{kI_{in}}{4\pi\, R} \left|\overrightarrow{l_{eff}}(\theta_{rec}, \phi_{rec})\right| \left|\overrightarrow{l_{eff}}(\theta_{tran}, \phi_{tran})\right| \left(\overrightarrow{\rho^*_{eff}}(\theta_{rec}, \phi_{rec}) \cdot \overrightarrow{\rho_{eff}}(\theta_{tran}, \phi_{tran})\right) e^{-jkR} \tag{5.8}$$

Equation (5.8), explicitly separates the antenna gains $|l_{eff}|$and polarization $\overrightarrow{\rho}(\theta, \phi)$ in to a form more useful for the model. All calculations are done at a single frequency with narrowband systems in mind.

Using Equations (5.5, 5.6, and 5.8) all the electric fields on the lens array and focal surface can be calculated. For example, assume we have a source antenna in the near field of the lens array and a far-field plane wave incident on the lens array.

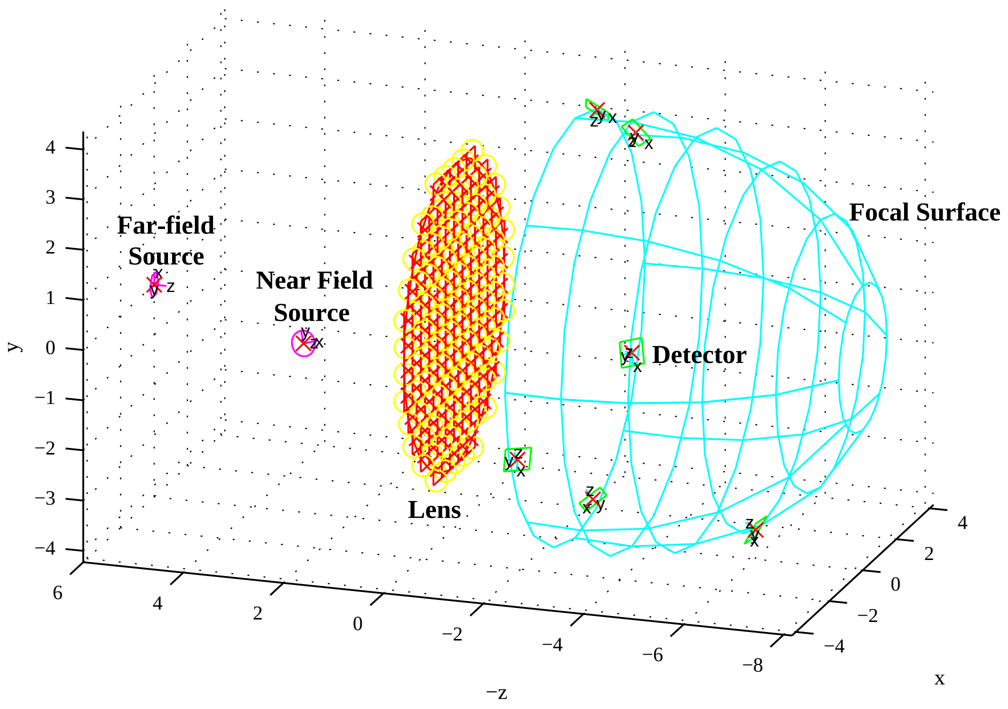


Figure 5.6. Example of lens array system with far-field source (magenta hexagon), near-field source (magenta circle) and a detectors on the the focal surface (green squares).

The lens array itself is populated with antenna elements on both sides, each having their own orientation, connected through delay lines and amplifiers. The antenna elements on the feed side of the array radiate to antennas placed on the focal surface, or other points, behind the array. Figure 5.6 illustrates the lens system with the near field source represented by a magenta circle and the far-field source represented by a magenta hexagon in the direction of arrival 6 λ from the center of the lens in the direction of reception. The non-feed side antennas are represented by red x's and feed side antennas represented by yellow o's at their positions in the array. The detector antenna (green) is on the focal surface (blue). Each source, array element, and detector has its own orientation represented by its own local coordinate system.

Using Jacobean matrices, [J], to perform coordinate transformations from local to global coordinate systems and from spherical to Cartesian coordinate systems

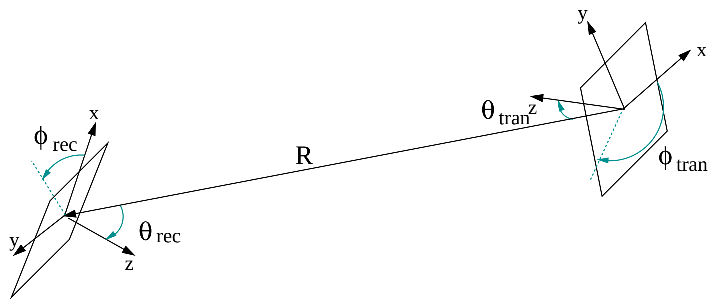


Figure 5.7. The coupling between two antennas, each with its own local coordinate systems

referring to Figure 5.7, Equation (5.8) can be rewritten in a form as

$$V_{oc} = j\eta \frac{kI_{in}}{4\pi\, R} \left|\overrightarrow{l_{eff}}(\theta_{rec}, \phi_{rec})\right| \left|\overrightarrow{l_{eff}}(\theta_{tran}, \phi_{tran})\right|$$
$$\overrightarrow{\rho^*_{eff}}(\theta_{rec}, \phi_{rec}) \cdot [J_{rec,sphere}(\theta_{rec}, \phi_{rec})] \cdot [J^*_{rec,globe}] \cdot$$
$$[J_{tran,globe}] \cdot [J^*_{tran,sphere}(\theta_{tran}, \phi_{tran})] \cdot \overrightarrow{\rho_{eff}}(\theta_{tran}, \phi_{tran}) e^{-jkR} \quad (5.9)$$

$[J_{rec,globe}]$ transform local receive Cartesian coordinates to global receive Cartesian coordinates. $[J_{rec,sphere}(\theta_{rec}, \phi_{rec})]$ transforms receive Cartesian coordinates to receive spherical coordinates in a direction (θ_{rec}, ϕ_{rec}). Since Jacobian matrices are unitary, their inverse is the conjugate transpose represented by $[J^*]$ and performs the inverse coordinate transformation, [53].

Using a circuit model to describe the connection of non-feed side elements to feed side elements, Figure 5.8, the input current for the feed side antenna can be calculated from the induced open circuit voltage at the non-feed side antenna as:

$$I_{in,NFD} = \frac{V_{oc} G_{ain}}{R_{FD} + R_{NFD}} \quad (5.10)$$

where R_{FD} and R_{NFD} are the feed side and non-feed side antennas resistances, respectively. Here the amplifier gain is a current gain and the antenna impedances are assume to be real and matched to the amplifiers.

By cascading Equation (5.9) for a near-field source or Equation (5.6) for a far-field planes with Equation (5.10) for propagating through the lens array and

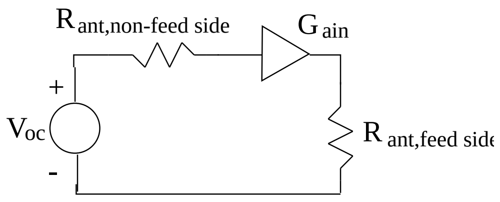


Figure 5.8. Circuit model for the non-feed side and feed side antennas in a lens array.

Equation (5.9) for feed side elements radiating to a detector, the open circuit voltage at the detector can be determined. From Equation (5.10), lens arrays fundamentally have 3 dB more loss than an equivalent planar array due to the conversion of radiated power to voltage and back to radiated power prior to detection (reception).

5.2.2 Antenna Element Models The equations presented in section 5.2.1 describe the radiation in terms of the antenna effective lengths. One of the requirements for a numerical model for a lens array, is the ability to describe arbitrary antennas. A standard way to describe an antenna's effective length is through an integral of its surface currents [45, 52]. Depending on the antenna, this could be an electric current (e.g. in a dipole) or a magnetic current (e.g. in a cavity model for a patch antenna). in a lossless antenna, the total radiated power is equal the total input power (P_{in}) at the feed. This is sometimes written as a relationship between input resistance, R_{ant}, and radiation resistance, R_{rad}, [45]:

$$P_{rad} = |I_0|^2 R_r \tag{5.11}$$

$$P_{in} = |I_{in}|^2 R_a \tag{5.12}$$

$$\frac{R_a}{R_r} = \frac{|I_0|^2}{|I_{in}|^2} \tag{5.13}$$

where the I_0 and I_{in} are the electric current density for the antenna and input current at the feed of the antenna respectively.

The radiation resistance is essentially a normalization of the total power radiated:

$$\begin{aligned} R_{rad} &= \frac{1}{|I_{in}|^2} \iint\limits_S \overrightarrow{E} \times \overrightarrow{H}^* \cdot \mathrm{d}S \\ &= \frac{1}{\eta\,|I_{in}|^2} \iint\limits_S \left|\overrightarrow{E}\right|^2 \cdot \mathrm{d}S \end{aligned} \tag{5.14}$$

where η is the free space impedance, 377Ω.

If the model has to perform a power normalization integral each time the effective length of the antenna is needed, it will run prohibitively slow. Fortunately, many antennas like dipoles, microstrip patch antennas and slot antennas have a structure to their electric and magnetic surface currents that determine their radiation patterns. For dipole antennas, all the calculations including the normalization constant can be done ahead of time and stored in a file, however for patch and slot antennas whose radiation pattern changes for different shape antennas, this is not true. For patch and slot antennas it is possible to calculate the general shape of the radiation pattern as a function of the patch shape apriori and then perform the surface integral once to calculate the normalization constant.

It is possible to relate the integral of the radiated power to an integral of the prototypes effective length. Then by calculating the normalization constant from the integral of the prototype's effective length a single set of equations can be used for the power normalization and the radiation coupling calculation. The total radiated power by an antenna is given by

$$\begin{aligned} P_{rad} &= \iint\limits_S \overrightarrow{E} \times \overrightarrow{H}^* \cdot \mathrm{d}S \\ &= \frac{1}{\eta} \iint\limits_S \left|\overrightarrow{E}\right|^2 \mathrm{d}S \end{aligned} \tag{5.15}$$

The electric field is calculated from the electric current density vector, $\overrightarrow{J}$, as

$$\begin{aligned}\overrightarrow{E_J} &= \frac{-k\eta}{4\pi R} e^{-jkR} \iint\limits_S \overrightarrow{J} e^{-jkR\cos\psi}\,\mathrm{d}S \\ &= \frac{-k\eta}{4\pi R} e^{-jkR} \overrightarrow{N}\end{aligned} \tag{5.16}$$

$$\overrightarrow{N} = \iint\limits_S \overrightarrow{J} e^{-jkR\cos\psi}\,\mathrm{d}S \tag{5.17}$$

where ψ is the angle between the vectors that point to the current element and the observation point in space. For a magnetic current density vector, $\overrightarrow{M}$, the electric field is

$$\begin{aligned}\overrightarrow{E_M} &= \frac{k}{4\pi R} e^{-jkR} \widehat{a}_r \times \iint\limits_S \overrightarrow{M} e^{-jkR\cos\psi}\,\mathrm{d}S \\ &= \frac{-k}{4\pi R} e^{-jkR} \widehat{a}_r \times \overrightarrow{L},\end{aligned} \tag{5.18}$$

$$\overrightarrow{L} = \iint\limits_S \overrightarrow{M} e^{-jkR\cos\psi}\,\mathrm{d}S, \tag{5.19}$$

where $\widehat{a}_r$ is the unit vector for the direction of observation. Combining Equation (5.5) with equations (5.16), (5.17), (5.18), and 5.19) gives the effective length in terms of electric and magnetic currents:

$$\overrightarrow{L_{eff,J}(\theta,\phi)} = \frac{\overrightarrow{N}}{I_{in}} = \frac{J_0}{I_{in}} \overrightarrow{N'}(\theta,\phi) \tag{5.20}$$

$$\overrightarrow{L_{eff,M}(\theta,\phi)} = \frac{-\widehat{a}_r \times \overrightarrow{L}}{\eta I_{in}} = \frac{-\widehat{a}_r M_0 \times \overrightarrow{L'}(\theta,\phi)}{\eta I_{in}} \tag{5.21}$$

$N'(\theta,\phi)$ and $L'(\theta,\phi)$ represent the surface integral for the antennas per Equations (5.17 and 5.19) and J_0 and M_0 are normalization constants for power conservation. If both electric and magnetic currents are present then the effective length is give by

$$\overrightarrow{L_{eff,J,M}} = \overrightarrow{L_{eff,J}} + \overrightarrow{L_{eff,M}} \tag{5.22}$$

Combining equation (5.15) with (5.16) results in an equation for the input antenna resistance, as follows:

$$P_{rad} = \frac{J_0^2}{2\eta} \iint\limits_S \left| \frac{-jk\eta}{4\pi R} e^{-jkR} \overrightarrow{N'} \right|^2 dS \tag{5.23}$$

$$\frac{|I_{in}|^2}{2} R_{ant} = \frac{J_0^2 k^2 \eta}{32\pi^2 R^2} \iint\limits_S \left| \overrightarrow{N'} \right|^2 dS \tag{5.24}$$

$$R_{ant} = \frac{J_0^2 k^2 \eta}{16\pi^2 R^2} \iint\limits_S \left| \frac{\overrightarrow{N'}}{I_{in}} \right|^2 dS \tag{5.25}$$

$$\tag{5.26}$$

Note that Equation (5.25) contains the effective length in the integral. Now it is possible to solve for the normalization constant J_0 in terms of the antenna input resistance, and the input current. For the lens model both the input resistance and input current are taken to be unity for unit input power in the calculation of the normalization constant, J_0:

$$\left| \frac{1}{J_0} \right|^2 = \frac{k^2 \eta}{16\pi^2 R^2} \iint\limits_S \left| \overrightarrow{N'} \right|^2 dS \tag{5.27}$$

A similar set of equation can be written for magnetic currents:

$$P_{rad} = \frac{M_0^2}{2\eta} \iint\limits_S \left| \frac{jk}{4\pi R} e^{-jkR} \widehat{a}_r \times \overrightarrow{L'} \right|^2 dS \tag{5.28}$$

$$\frac{|I_{in}|^2}{2} R_{ant} = \frac{M_0^2 k^2}{32\eta\pi^2 R^2} \iint\limits_S \left| \widehat{a}_r \times \overrightarrow{L'} \right|^2 dS \tag{5.29}$$

$$R_{ant} = \frac{M_0^2 k^2 \eta}{16\pi^2 R^2} \iint\limits_S \left| \frac{-\widehat{a}_r \times \overrightarrow{L'}}{\eta I_{in}} \right|^2 dS \tag{5.30}$$

$$\left| \frac{1}{M_0} \right|^2 = \frac{k^2}{16\eta\pi^2 R^2} \iint\limits_S \left| \widehat{a}_r \times \overrightarrow{L'} \right|^2 dS \tag{5.31}$$

When both electric and magnetic currents are present and their scaling constants are related by $J_0 = CM_0$, then the new normalization equation is given by

$$\left| \frac{1}{J_0} \right|^2 = \frac{k^2 \eta}{16\pi^2 R^2} \iint\limits_S \left| \overrightarrow{N'} - C \frac{\widehat{a}_r \times \overrightarrow{L'}}{\eta} \right| dS \tag{5.32}$$

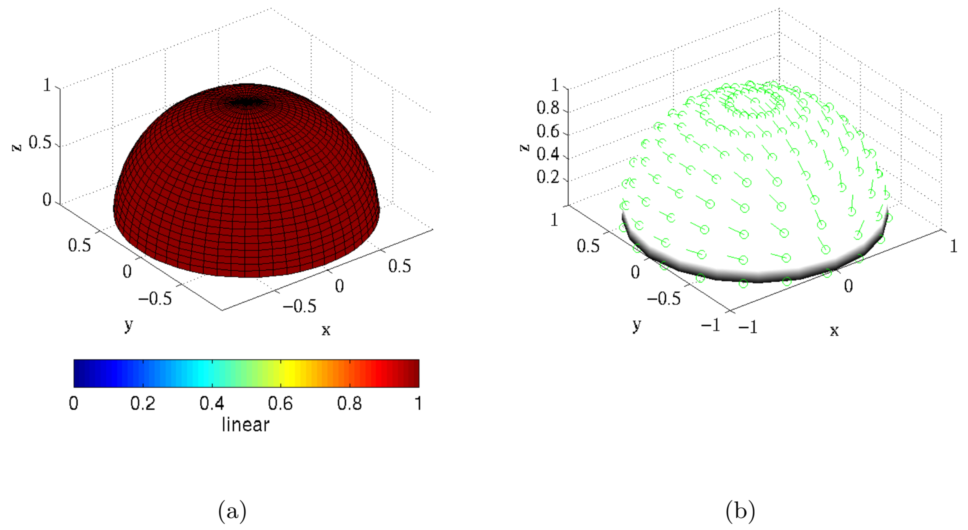


Figure 5.9. The radiation pattern (a) and far-field electric field pattern (b) for a semi-directional antenna. The electrical field vectors are plotted tangent to a unit sphere with their tails denoted as circles.

.

There are two principle antennas used in the model simulations, semi-directional and patch antenna. A semi-directional antenna radiates equally well in all directions in a hemisphere and nowhere else, Figure 5.9(a). The electric field polarization is chosen to be consistent with a $\widehat{a}_y$ oriented electric surface currents density vectors, Figure 5.9(b). When Equation (5.27)is applied to the semi-directional, antenna the normalization constant becomes $J_0 = \sqrt{\frac{2}{\eta\pi}}$

To calculate the normalization constant for the patch antenna, the cavity model is used [45], Figure 5.10. The cavity model represents the radiation from fringing electric fields off the patch as two magnetic currents. To be consistent with the semi-directional antenna, the magnetic currents are chosen to be in the $\widehat{a}_x$ direction, $\overrightarrow{M_x}$, so that the radiated electric field is polarized along $\widehat{a}_x$. By using the cavity model for the patch antenna, the numerical model only needs the electric

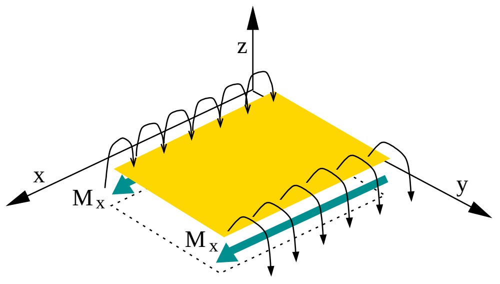


Figure 5.10: Cavity model for patch antenna.

length (L_e), width (W_e) and height (h_e) to calculate the normalization constant and the all subsequent effective lengths. The relevant equations are given below.

$$l_{eff,\theta} = \frac{-L_\phi}{\eta I_{in}} \tag{5.33}$$

$$l_{eff,\phi} = \frac{L_\theta}{\eta I_{in}} \tag{5.34}$$

$$\begin{aligned} L_\theta =& 2|\overrightarrow{M_x}|h_e W_e \cos\theta\, \cos\phi \mathrm{sinc}\, (kh_e \cos\theta) \\ & \mathrm{sinc}\left(k\frac{W_e}{2}\sin\theta\cos\phi\right)\cos\left(k\frac{L_e}{2}\sin\theta\, \sin\phi\right) \end{aligned} \tag{5.35}$$

$$\begin{aligned} L_\phi =& -2|\overrightarrow{M_x}|h_e W_e \sin\phi \mathrm{sinc}\, (kh_e \cos\theta) \\ & \mathrm{sinc}\left(k\frac{W_e}{2}\sin\theta\cos\phi\right)\cos\left(k\frac{L_e}{2}\sin\theta\sin\phi\right) \end{aligned} \tag{5.36}$$

a similar set of equations are derived for $\widehat{a}_y$ oriented currents,$\overrightarrow{M_y}$, the equations are,

$$\begin{aligned} L_\theta =& 2|\overrightarrow{M_y}|h_e L_e \cos\theta\sin\phi \mathrm{sinc}\, (kh_e \cos\theta) \\ & \mathrm{sinc}\left(k\frac{L_e}{2}\sin\theta\sin\phi\right)\cos\left(k\frac{W_e}{2}\sin\theta\cos\phi\right) \end{aligned} \tag{5.37}$$

$$\begin{aligned} L_\phi =& 2|\overrightarrow{M_y}|h_e L_e \cos\phi \mathrm{sinc}\, (kh_e \cos\theta) \\ & \mathrm{sinc}\left(k\frac{W_e}{2}\sin\theta\sin\phi\right)\cos\left(k\frac{L_e}{2}\sin\theta\cos\phi\right) \end{aligned} \tag{5.38}$$

With equations (5.33) to (5.38) the model can simulate any polarize patch antenna including elliptical polarization by relating $\overrightarrow{M_y} = C\overrightarrow{M_x}$, where C is a complex constant. The normalization constants, M_0, needs to be calculated only once and them stored in memory for future reference. For example, the patch antenna used in

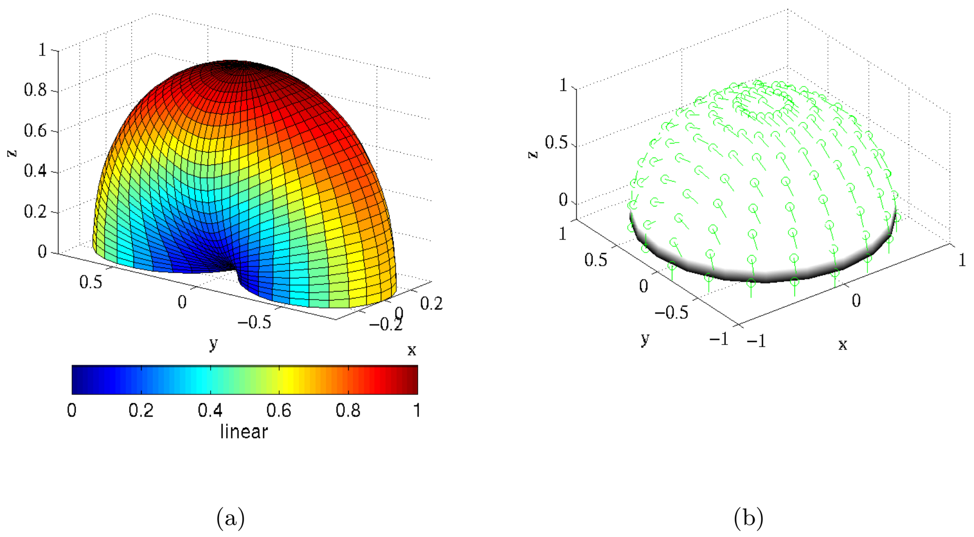


Figure 5.11. The radiation pattern (a) and far-field electric field pattern (b) for a patch antenna. The electric field vectors are plotted tangent to a unit sphere with their tails denoted as circles.

Chapter 4 has a radiation pattern, as shown in Figure 5.11(a) and electric field polarization plot, Figure 5.11(b) for a magnetic current density in the $\widehat{a_x}$ direction.

5.2.3 Numerical Model Structure

MatLab 5.3 produced by MathWorks is chosen as the programming tool for a design-oriented lens array modeling tool. A complete listing of the MatLab functions are listed in Appendix A, with a basic description of the objects listed below. Many of the objects and functions have names taken from microwave and optical imaging systems.

Object 1: **antenna**

The antenna object describes the type of antenna, its position and its orientation. Stored with the antenna object is a set of coordinate transformation matrixes that allow easy calculation of coupling between antenna elements per Equation (5.9).

Object 2: **Lens**

The lens object contains all the information to describe the lens: size, shape (circular or rectangular), number of unit cells and the design parameters (F/D and θ_0) as well as a complete description of each unit cell (gain, delay, feed side and non-feed side antenna objects). Since the lens array is an approximation to a lens, the way the aperture is populated with unit cells can effect its performance. To explore the effects of the unit cell placement, two different lattices and aperture approximations are used. The unit cells may be packed with a triangular lattice or maximally fill the aperture. The aperture may be approximated by constraining the unit cells to be within the aperture boundary, or letting them cross the boundary by $\frac{1}{4}$ unit cell width. Figure 5.12 shows four examples of a $3\,\lambda$ radius circular lens with different lattices and aperture approximations.

Object 3: **Array**

For comparison to a lens array, a planar array object is develop under the same conditions as the lens array.

Object 4: **Detector**

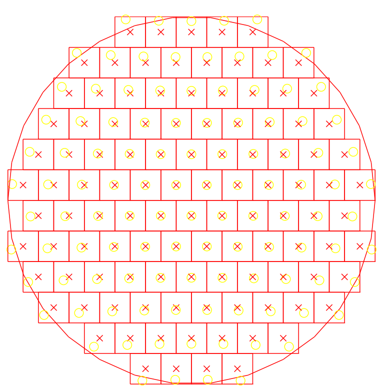
A detector is an antenna element (or antenna array with corporate feed) placed behind the used lens to detect the focal image of the lens. Each element in the detector is an antenna object with a delay and gain associated with it. The output of the corporate feed network may also have a gain associated with it.

Object 5: **Imager**

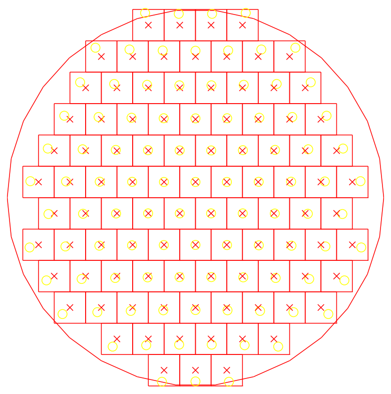
An imager is a collection of detectors used to sample the image produced by the lens.

Object 6: **Source**

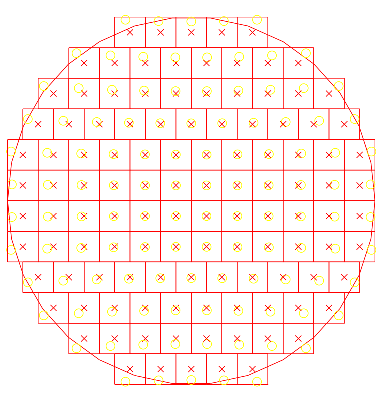
There are two different types of sources: near-field and far-field. A near-field source is an antenna object radiating towards the array. A far-field source is a plane wave incident upon the array. Both sources have input magnitudes, I_{in} for near-field and E_{in} for far-field, and phase delays.



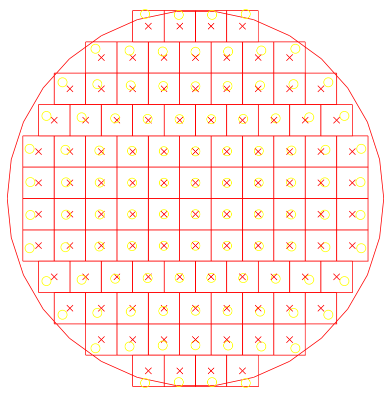
(a)



(b)



(c)



(d)

Figure 5.12. Four examples of unit cell layouts. Triangular lattice and unconstrained aperture approximation (a) Triangular lattice and constrained aperture approximation (b) Maximumly packed lattice and unconstrained aperture approximation (c) Maximumly packed lattice and constrained aperture approximation (d)

Object 7: **Channel**

The channel describes how the signals for a user is received by the array. It is a collections of near-field and far-field sources that represent the paths that the users signal took to the array.

Object 8: **QOLTF**

The Quasi-Optical Lens Transfer Function is a matrix that describes the induced open circuit voltages at the outputs of each detector in an imager due to a channel through a lens array.

Object 9: **NoiseTF**

The Noise Transfer Function is a matrix that relates the noise introduced by gain elements in a lens and noise in the detectors to the outputs of a “imager”.

Object 10: **Radiation Pattern**

Radiation pattern is the far-field radiation pattern of a lens array as seen in reception by a detector feed.

Object 11: **Image Pattern**

The Image pattern is the received power that a detector with unit load will receive when placed on the focal surface for a given “ channel”.

For easier interpretation of radiation patterns and image patterns, two different coordinate system are used in their calculation, Figure 5.13. The (α, χ) coordinate system is the negative of the (θ, ϕ) coordinate system.

Combining the above objects makes it possible to model communication systems that have mobile sources, multiple users, multi-path propagation channels, and time varying polarization.

5.3 Lens Simulation Results

In designing planar lens arrays, there are only two degrees of freedom: the relative antenna positions and unit cell delays. These two degrees of freedom

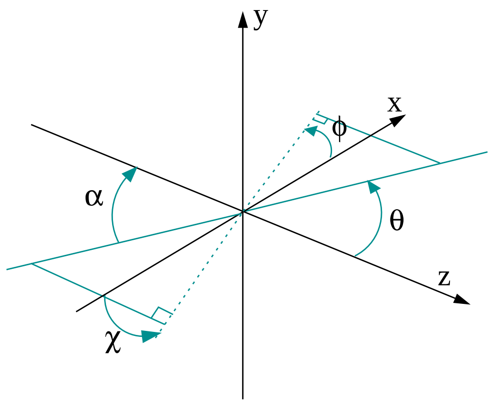


Figure 5.13. The two coordinate systems used to plot radiation and image patterns. By plotting the results in these two coordinate system, the inverse image introduced by the lens is reversed and a source at (α_s, χ_s)=(C_1,C_2) is detected at (θ_d,ϕ_d)=(C_1,C_2).

are uniquely specified by two design parameters, F/D and cone of best focus, θ_0. These two parameters only define the structure of the lens and do not define the antenna elements nor how the unit cells are placed in the array. To understand how the lens design parameters and unit cell design lens' overall performance, several numerical experiments are performed. The metrics that define a good lens design are application-specific. For example, in a spatial power combining application, the loss in the lens system affects the combining efficiency and is therefore most important. In an angle diversity communication system, the aberrations (distortions in the image) may be more important than the loss. Since it is not known which performance metrics are most important for angle diversity communication systems, a general characterization of focusing ability of the lens is performed for different lens designs. In most communication systems, it is the average performance that matters, and in an imaging system this average is over spatial angles. In the numerical experiments, fifty far-field sources are used, uniformly spread over the field of view for 0^o to 90^o off axis, Figure 5.14. The sources are arrange on concentric circles of 0^o (source 1), 20^o (sources 2-7), 40^o (sources 8-18), 60^o (sources 19-33), and 80^o (sources 34-50) off axis.

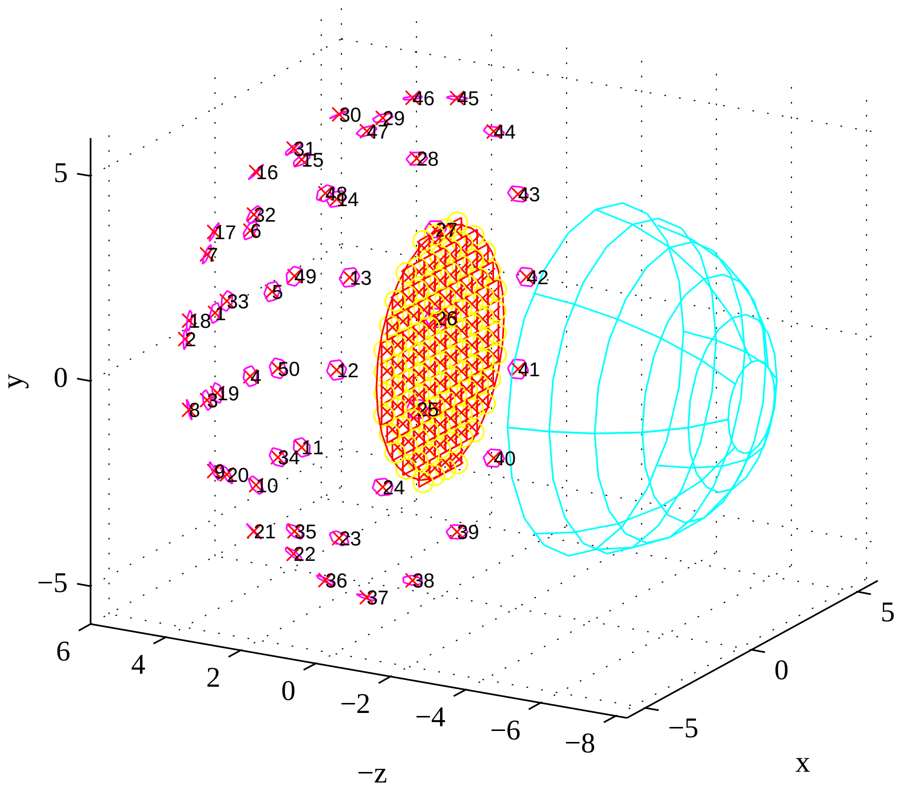
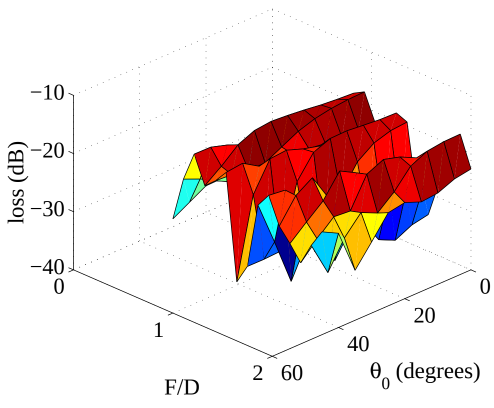
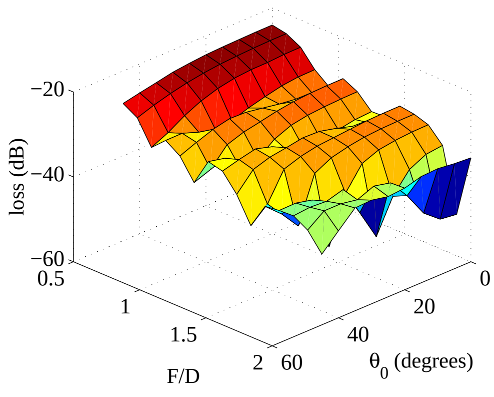


Figure 5.14: Fifty far-field sources used in numerical lens array simulations.

5.3.1 F/D versus θ_0 The first experiment examines the loss in a lens as a function of F/D and θ_0. Eight prototype lens designs are based on all combinations (patch or semi-directional antennas; triangle or maximumly packing lattice; and constrained or unconstrained aperture approximation). The lenses are 4λ in diameter and use $\frac{\lambda}{2}$ by $\frac{\lambda}{2}$ unit cells. For each prototype lens, the F/D and θ_0 are swept from 0.5 to 2 and 0^o to 45^o, respectively. Since some F/D and θ_0 combinations can lead to lens designs where the feed-sided antennas are outside their respective unit cell boundaries, in each simulation this physical limitation is verified. The loss is calculated for each source as the lens focuses the radiation onto a detector on the focal surface. The loss is the amount of collected power by the detector compared to the total power incident on the array aperture (the area of the lens aperture times



(a)



(b)

Figure 5.15. The average (a) and variance (b) for the loss of a $4\,\lambda$ lens with semi-directional antennas over all fifty sources showing the trends in loss as a function of F/D and θ_0.

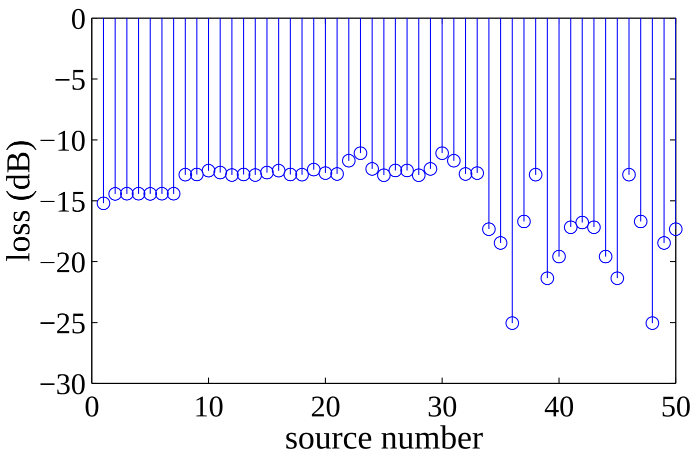
the incident power flux density). Therefore reduction in antenna element gain for scan angles off the optical axis is considered to be loss.

Since the amount of data obtained in this numerical experiment is large, only the trends will be discussed here. These trends are independent of the antenna elements used, size of the lens and its field of view. However, the optimal design for the lens (the optimal values for F/D and θ_0) are highly dependent on these parameters and the lens design needs to be optimize for each set of parameters. Figure 5.15(a) shows the average loss over the 50 sources as a function of F/D and θ_0 for a maximally packed unconstrained lens with semi-directional antennas. The loss dose not monotonically decrease as F/D increases but oscillates. This oscillation is not a result of standing waves between the lens array and detector, since the model does not include these effects. The oscillation most be a result of sinusoidal variation in the phase error as a function of F/D. For a constant F/D, the loss does monotonically decrease as θ_0 increases. The mean loss is minimize by having the smallest possible F/D and θ_0. This is equivalent to having the focal surface as close as possible to the lens, suggesting that path loss between lens and detector is the dominant loss mechanism. Simulations are performed where path loss and polarization loss are calculated independently. The simulations show that almost all the loss can be account for by the path loss, while the polarization loss contributes only 2% to 5% of the total loss.

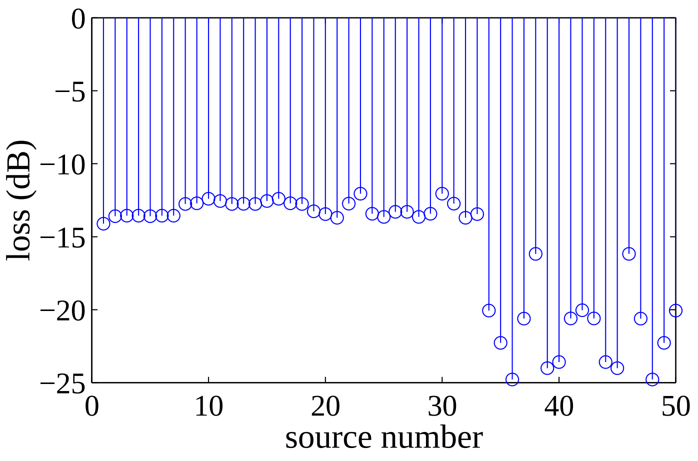
The variance of the loss over the 50 sources describes the how much the loss changes as a source moves off axis. Ideally, the loss should remain constant and the variance would be zero. The variance of the loss decreases as the cone of perfect focus moves away from the optical axis, Figure 5.15(b). This is consistent with the phase error for the unit cells being averaged over the field of view (section 5.1).

Assuming the phase error is a polynomial function of scan angle, Shelton [49] suggests that the optimal positions for the perfect focal points in a two-dimensional lens array occur at the roots of a Chebyshev polynomial whose order is the number of degrees of freedom. For example, a McGrath lens has two degrees of freedom, and the points of perfect focus should be placed at $\pm 45^o$ for minimum error up to 60^o off axis. Figures 5.16(a) and 5.16(b) show the loss for a $4\,\lambda$ lens with (F/D=.82, $\theta_0 = 14^o$) and (F/D=0.5, $\theta_0 = 45^o$), respectively. These plots illustrate that it is possible to design a lens with large or small θ_0 and still achieve low variance in loss if the F/D is chosen properly. It appears that the semi-directional lenses have less variation in their loss than patch-antennas lenses. This is probably due to the reduced gain of the patch antennas when the sources are far from the optical axis. It also appears that the patch antenna limits the useful scan angle of the lens to about $\pm 60^o$, independent of θ_0. Another interesting phenomenon is the variation in loss for sources that have the same angle from the optical axis (e.g. sources 19-33 that are 60^o of axis). The loss increases and decreases in a cosine fashion, showing the affect polarization loss in the system.

The lattice and aperture approximation seems to have little effect. However,



(a)



(b)

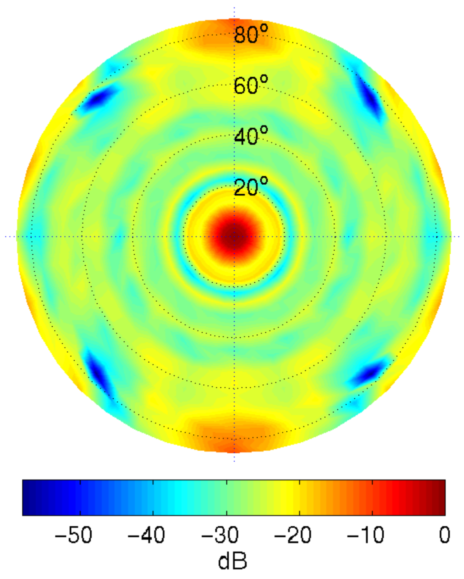
Figure 5.16. The loss for a $4\,\lambda$ diameter lens with semi-directional antennas and design parameters F/D=0.82 $\theta_0 = 14^o$ (a) and F/D=0.5 $\theta_0 = 45^o$ (b).

the larger the number of unit cells in the array, the more power it can collect, which argues for maximally packed unconstrained lens designs.

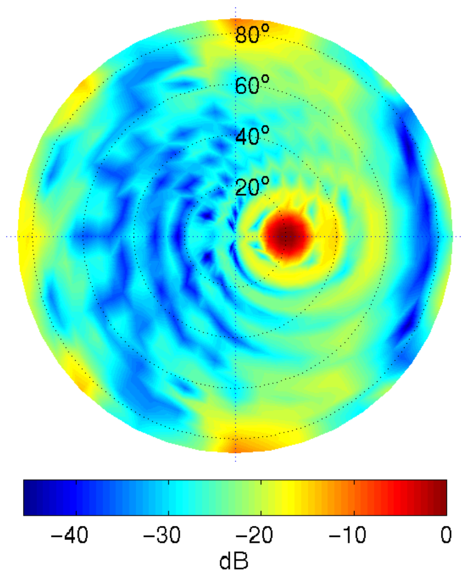
5.3.2 Lens Abberations The next experiment looks at aberration in the image for different lens designs as the source moves off axis. For an ideal lens, the image produced by a far-field source off axis should be identical to the one produced by a source on axis, only shifted. Aberrations are distortions of the image, as illustrated in Figure 5.17. The mean loss is minimized when F/d and θ_0. The aberrations can reduce the performance of a communication system by spreading the received power over a larger area on the focal surface. This will require more detectors to sample the image, which adds more noise sources to the angle diversity communication system and reduces the directivity of the lens.

Again, eight prototype $4\,\lambda$ lens designs with unit cell size of $\frac{\lambda}{2}$ by $\frac{\lambda}{2}$ are used. Each is a combination of the semi-directional or patch antennas, triangular or maximally packing lattices, and (F/D=0.8, $\theta_0 = 14^o$) or (F/D=0.5, $\theta_0 = 45^o$). For each of the prototype lenses, image patterns are calculated for each of the 50 sources. The half power beam radius (HPBR) for each pattern is calculated in eight different directions. The first lobe (ring) is also measured in a similar manner. The metrics of interest are: the average HPBR, the average radius of the first lobe ring and the ratio of the first lobe average power to the main beam power. Another metric is the average difference between the phase at the center of the main beam and the HPBR points. This metric is not important for adaptive systems, but has value for non-adaptive systems and this is discussed in the future work section 7.3.1.

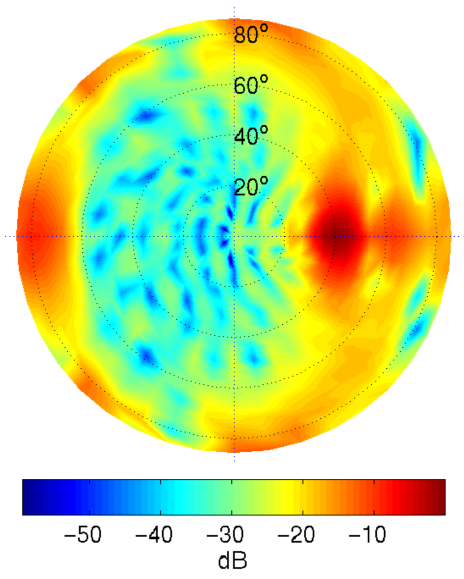
Tables 5.1, 5.2, 5.3, and 5.4 show the metrics for the lens designs. The percent standard deviation (STD) is the percent ratio of the STD to the mean. It gives an estimate of how much the HPBR and first lobe radius distort from the ideal circle. The percent STD for the HPBR phase is the ratio of the STD of delta phase at the HPBR points to 360^o. For a uniformly distributed phase from -180^o to 180^o,



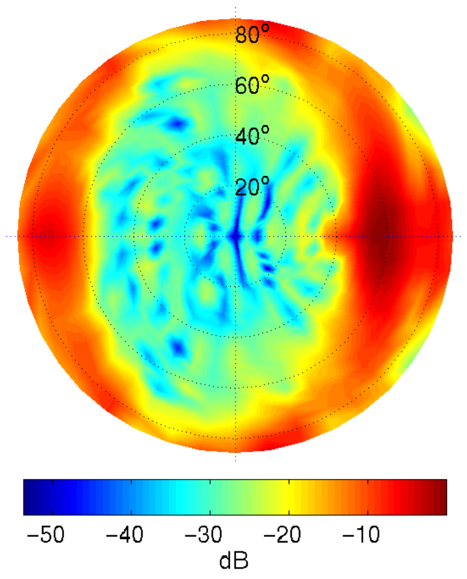
(a)



(b)



(c)



(d)

Figure 5.17. Image patterns for sources at 0^o (a), 20^o (b), 40^o (c) and 60^o (d) off axis showing the aberrations in the image.

Table 5.1. Metrics for F/D=0.82, $\theta_0 = 14^o$, and a triangular lattice using semi-directional antennas.

metric	0^o	20^o	40^o	60^o	80^o
Average HPBR (degrees)	8.3	8.6	9.50	10.76	9.88
STD (%)	2.2e-14	19.6	13.93	18.76	19.45
Average HPBR Phase	20	18.54	-148.6	1.78	-45
STD (%)	0	14.6	32	22.6	20.92
Average First Lobe Radius (degrees)	27.2	21.14	21.8	21.8	21.85
STD (%)	0	51.5	77.07	86.9	75.23
Average First Lobe Power (dB)	-18	-5.68	-4.10	-3.22	-4.1

Table 5.2. Metrics for F/D=0.5, $\theta_0 = 45^o$, and a triangular lattice using semi-directional antennas.

metric	0^o	20^o	40^o	60^o	80^o
Average HPBR (degrees)	7.9	8.34	9.5	11.84	10.32
STD (%)	0	21.93	24.27	40.34	37.34
Average HPBR Phase	3.5	-16.97	42.56	8.38	15.14
STD (%)	0	2.89	4.59	6.32	5.2
Average First Lobe Radius (degrees)	25.7	23.51	20.85	15.6	19.49
STD (%)	0	62.42	83.24	87.08	79.0
Average First Lobe Power (dB)	-25.77	-5.43	-2.86	-1.77	-3.13

Table 5.3. Metrics for F/D=0.82, $\theta_0 = 14^o$, and a maximally packed lattice using semi-directional antennas.

metric	0^o	20^o	40^o	60^o	80^o
Average HPBR (degrees)	7.84	8.32	9.2	10.58	9.6
STD (%)	0	20.24	13.2	18.52	19.7
Average HPBR Phase	18.1	17.37	-141.1	2.37	-42.47
STD (%)	0	14.03	32.58	22.32	20.43
Average First Lobe Radius (degrees)	25.7	22.83	21.9	19.04	21.06
STD (%)	0	38.57	75.12	92.25	73.53
Average First Lobe Power (dB)	-17.1	-7.32	-4.14	-2.80	-4.02

Table 5.4. Metrics for F/D=0.5, $\theta_0 = 45^o$, and a maximally packed lattice using semi-directional antennas.

metric	0^o	20^o	40^o	60^o	80^o
Average HPBR (degrees)	7.33	8.12	9.67	11.09	10.56
STD (%)	1.3e-14	23.54	23.4	41.13	37.48
Average HPBR Phase	1.07	-18.7	28.89	1.50	9.9
STD (%)	0	2.54	4	5.8	5.28
Average First Lobe Radius (degrees)	34	23.71	19.5	18.0	18.6
STD (%)	0	69.9	82.0	88. 32	86.0
Average First Lobe Power (dB)	-25.53 7	-4.81 7i	-2.71	-2.41	-2.48

the STD is 104^o (or 29%).

Several trends can be deduced from the data. First of all, the comparison between like lenses with triangular lattices versus maximally packed lattices, shows almost identical performances, again arguing for building lenses with as many unit cells as possible. All four lens have the same increase in average HPBR size, starting with about 16^o beamwidth at 0^o off axis to about 22^o beamwidth at 60^o off axis. However, the variance on the HPBR is lower for the lenses with $\theta_0 = 14^o$, especially for large scan angles. The phase stability for the lenses with $\theta = 45^o$ is nearly constant over scan angle, while the lenses with $\theta = 14^o$ appear to approach uniform distribution at large scan angles. The average power in the first side lobe approaches the power in the main lobe at large scan angles for all four lens designs, however the lenses with $\theta_0 = 45^o$ have better side lobe levels, especially at small scan angles. All four of the lenses have large variations in the first lobe power which is evident from Figure 5.17, where the first lobe power is larger on the side farthest from the center.

5.3.3 Lens Array Loss under Scaling We have seen in Section 5.3.1 that the loss in lens arrays can be large but it is also important to see if the loss in the lens arrays changes with size. The arguments in section 1.3 for improved SNR in communication systems relies on the assumption that the loss in a lens array does not increase with size. This assumption is based on a geometric argument for a lens operating in the paraxial region. Let us assume that there is an aperture antenna whose effective area equals its physical area and is $2x_0$ by $2y_0$, on the focal surface of a lens with diameter, D, Figure 5.18(a). The angular size of this antenna viewed from the lens is $\theta_{antenna} \approx 2\frac{x_0}{F} = 2\frac{x_0}{C_{onst}D}$, where $F/D = C_{onst}$. The angle that includes the HPBW spot focused on the lens is $\theta_{HPBW} \approx 2\frac{0.51\,\lambda}{D}$, [54], Figure 5.18(b). Both the aperture size and the focal spot size scale as the inverse of the lens diameter, resulting in loss independent of lens size.

Two prototype lenses with semi-directional antennas, $\frac{\lambda}{2}$ by $\frac{\lambda}{2}$ unit cells

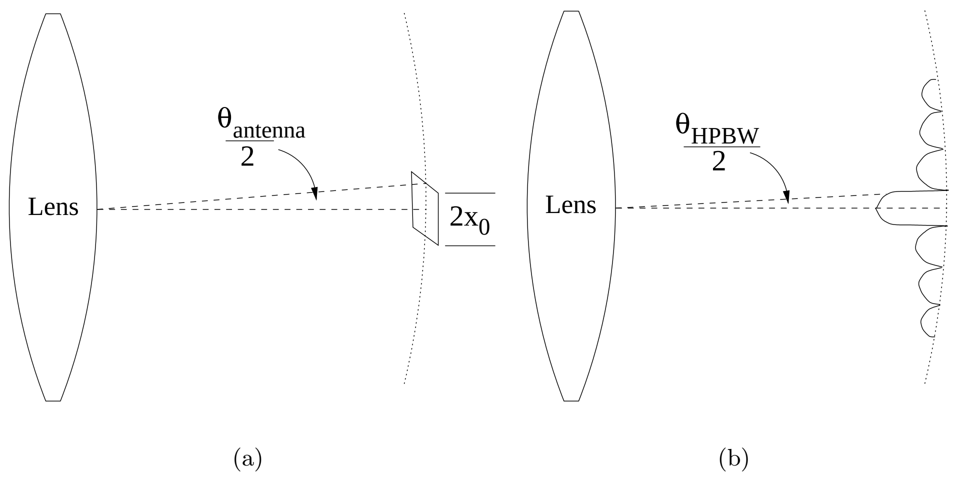


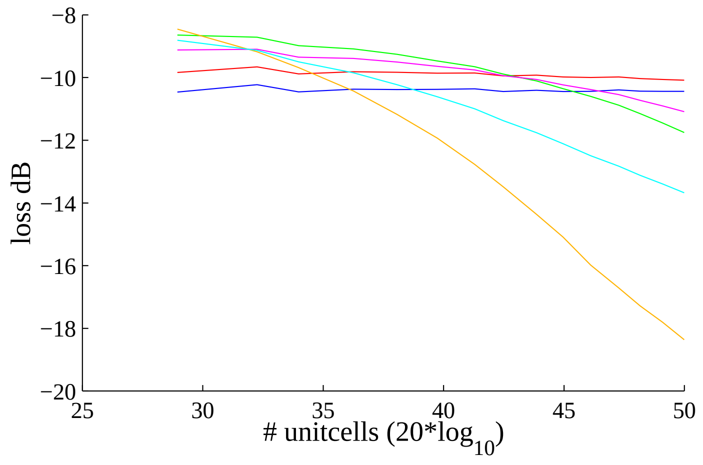
Figure 5.18. Illustration showing the angular size of an antenna on the focal surface (a) and the HPBW spot (b) as viewed from the lens.

spacing and, (F/D=0.8, $\theta_0 = 14^o$) or (F/D=0.5, $\theta_0 = 45^o$) are used to estimate the loss of a lens under scaling. The lenses are scaled from $3\,\lambda$ (35 unit cells) to $10\,\lambda$ (306 unit cells) in diameter. The metric of interest is how the loss scales with the number of unit cells, N_{cells}.

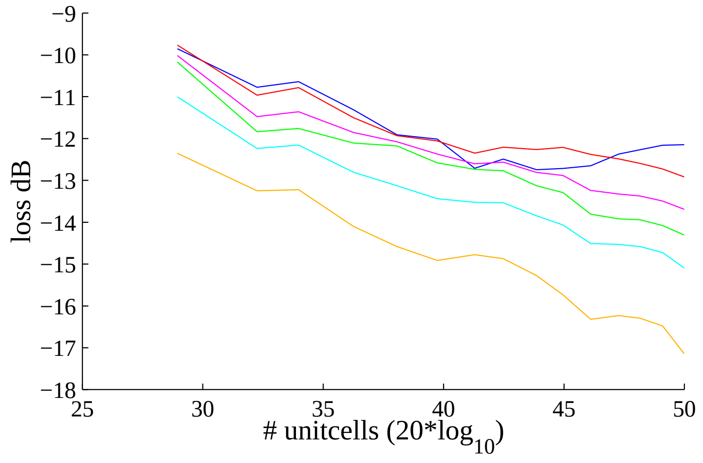
$$loss = N_{cells}^{\alpha}. \tag{5.39}$$

By taking $20log_{10}(\cdot)$ of both sides in Equation (5.39), α becomes the slope of the curve.

Figure 5.19 show the average loss for the lens sources. For the lens with F/D=0.8 and $\theta_0 = 14^o$, the loss is constant for sources within 20^o of the optical axis. For $\theta_0 = 14^o$, the phase error is minimized for angles less than 20^o from Shelton's approximation, showing the lens array to be a good approximation to a lens. For sources 20^o or more off axis, the loss scales as $N_{cells}^{0.11}$ to $N_{cells}^{0.4}$. For the lens with F/D=0.5 and $\theta_0 = 45^o$ the loss is increases for all source angles but at a smaller rate. The loss scales as $N_{cells}^{0.08}$ to $N_{cells}^{0.22}$. In both cases, the average scaling power over all sources within 60^o of the optical axis is $N_{cells}^{0.16}$.



(a)



(b)

Figure 5.19. The average loss for a lens with F/D=0.82 and $\theta_0 = 14^o$ (a) and F/D=0.5 and $\theta_0 = 45^o$ (b) lens array as it is scaled from $3\,\lambda$ to $10\,\lambda$. The source at 0^o is in blue, sources at 20^o are in red, sources at 40^o are in green, sources at 60^o are in yellow, sources from 0^o to 40^o are in magenta and sources from 0^o to 60^o are in cyan.

CHAPTER 6

SIMULATION OF BEAM-SPACE ADAPTIVE LENS ARRAY SYSTEMS

6.1 The LMS Algorithm

Adaptive arrays communication systems date back to 1950's the when Howells and Applebaum demonstrated the nulling of an interfering signal using a two element array [13]. Shortly afterwards, Widrow developed the LMS (Least Mean Squares) algorithm which is one of the simplest and most common algorithms used in adaptive systems. The LMS algorithm has been used in many different applications like adaptive arrays [18, 15], adaptive noise canceling [55], recursive filters [19], etc. LMS belongs to the class of algorithms that use gradient descent methods to find a minimum on an error surface. Two other popular algorithms, Newton's and steepest descent, also belong to the this class of algorithms.

An excellent tutorial on the LMS algorithm is give in [19], briefly repeated here for convenience. Let us assume that the LMS algorithm is applied to the adaptive array in Figure 6.1. Independent white noise, $n_{i,k}$, is added to each of the antenna signals, $\alpha_i d_k$. The subscript i is used to indicated the i-th signal and the subscript k is used to indicated the k-th sample in time. α_i is a complex coefficient used to describe the amplitude and phase changes of the original transmitted signal d_k that each antenna receives. For convenience, the received signals and noise are expresses in vector form, $d_k\mathbf{A}$, and $\mathbf{N_k}$, respectively, where a matrix or a vector are in bold face. The output of the adaptive array, y_k, is a linear weighted sum of the

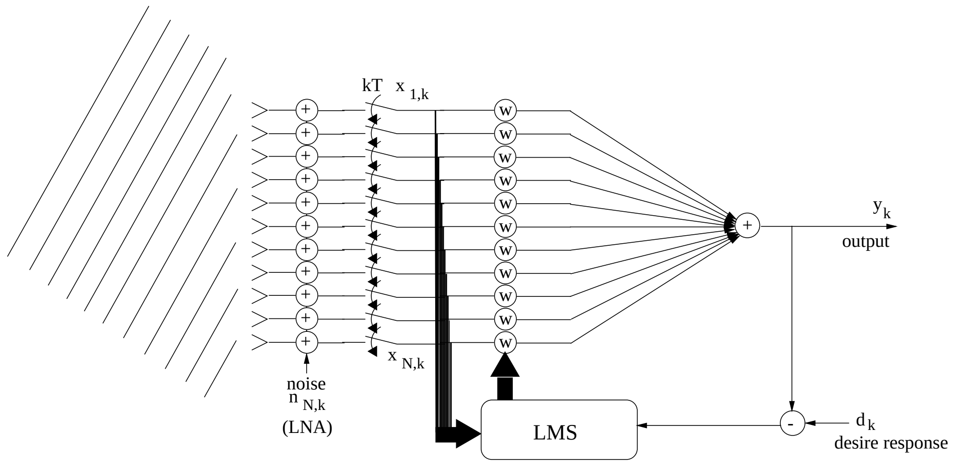


Figure 6.1. An example of a narrowband planar array adaptive receiver. The noise is added after each antenna element to model the effects of the LNAs and switches represent the sampling of the received signals with a sampling period of T.

received antenna signals and noise:

$$y_k = \mathbf{W_k}^* \mathbf{X_k}, \tag{6.1}$$

$$\mathbf{X_k} = d_k \mathbf{A} + \mathbf{N_k}, \tag{6.2}$$

The vector $\mathbf{W_k}$ is a vector contain the weights for the linear sum, while * denotes complex conjugate transpose. $\mathbf{W_k}, \mathbf{X_k}, \mathbf{N_k}$ and $\mathbf{A}$ are column vectors.

The LMS algorithm adjusts the weights such that the error,

$$e_k = d_k - y_k, \tag{6.3}$$

is minimized. The mean square error, $E[|e|_k^2]$, is a quadratic function of $\mathbf{W}$:

$$\zeta = E[|e|_k^2] = E[d_k^2] + \mathbf{W}^* \mathbf{R} \mathbf{W} - 2\mathbf{P}^* i \mathbf{W}, \tag{6.4}$$

where

$$\mathbf{R} = E[\mathbf{X_k} \mathbf{X_k^*}] \tag{6.5}$$

$$\mathbf{P} = E[d_k^* \mathbf{X_k}] \tag{6.6}$$

Here the $E[\cdot]$ denotes the expectation. Since the error function, ζ, is quadratic in $\mathbf{W}$, it has one minimum given by

$$\mathbf{W_{opt}} = \mathbf{R}^{-1}\mathbf{P} \tag{6.7}$$

which is sometimes known as the Wiener solution.

The LMS algorithm, just like the Newton Raphson and steepest descent algorithms, use an estimate of the gradient of the error function, ζ, to transverse down the slope to the local minimum. In the LMS case, the algorithm uses the instantaneous value for the squared error as the estimate for ζ. By taking the partial derivative of the $|e|_k^2$ in terms of weights, $\mathbf{W}$, the gradient is estimated to be

$$\widehat{\nabla_{\mathbf{k}}} = \frac{\partial |e|_k^2}{\partial \mathbf{W}} = -2e_k^* \mathbf{X_k} \tag{6.8}$$

The next set of weights is found by using the current set of weights and progressing in the negative gradient direction:

$$\begin{aligned} \mathbf{W_{k+1}} &= \mathbf{W_k} - \mu\widehat{\nabla} \\ &= \mathbf{W_k} - 2\mu e_k^* \mathbf{X_k} \end{aligned} \tag{6.9}$$

The constant μ is a gain constant that regulates the step size to insure convergence.

A convenient way to express the weights is in terms of the optimal weight solution and the weight error variable, $\mathbf{V_k}$,

$$\mathbf{W_k} = \mathbf{V_k} + \mathbf{W_{opt}} \tag{6.10}$$

and examine the expected decay of $\mathbf{V_k}$ by calculating and neglecting noise in the system

$$\begin{aligned} E[\mathbf{W_{k+1}}] &= E[\mathbf{W_k}] + 2\mu E[e_k \mathbf{X_k}] \\ &= E[\mathbf{W_k}] + 2\mu \left(E[d_k \mathbf{X_k}] - E[\mathbf{X_k}\mathbf{X_k^*}\mathbf{W_k}]\right) \\ &= E[\mathbf{W_k}] + 2\mu \left(\mathbf{P} - \mathbf{R}\, E[\mathbf{W_k}]\right) \\ &= \left(\mathbf{I} - 2\mu\mathbf{R}\right) E[\mathbf{W_k}] + 2\mu\mathbf{R}\mathbf{W_{opt}} \end{aligned} \tag{6.11}$$

Using the eigenvectors of $\mathbf{R}$ to rotate the system to its principal-axis coordinates (where the signal correlation matrix, $\mathbf{R}$, is diagonal) expressed using a primed variable, it can be shown that

$$\mathbf{V'_k} = (\mathbf{I} - 2\mu\mathbf{\Lambda})^k \, \mathbf{V_0}' \tag{6.12}$$

where Λ is the diagonal matrix containing the eigenvalues of $\mathbf{R}$ and $\mathbf{V_0}'$ is the initial weight vector. From Equation (6.12), the algorithm learns as a decay of a set of exponential functions. It shows that the mean vector converges slower for more weights. It also shows that for the mean vector to converge, the gain constant μ needs to be

$$0 < \mu < \frac{1}{\lambda_{max}}, \tag{6.13}$$

where λ_{max} is the largest eigenvalue of $\mathbf{R}$. Since the eigenvalues of $\mathbf{R}$ are difficult to calculate, μ is often set to be

$$\mu = \frac{\mu_0}{Tr[\mathbf{R}]} \tag{6.14}$$

where $Tr[\cdot]$ is the trace of $\mathbf{R}$. Since $Tr[\mathbf{R}]$ is the sum of the eigenvalues, this condition guarantees convergence.

Equation (6.12) shows the mean of the weights decay to the optimal solution. The noise in the system causes the weights to dither around the optimal value. The amount that the weights dither depends on the gain constant, μ. Let us assume the gradient estimation $\widehat{\nabla_\mathbf{k}}$ has independent noise added, $\mathbf{Z_k}$, such that

$$\widehat{\nabla_\mathbf{k}} = \nabla_\mathbf{k} + \mathbf{Z_k} \tag{6.15}$$

Assuming that the algorithm has already converged to steady-state solution, $\mathbf{W_{opt}}$, then $\nabla_\mathbf{k} = \mathbf{0}$. Then in accordance with Equations (6.15) and (6.8),

$$\mathbf{Z_k} = -2e_k^*\mathbf{X_k} \tag{6.16}$$

The covariance of the noise is given by

$$\begin{aligned} cov[\mathbf{Z_k}] =& E[\mathbf{Z_k}\mathbf{Z_k^*}] = 4E[|e|_k^2\mathbf{X_k}\mathbf{X_k^*}] \\ &\approx 4\zeta_{min}\mathbf{R} \end{aligned} \tag{6.17}$$

When the weights are near the optimal solution, the error e_k is approximately independent of the received signals, $\mathbf{X_k}$.

ζ_{min} is the minimum value for the error function given by

$$\zeta_{min} = E[|d|_k^2] - \mathbf{P}^*\mathbf{W_{opt}} \tag{6.18}$$

Using the eigenvector matrix of $\mathbf{R}$, denoted $\mathbf{Q}$, to rotate the system to the principal-axis coordinates, we have

$$\begin{aligned} E[\mathbf{Z_k}'] = cov[\mathbf{Z_k}'\mathbf{Z_k'}^*] =& E[\mathbf{Q}^{-1}\mathbf{Z_k}\mathbf{Z_k^*}\mathbf{Q}] \\ &\approx 4\zeta_{min}\mathbf{\Lambda} \end{aligned} \tag{6.19}$$

It can be shown that from Equation (6.19), the covariance of $\mathbf{V_k}$, (weight noise), is given by

$$\begin{aligned} cov[\mathbf{V_k}'\mathbf{V_k'}^*] &= \frac{\mu}{4}\left(\Lambda - \mu\Lambda^2\right)^{-1} E[\mathbf{Z_k}'\mathbf{Z_k'}^*] \\ &\approx \mu\zeta_{min}\left(\Lambda - \mu\Lambda^2\right)^{-1}\Lambda \\ &\approx \mu\zeta_{min}\mathbf{I} \end{aligned} \tag{6.20}$$

assuming μ is relatively small. From Equation (6.12), each weight relaxes with a time constant

$$\tau_i \approx \frac{1}{2\mu\lambda_i} \tag{6.21}$$

showing the natural tradeoff between the weight noise and adaptation rate.

6.2 Sampling the Lens Array Focal Surface

To insure that all the information contained in the image is preserved, the detectors are placed on the focal surface in accordance with the two-dimensional sampling theorem. The sampling theorem dictates the sampling period and pattern for a finite bandwidth function that is linear and shift invariant. Given a function $f(x, y)$ and its two-dimensional Fourier transform $F(\omega_x, \omega_y)$, then the sampled version of $f(x, y)$, $f(n, m)$, given by

$$f(n,m) = f(x,y)|_{(x,y)=\mathbf{V}(m,n)} \tag{6.22}$$

$$\tag{6.23}$$

where (m,n) samples are given by,

$$\begin{aligned} \begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} \mathbf{V} \end{bmatrix} \begin{bmatrix} n \\ m \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{V_1 V_2} \end{bmatrix} \begin{bmatrix} n \\ m \end{bmatrix} \end{aligned} \tag{6.24}$$

where $\mathbf{V}$ is a two dimensional matrix containing the vectors $\mathbf{V_1}, \mathbf{V_2}$ for the sampling pattern. An example of sampling is given in Figure 6.2 when $f(x, y)$ is a circle function, which is a hexagonal pattern. The Fourier transform of $f(n, m)$ is the sum of the aliased copies of $F(\omega_x, \omega_y)$ given by

$$\mathcal{F}\left\{f(n,m)\right\} = \frac{1}{|det\mathbf{V}|} \sum_{k,l} F\left(\omega_x - \omega_{x0}, \omega_y - \omega_{y0}\right) \tag{6.25}$$

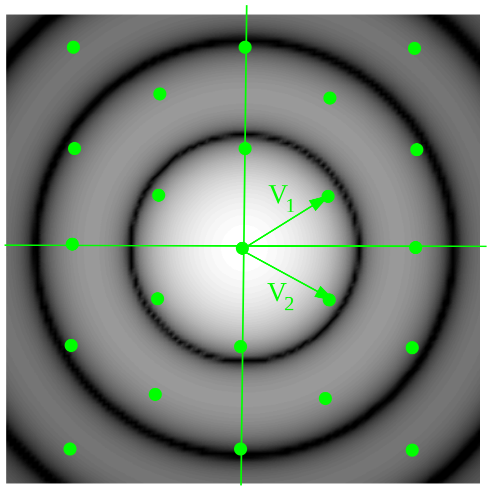
$$\begin{bmatrix} f_{x_o} \\ f_{y_o} \end{bmatrix} = \begin{bmatrix} \mathbf{U} \end{bmatrix} \begin{bmatrix} k \\ l \end{bmatrix}, \tag{6.26}$$

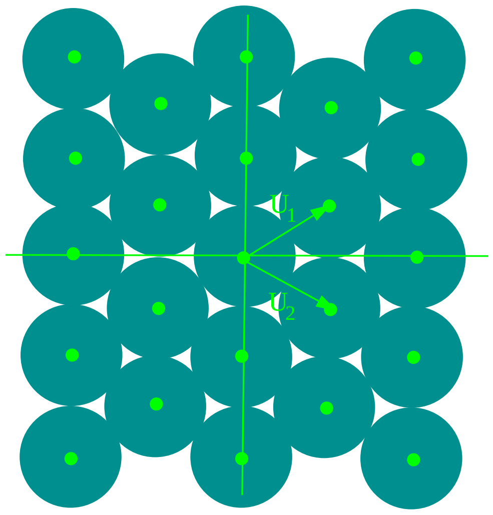
$$\tag{6.27}$$

where $\mathbf{V}$ is related to $\mathbf{U}$ as

$$\mathbf{U^T V} = 2\pi \mathbf{I} \tag{6.28}$$

From [54], a lens in its paraxial region approximates the two dimensional Fourier transform. If a plane wave is assumed to be the source, then the lens size and





(a) (b)

Figure 6.2. An example of two-dimensional hexagon sampling of an Airy pattern (a) and the aliases Fourier transform (b).

shape (pupil function) determine the image pattern. The lens shape is the inverse Fourier transform of the image pattern. The sampling vectors of the image must be short enough to prevent the aliased copies of the pupil function from overlapping.

For a rectangularly shaped lens, Figure 6.3, of size $2A_x$ by $2A_y$, the image intensity pattern is

$$\mathbf{I_0} \propto \text{sinc}^2\left(\frac{2A_x x}{\lambda z}\right)\text{sinc}^2\left(\frac{2A_y y}{\lambda z}\right) \tag{6.29}$$

The sample vectors for the rectangular lens are

$$\mathbf{V} = \begin{bmatrix} \frac{\pi\lambda z}{2A_x} & 0 \\ 0 & \frac{\pi\lambda z}{2A_y} \end{bmatrix} \tag{6.30}$$

Combining Equations (6.29) and (6.30) gives the samples for the image function

$$\begin{aligned} I_0(m,n) &\propto \text{sinc}^2\left(\frac{2A_x m\pi\lambda z}{2\lambda z A_x}\right)\text{sinc}^2\left(\frac{2A_y n\pi\lambda z}{2\lambda z A_x}\right) \\ &\propto \text{sinc}^2(\pi m)\,\text{sinc}^2(\pi n) \end{aligned} \tag{6.31}$$

For a source on the optical axis, the samples fall on the center of the main lobe and in its nulls, as shown in Figure 6.4.

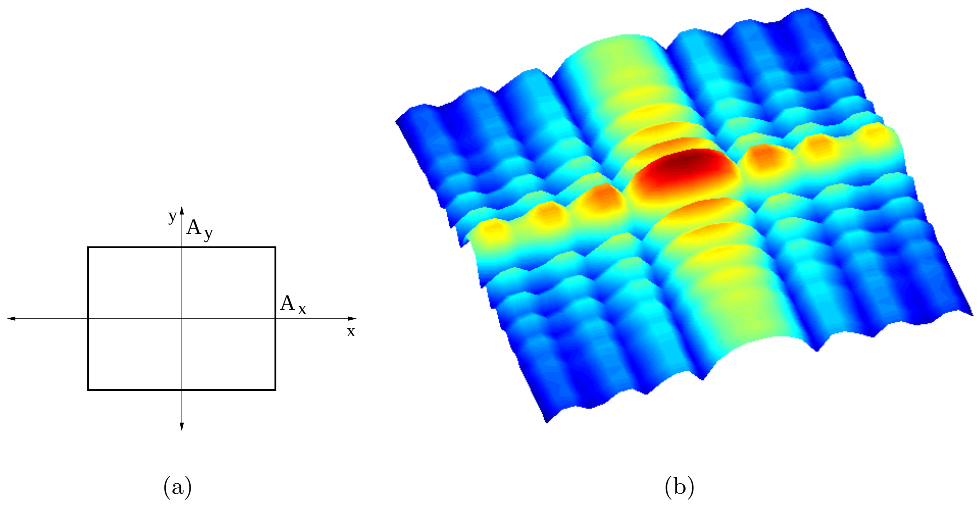


Figure 6.3. A rectangular aperture (a) and its crossed $sinc(x)$ Fourier transform (b).

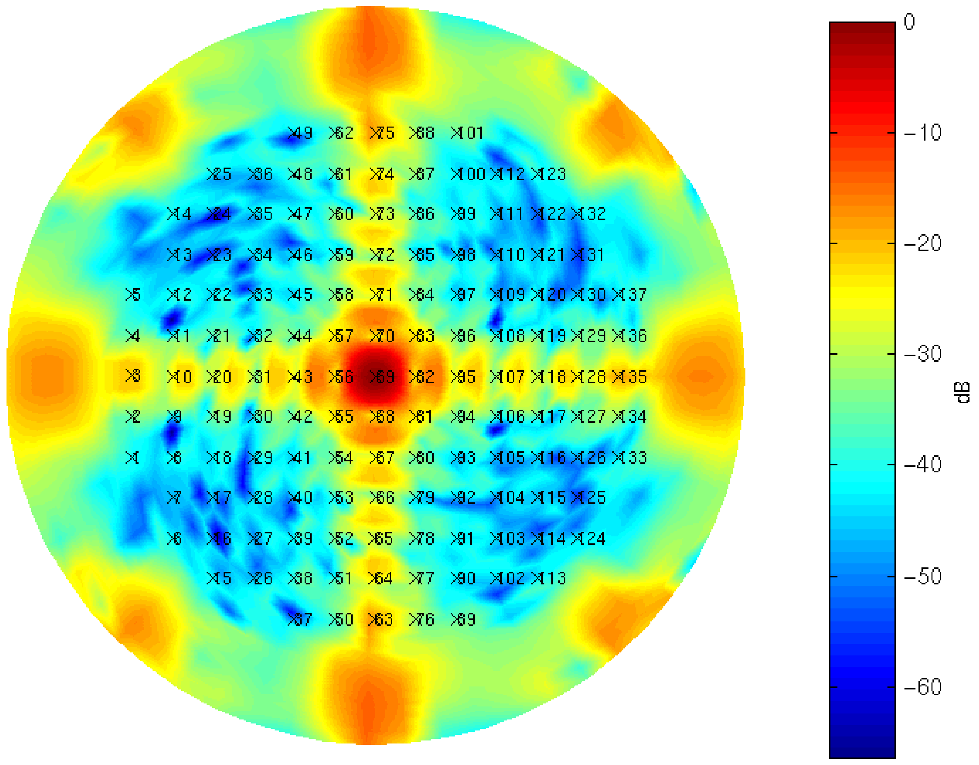


Figure 6.4. The sampling (numbered x's) of a rectangular aperture lens intensity image for a 60^o field of view.

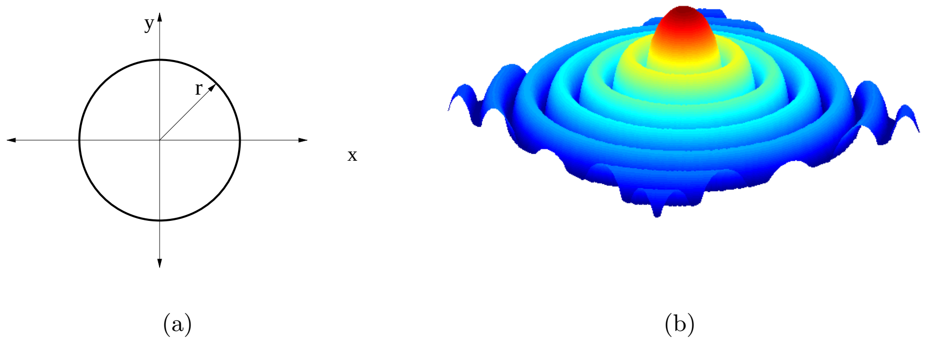


Figure 6.5: A circular aperture (a) and its Airy pattern Fourier transform (b).

For a circular lens with radius r the image pattern is

$$I_0(\rho) \propto \frac{2J_1(2\pi r\rho)}{2\pi r\rho}\rho = \sqrt{x^2+y^2} \tag{6.32}$$

and J_1 is the Bessel function of the first kind, Figure 6.5. Equation (6.32) is referred to as the Airy pattern. A hexagonal pattern is the optimal sampling pattern that minimizes the number of samples [56] for a circular aperture with sampling vectors

$$\mathbf{V} = \frac{1}{2r}\begin{bmatrix} 1 & \frac{1}{\sqrt{3}} \\ 1 & -\frac{1}{\sqrt{3}} \end{bmatrix} \tag{6.33}$$

For a source on the optical axis, the samples also fall on the center of the main lobe and near the nulls, Figure 6.6. It is interesting to note that for both the rectangular and circular lenses, the ratio of the HPBW and the sampling vector lengths is 1.13.

Simulations of the lens array image patterns agree well with Equations (6.29) and (6.32) in the paraxial region, however, it is important to point out that not all of the conditions for the sampling theorem are met by the lens array system: the lens system is noisy, and the aberration in the lens cause the system to be shift-variant. The noise has theoretically infinite bandwidth which contradicts the finite bandwidth condition. However, the noise in the simulations is added after

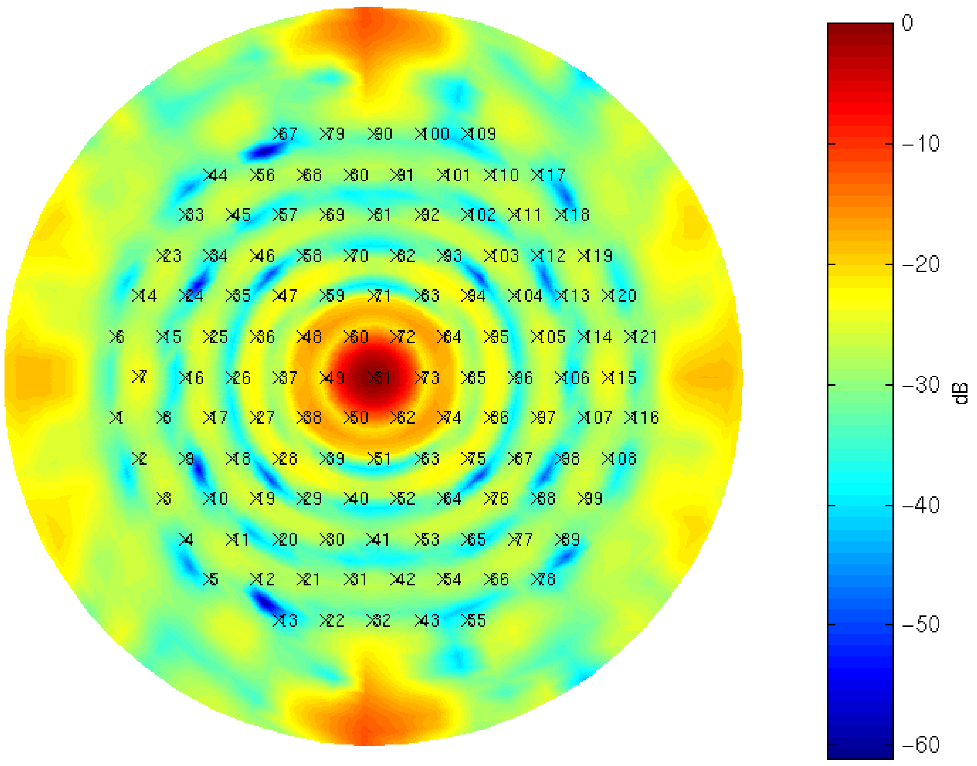


Figure 6.6. The sampling (numbered x's) of a circular aperture lens intensity image for a 60^o field of view.

the image for the desired signal is formed, and it is the information in the image pertaining to the desired signal that needs to be retained. The beamwidth of the image increases as the source moves off axis, suggesting that the effective bandwidth of the system decreases. Therefore the detectors oversample the image for signals far off the optical axis. The detectors are placed in an hexagonal pattern on the projection of the focal surface onto a disk, Figure 6.6. When the position of the detectors is inverse transformed back to the focal surface, the outer detectors end up closer together than those in the center, resulting in over sampling of the sources that are far off axis. This projection results in the detectors being placed closer together on the edges of the disk than in the center of the disk, Figure 6.7, again causing over sampling of signals far of the optical axis.

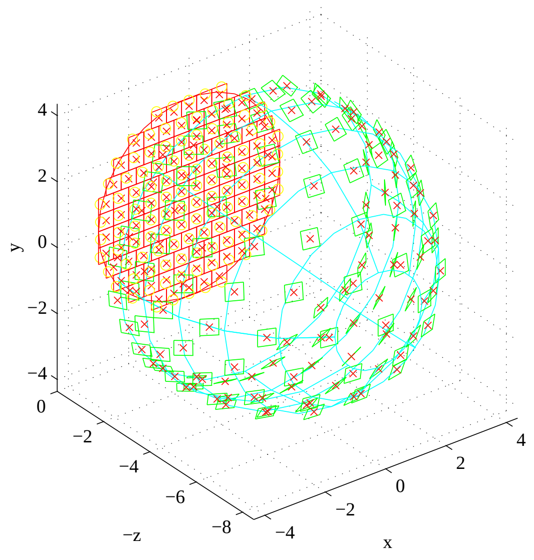


Figure 6.7: An example of a 6 λ diameter lens with a sampled focal surface

6.3 Numerical Experiments on Lens Array Systems

The goal of using a lens array as the front end for an adaptive array system is to show that a lens array system has better SNR under scaling and adapts faster since it requires fewer weights. A set of numerical experiments is performed to compare a lens array to a planar array in an adaptive system. For the lens array system, the signals for the detectors are ranked from strongest SNR to weakest SNR. Then the lens system is simulated using the strongest signal, followed by the two strongest signals, etc. For simplicity, the LMS training signal d_k is set equal to the transmitted data.

The output SNR is the principle metric for a communication system. Three different techniques are used to estimate the output SNR of the system. The first is a measure of the bit error rate (BER). This is the average number of wrong guesses a system makes in decoding the information . These experiments use QPSK complex baseband modulated data for the transmitted signals and the algorithm samples at twice the symbol rate. For QPSK modulation, the decisions boundaries are shown in Figure 6.8. The second technique is to estimate the noise in the output signal as the

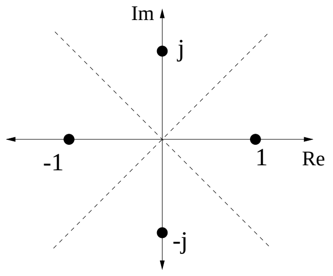


Figure 6.8: QPSK modulation and slicer decision regions.

part of the output signal that is not the transmitted data, d_k. From Equation (6.3), the algorithm tries to make the output signal look like the training signal (data). Therefore the noise at the output is

$$y_{noise,k} = y_k - d_k \tag{6.34}$$

Then the SNR is given by

$$SNR_{data} = \frac{var[d_k] + E[d_k]^2}{var[y_{noise,k}] + E[y_{noise,k}]^2} \tag{6.35}$$

For unit power QPSK modulation, the $var[d_k] = 1$, $E[d_k] = 0$ and equation 6.35 reduces to

$$SNR_{data} = \frac{1}{var[y_{noise,k}] + E[y_{noise,k}]^2} \tag{6.36}$$

The third technique is based on representing the output as the product of the received signals and the weights:

$$y_k = \mathbf{W_k}^* \mathbf{X_k} \tag{6.37}$$

$$= (\mathbf{W_{opt}} + \mathbf{V_k})^* (d_k \mathbf{A} + \mathbf{N_k}) \tag{6.38}$$

$$= \mathbf{W_{opt}}^* d_k \mathbf{A} + \mathbf{V_k}^* d_k \mathbf{A} + \mathbf{W_{opt}}^* \mathbf{N_k} + \mathbf{V_k}^* \mathbf{N_k}, \tag{6.39}$$

where the first term is the signal and the last three terms are the noise. This SNR is calculated in two different ways. One uses the calculated $\mathbf{W_{opt}}$ given by Equation (6.7) and the second uses the estimate of the optimal weights, $\widehat{\mathbf{W_{opt}}} = E[\mathbf{W_k}]$, as in Equation (6.12). Then the noise is calculated as

$$y_{noise,k} = y_k - \mathbf{W^*_{opt}} d_k \mathbf{A} \tag{6.40}$$

$$y_{\widehat{noise},k} = y_k - \widehat{\mathbf{W^*_{opt}}} d_k \mathbf{A} \tag{6.41}$$

and the SNR is

$$\begin{aligned} SNR_{\mathbf{W_{opt}}} &= \frac{cov[\mathbf{W^*_{opt}} d_k \mathbf{A}] + E[\mathbf{W^*_{opt}} d_k \mathbf{A}]^2}{cov[y_{noise,k}] + E[y_{noise,k}]^2} \\ &= \frac{\left|\mathbf{W^*_{opt}} \mathbf{A}\right|^2}{cov[y_{noise,k}] + E[y_{noise,k}]^2} \end{aligned} \tag{6.42}$$

$$\begin{aligned} SNR_{\widehat{\mathbf{W^*_{opt}}}} &= \frac{cov[\widehat{\mathbf{W^*_{opt}}} d_k \mathbf{A}] + E[\widehat{\mathbf{W^*_{opt}}} d_k \mathbf{A}]^2}{cov[y_{\widehat{noise},k}] + E[y_{\widehat{noise},k}]^2} \\ &= \frac{\left|\widehat{\mathbf{W^*_{opt}}} \mathbf{A}\right|^2}{cov[y_{\widehat{noise},k}] + E[y_{\widehat{noise},k}]^2} \end{aligned} \tag{6.43}$$

for QPSK modulation. As a comparison for the algorithms performance, the theoretical output $y_{k,opt}$ is calculated using the optimal weights, $\mathbf{W_{opt}}$, as in Equation (6.1).

6.3.1 SNR vs. Number of Signals The first experiment focuses on the change in SNR as more received signals are added to the lens array system, or in other words, more detector signals are used. Given the arguments in section 1.3, the SNR should increase as more signals are added, until the SNR of the next signal is so small that it reduces the system performance. From Equation (6.39), as each signal is added to the system, the vectors $\mathbf{W_{opt}}$, $\mathbf{A}$, $\mathbf{V_k}$ and $\mathbf{N_k}$ increase by one. Since the signals are added in decreasing order of SNR, at some point the added value to $\mathbf{A}$ is very small. Since the value added to $\mathbf{W_{opt}}$ is proportional to the value added to $\mathbf{A}$, Equation (6.7), the new value in $\mathbf{W_{opt}}$ is also very small. Thus the first three terms in Equation (6.39) do not change significantly for the added signal. However, the fourth term is independent of $\mathbf{A}$ and should add a significant amount of noise to the system.

The simulations are performed using a 6λ diameter lens that has 110 unit

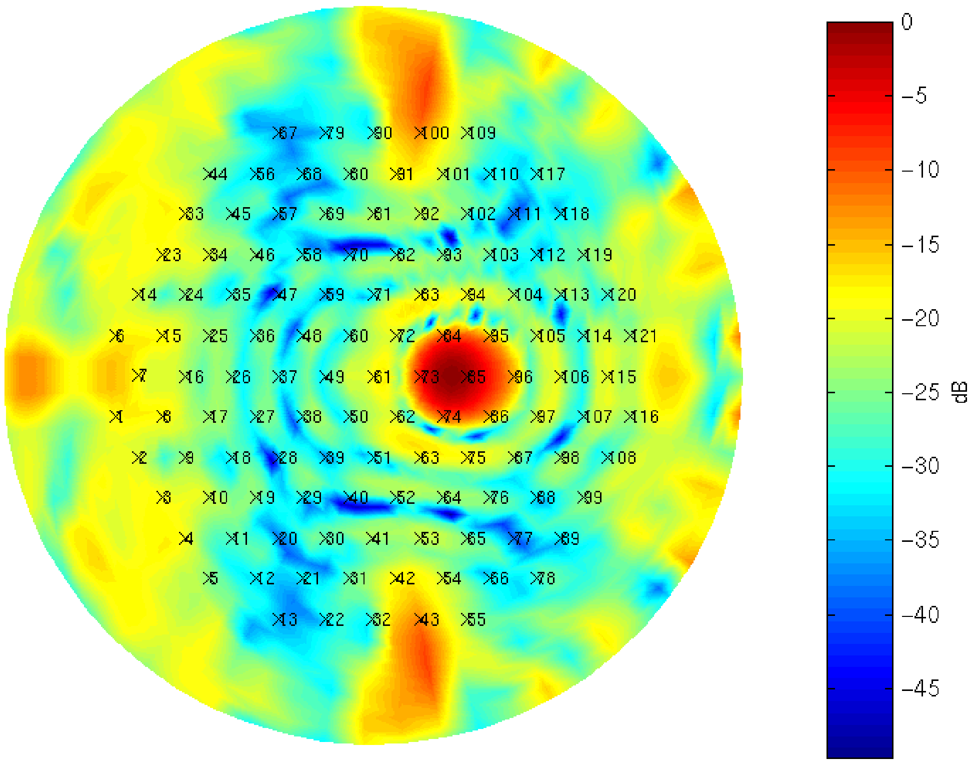


Figure 6.9. Sampling of a 6 λ diameter circular lens with a source 20^o off optical axis.

cells. Semi-directional antennas are used, $F/D = 0.5$, and $\theta_0 = 45^o$ to minimize aberrations. The source of the signal is 20^o off axis with detectors placed as shown in Figure 6.9. Figure 6.10 shows the distribution of normalized received signal power across all the detectors. The only noise sources in the system are at the detectors. Simulations are run such that the SNR of the peak received signal at the detector, $SNR_{\mathbf{X}}$, varies from 0.1 dB to 10 dB, by adjusting the power of the noise sources. The gain constant for the system is determine by Equation (6.14), with $\mu_0 = 0.05$.

Figure 6.11 to 6.13 show the simulated BER, SNR_{data} and $SNR_{\mathbf{W}}$ for the different cases of $SNR_{\mathbf{X}}$. Under all three metrics, the system performance improves as the $SNR_{\mathbf{X}}$ increases. This seems like an obvious result, but it gives important insight to the operation of the Wiener solution. In the Wiener solution, the weights

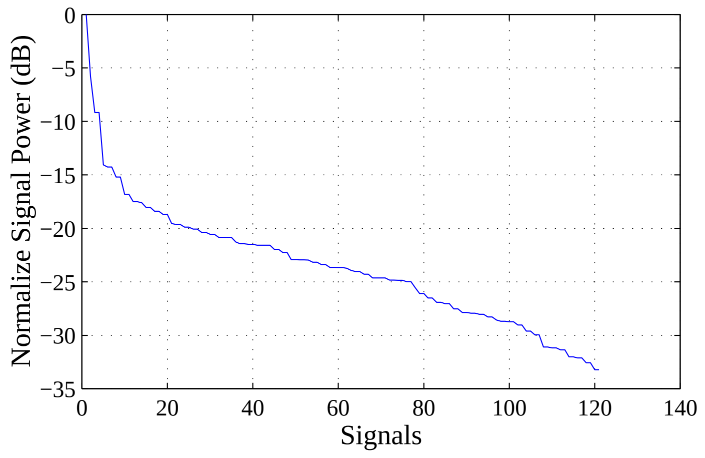


Figure 6.10. The distribution of received signal power across the 121 detectors for the image in Figure 6.9.

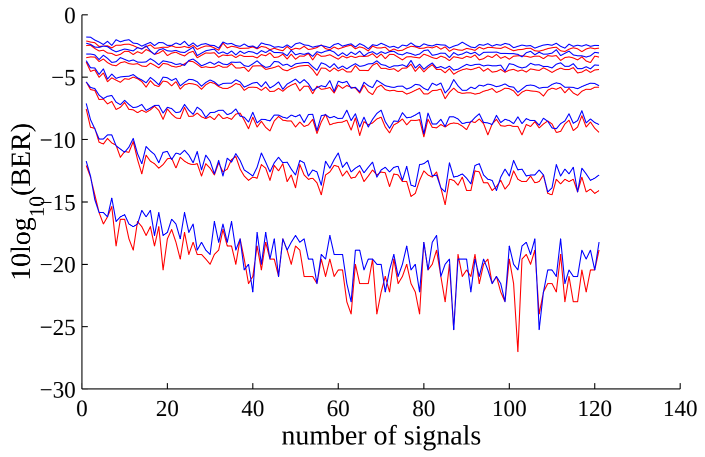


Figure 6.11. The BER for $SNR_{\mathbf{X}}$ 0.1, 0.178, 0.316, 1, 1.78, and 3.162 (top to bottom). The BER for the adaptive lens array systems is shown in blue and BER for the optimal weight solution is shown in red. The noise in the data is due to different noise and data realizations for the different simulations.

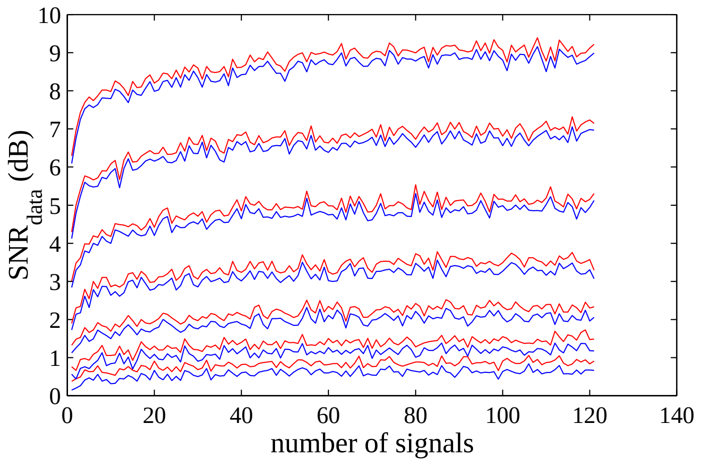


Figure 6.12. The SNR_{data}, Equation (6.35), for $SNR_{\mathbf{X}}$=0.1, 0.178, 0.316, 1, 1.78, and 3.162 (top to bottom). The adaptive lens array systems is shown in blue and the optimal weight solution is shown in red.

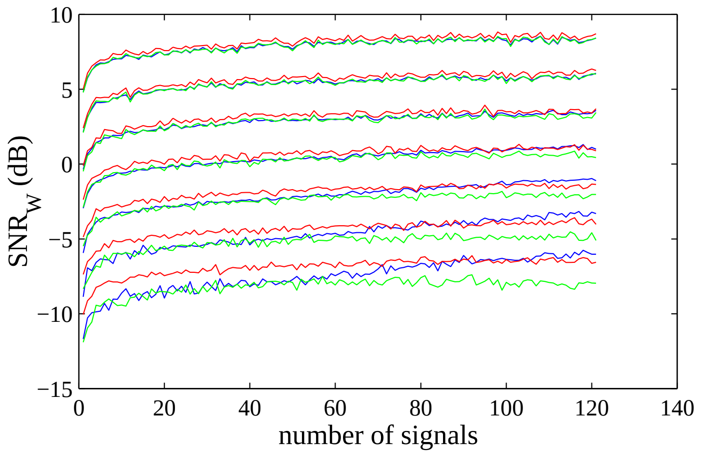


Figure 6.13. The $SNR_{\mathbf{W}}$ for $SNR_{\mathbf{X}}$= 0.1, 0.178, 0.316, 1 1.78, and 3.162 (top to bottom). The $SNR_{\mathbf{W_{opt}}}$ (6.42) for lens array systems is shown in blue, the $SNR_{\widehat{\mathbf{W_{opt}}}}$ (6.43) is shown in green and the optimal weight solution is shown in red.

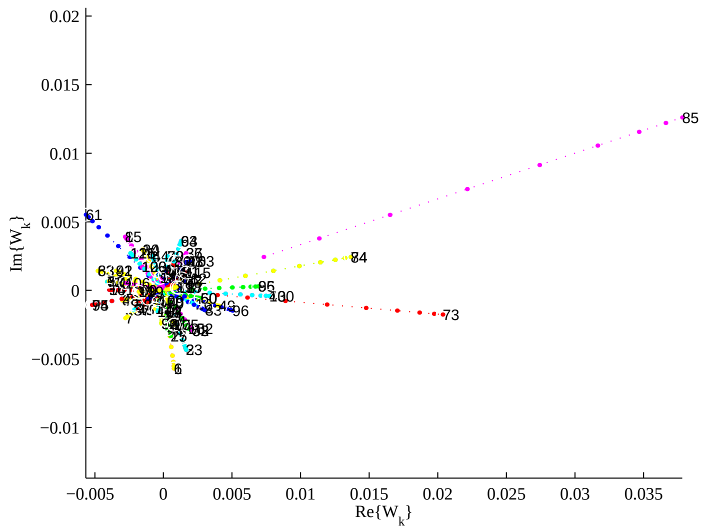


Figure 6.14. The Wiener solution for the weights with received signal in Figure 6.9 and $SNR_{\mathbf{X}}$=0.1, 0.178, 0.316, 0.562, 1, 1.78, 3.162, 5.62, and 10. Each weight track starts with the lowest $SNR_{\mathbf{X}}$ and ends with the largest $SNR_{\mathbf{X}}$ at the weight number.

are a scaled and decorrelated ($\mathbf{R^{-1}}$) version of the received desired signal ($\mathbf{P}$). The scaling of the weights depends on the relative magnitudes of the noise power to the signal power, and the received signals is found as

$$R = E\left[\mathbf{X_k}\mathbf{X_k}^*\right] \tag{6.44}$$

$$= E\left[\left(d_k\mathbf{A} + \mathbf{N_k}\right)\left(d_k\mathbf{A} + \mathbf{N_k}\right)^*\right] \tag{6.45}$$

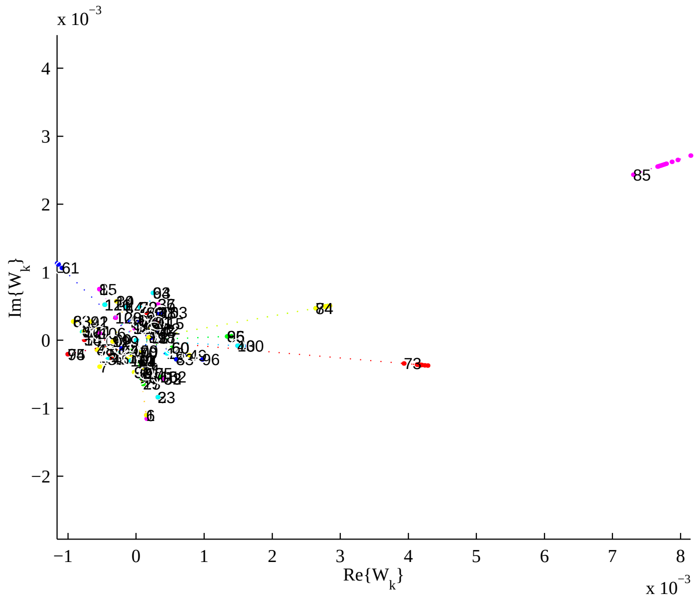
$$= \sigma^2_{signal}\mathbf{A}\mathbf{A}^* + \sigma_{noise}\mathbf{I} \tag{6.46}$$

$$= \mathbf{Q}^{-1}\left(\sigma^2_{signal}\mathbf{\Lambda_A} + \sigma^2_{noise}\mathbf{I}\right)\mathbf{Q} \tag{6.47}$$

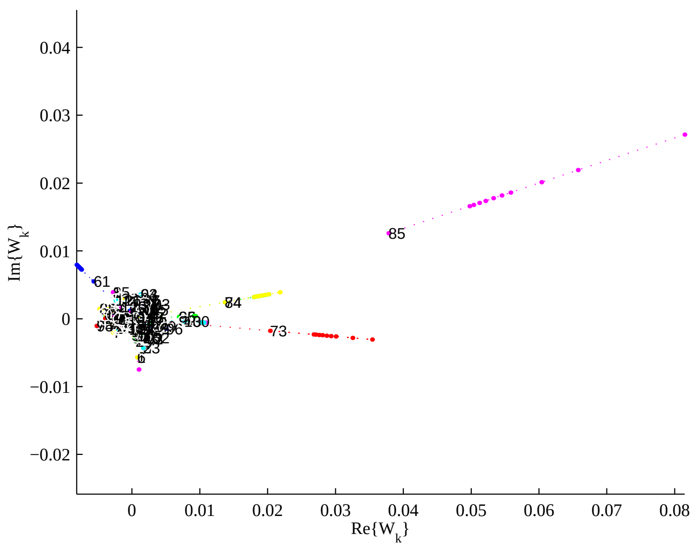
For a strong input $SNR_{\mathbf{X}}$, the weights illuminated by the desired signal have the largest magnitude. However, as the noise in the system increases, the weights adjust inwards to minimize the amount of noise at the output, Equation (6.47) as illustrated by Figure 6.14. The measured performance of the adaptive system is slightly less that that of the optimal weight system showing the effects of the weight noise in the system. Since the difference between the optimal weight system SNR and the adaptive weight system SNR is small, the weight noise does not contribute considerable noise to the output, for step size, $\mu = \frac{0.05}{Tr[\mathbf{R}]}$.

In all cases, the system performance improves as more signals are added to the system, which is expected. The added signal and noise powers cause the weights' magnitude to adjust inwards. Figure 6.15 shows the change in the optimal weights for the $SNR_{\mathbf{X}} = 0.1$ and $SNR_{\mathbf{X}} = 10$ as more signals are added to the system. In the $SNR_{\mathbf{X}} = 10$ case, the signal power dominates the weight scaling when a few signals are used, but the noise quickly begins to dominate. This is why the high $SNR_{\mathbf{X}}$ case weights scale faster than the low $SNR_{\mathbf{X}}$ case where the noise always dominates. The weights are constantly being changed as the additional signals are added so that the output SNR is a monotonically increasing function. Another way of looking at the weights are diagrams for the output signals. For a system with good SNR, the output signal should form clusters on the modulation constellation diagrams, Figure 6.16. As the noise increases, the center of the clusters move inwards. Figure 6.17 shows the centers of the clusters and there STD radii for various $SNR_{\mathbf{X}}$. It is interesting to point out that the STD radius do not vary significantly for different numbers of signals used.

Figure 6.18 shows the weight noise power for the different $SNR_{\mathbf{X}}$ cases and variance on the weight noise power which agrees well with calculations based on Equations (6.20), Figure 6.19. The simulated noise power is about 7 dB lower than that calculated and may be related to the fact that the samples of the modulated data are not a uniformly distributed, but are cyclostationary [57, 58]. It is interesting to point out that the weight noise increases for larger $SNR_{\mathbf{X}}$. Since the increased $SNR_{\mathbf{X}}$ means a decrease in total noise power and input power for the system, Equation (6.5), the gain constant is larger, Equation (6.14), accounting for the increase in weight noise. The variation in weight noise between the weights is small and consistent with theory. These simulations still raise the question: "Is it possible to have a signal whose SNR is so small that the addition of this signal to



(a)



(b)

Figure 6.15. The Wiener solution for the weights with received signal in Figure 6.9 with $SNR_{\mathbf{X}} = 0.1$ (a) and $SNR_{\mathbf{X}} = 10$ (b) as signals are added to the adaptive lens system. Each point in the weight tracks represents the Wiener solution for 1, 2, 3,…, 10, all (121) received signals used, starting with 1 signals and ending with all signals (the numbered point).

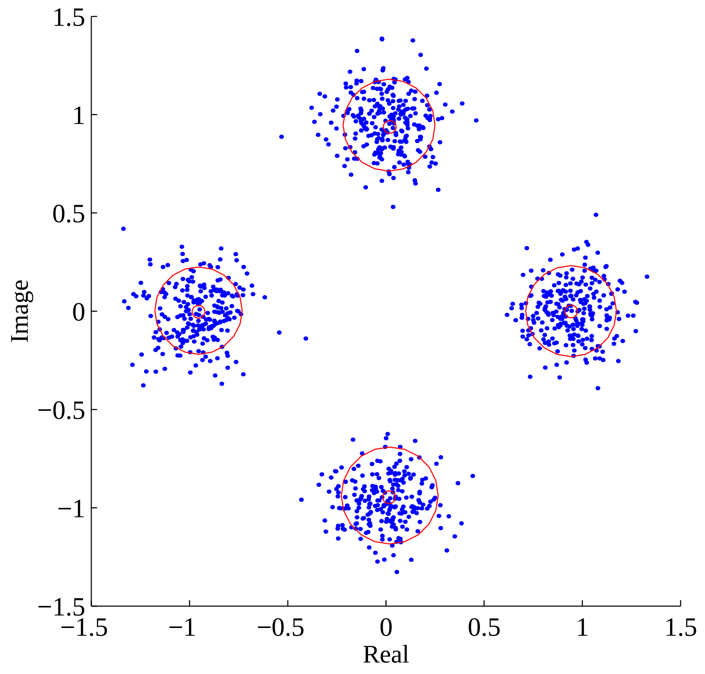


Figure 6.16: Eye diagram for adaptive lens array system with $SNR_{\mathbf{X}} = 10$.

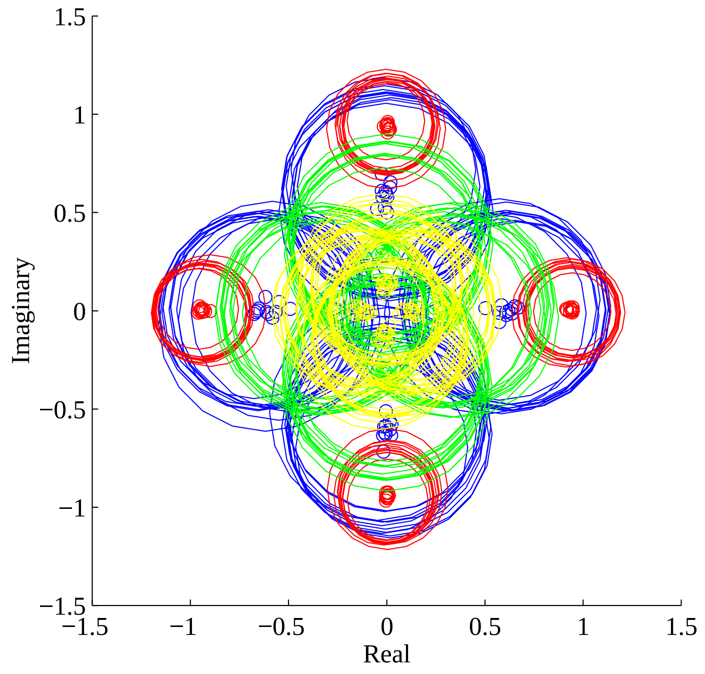
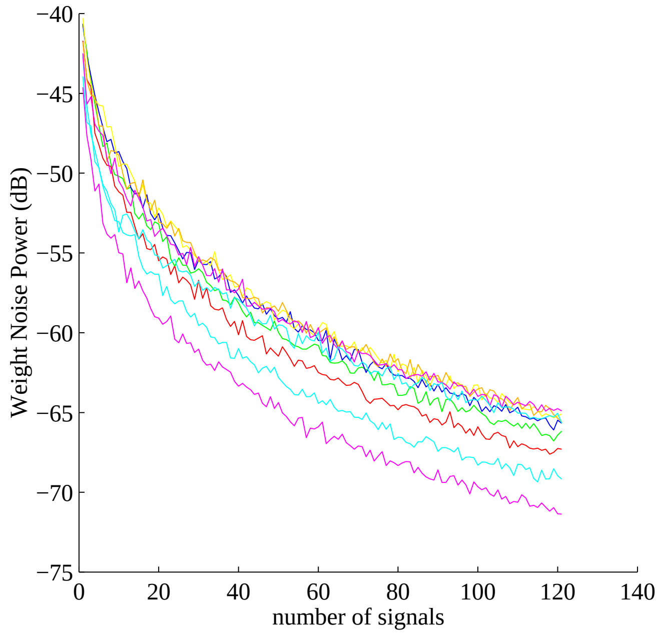
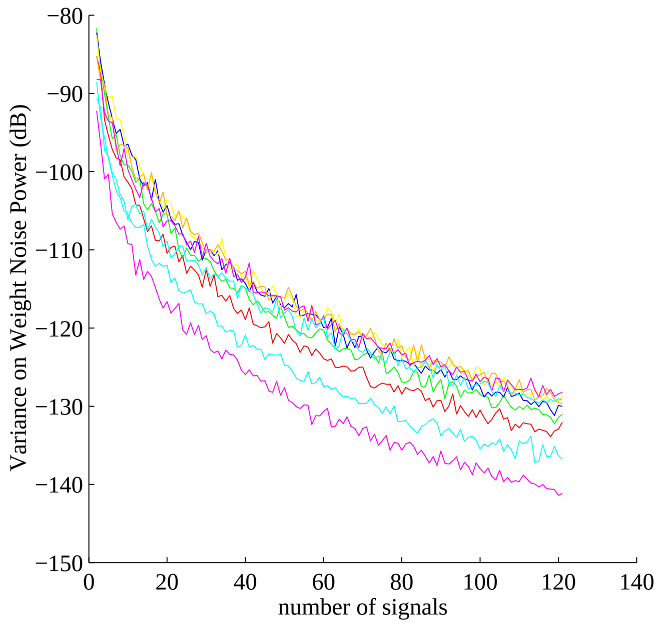


Figure 6.17. Eye diagram for adaptive lens array system with $SNR_{\mathbf{X}} = 10$ (red), $SNR_{\mathbf{X}} = 1$ (blue), $SNR_{\mathbf{X}} = 0.3$ (green) and $SNR_{\mathbf{X}} = 0.1$ (yellow). The centers (o's) and STD rings are plotted for 1, 2, 3,. . . , 10 and all (121) signals used.

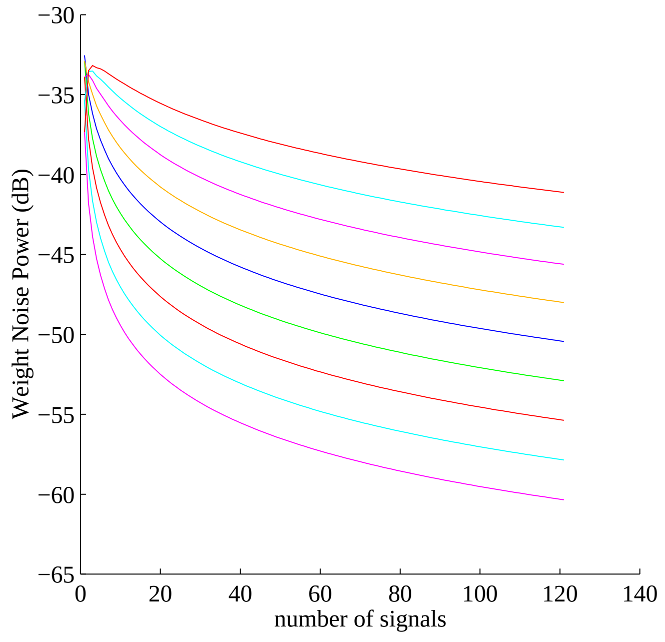


(a)

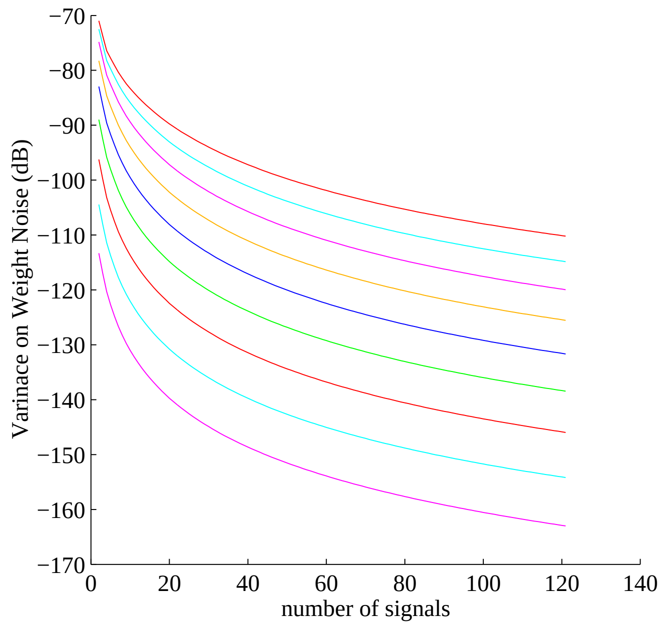


(b)

Figure 6.18. The measured weight noise (a) and variation in weight noise between the weights (b) for $SNR_{\mathbf{X}} = 0.1$ (magenta), 0.17 (cyan), 0.32 (red), 0.56 (green), 1 (blue), 1.78 (yellow), 3.162 (magenta), 5.6 (cyan) and 10 (red).



(a)



(b)

Figure 6.19. The calculated weight noise (a) and variation in weight noise between the weights (b) for $SNR_{\mathbf{X}} = 0.1$ (magenta), 0.17 (cyan), 0.32 (red), 0.56 (green), 1 (blue), 1.78 (yellow), 3.162 (magenta), 5.6 (cyan) and 10 (red).

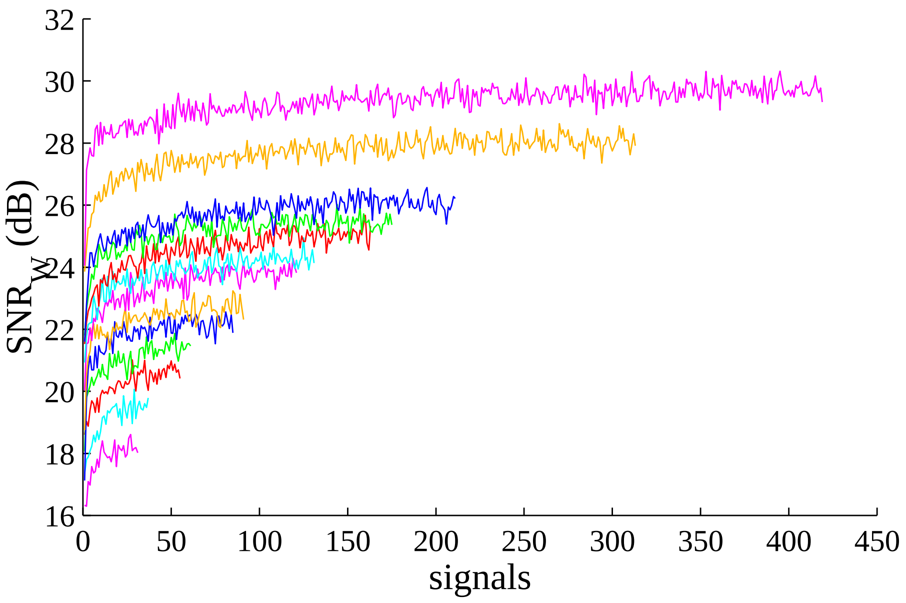


Figure 6.20. The $SNR_{\widehat{\mathbf{W}_{\mathbf{opt}}}}$ for increasing lens array diameters ($3\,\lambda$, $3.5\,\lambda$, $4\,\lambda$, $4.5\,\lambda$, $5\,\lambda$, $5.5\,\lambda$, $6\,\lambda$, $6.5\,\lambda$, $7\,\lambda$, $7.5\,\lambda$, $8\,\lambda$, $10\,\lambda$ and $12\,\lambda$, bottom to top).

the system would reduce the overall performance?". In these simulations, the signals have a range of 30 dB in signal power and the output SNR appears to be a monotonically increasing function. Since the step size changes in accordance to the amount of signal and noise power, the algorithm is still able to estimate the signal components in all of the detectors. We can see the improved estimation of the signals by the reduction in weight noise, Figure 6.18.

6.3.2 SNR vs. Lens Array Size The next experiment focuses on the change in output SNR as the lens size increases. The same basic setup is used in the simulations as in section 6.3.1 with a lens of $F/D = 0.5$, $\theta_0 = 45^o$ and a diameter varying from 3λ (28 unit cells) to 12λ (454 unit cells). The detector SNR is set at $SNR_{\mathbf{X}} = 112$. Figure 6.20 shows each system $SNR_{\widehat{\mathbf{W}}}$ as the number of signals increases. These curves have the same monotonically increasing nature as those in section 6.3.1. Figure 6.21 shows the maximum SNR for the lens array and the planar array. It is expected that the lens array SNR would grow faster than the

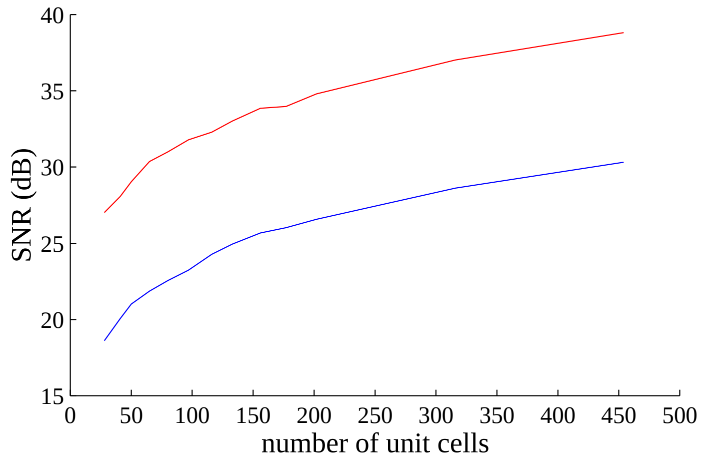


Figure 6.21. Maximum $SNR_{\widehat{\mathbf{W_{opt}}}}$ for adaptive lens array (blue) and planar array (red).

SNR for planar array as the lens size increases. Instead, the difference between the two curves appears constant, which suggests the loss increasing with size does have a large effect on performance, Section 5.3.3.

6.3.3 Adaptation Rate Vs. Number of Signals The final asserted advantage of the lens arrays in adaptive systems is its increased adaptation rate due to fewer weights. The same setup is used as in section 6.3.1 to estimate the adaptation rates for a lens array with $SNR_{\mathbf{X}} = 1, 3 and 10$. For each configuration of the system, 500 simulations are performed averaging their error sequences. Figure 6.22 shows the error curves for the $SNR_{\mathbf{X}} = 10$ case with the planar array adapting almost as fast as the lens cases with 9 signals. The lens case where all signals are used is slower than the planar array indicating the weights corresponding to low received signal power need a long time to adapt, as in Equation (6.21). Table 6.1 show the estimated adaptation rates for different lens and planar array systems. Since the step size changes as the number of signals are used, the resulting SNR always increases at the cost of adaption rate. This is a classic tradeoff in adaptive systems and needs to be explored further.

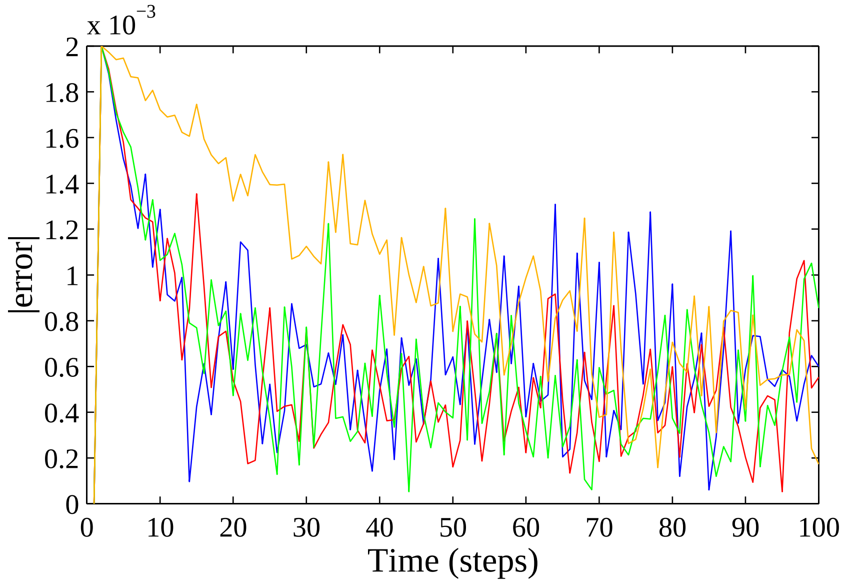


Figure 6.22. Error curve tracks for 5 (blue), 9 (red) and all (yellow) signal used in adaptive lens array compared to a planar array (green).

Table 6.1. The number of time steps required for planar and lens array systems to adapt.

	Number of Signals in Lens Array					
$SNR_{\mathbf{x}}$	1	3	5	9	all	array
1	10	15	30	40	120	70
3.2	8	13	18	27	80	60
10	13	20	22	25	80	40

CHAPTER 7

FUTURE WORK AND CONCLUSIONS

7.1 Optically Controlled Lens Arrays

7.1.1 Future Work for Optically Controlled Lens Arrays

There are three principle problems that have to be overcome for optically controlled lens arrays to become practical. First, the size of the SPDT switches should be reduced. Second, a better optical circuit for delivering optical signals to the photodiodes needs to be designed. Third, the overall construction complexity should be minimized further..

Looking at the layout of the unit cell, most of the SPDT switch size comes from the packages of the MESFETs and photodiodes. Using chips or MMICs, the switch size could be reduced by about a factor of 2. If a MMIC SPDT switch is optically controlled through its bias, the size would be reduced further. An all-MMIC solution for the optical controlled microwave switch would significantly reduce the size and complexity of the array manufacturing, but require a special process..

A significant amount of the effort and complexity in the array is in the design and fabrication of the optical fiber mount. By using ridge waveguides [59] stamped or etched into a translucent material to replace the optical mount and the fibers, the optical system can be made in a single fabrication step. In Chapter 4, the electrical and optical circuit could be separated at the photodiode and optical fiber interface. It is also this interface that causes most of the lost in the optical circuit. Instead of separating the two circuits at the photodiode/fiber interface, it would be

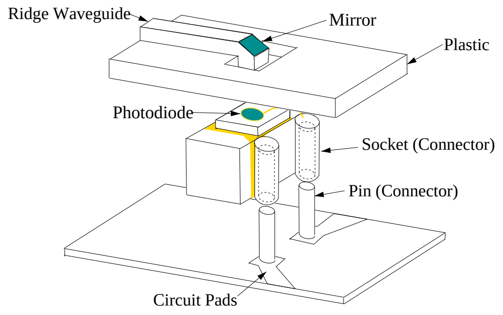


Figure 7.1: New optical feed system for optically controlled arrays.

more efficient to have the connection at the photodiode/MESFET interface, as illustrated in Figure 7.1. By directly gluing the photodiodes to the optical waveguides, the loss in the optical circuit could be reduced to about 1 dB. The photodiode is connected to the MESFETs through a connector, shown as a pin/socket connector in the figure. Since most of the spectral energy in the switch currents is 500 MHz or below, the added inductance and capacitance of a pin/socket connector should have negligible affect.

7.1.2 Conclusion for Optically Controlled Lens Arrays In this thesis, a high-speed optically controlled T/R active antenna is demonstrated. This active antenna is designed as an element of a $6\,\lambda_0$ by $3\,\lambda_0$ cylindrical active lens array with a F/D=1, a directivity of 20 dB and a 10-degree beamwidth in the focusing plane. The lens array is capable of phase-shifterless beamforming in both transmition and reception.

The optically controlled SPDT switch used to route the signal in the active antenna exhibits 0.31 dB insertion loss, 36 dB isolation and -10 dB return loss from 8.36 to 10.8 GHz. Unlike in previously-demonstrated optically-controlled microwave switches, these microwave parameters do not change with the level of incident control (optical) power. However, the optical power level conveniently controls the switching speed alone, making the same switch design easily integrated into applications with different switching speed requirements. Microsecond switching required in most T/R applications can be accomplished with only microWatts of optical power, but some applications, such as phase shifters in phased array, polarization switching in multi-path environments and wide bandwidth systems would benefit from nanosecond switching speeds. Even though the 2.6 ns speed demonstrated in this paper requires relatively large optical power (about 10 mW), the optical energy is quite low, 21 pJ. For comparison, the fastest reported MEMS microwave switch has a rise time of about 1 μs to 5 μs and requires about 2 nJ of control energy [29]. The switch presented here draws more electrical power (3 mA per PIN diode) than a MEMS switch, but the required DC current is a small fraction of the 160 mA drawn by the LNA and PA. However, the optically controlled microwave switch requires significantly less control energy than a MEMS switch, since the energy distribution is fundamentally different. In some respects, it is easier to generate fast optical pulses than to generate fast high voltage (30 V - 50 V) electrical pulses required for MEMMS switches.

The active array element is designed with off-the-shelf components not optimized for speed or low power, and the coupling of light from the fibers is not optimal. Therefore, the presented results are by no means fundamentally limited and we expect that significant improvements can be made by using PDs with better placement tolerances, using chip MESFETs for the pin diode bias control and by improving the laser diode to photodiode coupling efficiency (using, e.g. microlenses and printed optical waveguide structures). Ultimately, a large portion of the switch, and in principle the entire switch circuit, can be implemented monolithically.

7.2 Modeling of Lens Arrays

7.2.1 Future Work in Modeling Lens Arrays Most of the errors in the numerical model in Chapter 5 are due to only accounting for direct radiation. Due to the close proximately of the lens array elements and the detectors to the lens, mutual coupling will play an important role in the lens array performance.

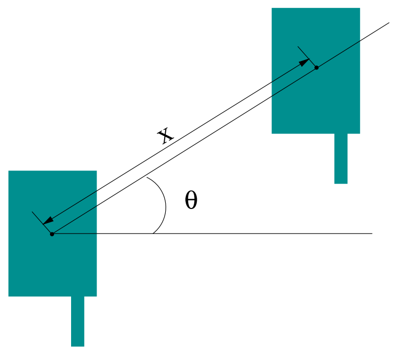


Figure 7.2: Mutual coupling for patch antenna with arbitrary orientations.

In physical measurements with real active lens arrays, standing waves have been noticed between the lens and the detector elements that can not be explained by simple radiation, [1]. In an attempt to estimate some of the mutual coupling, the numerical model is modified to include the calculation of induced currents in each antenna due to other antennas in its neighborhood, [60, 61, 62]. [62] shows that the mutual coupling between arbitrary orientated patch antennas, Figure 7.2, can be estimated from several full-wave simulations and the following equation

$$|S_{21}| = \frac{\rho}{x^{1+\sin\theta}}, \tag{7.1}$$

where ρ is a fitting parameter. Currents induced by mutual coupling can be calculated from the mutual coupling impedances. With these current, the mutual coupling image is calculated to estimate the error in the image calculations. Averaging the error over all fifty sources (section 6.3) the mean error is about 1.5% suggesting that the image pattern is at best accurate to 18 dB below the main lobe. Hopefully, in the near future it will be possible to perform a full wave model of a lens array.

Besides improving the accuracy of the model, work needs to be done to determine the best lens array design for both circular and rectangular lenses. A question of whether of not the lens array transformation fundamentally limits the achievable radiation patterns for a lens array should be addressed. Work in [25, 26, 24] on the limitations of network lenses may be applicable to lens arrays since the sampling pattern presented in section 6.2 appear to corresponds with the beams produce by network lenses. Finally, an analytical model for the lens array transformation should be developed that describes the lens array loss, aberrations and antenna element effects so that a communication model can be developed.

7.2.2 Conclusion for Modeling of Lens Arrays The preliminary work done in chapter 5 shows that design of lens arrays can be optimized for different performance criteria (i.e. loss, aberration, etc.). Depending on the choice of design parameters, the loss or aberrations in the lens can optimized. It is still unknown what characteristic is most important to lens array communication systems and more work needs to be done to answer this question.

7.3 Adaptive Lens Array Systems

7.3.1 Future Work for Adaptive Lens Array Systems The work presented in chapter 6 is for the simplest communication problem, a single user. More work needs to be done in determining the affects of loss and aberrations on the pre-noise beam-forming process gain. Also the tradeoffs between adaption rate and weights noise needs to be explored and compared to a planar array system. The simulations should be expanded to more complicated environments like the multi-user and multi-path environments. Besides simulating more complicated environments, the receiver model should be modified to incorporate practical limitations, such as imperfect training signals. Lens array systems may also have advantages with other algorithms such as angle of arrival algorithms.

7.3.2 An Arguement for Improved SNR with Lens Arrays Lens arrays may have an advantage over planar arrays by performing some beam-forming before noise in added by gain devices such as low noise amplifiers (LNAs) and down

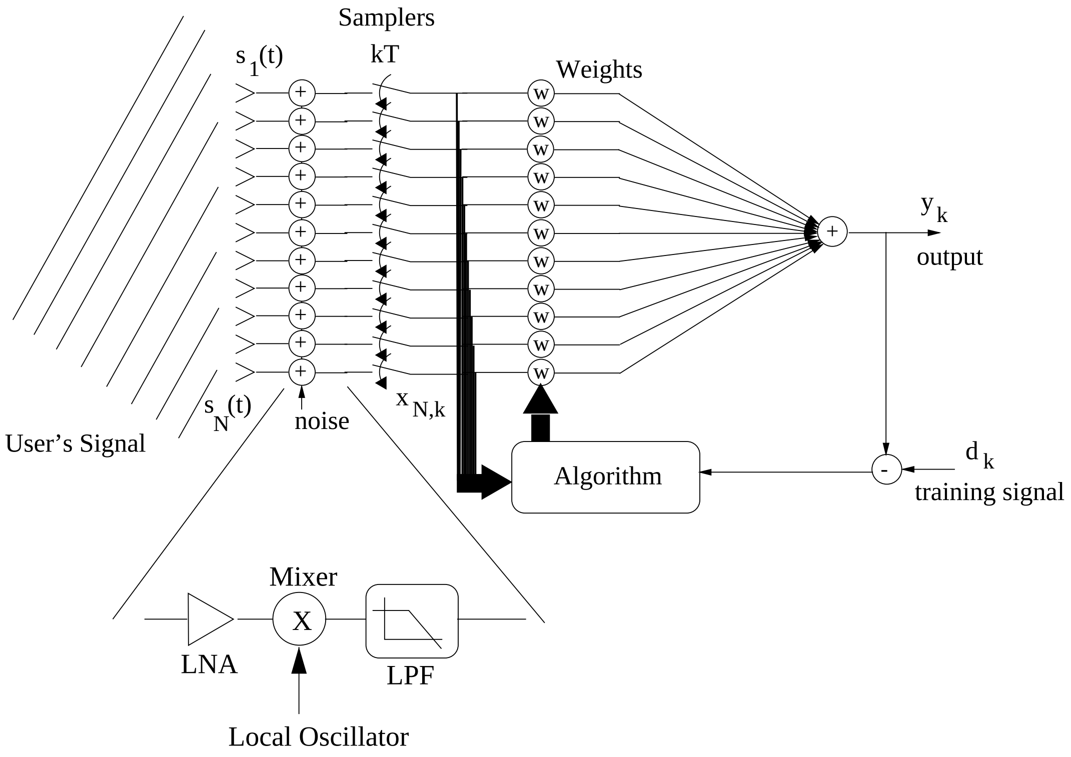


Figure 7.3. An example of a narrowband planar array adaptive receiver shown for the single user case. The received signals, $s_i(t)$, are amplified with LNAs and down-converted (mixer, local oscillator and LPF) before being sampled (represented by switches). For this model, the effect of the LNAs and down-converters is the addition of noise. The received signals plus noise are sampled with a sampling period T, giving signals, $x_{i,k}$, used by the algorithm. The algorithm uses the training signal, d_k, to adjust the weights, W, such that the SNR at the output, y_k, is maximized.

converters which translate the modulated signal to a lower frequency. In most "standard" adaptive systems, there is a LNA place directly after each array element, Figure 7.3. By placing the amplifiers close to the array elements, the affect of the noise introduced by cable loss can be minimized. In systems that are not limited by "sky" noise, the noise generated by the LNAs becomes the dominate noise source and is usually considered to be white and uncorrelated. After the LNA, a down-converter are usually added to mix the carrier signal to a lower frequency which is easier to process. Down-converters, which contain mixers (multipliers), local oscillators (LOs) and low pass filters (LPFs) do add additional noise and can have gain greater than unity. For our purpose a complex-baseband model will be used and the noise generated by a LNAs and down-converters are modeled as uncorrelated and white noise sources. The output signal, y_k, can be written in terms of sampled versions of the received signals ($s_i(k)$), the noise sources ($n_i(k)$) and the optimal weights (w_i):

$$y(k) = \left(\sum_{i=1}^{N} w_i s_i(k)\right) + \left(\sum_{i=1}^{N} w_i n_i(k)\right), \tag{7.2}$$

where i is the ith antenna element. The first term is the signal term, and the second term is the noise term. For a planar array receiving a signal from a far-field source, each element sees a phase shifted version of the original signal,

$$s_i(k) = \alpha_i s_0(k) \tag{7.3}$$

$s_0(k)$ is the sampled complex baseband representation of the transmitted signal, and α_i is a complex constant representing the phase shift of the signal and taken to have unit magnitude. Under these conditions, $y(k)$, reduces to

$$y(k) = \left(\sum_{i=1}^{N} w_i \alpha_i s_0(k)\right) + \left(\sum_{i=1}^{N} w_i n_i(k)\right), \tag{7.4}$$

It can be shown from Equation (1.1) that the optimal solution for the weights is $w_i = w_0 \alpha_i^*$, where w_o is a real constant, Figure 7.4. Substituting the weights into Equation (7.4), the output reduces to

$$\begin{aligned} y(k) &= \left(\sum_{i=1}^{N} w_0 |\alpha_i|^2 s_0(k)\right) + \left(\sum_{i=1}^{N} w_0 \alpha_i^* n_i(k)\right) && (7.5)\\ &= \left(w_0 s_0(k) \sum_{i=1}^{N} 1\right) + \left(w_0 \sum_{i=1}^{N} e^{-\phi_i} n_i(k)\right) && (7.6)\\ &= (w_0 s_0(k) N) + \left(w_0 \sum_{i=1}^{N} e^{-\phi_i} n_i(k)\right) && (7.7) \end{aligned}$$

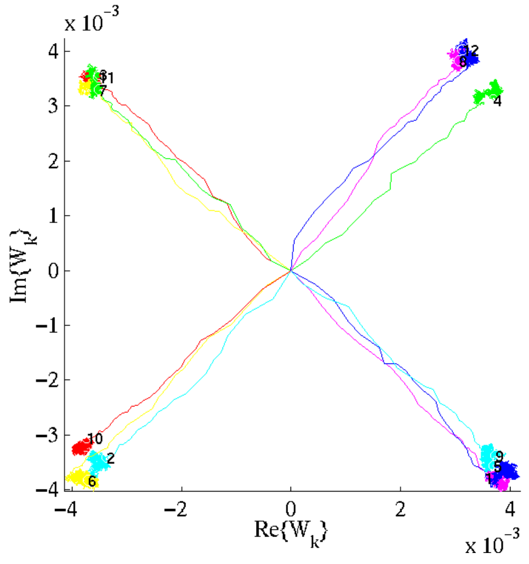


Figure 7.4. The weight trajectories in the complex plane for a planar array adaptive receiver converging to the optimal weights.

ϕ_i is the phase shift seen by each array element. The SNR for this system is

$$SNR_{planar} = \frac{E\left[\left(Nw_0 s_0(k)\right)\left(Nw_0 s_0(k)\right)^*\right]}{E\left[\left(w_0 \sum_{i=1}^{N} e^{-\phi_i} n_i(k)\right)\left(w_0 \sum_{i=1}^{N} e^{-\phi_i} n_i(k)\right)^*\right]} \tag{7.8}$$

$$= \frac{N^2 w_0^2 \sigma_s^2}{w_0^2 \sum_{i=1}^{N} E\left[\left(e^{-\phi_i} n_i(k)\right)\left(e^{-\phi_i} n_i(k)\right)^*\right]} \tag{7.9}$$

$$= \frac{N^2 \sigma_s^2}{N \sigma_n^2} \tag{7.10}$$

$$= \frac{N \sigma_s^2}{\sigma_n^2} \tag{7.11}$$

Since the signals add coherently and the noise adds incoherently the SNR for this system scales linearly with the number of the elements in the array.

The system in Figure 7.3 is the "typical way" in which planar arrays adaptive systems are implemented; however, if the beamforming could be done before the noise is added, then the resulting SNR is greater than showed previously, Figure 7.5. In this case, the output signal is

$$y(k) = \left(\sum_{i=1}^{N} w_i s_i(k)\right) + n_i(k) \tag{7.12}$$

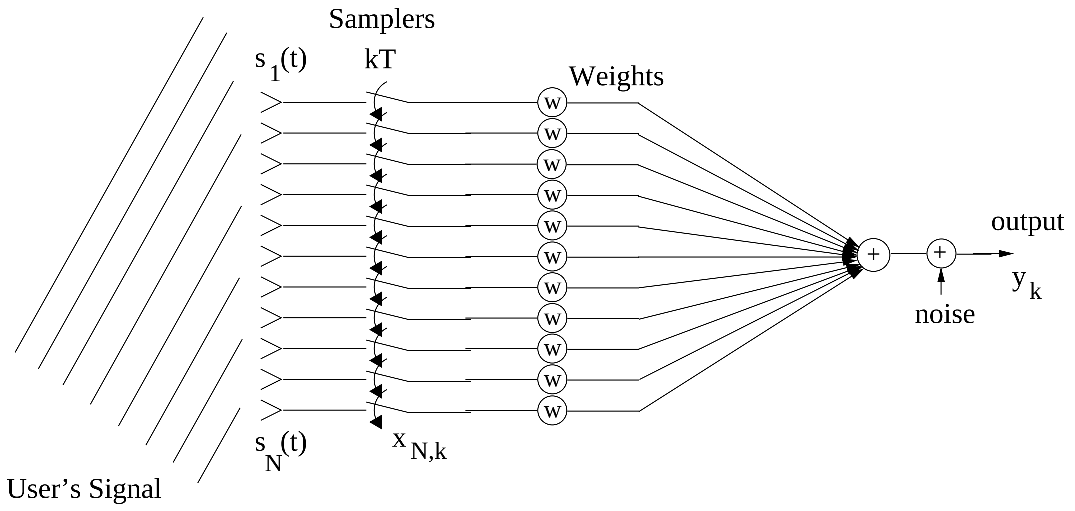


Figure 7.5. An example of a planar array adaptive system where the beam-forming is performed before any amplifiers of down-converters (represented by the noise source).

Again, using the previous assumptions for the noise, weights and α_i's, $y(k)$ reduces to

$$y(k) = Nw_0s_0(k) + n_i(k) \tag{7.13}$$

The resulting SNR is

$$\begin{aligned} SNR_{planar,ideal} &= \frac{E\left[(Nw_0s_0(k))\,(Nw_0s_0(k))^*\right]}{E\left[(n_i(k))\,(n_i(k))^*\right]} && (7.14)\\ &= \frac{N^2w_0^2\sigma_s^2}{\sigma_n^2} && (7.15)\\ &= \frac{N^2w_0^2\sigma_s^2}{\sigma_n^2} && (7.16)\\ & && (7.17)\end{aligned}$$

If it is possible to perform the beam-forming (weights) before the down conversion (noise) then the SNR grows as N^2. In addition, gain in the weights (w_0) can be added without penalty. However, the system in Figure 7.5 is not practical. There are several important physical limitations that are over looked in the model. First of all, the samplers in the system need to operate at twice the carrier frequency,

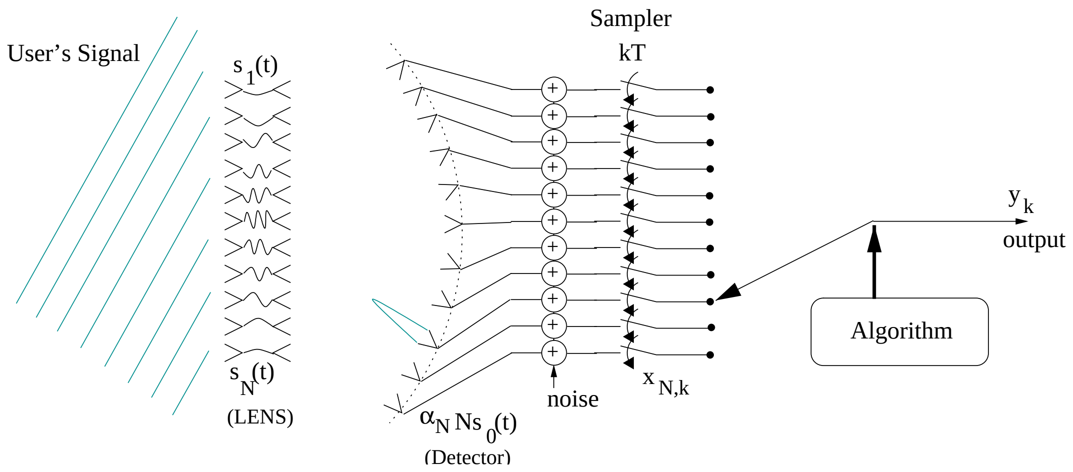


Figure 7.6. An example of a narrowband lens array adaptive receiver shown for the single user case. The received signal, $s(t)$, is perfectly focused onto a detector on the focal surface. The detected signal, $e^{-j\phi}Ns(t)$, are amplified and down converted (represented by the noise source). The down converted signals plus noise are sampled (represented by switches) with a sampling period T. Since the lens perfectly focuses the received signal, the algorithm only chooses which detector to switch to the output.

since the act of down conversion adds noise to the system. Secondly, the beam-forming hardware needs to located very close to the antenna array, since cable loss reduces signal power and adds thermal noise. It is for this reason that LNAs are placed close to the antenna elements and become one of the dominate noise sources.

A lens array, Figure 7.6, focuses the received signal, $s(t)$, to a point on the focal surface. The transmission line delays in the lens and the spatial delays between lens elements and the focal point act as the beam-forming network (weights) shown in Figure 7.5. For an ideal lens, all of the lens array received signals, $s_i(k)$, sum coherently at the focal point, $e^{j\phi}Ns_0(k)$. Here the ϕ indicates that in general, the coherent sum of the signals is out of phase from the original signal. In the ideal lens system, the algorithm decision is to choose which detector is illuminated by the lens.

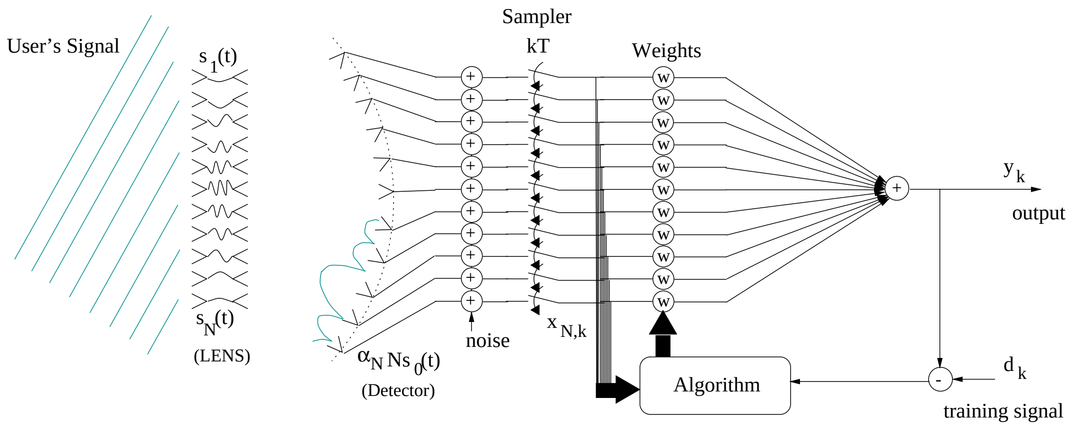


Figure 7.7. An example of a narrowband lens array adaptive receiver shown for the single user case. The received signal, $s(t)$, is focused onto detectors on the focal surface. The detected signals, $\alpha_i N s(t)$, are amplified and down converted (represented by the noise source). The down converted signals plus noise are sampled (represented by switches) with a sampling period T, giving signals, $x_{i,k}$, used by the algorithm. The algorithm uses the training signal, d_k, to adjust the weights, W, such that the SNR at the output, y_k, is maximized.

Just like the pre-noise beam-forming planar array system, the SNR is

$$SNR_{lens,ideal} = \frac{E\left[\left(e^{j\phi} N s_0(k)\right)\left(e^{j\phi} N s_0(k)\right)^*\right]}{E\left[(n(k))(n(k))^*\right]} \tag{7.18}$$

$$= \frac{N^2\sigma_s^2}{\sigma_n^2} \tag{7.19}$$

Unfortunately, it is not possible to build an ideal lens array. The array can not focus all of the received signals to a single point, but to a focal spot, Figure 7.7. This means that the signal signal power is distributed over many detectors. For the formulation of the performance of the system in Figure 7.7, let us redefine the α_i's as a measure of how well the lens focuses. Assuming that the lens array elements receive signals from a far-field source of the form,

$$s_i(k) = e^{j\phi_i} s_0(k) \tag{7.20}$$

where ϕ_i describes the phase shift seen by each array element, the signals received by the detectors on the focal surface is given by

$$s_{i,det}(k) = \alpha_i N s_0(k) \tag{7.21}$$

The α_i's shows how well the lens coherently focuses the re-radiated signals to each detectors and are constrained such that

$$\sum_{i=1}^{M} |\alpha_i| \leq 1 \tag{7.22}$$

Here the summation is taken from 1 to M for the M detectors on the focal surface. Interestingly, if the detectors are placed on the focal surface in accordance to the 2-dimensional sampling theorem, then $M \approx N$. For this analysis, we will assume $M = N$ so that both the planar array system and lens array system have the same number of degrees of freedom (weights).

The output for this system is given by

$$y(k) = \left(\sum_{i=1}^{N} w_i \alpha_i N s_0(k) \right) + \left(\sum_{i=1}^{N} w_i n_i(k) \right), \tag{7.23}$$

Again, using $w_i = w_0 \alpha_i^*$, the output becomes

$$y(k) = \left(w_0 N s_0(k) \sum_{i=1}^{N} |\alpha_i|^2 \right) + \left(w_0 \sum_{i=1}^{N} \alpha_i^* n_i(k) \right), \tag{7.24}$$

The SNR for the system is

$$SNR_{lens} = \frac{E\left[\left(w_0 N s_0(k) \sum_{i=1}^{N} |\alpha_i|^2 \right) \left(w_0 N s_0(k) \sum_{i=1}^{N} |\alpha_i|^2 \right)^* \right]}{E\left[\left(w_0 \sum_{i=1}^{N} \alpha_i^* n_i(k) \right) \left(w_0 \sum_{i=1}^{N} \alpha_i^* n_i(k) \right)^* \right]} \tag{7.25}$$

$$= \frac{w_0^2 N^2 \sigma_s^2 \left| \sum_{i=1}^{N} |\alpha_i|^2 \right|^2}{w_0^2 \sum_{i=1}^{N} E\left[(\alpha_i^* n_i(k)) (\alpha_i^* n_i(k))^* \right]} \tag{7.26}$$

$$= \frac{w_0^2 N^2 \sigma_s^2 \left(\sum_{i=1}^{N} |\alpha_i|^2 \right)^2}{w_0^2 \sigma_n^2 \sum_{i=1}^{N} |\alpha_i|^2} \tag{7.27}$$

$$= \frac{N^2 \sigma_s^2}{\sigma_n^2} \sum_{i=1}^{N} |\alpha_i|^2 \tag{7.28}$$

The Equation (7.28), describes the SNR for an imperfect lens array system. The system can be considered as a hybrid system with beam-forming before noise is added (lens) and beam-forming after noise is added (weights). When the α_i's have equal magnitude, $|\alpha_i| = \frac{1}{N}$, then beam-forming occurs after the addition of the noise.

This is as if the lens array does not focus the signals but illuminate the focal surface equally. In this case, the SNR of Equation (7.28) reduces to

$$SNR_{lens} = \frac{N\sigma_s^2}{\sigma_n^2} \tag{7.29}$$

which is the same as for the planar array case in Figure 7.3. When all the α_i's are equal to zero except for one which has unit magnitude, then in this case all the beam-forming in done before the noise is added (the ideal lens case). Then Equation (7.28) becomes

$$SNR_{lens} = \frac{N^2\sigma_s^2}{\sigma_n^2} \tag{7.30}$$

and equals the result for the ideal lens case.

Given the two previous examples, the hybrid lens system should have a better performance than the "typical" planar array system, Figure 7.3, from the partial beam-forming done before the noise is added. Is this true in general? The SNR Equation (7.28) depends on the distribution of the signal among the detectors described by the summation

$$\sum_{i=1}^{N} |\alpha_i|^2 \tag{7.31}$$

Let us start with the no pre-noise beam-formingi, where $|\alpha_i| = \frac{1}{N} = \alpha_0$. Equation (7.31) becomes

$$\sum_{i=1}^{N} |\alpha_0|^2 \tag{7.32}$$

Then re-distribute the signal by letting $\alpha_m = \alpha_0 - c$ and $\alpha_n = \alpha_0 + c$ where m, n are the indexes of two detectors. The Equation (7.32) becomes

$$\left(\sum_{i=1, i\neq m,n}^{N} |\alpha_0|^2\right) + (\alpha_0 - c)^2 + (\alpha_0 + c)^2 = \left(\sum_{i=1, i\neq m,n}^{N} |\alpha_0|^2\right) + 2\alpha_0^2 + 2c^2 \tag{7.33}$$

$$= \left(\sum_{i=1}^{N} |\alpha_0|^2\right) + c^2 \tag{7.34}$$

$$> \sum_{i=1}^{N} |\alpha_0|^2 \tag{7.35}$$

Therefore the re-distribution of the signal, pre-noise beam-forming, increases the processing gain of the system. The limiting case is the perfect lens system.

There are several practical issues in adaptive system that still need to be addressed. It is true that the optimal solution for an adaptive system adds in a signal no matter how small its signal power. For a lens system, most of the signal power is concentrated on a few detectors. If a system only processes the detector signals with the most signal power, the resulting SNR will only be slightly less than the optimal SNR,

$$SNR_{lens,M} = \frac{N^2\sigma_s^2}{\sigma_n^2}\sum_{i=1}^{M}|\alpha_0|^2 \tag{7.36}$$

$M < N$ and determines the number of detector signals to be used. Practically speaking, a processor operating on a subset of the detector signals will have nearly optimal SNR with far less computation load. In real adaptive systems, the samplers also quantize the detector signals. If these quantizers do not have enough dynamic range (bits), then the quantization noise may obscure the algorithms ability to estimate very weak signals. Finally, there is a fundamental tradeoff between the weight noise (how well the algorithm estimates the optimal weights) and convergence to the optimal weight solution. The faster the algorithm converges to the solution, the more weight noise there is at the output, which reduces the systems SNR. For the lens array system, this tradeoff is different than for the "typical" planar array system and may have some addition practical advantages.

It has been shown that lens array can improve a adaptive systems SNR though pre-noise beam-forming. How this SNR increases as the number of elements in the lens array increase is still unknown. Unfortunately, real lens arrays are lossy,

$$\sum_{i=1}^{N}|\alpha_i| < 1 \tag{7.37}$$

and the loss increases with lens size,

$$|\alpha_i| \propto N^{-b}, \qquad 0 < b < 1 \tag{7.38}$$

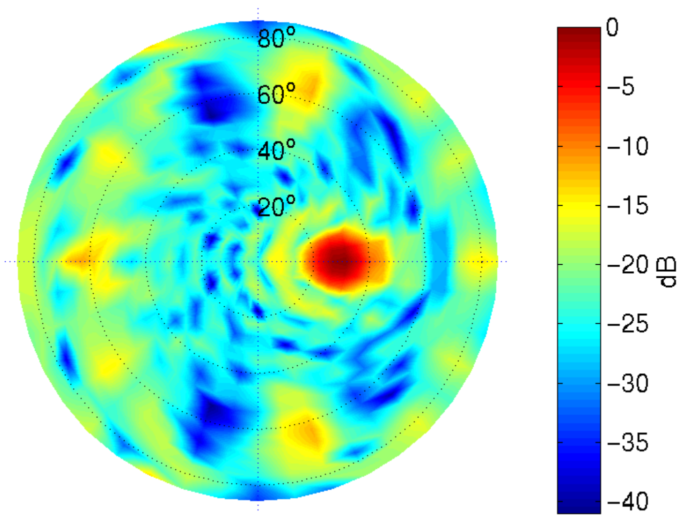
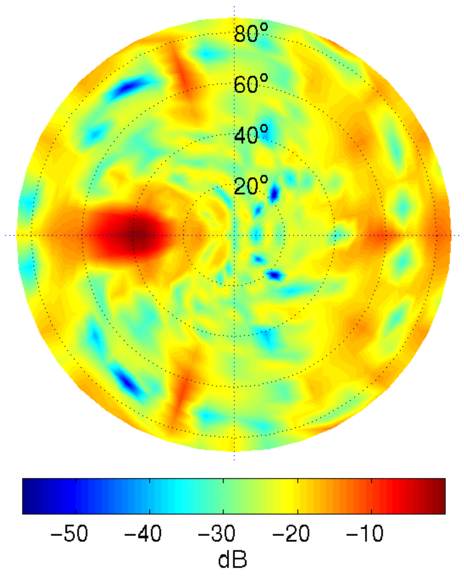


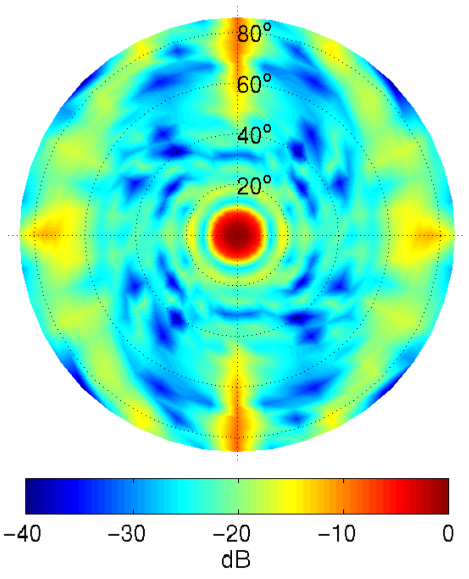
Figure 7.8: Image pattern for "desired" user at 30^o in multi-user model.

This loss is a result of path loss between lens array elements and elements on the focal surface. It does not add any additional noise to the system but does reduce signal power. The affect of this loss on adaptive lens array systems is still unknown.

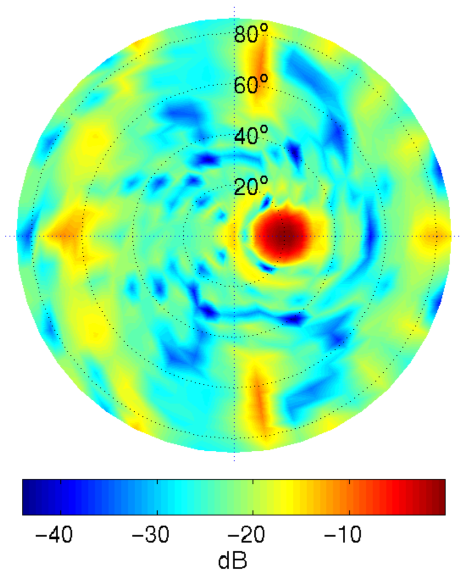
7.3.3 Adaptive Lens Array Systems with Multiple Users The multi-user case and the user/jammer case are examples of a "desired" user being interfered by an "other" user. Preliminary simulations for the multi-user case are done with the "other" user at twice the QPSK modulation data rate and equal transmit power moving towards the "desired" user. The user is 20^o off axis, Figure 7.8. When the "other" user moves close to the "desired" user as in Figure 7.9, then the desired user's weight adjust to null out the "other" user, Figure 7.10(a) and 7.11. The weight plot, Figure 7.10(b) and the radiation plots, Figure 7.13, for the planar array show similar results as in the lens array case. Cuts in the radiation patterns along $\chi = 0$ show that the lens and planar arrays create the same depth nulls when the "other" user is close to the "desired" user showing the limitation in the array layout, Figure 7.13. The planar array creates deeper nulls when the "other" user is far from the "desired" user to reduce its power below the noise floor since planar



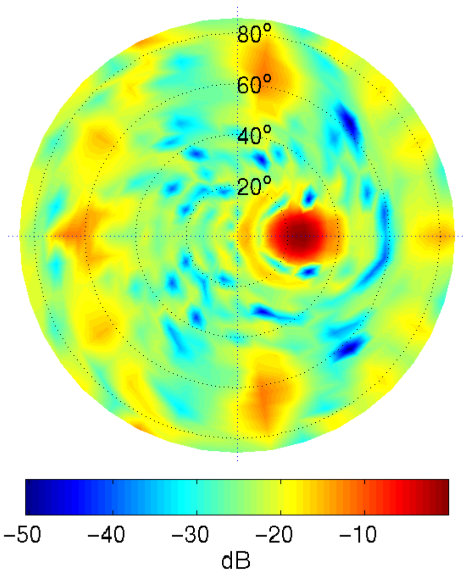
(a)



(b)

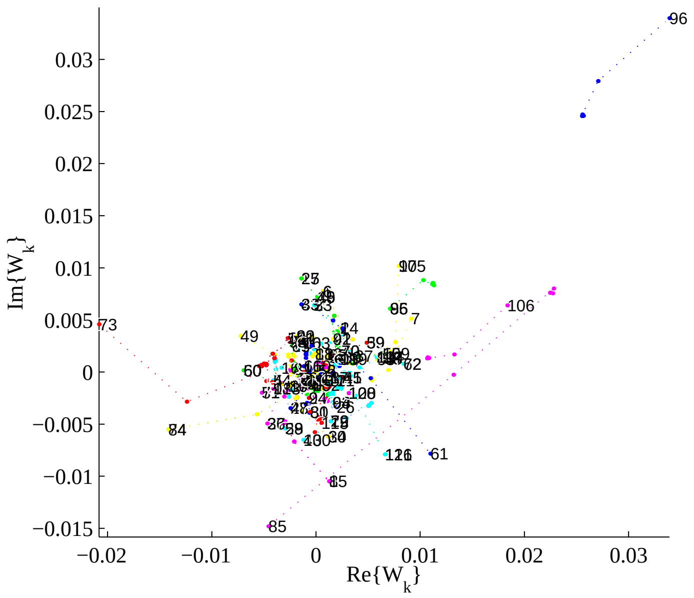


(c)

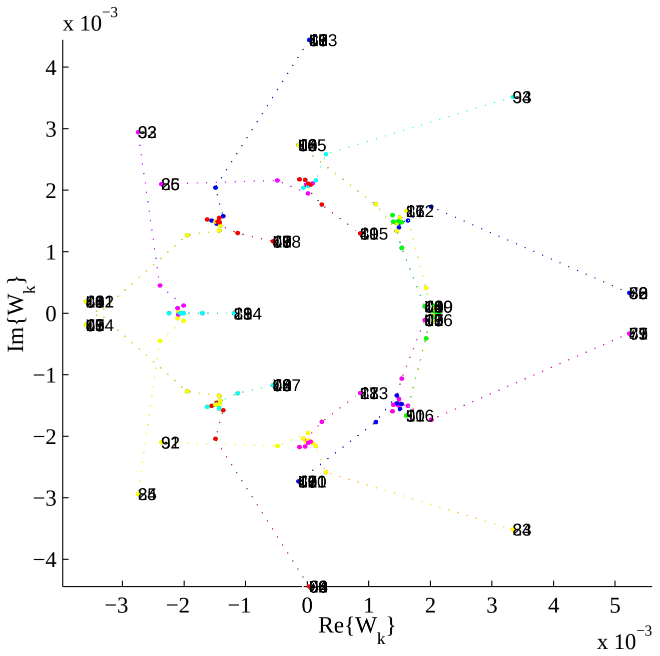


(d)

Figure 7.9. Image pattern for "other" user at -40^o (a), 0^o (b), 20^o (c) and 25^o (d) in multi-user simulation.

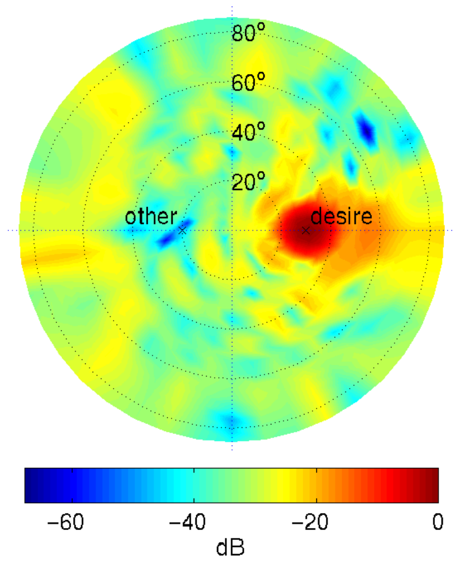


(a)

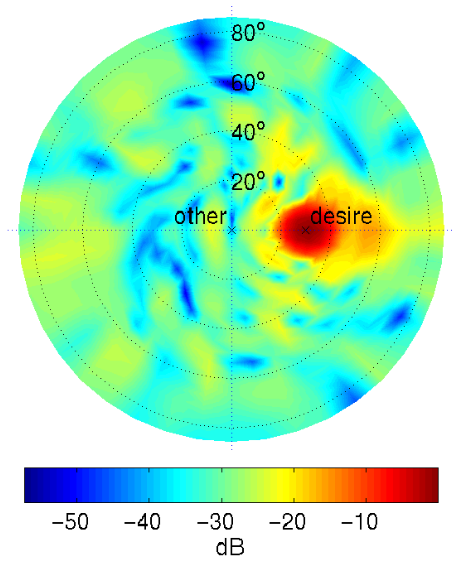


(b)

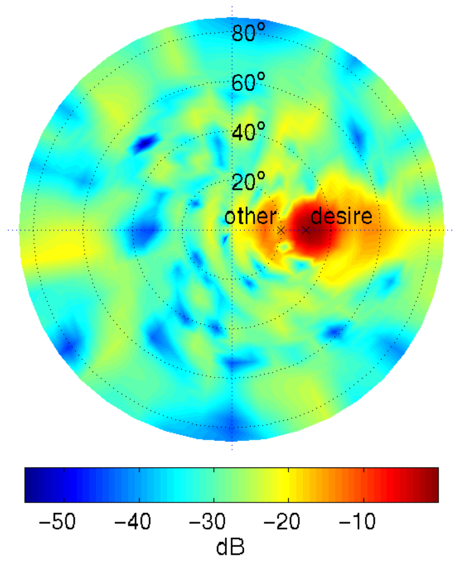
Figure 7.10. The movement of the optimal weight for lens array (a) and planar array (b) adaptive systems as the “other” user moves closer to the “desired” user. The final realization of the optimal weights are the numbered data points.



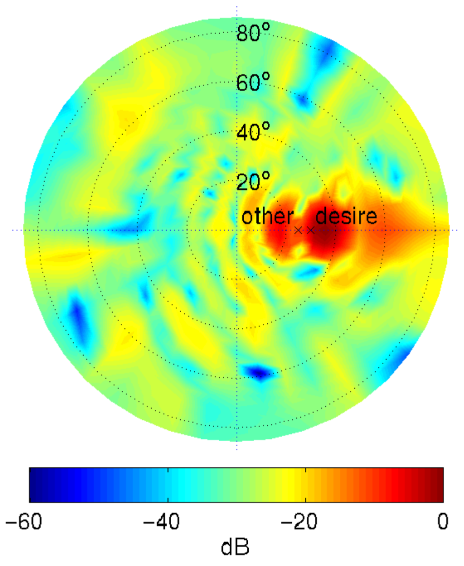
(a)



(b)

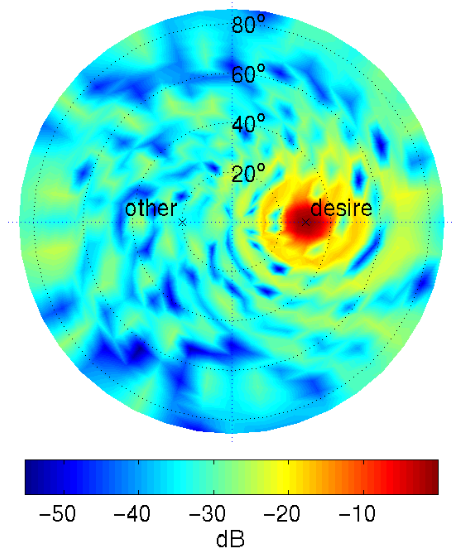


(c)

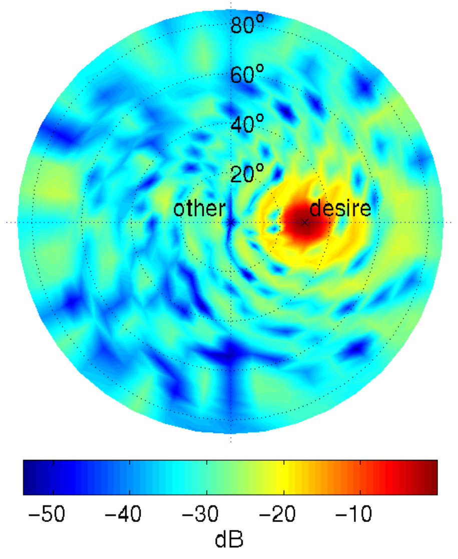


(d)

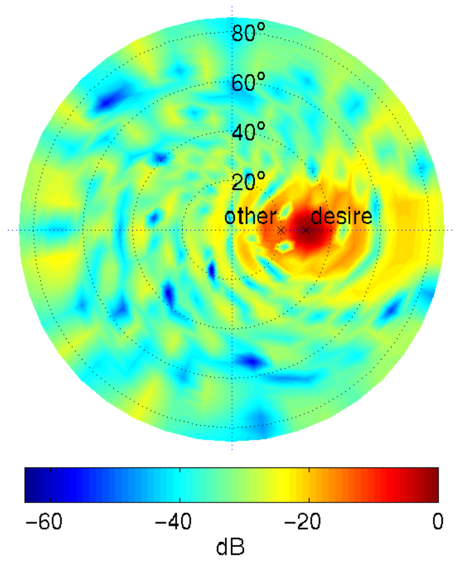
Figure 7.11. The radiation patterns for lens array system in multi-user environment with "desired" user at 30^o and the "other" user at -40^o (a),0^o (b), 20^o (c) and 25^o (d).



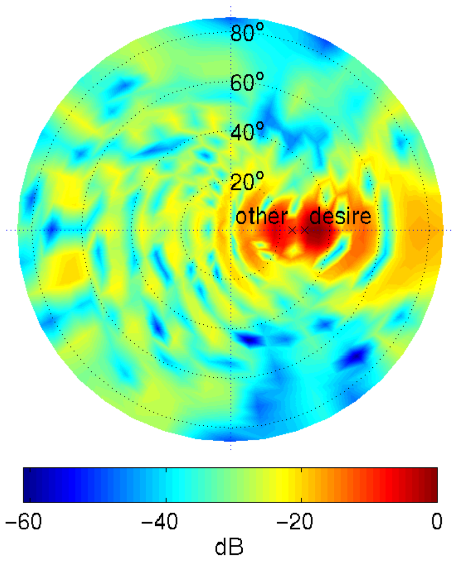
(a)



(b)

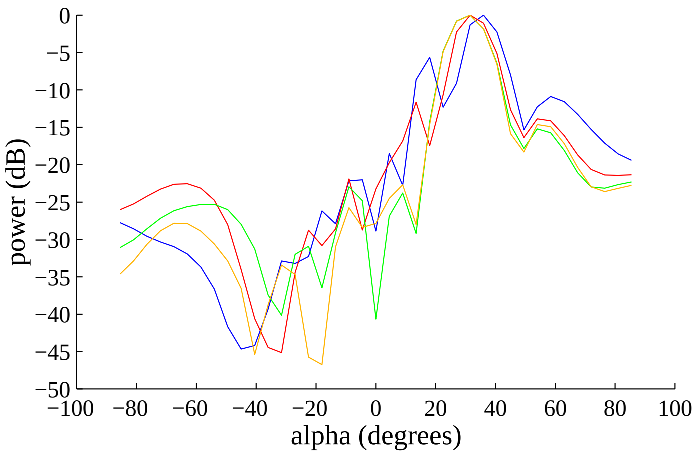


(c)

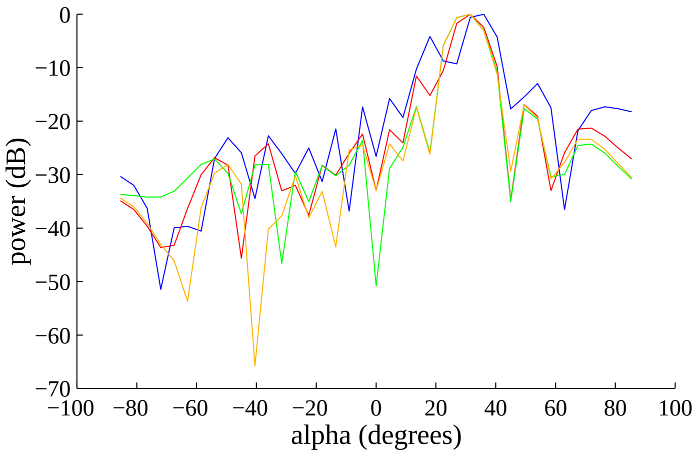


(d)

Figure 7.12. The radiation patterns for planar array system in multi-user environment with "desired" user at 30^o and the "other" user at -40^o (a),0^o (b), 20^o (c) and 25^o (d).



(a)



(b)

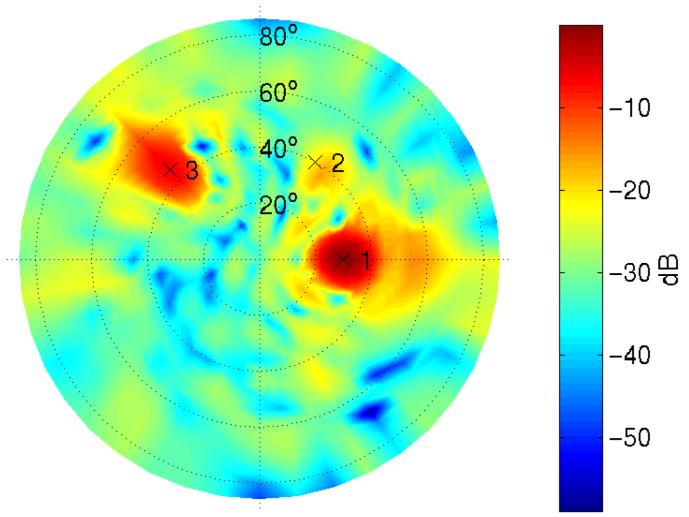
Figure 7.13. $\chi = 0$ cuts for the radiation patterns in figure 7.11 (a) and figure 7.13 (b). The cuts are shown for the “other” user at -40^o (yellow), 0^o (green), 20^o (red) and 25^o (blue).

arrays have less loss than lens arrays. These results suggest that lens array and a planar array have the same ability to null out "other" users and that there is not a fundamental limitation in achievable radiation patterns.

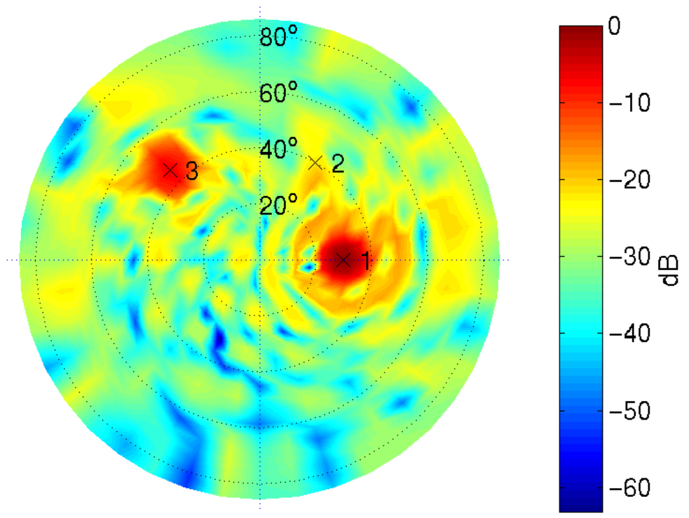
7.3.4 Adaptive Lens Array Systems with Multi-path For the multi-path case, both lens and planar arrays steer beams in the directions of the received signals for the "desired" user, Figure 7.14. The gain in each beam is proportional to the strength of the signal in each path. For example, path 2 has 10 dB less power that path 1, and the difference in there respective array gains is about 10 dB. Again, the lens array system (Figure 7.15) only uses the detectors illuminated by the "desired" user, Figure 7.16(b). Since the signals for the three paths are correlated, they form a standing wave on the surface of the arrays causing the weighs in the planar array form on sets of of circles instead of just one, Figure 7.16(a). The planar arrays still have a better SNR of 23.5 dB, versus the lens arrays of 15.1 dB.

The simulations in section 6.3 are impractical from the point of view that the LMS algorithm knows the transmitted data for the training signal and that the sources are stationary. In a realistic situations, neither of theses conditions are valid. The simulations need to be repeated with decision-directed feedback used for the training signal and with the transmitting sources moving. An important question is whether the weights that are turned "off" will turn back "on" when needed. One simple solution is a second algorithm which uses a larger set of weights and check if new weights are needed. Another solution is to devise a means of estimating the amount of "desired" user signal at each detector to determine which weights should be "on".

A lens array may have an advantage in statistical diversity systems where it naturally decorrelates signals. Previous statistical diversity systems use antennas that are separated by many wavelengths (on the order of $10\,\lambda$) to decorrelate the signals, [8, 9, 10, 12, 63]. By using a lens array, the same order of diversity can



(a)



(b)

Figure 7.14. The radiation patterns for an adaptive lens array (a) and the planar array (b) for three paths of power 0 dB (1), -10 dB (2) and -3 dB (3) in a multi-path environment.

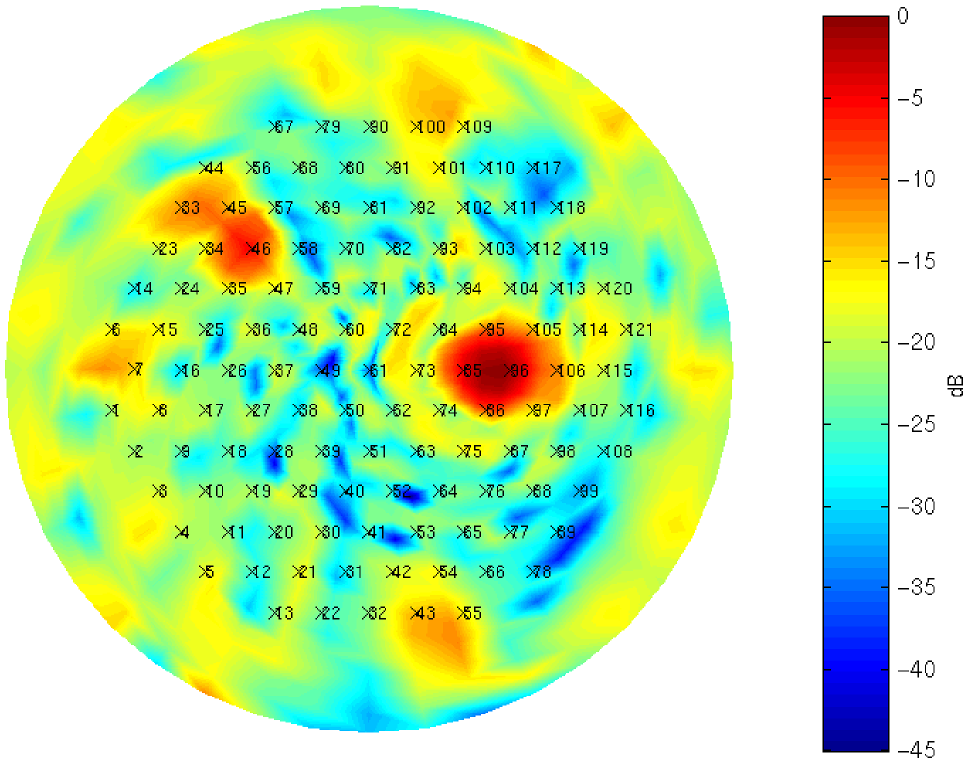
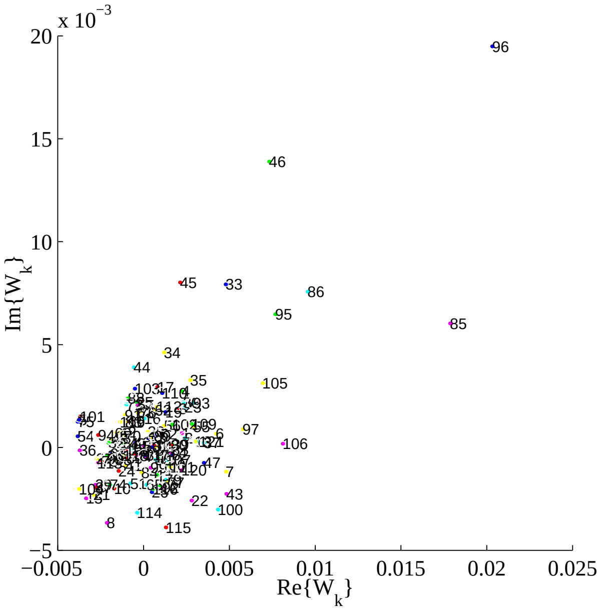
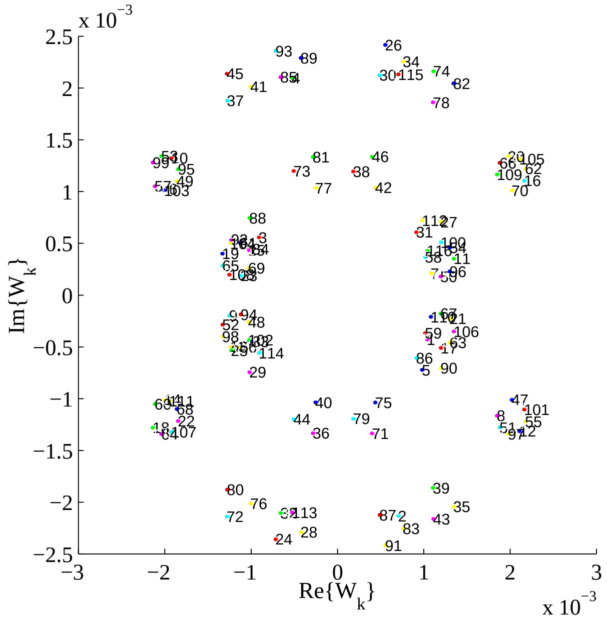


Figure 7.15: The image pattern and detector layout for multi-path environment.

be achieved in a much smaller space. In most statistical diversity systems, the received signals are first decorrelated and then combined. Since a lens array already partially decorrelates the signals, the processor can finish decorrelating the signals with minimal computational power. Turin [63] shows that the optimal combination of the received signals depends on the first- and second-order statistics of the received signals. In a lens array, this is the average of the mean and correlation between the detector signals over all incident angles. If both the mean and covariance of the detectors are non-zero, then the optimal sum of the signals is a mixture of coherent (for non-zero mean) and non-coherent (for non-zero covariance) detection. From the simulations in section 5.3.2, the phase variations on the signals can be uniformly distributed depending on the lens design. A uniform phase distribution has a non-zero mean and only a non-coherent sum of the signals, which has lower SNR than a



(a)



(b)

Figure 7.16. The optimal weights for an adaptive lens array (a) and the planar array (b) in a multi-path environment.

coherent summation of signals. Most of the work in this problem centers around the characterization of the lens correlation function. Once this is know, then it should be a straightforward analysis to determine if lens arrays have an advantage in statistical diversity systems.

7.3.5 Conclusion for Adaptive Lens Array Systems More work needs to be done to show whether lens arrays have clear advantages in adaptive systems. It is clear that they have the advantage of reduced computational complexity at the cost of SNR. The main limitation of lens arrays are their losses. However, for a small enough aperture, network lens arrays and dielectric lenses may be used with less loss. Even though the sampling of the image surface is consistent with the beam spacing for network lenses, it might be important to over-sample of the image surface to recover some of the lost power. Lens arrays have built-in direction of arrival (DOA) information which may give them advantages with some types of algorithms [64, 16, 17]. In this thesis, the analysis of the LMS algorithm applied to lens antenna arrays opened up a new area of research in adaptive (smart) antennas. A lot of work in the future, both at the algorithm and front-end hardware end, remains to be done.

BIBLIOGRAPHY

[1] S. Hollung, **Quasi-optical Transmit/Receive Lens Amplifier Arrays**, Ph.D. thesis, University of Colorado at Boulder, 1998.

[2] D. T. McGrath, “Planar three-dimensional constrained lens,” **IEEE Transactions on Antenna and Propagation**, vol. 34, Jan. 1986.

[3] J. G. Proakis, **Digital Communications**, McGraw-Hill, 3rd edition, 1995.

[4] L. W. Couch II, **Digital and Communication Systems**, Macmillan Publishing, 4th edition, 1993.

[5] E. A. Lee and D. G. Messerschmitt, **Digital Communication**, Kluwer Academic Publisher, 1st edition, 1994.

[6] R. Klemm, **Space-Time Adaptive Processing**, Institution of Electrical Engineers, 1st edition, 1998.

[7] R. Nitzberg, **Radar Signal Processing and Adaptive Systems**, Artech House, 1st edition, 1999.

[8] D. G. Brennan, “Linear diversity combining techniques,” **Proceedings of IRE**, vol. 47, pp. 1075–1102, 1959.

[9] D. G. Brennan, “On the maximum signal-to noise ratio realizable from several noisy signals,” **Proceedings of IRE**, vol. 43, pp. 1530, Oct. 1953.

[10] J. N. Pierce, "Theoretical diversity improvement in frequency-shift keying," **Proceedings of IRE**, vol. 46, pp. 903–910, May 1958.

[11] G. L. Turin, "Communication through noisy, random-multipath channels," **IRE Convention Record**, pp. 154–166, 1956.

[12] Price and Green, "A communication technique for multipath channels," **Proceedings of IRE**, vol. 46, pp. ?, Mar. 1958.

[13] P. Howells, "Intermediate frequency side-lobe canceller," **U. S. Patent 3,202,990**, Aug. 1965.

[14] B. Widrow and J. McCool, "A comparison of adaptive algorithms based on the methods of steepest descent and random search," **IEEE Transactions on Antenna and Propagation**, vol. 24, pp. 615–637, Sept. 1976.

[15] L. J. Griffiths, "A simple adaptive algorithm for real-time processing in antenna arrays," **Proceedings of IEEE**, pp. 1696–1704, Oct. 1969.

[16] R. O. schmidt, "Multiple emitter location and signal parameter estimation," **IEEE Transactions on Antenna and Propagation**, pp. 276–280, Mar. 1986.

[17] R. Kumaresan and D. W. Tufts, "Estimating the angle of arrival of multiple plane waves," **IEEE transactions on Aerospac Electronic Systems**, pp. 134–139, Jan. 1983.

[18] B. Widrow, P. E. Mantey, L. J Griffiths, and B. B. Goode, "Adaptive antenna systems," **Proceedings of IEEE**, vol. 12, pp. 2143–2159, Dec. 1967.

[19] B. Widrow and S. D. Stearns, **Adaptive Signal Processing**, Prentice-Hall, 1st edition, 1985.

[20] G. Kriehn, A. Kirulata, P. E. X. Silveira, S. Weaver, S. Kraut, K. Wagner, R. T. Weverka, and L. Griffiths, "Optical beamtap beam-forming and jammer-nulling

system for broadband phase-array antennas," **Applied Optics**, pp. 212–230, Jan. 2000.

[21] R. C. Hansen, **Phased Array Antennas**, John Wiley and Sons, 1st edition, 1998.

[22] J. L. Bulter and R. Lowe, "Beam forming matrix simplifies design of electronically scanned antennas," **Electronic Design**, vol. 9, pp. 170–173, 1961.

[23] J. N. Nolen, **Synthesis of Multiple Beam Networks for Arbitrary Illuminations**, Ph.D. thesis, 1965.

[24] W. D. White, "Pattern limitations in multiple-beam antennas," **IEEE Transactions on Antenna and Propagation**, vol. 18, pp. 430–436, July 1962.

[25] J. L. Allen, "A theorical limitation on the formation of lossless multiple beams in linear arrays," **IEEE Transactions on Antenna and Propagation**, vol. 9, pp. 350–352, July 1961.

[26] J. P. Shelton, "Multibeam planar arrays," **Proceedings og IEEE**, vol. 56, pp. 430–436, Nov. 1968.

[27] R. A. York and Z. B. Popović, **Active and Quasi-Optical Arrays for Solid-State Power Combining**, John Wiley and Sons, 1st edition, 1997.

[28] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, **Feedback Control of Dynamic Systems**, Addison-Wesley Publishing Company, 3rd edition, 1994.

[29] Elliott R. Brown, "Rf-mems switches for reconfigurable integrated circuits," **IEEE Transtransactons on Microwave Theory Techniques**, vol. 46, pp. 1868–1880, 1998 Nov.

[30] A.M. Johnson and D.H. Austin, "Microwave switching by picosecond photoconductivity," **IEEE Quantum Electronics**, June 1975.

[31] **Millimeter-Wave Switching by Optically Generated Plasma in Silcon**, vol. 14, Nov. 1978.

[32] P.R. Herczfeld, A. S. Daryoush, P.Stabile A. Rosen, , and V.M. Contarino, "Optically controlled microwave devices and circuits," **RCA Review**, p. 528, Dec. 1985.

[33] P. J. Stabile, A. Rosen, and P. R. Herczfeld, "Optically controlled lateral pin diodes and microwave control circuits," **RCA Review**, pp. 443–456, Dec. 1986.

[34] P. Stabile Rosen, W. Janton, P. Basile A. Gombar, J. Delmaser, and R. Hurwitz, "Laser-activated pin diode switch for rf applications," **IEEE Transtransactons on Microwave Theory Techniques**, vol. 37, pp. 1255–1257, Aug. 1989.

[35] A. Guenther, M. Kristian, and T. Martin, **Opening Switches**, Plenum, 1987.

[36] Rosen, P. J. Stabile, A. M. Gomar, A. M. Gombar, W. M. Janton, A. Bahasafri, and P. Hercfeld, "100 kw dc-biased, all semiconductor switch using si pin diodes and algaas 2-d laser arrays," **IEEE Photonics Technology Letters**, pp. 132–134, June 1989.

[37] J.L. Freeman, S. Ray, D.L. West, A. G. Thompson, and M. J. LaGasse, "Microwave control using a high-gain bias-free optoelectronic switch," **Optical Technology for Microwave Applications SPIE**, vol. 5, pp. 320–325, 1991.

[38] S. S. Gevorgian, "Short-circuit photocurrent-controlled microwave pin diode switch," **Microwave and optical Technology Letters**, vol. 7, no. 12, pp. 553–555, Aug. 1994.

[39] B. G. Streetman, **Solid State Electronic Devices**, Prentice Hall, 3rd edition, 1990.

[40] A. Yariv, **Optical Electronics**, Saunders College, 4th edition, 1991.

[41] M. L. VanBlaricum, "Photonic antenna reconfiguration: A status survey," **Proceedings of the SPIE, Photonics and Radio Frequency II**, pp. 180–189, 1998 July.

[42] A. S. Nagra, O. Jerphagnon, P. Chavarka, M. L. VanBlaricum, and R. A. York, "Bias free optical control of microwave circuits and antenna using improved optically variable capacitors," in **Internation Microwave Symposium Digest**, June 2000, pp. 687–690.

[43] P. R. Gray and R. G. Meyer, **Analysis and Design of Analog Integrated Circuits**, John Wiley and Sons, 3rd edition, 1993.

[44] T. Marshall, M. Forman, and Z. Popovic, "Two ka-band quasi-optical amplifier arrays," **IEEE Transtransactons on Microwave Theory Techniques**, Dec. 1999.

[45] C. A. Balanis, **Antenna Theory Analysis and Design**, John Wiley and Sons, 2rd edition, 1997.

[46] C. C. Lin, "A microwave coupler throught a narrow slot in the common groung plane of a two-sided microstrip circuit," **Microwave and Optical Technology Letters**, pp. 543–547, Nov. 1991.

[47] J. Brown, "Microwave lenes," 1953.

[48] H.Gent, "Bootlace aerial," **Royal Radar Establishment Journal**, pp. 47–57, Oct. 1957.

[49] J. B. L. Roa, "Multifocal three-dimensional bootlace lenses," **IEEE Transactions on Antenna and Propagation**, vol. 30, pp. 1050–1056, 1982.

[50] C. M. Rappaport and A. I. Zaghloul, "Optimized three-dimmensional for wide-angle scanning," **IEEE Transactions on Antenna and Propagation**, vol. 33, no. 11, pp. 1227–1236, Nov. 1985.

[51] W. Rotman and R. F. Turner, “Wide-angle microwave lens for line source applications,” **IEEE Transactions on Antenna and Propagation**, vol. 11, pp. 623–632, Nov. 1963.

[52] W. L. Stutzman and G. A. Thiele, **Antenna Theory and Design**, John Wiley and Sons, 1st edition, 1981.

[53] T. Kailath, **Linear Systems**, Prentice-Hall, 2nd edition, 1997.

[54] Goodman, **Introduction to Fourier Optics**, McGraw–Hill, 2nd edition, 1996.

[55] B. Widrow, J. McCool, J. R. Glover jr., J. M. McCool, J. Kaunitz, C. S. Wiliams, R. H. Hearn, J. R. Zeidler, E. Dong jr., and R. C. Goodlin, “Adaptive noise cancelling: Principles and applications,” **Proceedings of IEEE**, vol. 63, pp. 1692–1716, Dec. 1975.

[56] D. E. Dudgeon and R. M. Mersereau, **Multidimensional Digital Signal Processing**, Prentice-Hall, 1st edition, 1984.

[57] B. Widrow and E. Walach, “On the statistical efficiency of the lms algorithm with nonstationary inputs,” **IEEE Transactions on Information Theory**, pp. 211–221, Mar. 1984.

[58] D. C. McLernon, “Analysis of lms algorithm with inputs from cyclostationary random processes,” **Electronics Letters**, pp. 136–138, Jan. 1991.

[59] D. H. Hartman and G. R. Lalk, “Radiant cured optical waveguides on printed circuit boards for photonic interconnection use,” **Applied Optics**, p. 40, Jan. 1989.

[60] L. D. Bamford, P. S. Hall, and A. Fray, “Calculation of antenna mutual coupling form far radiated fields,” **Electronic Letters**, pp. 1299–1301, July 1993.

[61] J. A. G. Malherbe, "Analysis fo a linear antenna array including the effects of mutual coupling," **IEEE Transactions on Education**, pp. 29–34, Feb. 1989.

[62] D. E. J. Humphrey and V. F. Fusco, "A mutual coupling model for microstrip patch antenna pairs with arbitrary orientation," **Microwave and Optical Technology Letters**, pp. 230–233, June 1998.

[63] G. L. Turin, "On optimal diversity reception," **IRE Transactions on Information Theory**, pp. 154–166, July 1961.

[64] S. N. Diggavi, J. J. Shynk, and A. J. Laub, "Directional-of-arrival estimation for a lens-base array," **IEEE Transactions on Antenna and Propagation**, pp. 666–675, May 1994.

APPENDIX A

MATLAB CODE

This appendix contains all the *MatLab* code used in the numerical modeling of the lens arrays and LMS adaptive systems.

Numerical Model Main Code

antenna_pattern.m

```
function [antenna_ptrn]=antenna_pattern(antenna,alpha_range, ki_range)

%this function calculates the antenna pattern in effective length


alpha=[alpha_range(1):(alpha_range(2)-alpha_range(1))/(alpha_range(3)-1):...
   alpha_range(2)];
ki=[ki_range(1):(ki_range(2)-ki_range(1))/(ki_range(3)-1):ki_range(2)];

for k=1:length(alpha)
   for l=1:length(ki)
      [le(k,l),ro{k,l}]=feval(antenna.name,alpha(k),ki(l),antenna.polar);
   end
end

antenna_ptrn.le=le;
antenna_ptrn.polar=ro;
antenna_ptrn.alpha=alpha;
antenna_ptrn.ki=ki;
antenna_ptrn.create='antenna_pattern';
```

arrayTF.m

```
function [array_TF]=arrayTF(array,channel,optional)

%this function creates a tranfer function for channel to array
```

```
% the channel inputs are all unity
%optional.polar_source use polarization for source coupling
%                    default is 'yes'
%optional.polar_feed use polarization for feed coupling
%                    default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                    default is 'yes'
%optional.pathloss_feed use pathloss for feed coupling
%                    default is 'yes'

optional_focus.polar_source='yes';
optional_focus.polar_feed='yes';
optional_focus.pathloss_source='yes';
optional_focus.pathloss_feed='yes';

if nargin == 3
   if isfield(optional,'polar_source')
         optional_focus.polar_source=optional.polar_source;
   end
   if isfield(optional,'polar_feed')
      optional_focus.polar_feed=optional.polar_feed;
   end
   if isfield(optional,'pathloss_source')
      optional_focus.pathloss_source=optional.pathloss_source;
   end
   if isfield(optional,'pathloss_feed')
      optional_focus.pathloss_feed=optional.pathloss_feed;
   end
end



for k=1:length(channel)

   source=channel{k};
      [array_TF(:,k)]=array_focus(array,source,optional_focus);
end
```

array_flux.m

```
function [flux]=array_flux(array,I_in,alpha_range,ki_range)

%this function calculates the total radiated flux from an array whose
%elements have input currents I_in
%alpha_range is the range of alpha angles [alpha_min,alpha_max,alpha_N]
%ki_range is the range of ki angles [ki_min,ki_max,ki_N]
```

```
alpha=[alpha_range(1):(alpha_range(2)-alpha_range(1))/(alpha_range(3)-1):...
   alpha_range(2)];
ki=[ki_range(1):(ki_range(2)-ki_range(1))/(ki_range(3)-1):ki_range(2)];

flux.alpha=alpha;
flux.ki=ki;

for k=1:alpha_range(3)
   comments(['Radiation Pattern ',num2str(k/alpha_range(3),2)]);
   for l=1:ki_range(3)
      if alpha(k)==0 & l~=1
         flux.E_rad{k,l}=flux.E_rad{1,1};
         flux.P_rad(k,l)=flux.P_rad(1,1);
      else
         a_rad=[-1*sin(alpha(k))*cos(ki(l));-1*sin(alpha(k))*sin(ki(l));...
            -1*cos(alpha(k))];

         E_rad_kl=[0;0;0];
         for m=1:array.N_c
            delay_klm=-dot(a_rad,array.cell.nfd.pos{m});

            [antenna_m]=pull_antenna(array,m);
            receive.pos=antenna_m.pos+a_rad;
            [theta_tran,phi_tran]=get_direction(antenna_m,receive);


            [le_klm,ro_klm]=feval(antenna_m.name,theta_tran,phi_tran,...
               antenna_m.polar);
            E_rad_klm=377*((-j*2*pi)/(4*pi))*le_klm*exp(-j*2*pi*delay_klm)*...
               ro_klm;
            E_rad_kl=E_rad_kl+E_rad_klm;
         end
         flux.E_rad{k,l}=E_rad_kl;
         flux.P_rad(k,l)=E_rad_kl'*E_rad_kl/377;
      end
   end
end

flux.total=0;
alpha_step=abs(alpha(2)-alpha(1));
ki_step=abs(ki(2)-ki(1));

for k=1:alpha_range(3)-1
   for l=1:ki_range(3)-1
      E_rad_temp=(flux.E_rad{k,l}+flux.E_rad{k+1,l}+flux.E_rad{k,l+1}+...
            flux.E_rad{k+1,l+1})/4;
      pwr_avg=E_rad_temp'*E_rad_temp/377;
      area=sin((alpha(k)+alpha(k+1))/2)*alpha_step*ki_step;
      flux.total=flux.total+pwr_avg*area;
   end
```

```
end
```

directivity.m

```
function [direct,flux]=directivity(rad_ptrn,optional)

%this function calculates the directivity of a radiation pattern
%it assumes that all points not taken in the pattern have zero power
%flux is the result of the power flux integral
%optional.freq tells which 'measured radiation' pattern to use
switch (rad_ptrn.create)
   case('plot_radiation')
      disp('Please use output of rad_pattern')

   case{'rad_pattern','antenna_pattern','image_pattern'}
      flux=0;
      switch (rad_ptrn.create)
         case 'rad_pattern'
            Voc=rad_ptrn.Voc;
         case 'image_pattern'
            Voc=rad_ptrn.Voc;
            rad_ptrn.alpha=rad_ptrn.theta;
            rad_ptrn.ki=rad_ptrn.phi;
         case 'antenna_pattern'
            Voc=rad_ptrn.le;
      end

      flux=0;
      alpha=rad_ptrn.alpha;
      ki=rad_ptrn.ki;
      alpha_step=abs(alpha(2)-alpha(1));
      ki_step=abs(ki(2)-ki(1));

      for k=1:length(rad_ptrn.alpha)-1
         for l=1:length(rad_ptrn.ki)-1
            Voc_temp=[Voc(k,l),Voc(k+1,l),Voc(k,l+1),Voc(k+1,l+1)];
            pwr_avg=Voc_temp*Voc_temp'/4;
            area=sin((alpha(k)+alpha(k+1))/2)*alpha_step*ki_step;
            flux=flux+pwr_avg*area;
         end
      end


   case('measured')
      flux=0;
      pwr=rad_ptrn.pwr;
      alpha=rad_ptrn.alpha;
      ki=rad_ptrn.ki;
      alpha_step=abs(alpha(2)-alpha(1));
      ki_step=abs(ki(2)-ki(1));

      for k=1:length(rad_ptrn.alpha)-1
         for l=1:length(rad_ptrn.ki)-1
            pwr_avg=sum([pwr(k,l),pwr(k+1,l),pwr(k,l+1),pwr(k+1,l+1)])/4;
            area=sin((alpha(k)+alpha(k+1))/2)*alpha_step*ki_step;
            flux=flux+pwr_avg*area;
         end
      end


      disp('not tested yet need a function to strip out a rad_ptrn from data')
end

direct=(4*pi*max(max(abs(Voc)))^2)/flux;
if strcmp(rad_ptrn.create,'antenna_pattern')
   flux=flux*((2*pi*377)^2)/((4*pi)^2*377);
end
```

get_first_lobe.m

```
function [first_lobe]=get_first_lobe(ptrn,optional)


%this function calculates the peak power of the first side lobe  of
%the a pattern in eigth direction.
%optional.center

if strcmp(ptrn.create,'rad_pattern') | strcmp(ptrn.create,'image_pattern')

   pwr=abs(ptrn.Voc).^2;
   if strcmp(ptrn.create,'rad_pattern')
      ptrn.theta=ptrn.alpha;
      ptrn.phi=ptrn.ki;
   end

   if exist('optional')
      HPBW.center=optional.center;
   else
      [k_list,k_index]=max(pwr);
      [pwr_max,l_index]=max(k_list);
      k_index=k_index(l_index);
      HPBW.center=[ptrn.theta(k_index),ptrn.phi(l_index)];
   end

   x_max=HPBW.center(1)*cos(HPBW.center(2));
   y_max=HPBW.center(1)*sin(HPBW.center(2));
   x_y_points=j*ones(2,8);
```

```
for m=0:7

   r_temp=[0:1/40:2];
   x_temp=cos(m*pi/4)*r_temp+x_max*ones(size(r_temp));
   y_temp=sin(m*pi/4)*r_temp+y_max*ones(size(r_temp));

   for k=1:length(r_temp)
      theta(k)=sqrt(x_temp(k)^2+y_temp(k)^2);
      if theta>.001
         phi(k)=atan2(y_temp(k),x_temp(k));
         if phi(k) <0
            phi(k)=phi(k)+2*pi;
         end
      else
         phi(k)=0;
      end
   end

   [X,Y]=meshgrid(ptrn.phi,ptrn.theta);
   [r_intp]=interp2(X,Y,pwr,phi,theta);

   first=0;
   second=0;
   for k=1:length(r_intp)-1
      if r_intp(k) <= r_intp(k+1) & first==0
         first=1;
         rise=k;
      end

      if r_intp(k) > r_intp(k+1) & first==1 & second==0
         second=1;
         fall=k+1;
      end
   end

   if first==1 & second==1
      [first_lobe.pwr(m+1),index]=max(r_intp([rise:fall]));
      r_lobe=r_temp(rise+index-1);
   else
      r_lobe=0;
   end

   x_y_points(1,m+1)=r_lobe*cos(m*pi/4)+x_max;
   x_y_points(2,m+1)=r_lobe*sin(m*pi/4)+y_max;
end

vector1=[sin(HPBW.center(1))*cos(HPBW.center(2));...
         sin(HPBW.center(1))*sin(HPBW.center(2));...
         cos(HPBW.center(1))];
count=8;
for m=1:8
```

```
      if x_y_points(1,m)~=j
         theta=sqrt(x_y_points(1,m)^2+x_y_points(2,m)^2);
         if theta>.001
            phi=atan2(x_y_points(2,m),x_y_points(1,m));
            if phi < 0
               phi=phi+2*pi;
            end
         else
            phi=0;
         end

         vector2=[sin(theta)*cos(phi);sin(theta)*sin(phi);cos(theta)];
         first_lobe.angles(m)=acos(vector1'*vector2);
      else
         count=count-1;
         first_lobe.angles(m)=0;
      end
   end

   first_lobe.angle_avg=sum(first_lobe.angles)/count;
   first_lobe.pwr_avg=sum(first_lobe.pwr)/count;

else
   disp('use output of rad_pattern or image_pattern')
end
```

get_HPBW.m

```
function [HPBW]=get_HPBW(ptrn,optional)


%this function calculates the HPBW of the a pattern in eigth direction.
%the average angel is the angel from the center to the outward
%optional.center

if strcmp(ptrn.create,'rad_pattern') | strcmp(ptrn.create,'image_pattern')

   Voc=ptrn.Voc;
   if strcmp(ptrn.create,'rad_pattern')
      ptrn.theta=ptrn.alpha;
      ptrn.phi=ptrn.ki;
   end

   [X,Y]=meshgrid(ptrn.phi,ptrn.theta);

   if exist('optional')
      HPBW.center=optional.center;
      [Voc_max]=interp2(X,Y,Voc,HPBW.center(2),HPBW.center(1));
   pwr_max=Voc_max*Voc_max';
   HPBW.pwr_max=pwr_max;
else
   [k_list,k_index]=max(Voc);
   [Voc_max,l_index]=max(k_list);
   k_index=k_index(l_index);
   HPBW.center=[ptrn.theta(k_index),ptrn.phi(l_index)];
   pwr_max=Voc_max*Voc_max';
   HPBW.pwr_max=pwr_max;
end

x_max=HPBW.center(1)*cos(HPBW.center(2));
y_max=HPBW.center(1)*sin(HPBW.center(2));
x_y_points=j*ones(2,8);
for m=0:7

   r_temp=[0:1/40:1];
   x_temp=cos(m*pi/4)*r_temp+x_max*ones(size(r_temp));
   y_temp=sin(m*pi/4)*r_temp+y_max*ones(size(r_temp));

   for k=1:length(r_temp)
      theta(k)=sqrt(x_temp(k)^2+y_temp(k)^2);
      if theta(k)>.001
         phi(k)=atan2(y_temp(k),x_temp(k));
         if phi(k)<0
            phi(k)=phi(k)+2*pi;
         end
      else
         phi(k)=0;
      end
   end

   [Voc_intp]=interp2(X,Y,Voc,phi,theta);
   r_intp=abs(Voc_intp).^2;

   first=0;
   for k=1:length(r_intp)-1
      if r_intp(k) >= pwr_max/2 & r_intp(k+1)<=pwr_max/2 & first==0
         first=1;
         slope=(r_temp(k+1)-r_temp(k))/(r_intp(k+1)-r_intp(k));
         intercept=r_temp(k)-slope*r_intp(k);
         r_3dB=slope*(pwr_max/2)+intercept;

         slope=(Voc_intp(k+1)-Voc_intp(k))/(r_temp(k+1)-r_temp(k));
         intercept=Voc_intp(k) -slope*r_temp(k);
         Voc_3dB=slope*r_3dB+intercept;
         HPBW.phase(m+1)=angle(Voc_3dB)-angle(Voc_max);

         x_y_points(1,m+1)=r_3dB*cos(m*pi/4)+x_max;
         x_y_points(2,m+1)=r_3dB*sin(m*pi/4)+y_max;
      end
      end
   end

   vector1=[sin(HPBW.center(1))*cos(HPBW.center(2));...
            sin(HPBW.center(1))*sin(HPBW.center(2));...
            cos(HPBW.center(1))];
   count=8;
   for m=1:8
      if x_y_points(1,m)~=j
         theta=sqrt(x_y_points(1,m)^2+x_y_points(2,m)^2);
         if theta>.001
            phi=atan2(x_y_points(2,m),x_y_points(1,m));
            if phi<0
               phi=phi+2*pi;
            end
         else
            phi=0;
         end

         vector2=[sin(theta)*cos(phi);sin(theta)*sin(phi);cos(theta)];
         HPBW.angles(m)=acos(vector1'*vector2);
      else
         count=count-1;
         HPBW.angles(m)=0;
      end
   end

   if count==0
      HPBW.avg=0;
   else
      HPBW.avg=sum(HPBW.angles)/count;
   end

else
   disp('use output of rad_pattern or image_pattern')
end
```

image_pattern.m

```
function [image_ptrn,mutual_ptrn]=image_pattern(lens,channel,theta_range,phi_range,optional)

%this function outputs the det_antenna Voc for an omni direction antenna
%polarized to the center element of the lens feed side for the 'channel'
%input over the specified ranges.
%mutual_prtn is the image produced by the mutual coupling currents
%theta_range is a vector [theta_start,theta_end,theta_N]
%phi_range is a vector [phi_start,phi_end,phi_N]
%optional.polar_source use polarization for source coupling
```

```
%                     default is 'yes'
%optional.polar_feed use polarization for feed coupling
%                     default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                     default is 'yes'
%optional.pathloss_feed {'yes' 'no' 'focal'} use pathloss for feed coupling
%                     'focal' uses the focal distance on the optical axis
%                     as the path loss for all antenna elements
%                     default is 'yes'
%optional.surface ('focal',radius) to use the focal surface or a surface of
%                     radius 'radius' default is 'focal'
%optional.det_antenna allows you to define your own detector antenna element
%optional.mutual.use ('yes', 'no','max') calculates the image do to mutual
%                  coupling.  'yes' uses the estimated magnitude and phase
%                  of the coupled currents and 'max' uses just the magnitude
%                  default is 'no'
%optional.mutual.extra is an open field for use by name_mutual.m mutual
%coupling calculations functions

mutual.use='no';
surface='focal';
focal='no';
optional_focus.polar_source='yes';
optional_focus.polar_feed='yes';
optional_focus.pathloss_source='yes';
optional_focus.pathloss_feed='yes';

%initalize default det_antenna
middle=round(lens.N_c/2);
[fd_antenna]=pull_antenna(lens,middle,'feed');
[le_nfd,ro_fd]=feval(fd_antenna.name,0,0,fd_antenna.polar);
J_sph_fd=xyz2sph(0,0);
J_gfd=fd_antenna.local;
ro_source=J_gfd'*J_sph_fd'*ro_fd;
det_antenna.polar=ro_source;
det_antenna.theta_r=0;
det_antenna.name='omni_antenna';
%

if nargin ==5
   if isfield(optional,'polar_source')
      optional_focus.polar_source=optional.polar_source;
   end
   if isfield(optional,'polar_feed')
      optional_focus.polar_feed=optional.polar_feed;
   end
   if isfield(optional,'pathloss_source')
      optional_focus.pathloss_source=optional.pathloss_source;
   end
   if isfield(optional,'pathloss_feed')
      if strcmp(optional.pathloss_feed,'focal')
```

```
            optional_focus.pathloss_feed='no';
            focal='yes';
         else
            optional_focus.pathloss_feed=optional.pathloss_feed;
            focal='no';
         end
      end
      if isfield(optional,'mutual')
         mutual=optional.mutual;
      end
      if isfield(optional,'surface')
         surface=optional.surface;
      end
      if isfield(optional,'det_antenna')
         det_antenna=optional.det_antenna;
      end
   end

   theta=[theta_range(1):(theta_range(2)-theta_range(1))/(theta_range(3)-1):...
         theta_range(2)];
   phi=[phi_range(1):(phi_range(2)-phi_range(1))/(phi_range(3)-1):phi_range(2)];

   Iin_fd=0;
   for k=1:theta_range(3)
      for l=1:phi_range(3)
         comments(['Calculating Image ',num2str(k/theta_range(3),2)]);
         if theta(k)==0 & l~=1
            Voc(k,l)=Voc(1,1);
         else
            if strcmp(surface,'focal')
               [rad]=get_focal(lens, theta(k));
            else
               rad=surface;
            end

            pos=[rad*sin(theta(k))*cos(phi(l));rad*sin(theta(k))*sin(phi(l));...
               rad*cos(theta(k))];
            [detector_kl]=build_det(det_antenna.name,pos,...
               det_antenna.theta_r,1,1,0.5,0.5,0,det_antenna.polar);

            Voc_temp=0;
            for m=1:length(channel)
               source_m=channel{m};

               [Voc_temp2,Iin_fd_m]=lens_focus(lens,source_m,detector_kl,...
                  optional_focus);

               Voc_temp=Voc_temp+Voc_temp2;

               if k==1 & l==1
                  switch(mutual.use)
```

```
                case {'yes','max'}
                    Iin_fd=Iin_fd+Iin_fd_m;
              end
            end
          end
          Voc(k,l)=Voc_temp;
        end

    end
end
if strcmp(focal,'yes')
    image_ptrn.Voc=Voc/get_focal(lens,0);
else
    image_ptrn.Voc=Voc;
end
image_ptrn.theta=theta;
image_ptrn.phi=phi;
image_ptrn.create='image_pattern';

if strcmp(mutual.use,'yes') | strcmp(mutual.use,'max')

    Voc=[];
    [mutual_couple]=get_mutual_Z(lens,optional.mutual);
    switch(mutual.use)
        case{'yes'}
            Iin_fd_mutual=mutual_couple*Iin_fd;
        case{'max'}
            Iin_fd_mutual=abs(mutual_couple)*abs(Iin_fd);
    end

    for k=1:theta_range(3)
        for l=1:phi_range(3)

            comments(['Calculating Mutual ',num2str(k/theta_range(3),2)]);

            if theta(k)==0 & l~=1
                Voc(k,l)=Voc(1,1);
            else
                if strcmp(surface,'focal')
                    [rad]=get_focal(lens, theta(k));
                else
                    rad=surface;
                end

                pos=[rad*sin(theta(k))*cos(phi(l));rad*sin(theta(k))*sin(phi(l));...
                    rad*cos(theta(k))];
                a_z=(-1*pos)/sqrt(pos'*pos);

                [detector_kl]=build_det(det_antenna.name,pos,...
                    det_antenna.theta_r,1,1,0.5,0.5,0,det_antenna.polar);
            [Voc(k,l)]=mutual_focus(lens,Iin_fd_mutual,detector_kl,...
               optional_focus);
         end
      end
   end

if strcmp(focal,'yes')
   mutual_ptrn.Voc=Voc/get_focal(lens,0);
else
   mutual_ptrn.Voc=Voc;
end
mutual_ptrn.theta=theta;
mutual_ptrn.phi=phi;
mutual_ptrn.create='image_pattern';

end
```

local_unit_vectors.m

```
function local_unit_vectors(device,optional)

%local_unit_vectors(device,optional)
%this function plots local unit vectors for antennas in the device
%(lens or detectors)
%optiona.fig_num is optional and default is 4
%optional.font_size is optional allows the letter font size to be changed
%           default is 10
%optional.length length of unit vectors
%           default is 0.25

fig_num=4;

optional_draw.font_size=10;
if nargin==2
   if isfield(optional,'fig_num')
      fig_num=optional.fig_num;
   end
   if isfield(optional,'font_size')
      optional_draw.font_size=optional.font_size;
   end
   if isfield(optional,'length')
      optional_draw.length=optional.length;
   end
end

figure(fig_num)

if findstr(class(device),'cell')
   for k=1:length(device)
```

```
      device_k=device{k};

         switch device_k.device
            case('detector')
               for m=1:device.N_c
                  drawunit(device_k.cell.local{m},device_k.cell.pos{m},'c',...
                     optional_draw)
               end
            case('source')
               switch device_k.type
                  case('nearfield')
                     drawunit(device_k.local,device_k.pos,'m',optional_draw)
                  case('farfield')
                     drawunit(device_k.local,device_k.pos,'m',optional_draw)
               end
         end
   end
else
   switch device.device
      case('lens')
         for m=1:device.N_c
            drawunit(device.cell.nfd.local{m},device.cell.nfd.pos{m},'g',...
               optional_draw)
            drawunit(device.cell.fd.local{m},device.cell.fd.pos{m},'c',...
               optional_draw)
         end
      case('array')
         for m=1:device.N_c
            drawunit(device.cell.nfd.local{m},device.cell.nfd.pos{m},'g',...
               optional_draw)
         end
      case('detector')
         for m=1:device.N_c
            drawunit(device.cell.local{m},device.cell.pos{m},'c',optional_draw)
         end
      case('source')
         switch device.type
            case('nearfield')
               drawunit(device.local,device.pos,'m',optional_draw)
            case('farfield')
               drawunit(device.local,device.pos,'m',optional_draw)
         end
   end
end
```

make_channel.m

```
function [channel]=make_channel(alpha_max,alpha_sample)
```

```
%this function uniformly places far field sources over a surfcae from 0
%to alpha_max

alpha=[0:alpha_max/(alpha_sample-0.5):alpha_max]
alpha_step=alpha(2)-alpha(1);


channel_num=1;
channel{channel_num}=build_far(0,0,pi/2,1,[0;1;0]);
for k=2:length(alpha)
   ki_sample=floor(sin(alpha(k))*2*pi/alpha_step);

   if floor(k/2)==k/2
      ki=[0:2*pi/ki_sample:2*pi-2*pi/ki_sample];
   else
      ki=[0+pi/ki_sample:2*pi/ki_sample:2*pi-pi/ki_sample];
   end

   for l=1:ki_sample
      channel_num=channel_num+1
      channel{channel_num}=build_far(alpha(k),ki(l),pi/2,1,[0;1;0]);
   end
end
```

make_imager.m

```
function [imager]=make_imager(lens,detector,theta_max,optional)

%this function places the detectors on the image_ptrnr surface spaced according
%the Nyquist sampling therom. I use a small angle approximation that results
%in alittle over sampling for lens smaller than radius=3 lambda
%'detector' is the element used
%( right now only written for single antenna for use in '2D' lens, arrays
% detectors are for '1D' lenses)
%'theta_max' range from 0 to theta_max to place the detectors
%optional.image_ptrn will use image pattern to calculate HPBW for sampling
%                 use image with beam on optical axis
%                 default is to estimate HPBW from lens size
%optional.sampling_scale   ratio of desired sampling rate to Nyquist sampling rate
%                    default is 1
%optional.surface ('focal',num) which focal surface to use focal surface or
%                       a ball of radius 'num'
%                 default is focal surface
%optional.fit ('ceil','floor') ceil packs the detectors such the whole angle
%           space is covered with max spacing no greater than Nyquist
%           floor packs the detectors starting at the center with Nyquist
%           spacing until theta_max is reached
%           default is floor
```

```
surface='focal';
sampling_scale=1;
estimate='yes';
fit='floor';


if nargin==4
   if isfield(optional,'sampling_scale')
      sampling_scale=optional.sampling_scale;
   end
   if isfield(optional,'surface')
      surface=optional.surface;
   end
   if isfield(optional,'image_ptrn')
      image_ptrn=optional.image_ptrn;
      estimate='no';
   end
   if isfield(optional,'fit')
      fit=optional.fit;
   end
end

switch(estimate)
   case('yes')
      switch(lens.lensing)
         case('2D')
            switch(lens.shape)
               case('circle')
                  HPBW=2*asin(.255/lens.dimension);
               case('rect')
                  HPBW(1)=2*asin(1.4/(lens.dimension(1)*pi));
                  HPBW(2)=2*asin(1.4/(lens.dimension(2)*pi));
            end
         case('1Dx')
            switch(lens.shape)
               case('circle')
                  disp('warning this code does not work with 1D circle lenses')
               case('rect')
                  HPBW(1)=2*asin(1.4/(lens.dimension(1)*pi));
                  HPBW(2)=0;
            end
         case('1Dy')
            switch(lens.shape)
               case('circle')
                  disp('warning this code does not work with 1D circle lenses')
               case('rect')
                  HPBW(1)=0;
                  HPBW(2)=2*asin(1.4/(lens.dimension(2)*pi));
            end
      end
   case('no')
      HPBW=get_HPBW(image_ptrn);
      switch(lens.lensing)
         case('2D')
            switch(lens.shape)
               case('circle')
                  HPBW=2*HPBW.avg;
               case('rect')
                  HPBW(1)=sum(HPBW.angles([1,5]))/2;
                  HPBW(2)=sum(HPBW.angles([3,7]))/2;
            end
         case('1Dx')
            switch(lens.shape)
               case('circle')
                  disp('warning this code does not work with 1D circle lenses')
               case('rect')
                  HPBW(1)=sum(HPBW.angles([1,5]))/2;
                  HPBW(2)=0;
            end
         case('1Dy')
            switch(lens.shape)
               case('circle')
                  disp('warning this code does not work with 1D circle lenses')
               case('rect')
                  HPBW(1)=0;
                  HPBW(2)=sum(HPBW.angles([3,7]))/2;
            end
      end
end

switch(lens.lensing)
   case('2D')
      switch(lens.shape)
         case('circle')
            sample=HPBW*1.13*0.5*[1 1 ;sqrt(3) -sqrt(3)];
         case('rect')
            sample(:,1)=sampling_scale*HPBW(1)*1.12*[1;0];
            sample(:,2)=sampling_scale*HPBW(2)*1.12*[0;1];
      end
   case('1Dx')
      switch(lens.shape)
         case('circle')
            disp('warning this code does not work with 1D circle lenses')
         case('rect')
            sample(:,1)=sampling_scale*HPBW(1)*1.12*[1;0];
            sample(:,2)=[0;0];
      end
   case('1Dy')
      switch(lens.shape)
         case('circle')
            disp('warning this code does not work with 1D circle lenses')
         case('rect')
            sample(:,1)=[0;0];
            sample(:,2)=sampling_scale*HPBW(2)*1.12*[0;1];
      end
end

switch(lens.lensing)
   case('2D')
      switch(lens.shape)
         case('circle')
            x_space=2*abs(sample(1,1));
            switch(fit)
               case('ceil')
                  num_u1=ceil(theta_max/x_space);
                  num_u2=num_u1;
                  sample_new=theta_max/num_u1;
                  sample=(sample_new/x_space)*sample;
               case('floor')
                  num_u1=floor(theta_max/x_space);
                  num_u2=num_u1;
            end
         case('rect')
            x_space=abs(sample(1,1))
            y_space=abs(sample(2,2))
            switch(fit)
               case('ceil')
                  num_u1=ceil(theta_max/x_space);
                  sample_new=theta_max/num_u1;
                  sample(:,1)=(sample_new/x_space)*sample(:,1);
                  num_u2=ceil(theta_max/y_space);
                  sample_new=theta_max/num_u2;
                  sample(:,2)=(sample_new/y_space)*sample(:,2);
               case('floor')
                  num_u1=floor(theta_max/x_space);
                  num_u2=floor(theta_max/y_space);
            end
      end

      pos=[];
      for k=-num_u1-1:num_u1+1
         for l=-num_u2-1:num_u2+1
            pos_temp=sample*[k;l];
            if sqrt(pos_temp'*pos_temp)/theta_max < 1.04
               pos=[pos,pos_temp];
            end
         end
      end

   case('1Dx')
      switch(lens.shape)
         case('circle')
```

```
            disp('warning this code does not work with 1D circle lenses')
         case('rect')
            x_space=abs(sample(1,1));
            switch(fit)
               case('ceil')
                  num_u1=ceil(theta_max/x_space);
                  sample_new=theta_max/num_u1;
                  sample(:,1)=(sample_new/x_space)*sample(:,1);
               case('floor')
                  num_u1=floor(theta_max/x_space);
            end
      end

      pos=[];
      for k=-num_u1-1:num_u1+1
         pos_temp=sample*[k;0];
         if sqrt(pos_temp'*pos_temp)/theta_max < 1.04
            pos=[pos,pos_temp];
         end
      end

   case('1Dy')
      switch(lens.shape)
         case('circle')
            disp('warning this code does not work with 1D circle lenses')
         case('rect')
            y_space=abs(sample(2,2));
            switch(fit)
               case('ceil')
                  num_u2=ceil(theta_max/y_space);
                  sample_new=theta_max/num_u2;
                  sample(:,2)=(sample_new/y_space)*sample(:,2);
               case('floor')
                  num_u2=floor(theta_max/y_space);
            end
      end

      pos=[];
      for k=-num_u2-1:num_u2+1
         pos_temp=sample*[0;k];
         if sqrt(pos_temp'*pos_temp)/theta_max < 1.04
            pos=[pos,pos_temp];
         end
      end
end

[m,n]=size(pos);

for k=1:n
   pos_temp=pos(:,k);
   if detector.N_c==1
```

```
      theta_temp=sqrt(pos_temp'*pos_temp);
      if theta_temp>.01
         phi_temp=atan2(pos_temp(2),pos_temp(1));
         if phi_temp<0
            phi_temp=phi_temp+2*pi;
         end
      else
         phi_temp=0;
      end

      focal_temp=get_focal(lens, theta_temp);
      pos_xyz=[focal_temp*sin(theta_temp)*cos(phi_temp);...
               focal_temp*sin(theta_temp)*sin(phi_temp);...
               focal_temp*cos(theta_temp)];
      a_z=(-1*pos_xyz')/sqrt(pos_xyz'*pos_xyz);
      detector_temp=detector;
      [antenna_temp]=build_ant(detector.cell.name{1},pos_xyz,...
         a_z,detector.cell.theta_r{1},detector.cell.polar{1});
      detector_temp.cell.pos(1)=antenna_temp.pos(1);
      detector_temp.cell.name(1)=antenna_temp.name(1);
      detector_temp.cell.base(1)=antenna_temp.base(1);
      detector_temp.cell.theta_r(1)=antenna_temp.theta_r(1);
      detector_temp.cell.local(1)=antenna_temp.local(1);
      detector_temp.cell.polar(1)=antenna_temp.polar(1);

      imager{k}=detector_temp;
   else
      disp('need to write code that places arrays')
   end
end
```

noiseTF.m

```
function [noise_TF]=noiseTF(lens,imager,optional)

%this function creates a transfer function that related the noise sources
%(amplifiers) to the detector outputs.
%noise_TF.lens is the lens sources
%noise_TF.det is cell structure containing the TF for each detectors unitcell
%noise_TF.imager is the noise source at each port of the imager before sampling
%The noise doesn't go through the amplifiers.
%optional.delay_lens ('before', 'after') default 'before'
%optional.delay_detector ('before', 'after') default 'before'
%optional.polar use polarization for feed coupling
%                     default is 'yes'
%optional.pathloss use pathloss for feed coupling
%                     default is 'yes'
%optional.flags flags the turn on/off the calculates of TF
%                  [lens,detector,imager]
%                  default is [1,1,1]

delay_lens='before';
delay_detector='before';
optional_feed.polar='yes';
optional_feed.pathloss='yes';
flag=[1,1,1];

if nargin==3
   if isfield(optional,'delay_lens')
      delay_lens=optional.delay_lens;
   end

   if isfield(optional,'delay_detector')
      delay_detector=optional.delay_detector;
   end

   if isfield(optional,'polar')
      optional_feed.polar=optional.polar;
   end

   if isfield(optional,'pathloss')
      optional_feed.pathloss=optional.pathloss;
   end

   if isfield(optional,'flags')
      flags=optional.flags;
   end
end

if flags(1)==1
   for k=1:lens.N_c
      %comments(['QOL noise TF ',num2str(k/lens.N_c,3)]);
      for m=1:length(imager)
         detector=imager{m};
         [fd_antenna]=pull_antenna(lens,k,'feed');

         switch(delay_lens)
            case 'before'
               Iin_fd=exp(-j*2*pi*lens.cell.delay(k));
            case 'after'
               Iin_fd=1;
         end

         Voc=0;
         for l=1:detector.N_c
            [det_antenna]=pull_antenna(detector,l);
            [Voc_l]=couple(fd_antenna,det_antenna,Iin_fd,optional_feed);
            switch(delay_detector)
               case 'before'
                  Voc=Voc+Voc_l*detector.g*detector.cell.g{l}*...
```

```
                     exp(-j*2*pi*detector.cell.delay{l});
            case 'after'
               Voc=Voc+Voc_l*detector.g*detector.cell.g{l};
            end
         end
         Voc=Voc*detector.g;

         noise_TF.lens(m,k)=Voc;
      end
   end
end


if flags(2)==1
   for k=1:length(imager)
      detector=imager{k};

      noise_det_temp=zeros(1,detector.N_c);

      for l=1:detector.N_c
         switch(delay_detector)
            case('before')
               noise_det_temp(l)=detector.g*exp(-j*2*pi*detector.cell.delay{l});
            case('after')
               noise_det_temp(l)=detector.g;
         end
      end
      noise_TF.det{k}=noise_det_temp;
   end
end

if flags(3)==1
   noise_TF.imager=eye(length(imager));
end
```

number_elements.m

```
function number_elements(device,optional)

%this function numbers the antennas in the device (lens detectors channel
%imager)
%optiona.fig_num is optional and default is 4
%optional.font_size this optional allows the fonts size to be changed
%              default is 10

fig_num=4;
font_size=10;
if nargin==2
   if isfield(optional,'fig_num')
```

```
      fig_num=optional.fig_num;
   end
   if isfield(optional,'font_size')
      font_size=optional.font_size;
   end
end

figure(fig_num)

if findstr(class(device),'cell')
   for k=1:length(device)
      device_k=device{k};

         switch device_k.device
            case('detector')
               for m=1:device_k.N_c
                  pos=device_k.cell.pos{m};
                  [H_temp]=text(pos(1),-1*pos(3),pos(2),...
                     [num2str(k,2),',',num2str(m,3)]);
                  set(H_temp,'FontSize',font_size)
               end
            case('source')
               pos=device_k.pos;
               switch device_k.type
                  case('nearfield')
                     [H_temp]=text(pos(1),-1*pos(3), pos(2),num2str(k,3));
                     set(H_temp,'FontSize',font_size)
                  case('farfield')
                     [H_temp]=text(pos(1),-1*pos(3),pos(2),num2str(k,3));
                     set(H_temp,'FontSize',font_size)
               end
         end
   end
else
   switch device.device
      case('lens')
         for m=1:device.N_c
            pos=device.cell.nfd.pos{m};
            [H_temp]=text(pos(1),-1*pos(3),pos(2),num2str(m,3));
            set(H_temp,'FontSize',font_size)
         end
      case('array')
         for m=1:device.N_c
            pos=device.cell.nfd.pos{m};
            [H_temp]=text(pos(1),-1*pos(3),pos(2),num2str(m,3));
            set(H_temp,'FontSize',font_size)
         end
      case('detector')
         for m=1:device.N_c
            pos=device.cell.pos{m};
            [H_temp]=text(pos(1),-1*pos(3),pos(2),num2str(m,3));
            set(H_temp,'FontSize',font_size)
         end
   end
end
```

per_error.m

```
function [error_ptrn]=per_error(image_ptrn, mutual_ptrn)

%this function plots the percent error do to mutual coupling


error_ptrn=image_ptrn;
error_ptrn.create='per_error';
error_ptrn.error_point=abs(mutual_ptrn.Voc./image_ptrn.Voc);
temp_max=max(max(abs(image_ptrn.Voc)));
error_ptrn.error_max=abs(mutual_ptrn.Voc./(temp_max*ones(size(mutual_ptrn.Voc))));
```

planar_array.m

```
function [array]=planar_array(shape,dimension,antenna,unitcellsize,latice)

%Shape is a string argument that discribes the lens shape 'circle','rect'
%dimension is a vector that contains the important dimensions for the array
%  circle [radius]
%  rect [width,height] width = x dimension and height = y dimension
%antenna is cell discribes the antenna types,
%polarization and orientation.  The polarization vector is for a model
%antenna in the oreintation for the theorical or measured radiation patterns.
%Orientation discribes the rotation of the antennas from the model antenna.
%unitcellsize defines the x/y dimensions for unitcell
%  [width,height] width = x  height = y
%latice is a cell of strings that discribes the how the unitcells are
%placed in the lens {pattern,boundary}
%  PATTERN
%  'triangle' uses a triangular lattice to fill the shape trying to maximise
%  the number of unit cells
%  'max'  uses a row structure with as many unitcells as possible in
%  each row
%  BOUNDARY
%  'best' is the tightest packing allowing for 1/4 of the unitcell
%  to extend outside the defined shape
%  'limit' is the tightest packing without letting the unitcells
%  extend outside the defined shape
%note: all dimensions are in wave length of the center frequency
%note: this code assumes for now that the lens is confined to a plane
```

```
%initialize array
array.device='array';
array.shape=shape;
array.dimension=dimension;
array.cell.width=unitcellsize(1);
array.cell.height=unitcellsize(2);

switch shape
   case {'circle'}
      [array]=packcircle(latice,array);
   case {'rect'}
      [array]=packrect(latice,array);
end

array.N_c=sum(array.R);
[array]=packant2(array,antenna);
```

plot_antenna.m

```
function plot_antenna(antenna_ptrn,parameter,optional)

%this function plots the antenna_ptrn pwr or polarization
%parameter {'pattern','polar'}  the pattern is on a linear scale
%                              the polarization is plotted 1/8 of effective
%                              length circular polarization plots incorrectly
%optional.data {'raw', 'norm' ,num} default is the  normalize the max power
%                                    to 0 dB or the max power can be normalize
%                                    to 'num' in dB
%optional.export {'screen', 'file'}  this option ajusts the colors for
%                                    screen viewing or file exporting

figure(7)
whitebg('k')
view(3)

data='norm';
export='screen';
if nargin==3
   if isfield(optional,'data')
      data=optional.data;
   end
   if isfield(optional,'export')
      export=optional.export;
   end
end

switch (parameter)
   case 'pattern'
      colormap('jet')
      pwr=antenna_ptrn.le.*antenna_ptrn.le;

      max_pwr=max(max(pwr));

      if strcmp(data,'norm')
         pwr=pwr/max_pwr;
      elseif strcmp(class(data),'double')
         data=10^(data/10);
         pwr=(data/max_pwr)*pwr;
      end
%      pwr=10*log10(pwr);

      for k=1:length(antenna_ptrn.alpha)
         for l=1:length(antenna_ptrn.ki)
            z(k,l)=pwr(k,l)*cos(antenna_ptrn.alpha(k));
            y(k,l)=pwr(k,l)*sin(antenna_ptrn.alpha(k))*sin(antenna_ptrn.ki(l));
            x(k,l)=pwr(k,l)*sin(antenna_ptrn.alpha(k))*cos(antenna_ptrn.ki(l));
         end
      end

      surf(x,y,z,x.^2+y.^2+z.^2)
      %colorbar
      xlabel('x')
      ylabel('y')
      zlabel('z')
      axis('equal')
      case 'polar'
         drawball(1)
         colormap('gray');
         switch(export)
            case 'screen'
               caxis([1,1.1]);
            case 'file'
               caxis([0,0.1]);
         end
         hold on
         for k=1:length(antenna_ptrn.alpha)
            for l=1:length(antenna_ptrn.ki)
               if antenna_ptrn.alpha(k)==0 & antenna_ptrn.ki~=0

               else
               z_temp=cos(antenna_ptrn.alpha(k));
               y_temp=sin(antenna_ptrn.alpha(k))*sin(antenna_ptrn.ki(l));
               x_temp=sin(antenna_ptrn.alpha(k))*cos(antenna_ptrn.ki(l));
               [J_sph]=xyz2sph(antenna_ptrn.alpha(k),antenna_ptrn.ki(l));
               polar_vector= J_sph'*antenna_ptrn.polar{k,l}/8;
               plot3([x_temp,x_temp+polar_vector(1)],...
                     [y_temp,y_temp+polar_vector(2)],...
                     [z_temp,z_temp+polar_vector(3)],'g');
               plot3(x_temp,y_temp,z_temp,'go');
               end
            end
         end
         hold off
         xlabel('x')
         ylabel('y')
         zlabel('z')
         axis('equal')
end
```

plot_array.m

```
function plot_array(array,optional)

%plot_array(array,optional)
%This function plots a 3d picture of the array in figure 4
%optional.fig_num default is 4

fig_num=4;
if nargin==2
   fig_num=optional.fig_num;
end

whitebg('k')
figure(fig_num)
view(3)

for m=1:array.N_c
   drawrect(array.cell.nfd.base{m},array.cell.nfd.pos{m},array.cell.height, ...
      array.cell.width,'r');

   hold on
   plot3(array.cell.nfd.pos{m}(1),-1*array.cell.nfd.pos{m}(3),  ...
      array.cell.nfd.pos{m}(2),'rx');

end

hold on
switch array.shape
   case{'circle'}
      angle=[0:pi/15:2*pi];
      D=2*array.dimension;
      x_circle=(D/2)*cos(angle);
      y_circle=(D/2)*sin(angle);
      z_circle=zeros(1,length(angle));
      plot3(x_circle,-1*z_circle,y_circle,'r')
   case {'rect'}
      w=array.dimension(1);
      h=array.dimension(2);
      x=[-w/2, -w/2, w/2, w/2, -w/2];
      y=[h/2, -h/2, -h/2, h/2, h/2];
      z=zeros(1,5);
      plot3(x,-1*z,y,'r')

end
hold off
axis('equal')

xlabel('x')
ylabel('-z')
zlabel('y')
```

plot_channel.m

```
function plot_channel(channel,optional)

%This function plots a 3d picture of the channel in figure 4
%optional.fig_num default is 4
%optional.radius changes the size of source symbol radius
%        default is 0.25
%optional.distance changes the distance the far field source symbols
%        are plotted default is 6

optional_plot.fig_num=4;
optional_plot.radius=0.25;
if nargin==2
   if isfield(optional,'fig_num')
      optional_plot.fig_num=optional.fig_num;
   end
   if isfield(optional,'radius')
      optional_plot.radius=optional.radius;
   end
   if isfield(optional,'distance')
      optional_plot.source.pos=(optional.distance/6)*source.pos;
   end
end

for k=1:length(channel)
   source_k=channel{k};
   plot_source(source_k,optional_plot);
end
```

plot_couple.m

```
function  plot_couple(transmit,receive,optional)

%function  plot_couple(transmit,receive,optional)
```

```
%this function plots the coupling between two antennas 'transmit' (red) and
%'receive' (blue). Only linear polarization and 1/r loss is used.  All
%antenna fields are required (name,polar,bae,local,pos,theta_r).
%optional.pathloss default is 'yes'
%optional.fig_num default is 5

fig_num=5;
pathloss='yes';
if nargin==3
   if isfield(optional,'pathloss')
      pathloss=optional.pathloss;
   end

   if isfield(optional,'fig_num')
      fig_num=optional.fig_num;
   end
end

figure(fig_num)
view(3)
whitebg('k')

drawrect(transmit.local,transmit.pos,.5,.5,'r')
drawrect(receive.local,receive.pos,.5,.5,'g')

drawunit(transmit.local,transmit.pos,'r')
drawunit(receive.local,receive.pos,'g')

[theta_rec,phi_rec]=get_direction(receive,transmit);
[theta_tran,phi_tran]=get_direction(transmit,receive);
J_sph_tran=xyz2sph(theta_tran,phi_tran);
J_sph_rec=xyz2sph(theta_rec,phi_rec);
J_gt=transmit.local;
J_gr=receive.local;

Rt=transmit.pos;
Rr=receive.pos;
r=Rr-Rt;
r_mag=sqrt(r'*r);

[le_t,ro_t]=feval(transmit.name,theta_tran,phi_tran,transmit.polar);
ro_t_l=J_gt'*J_sph_tran'*ro_t;

switch pathloss
   case ('no')
      Ei_g=((2*pi)/(4*pi))*le_t*ro_t_l;
   case('yes')
      Ei_g=((2*pi)/(4*pi*r_mag))*le_t*ro_t_l;
end

pos=receive.pos;
```

```
xpoint=(Ei_g)+pos;

hold on
plot3([pos(1),xpoint(1)],-1*[pos(3),xpoint(3)],[pos(2),xpoint(2)],'y')
plot3([transmit.pos(1),receive.pos(1)],-1*[transmit.pos(3),receive.pos(3)], ...
      [transmit.pos(2),receive.pos(2)],'b')

axis('equal')

hold off
grid on
```

plot_detector.m

```
function plot_detector(detector,optional)

%this function plots a 3d picture of the detector in figure 4
%optional.fig_num default is 4

fig_num=4;
if nargin==2
   fig_num=optional.fig_num;
end

whitebg('k')
figure(fig_num)
view(3)
connecting=[];
for m=1:detector.N_c
   drawrect(detector.cell.base{m},detector.cell.pos{m},detector.cell.height, ...
      detector.cell.width,'g');

   hold on
   plot3(detector.cell.pos{m}(1),-1*detector.cell.pos{m}(3),  ...
      detector.cell.pos{m}(2),'rx');
   hold off
   connecting=[connecting,detector.cell.pos{m}];
end
hold on
plot3(connecting(1,:),-1*connecting(3,:),connecting(2,:),'g')
hold off
axis('equal')
```

plot_detector_pos.m

```
function plot_detector_pos(imager)

%this function plots detectors positions on image plot

figure(2);
whitebg('w')
for k=1:length(imager)
   detector_k=imager{k};
   pos_k=detector_k.cell.pos{1};

   theta_k=atan2(sqrt(pos_k(1)^2+pos_k(2)^2),pos_k(3));

   if theta_k > .001
      phi_k=atan2(pos_k(2),pos_k(1));
      if phi_k < 0
         phi_k=phi_k+2*pi;
      end
   else
      phi_k=0;
   end

   x_k=theta_k*cos(phi_k);
   y_k=theta_k*sin(phi_k);
   hold on
   plot3(x_k,y_k,0,'kx')
   text(x_k,y_k,0,num2str(k,3))
   hold off
end
```

plot_error.m

```
function [error_ptrn_plot]=plot_error(error_ptrn,ref,optional)


%this function plots the error pattern and outputs the plotted data
%'ref' determines which percent error calculation to plot
%     'point' is point by point calculation
%     'max' percent error referenced to max power of image pattern
%optional.scale  ('linear','dB') default is dB
%optional.data ('raw',num) default  num is the max scale limit
%optional.format ('polar','rect') default is polar representation of
%                                  theta and phi where the imageius is
%                                  linear with theta
%optional.theta_range [theta_start,theta_end] if theta_start = theta_end
%                               it plots is a cut with theta=theta_start
%optional.phi_range [phi_start,phi_end] if phi_start = phi_end then it produces
%                                     the plot is a cut with phi=phi_start
%if either of the theta or phi ranges are of length one, then plot_image
```

```
%allows stacking of 2D plots and optional.line_color defines the line
%type and color (default is blue solid)
%if the output of one plot_image is feed into another plot_image
%then the data is reploted ignoring all optional's
%if measured data if feed is used, then the optional.freq determines
%which pattern is used

scale='linear';
data=100;
format='polar';
figure(5)
line_color='b-';

if strcmp(ref,'point')
   error_ptrn.error=error_ptrn.error_point;
elseif strcmp(ref,'max')
   error_ptrn.error=error_ptrn.error_max;
   end

if nargin==3
   if isfield(optional,'line_color')
      line_color=optional.line_color;
   end
end

switch (error_ptrn.create)
   case('plot_error')
      if isfield(error_ptrn,'x')
         surf(error_ptrn.x,error_ptrn.y,error_ptrn.error)
         title(['theta is from ',num2str(error_ptrn.theta_range(1)/pi,2),'-'...
            num2str(error_ptrn.theta_range(2)/pi,2),' (pi)'])
         xlabel('x')
         ylabel('y')
         zlabel(['percent error (',scale,')'])
         axis('square')
      else
         if length(error_ptrn.theta)==1
            hold on
            plot(error_ptrn.phi/pi,error_ptrn.error,line_color)
            title(['theta = ',num2str(error_ptrn.theta(1))]);
            xlabel('phi (pi)')
            ylabel(['pwr (',scale,')'])
            hold off
         elseif length(error_ptrn.phi)==1
            hold on
            plot(error_ptrn.theta/pi,error_ptrn.error,line_color)
            title(['theta = ',num2str(error_ptrn.phi(1))]);
            xlabel('theta (pi)')
            ylabel(['percent error (',scale,')'])
            hold off
         else
```

```
            surf(error_ptrn.phi/pi,error_ptrn.theta/pi,error_ptrn.error);
            xlabel('phi (pi)')
            ylabel('theta (pi)')
            zlabel(['percent error (',scale,')'])
         end
      end
   case{'per_error'}
      if nargin ==3
         if isfield(optional,'scale')
            scale=optional.scale;
         end
         if isfield(optional,'data')
            data=optional.data;
         end
         if isfield(optional,'format')
            format=optional.format;
         end
         if isfield(optional,'theta_range')
            theta_range=optional.theta_range;
            [Y,start_temp]=min(abs(error_ptrn.theta-theta_range(1)*...
               ones(size(error_ptrn.theta))));
            [Y,end_temp]=min(abs(error_ptrn.theta-theta_range(2)*...
               ones(size(error_ptrn.theta))));
            error_ptrn.theta=error_ptrn.theta([start_temp:end_temp]);
            error_ptrn.error=error_ptrn.error([start_temp:end_temp],:);
         end
         if isfield(optional,'phi_range')
            phi_range=optional.phi_range;
            [Y,start_temp]=min(abs(error_ptrn.phi-phi_range(1)*...
               ones(size(error_ptrn.phi))));
            [Y,end_temp]=min(abs(error_ptrn.phi-phi_range(2)*...
               ones(size(error_ptrn.phi))));
            error_ptrn.phi=error_ptrn.phi([start_temp:end_temp]);
            error_ptrn.error=error_ptrn.error(:,[start_temp:end_temp]);
         end
      end

      error_data=error_ptrn.error;

      if strcmp(class(data),'double')
         [m,n]=size(error_data);
         for k=1:m
            for l=1:n
               if error_data(k,l)>data
                  error_data(k,l)=data;
               end
            end
         end
      end

      if strcmp(scale,'dB')
         error_data=20*log10(error_data);
      end

      if length(error_ptrn.theta)==1
         hold on
         plot(error_ptrn.phi/pi,pwr,line_color)
         title(['theta = ',num2str(error_ptrn.theta(1))]);
         xlabel('phi (pi)')
         ylabel(['percent error (',scale,')'])
         error_ptrn_plot.error=error_data;
         error_ptrn_plot.theta=error_ptrn.theta;
         error_ptrn_plot.phi=error_ptrn.phi;
         hold off
      elseif length(error_ptrn.phi)==1
         hold on
         plot(error_ptrn.theta/pi,pwr,line_color)
         title(['theta = ',num2str(error_ptrn.phi(1))]);
         xlabel('theta (pi)')
         ylabel(['percent error (',scale,')'])
         error_ptrn_plot.error=error_data;
         error_ptrn_plot.theta=error_ptrn.theta;
         error_ptrn_plot.phi=error_ptrn.phi;
         hold off
      else
         switch (format)
            case ('polar')
               for k=1:length(error_ptrn.theta)
                  for l=1:length(error_ptrn.phi)
                     x(k,l)=error_ptrn.theta(k)*cos(error_ptrn.phi(l));
                     y(k,l)=error_ptrn.theta(k)*sin(error_ptrn.phi(l));
                  end
               end
               surf(x,y,error_data)
               title(['theta is from ',num2str(error_ptrn.theta(1)/pi,2),'-'...
                  num2str(error_ptrn.theta(length(error_ptrn.theta))/pi,2),...
                  ' (pi)'])
               xlabel('x')
               ylabel('y')
               zlabel(['percent error(',scale,')'])
               axis('square')
               error_ptrn_plot.error=error_data;
               error_ptrn_plot.x=x;
               error_ptrn_plot.y=y;
               error_ptrn_plot.theta_range=[error_ptrn.theta(1),...
                  error_ptrn.theta(length(error_ptrn.theta))];
            case('rect')
               surf(error_ptrn.phi/pi,error_ptrn.theta/pi,pwr);
               xlabel('phi (pi)')
               ylabel('theta (pi)')
               zlabel(['percent error (',scale,')'])
               error_ptrn_plot.error=error_data;
```

```
               error_ptrn_plot.theta=error_ptrn.theta;
               error_ptrn_plot.phi=error_ptrn.phi;
         end
      end
      error_ptrn_plot.create='plot_error';
end
```

plot_focal.m

```
function   plotfocl(lens,optional)

%plotfocl(lens,max_theta)
%This function plots the focal surface
%optional.surface ['focal',radius] chooses between the focal surface and
%                  a sphere of radius of radius 'radius'
%                  default is 'focal'
%optional.theta_max theta_max is the max angle for ploting the focal surface
%               default is theta_max=pi/4
%optional.rings  the number of ring plotted
%               default is 6
%optional.fig_num is the numb of the figure the surface will be plotted in
%               default is figure 4
%optional.color is the color for the surface
%               default is 'c' cyan

surface='focal';
theta_max=pi/4;
rings=6;
fig_num=4;
color='c';

if nargin==2
   if isfield(optional,'surface')
      surface=optional.surface;
   end
   if isfield(optional,'theta_max')
      theta_max=optional.theta_max;
   end
   if isfield(optional,'rings')
      rings=optional.rings;
   end
   if isfield(optional,'fig_num')
      fig_num=optional.fig_num;
   end
   if isfield(optional,'color')
      color=optional.color;
   end
end
```

```
figure(fig_num)
view(3)
phi=[0:pi/10:2*pi];
theta=[theta_max/rings:theta_max/rings:theta_max];
phi_line=[0:pi/4:2*pi];

X_line=[];
Y_line=[];
Z_line=[];
for k=1:length(theta)
   if strcmp(surface,'focal')
      focal_rad=get_focal(lens,theta(k));
   else
      focal_rad=surface;
   end

   x_circle=-1*focal_rad*sin(theta(k))*cos(phi);
   y_circle=-1*focal_rad*sin(theta(k))*sin(phi);
   z_circle=focal_rad*cos(theta(k))*ones(1,length(phi));

   hold on
   plot3(x_circle,-1*z_circle,y_circle,color)
   hold off

   x_temp=-1*focal_rad*sin(theta(k))*cos(phi_line);
   y_temp=-1*focal_rad*sin(theta(k))*sin(phi_line);
   z_temp=focal_rad*cos(theta(k))*ones(1,length(phi_line));

   X_line=[X_line;x_temp];
   Y_line=[Y_line;y_temp];
   Z_line=[Z_line;z_temp];

end

hold on

        plot3(X_line,-1*Z_line,Y_line,color)

hold off
xlabel('x')
ylabel('-z')
zlabel('y')
axis('equal')
```

plot_image.m

```
function [image_ptrn_plot]=plot_image(image_ptrn,optional)

%this function plots the image pattern and outputs the plotted data
```

```
%optional.scale  ('linear','dB') default is dB
%optional.data ('raw','norm',num) default is normalize to max unity
%                                 num is the normalizaton of the max power
%                                to this value in the scan range and
%                                scale specified
%optional.format ('polar','rect') default is polar representation of
%                                   theta and phi where the imageius is
%                                   linear with theta
%optional.theta_range [theta_start,theta_end] if theta_start = theta_end
%                                it plots is a cut with theta=theta_start
%optional.phi_range [phi_start,phi_end] if phi_start = phi_end then it produces
%                                      the plot is a cut with phi=phi_start
%if either of the theta or phi ranges are of length one, then plot_image
%allows stacking of 2D plots and optional.line_color defines the line
%type and color (default is blue solid)
%if the output of one plot_image is feed into another plot_image
%then the data is reploted ignoring all optional's
%if measured data if feed is used, then the optional.freq determines
%which pattern is used

scale='dB';
data='norm';
format='polar';
figure(2)
line_color='b-';

if nargin==2
   if isfield(optional,'line_color')
      line_color=optional.line_color;
   end
end

switch (image_ptrn.create)
   case('plot_image')
      if isfield(image_ptrn,'x')
         surf(image_ptrn.x,image_ptrn.y,image_ptrn.pwr)
         title(['theta is from ',num2str(image_ptrn.theta_range(1)/pi,2),'-'...
            num2str(image_ptrn.theta_range(2)/pi,2),' (pi)'])
         xlabel('x')
         ylabel('y')
         zlabel(['pwr (',scale,')'])
         axis('square')
      else
         if length(image_ptrn.theta)==1
            hold on
            plot(image_ptrn.phi/pi,image_ptrn.pwr,line_color)
            title(['theta = ',num2str(image_ptrn.theta(1))]);
            xlabel('phi (pi)')
            ylabel(['pwr (',scale,')'])
            hold off
         elseif length(image_ptrn.phi)==1
```

```
            hold on
            plot(image_ptrn.theta/pi,image_ptrn.pwr,line_color)
            title(['theta = ',num2str(image_ptrn.phi(1))]);
            xlabel('theta (pi)')
            ylabel(['pwr (',scale,')'])
            hold off
         else
            surf(image_ptrn.phi/pi,image_ptrn.theta/pi,image_ptrn.pwr);
            xlabel('phi (pi)')
            ylabel('theta (pi)')
            zlabel(['pwr (',scale,')'])
         end
      end
   case{'image_pattern','measured'}
      if strcmp(image_ptrn.create,'measured')
         %pull image pattern for optional.freq
         %squareroot the pwr and make field image_ptrn.Voc
         disp('measure data part of the code is not done yet')
      end

      if nargin ==2
         if isfield(optional,'scale')
            scale=optional.scale;
         end
         if isfield(optional,'data')
            data=optional.data;
         end
         if isfield(optional,'format')
            format=optional.format;
         end
         if isfield(optional,'theta_range')
            theta_range=optional.theta_range;
            [Y,start_temp]=min(abs(image_ptrn.theta-theta_range(1)*...
               ones(size(image_ptrn.theta))));
            [Y,end_temp]=min(abs(image_ptrn.theta-theta_range(2)*...
               ones(size(image_ptrn.theta))));
            image_ptrn.theta=image_ptrn.theta([start_temp:end_temp]);
            image_ptrn.Voc=image_ptrn.Voc([start_temp:end_temp],:);
         end
         if isfield(optional,'phi_range')
            phi_range=optional.phi_range;
            [Y,start_temp]=min(abs(image_ptrn.phi-phi_range(1)*...
               ones(size(image_ptrn.phi))));
            [Y,end_temp]=min(abs(image_ptrn.phi-phi_range(2)*...
               ones(size(image_ptrn.phi))));
            image_ptrn.phi=image_ptrn.phi([start_temp:end_temp]);
            image_ptrn.Voc=image_ptrn.Voc(:,[start_temp:end_temp]);
         end
      end

      pwr=(abs(image_ptrn.Voc)).^2;
```

```
max_value=max(max(pwr));
max_temp=max_value;

if strcmp(data,'norm')
   max_temp=1;
elseif strcmp(class(data),'double')
   if strcmp(scale,'linear')
      max_temp=data;
   else
      max_temp=10^(data/10);
   end
end

pwr=max_temp*pwr/max_value;

if strcmp(scale,'dB')
   pwr=10*log10(pwr);
end

if length(image_ptrn.theta)==1
   hold on
   plot(image_ptrn.phi/pi,pwr,line_color)
   title(['theta = ',num2str(image_ptrn.theta(1))]);
   xlabel('phi (pi)')
   ylabel(['pwr (',scale,')'])
   image_ptrn_plot.pwr=pwr;
   image_ptrn_plot.theta=image_ptrn.theta;
   image_ptrn_plot.phi=image_ptrn.phi;
   hold off
elseif length(image_ptrn.phi)==1
   hold on
   plot(image_ptrn.theta/pi,pwr,line_color)
   title(['theta = ',num2str(image_ptrn.phi(1))]);
   xlabel('theta (pi)')
   ylabel(['pwr (',scale,')'])
   image_ptrn_plot.pwr=pwr;
   image_ptrn_plot.theta=image_ptrn.theta;
   image_ptrn_plot.phi=image_ptrn.phi;
   hold off
else
   switch (format)
      case ('polar')
         for k=1:length(image_ptrn.theta)
            for l=1:length(image_ptrn.phi)
               x(k,l)=image_ptrn.theta(k)*cos(image_ptrn.phi(l));
               y(k,l)=image_ptrn.theta(k)*sin(image_ptrn.phi(l));
            end
         end
         surf(x,y,pwr)
         title(['theta is from ',num2str(image_ptrn.theta(1)/pi,2),'-'...
                  num2str(image_ptrn.theta(length(image_ptrn.theta))/pi,2),...
                  ' (pi)'])
               xlabel('x')
               ylabel('y')
               zlabel(['pwr (',scale,')'])
               axis('square')
               image_ptrn_plot.pwr=pwr;
               image_ptrn_plot.x=x;
               image_ptrn_plot.y=y;
               image_ptrn_plot.theta_range=[image_ptrn.theta(1),...
                  image_ptrn.theta(length(image_ptrn.theta))];
            case('rect')
               surf(image_ptrn.phi/pi,image_ptrn.theta/pi,pwr);
               xlabel('phi (pi)')
               ylabel('theta (pi)')
               zlabel(['pwr (',scale,')'])
               image_ptrn_plot.pwr=pwr;
               image_ptrn_plot.theta=image_ptrn.theta;
               image_ptrn_plot.phi=image_ptrn.phi;
         end
      end
      image_ptrn_plot.create='plot_image';
end
```

plot_imager.m

```
function plot_imager(imager,optional)

%This function plots a 3d picture of the imager in figure 4
%optional.fig_num default is 4

optional_plot.fig_num=4;
if nargin==2
   optional_plot.fig_num=optional.fig_num;
end

for k=1:length(imager)
   detector_k=imager{k};
   plot_detector(detector_k,optional_plot);
end
```

plot_lens.m

```
function plotlens(lens,optional)

%plotlens(lens,optional)
%This function plots a 3d picture of the quasi-optical lens in figure 4
```

```
%optional.fig_num default is 4

fig_num=4;
if nargin==2
   fig_num=optional.fig_num;
end

whitebg('k')
figure(fig_num)
view(3)

for m=1:lens.N_c
   drawrect(lens.cell.nfd.base{m},lens.cell.nfd.pos{m},lens.cell.height, ...
      lens.cell.width,'r');

   hold on
   plot3(lens.cell.nfd.pos{m}(1),-1*lens.cell.nfd.pos{m}(3),  ...
      lens.cell.nfd.pos{m}(2),'rx');

   plot3(lens.cell.fd.pos{m}(1),-1*lens.cell.fd.pos{m}(3),  ...
      lens.cell.fd.pos{m}(2),'yo');
   hold off

end

hold on
switch lens.shape
   case{'circle'}
      angle=[0:pi/15:2*pi];
      D=2*lens.dimension;
      x_circle=(D/2)*cos(angle);
      y_circle=(D/2)*sin(angle);
      z_circle=zeros(1,length(angle));
      plot3(x_circle,-1*z_circle,y_circle,'r')
   case {'rect'}
      w=lens.dimension(1);
      h=lens.dimension(2);
      x=[-w/2, -w/2, w/2, w/2, -w/2];
      y=[h/2, -h/2, -h/2, h/2, h/2];
      z=zeros(1,5);
      plot3(x,-1*z,y,'r')

end
hold off
axis('equal')

xlabel('x')
ylabel('-z')
zlabel('y')
```

plot_radiation.m

```
function [rad_ptrn_plot]=plot_radiation(rad_ptrn,optional)

%this function plots the radiation pattern and outputs the ploted data
%optional.scale  ('linear','dB') default is dB
%optional.data ('raw','norm',num) default is normalize to max unity
%                                 num is the normalizaton of the max power
%                                to this value in the scan range and
%                                scale specified
%optional.format ('polar','rect') default is polar representation of
%                                   alpha and ki where the radius is
%                                   linear with alpha
%optional.alpha_range [alpha_start,alpha_end] if alpha_start = alpha_end
%                                it plots is a cut with alpha=alpha_start
%optional.ki_range [ki_start,ki_end] if ki_start = ki_end then it produces
%                                      the plot is a cut with ki=ki_start
%if either of the alpha or ki ranges are of length one, then plot-radiati
%allows stacking of 2D plots and optional.line_color defines the line
%tye and color (default is blue solid)
%if the output of one plot_radiation is feed into another plot_radiation
%then the data is reploted ignoring all optional's
%if measured data if feed is used, then the optional.freq determines
%which pattern is used

scale='dB';
data='norm';
format='polar';
figure(3)
line_color='b-';

switch (rad_ptrn.create)
   case('plot_radiation')
      if isfield(rad_ptrn,'x')
         surf(rad_ptrn.x,rad_ptrn.y,rad_ptrn.pwr)
         title(['alpha is from ',num2str(rad_ptrn.alpha_range(1)/pi,2),'-'...
            num2str(rad_ptrn.alpha_range(2)/pi,2),' (pi)'])
         xlabel('x')
         ylabel('y')
         zlabel(['power (',scale,')'])
         axis('square')
      else
         if length(rad_ptrn.alpha)==1
            hold on
            plot(rad_ptrn.ki/pi,rad_ptrn.pwr,line_color)
            title(['alpha = ',num2str(rad_ptrn.alpha(1))]);
            xlabel('chi (pi)')
            ylabel(['power (',scale,')'])
            hold off
         elseif length(rad_ptrn.ki)==1
            hold on
            plot(rad_ptrn.alpha/pi,rad_ptrn.pwr,line_color)
            title(['alpha = ',num2str(rad_ptrn.ki(1))]);
            xlabel('alpha (pi)')
            ylabel(['power (',scale,')'])
            hold off
         else
            surf(rad_ptrn.ki/pi,rad_ptrn.alpha/pi,rad_ptrn.pwr);
            xlabel('chi (pi)')
            ylabel('alpha (pi)')
            zlabel(['power (',scale,')'])
         end
      end
   case{'rad_pattern','measured','antenna_pattern'}
      switch rad_ptrn.create
         case('measured')
            %pull radiation pattern for optional.freq
            %squareroot the pwr and make field rad_ptrn.Voc
            disp('measure data part of the code is not done yet')
         case('antenna_pattern')
            rad_ptrn.Voc=rad_ptrn.le;
      end

      if nargin==2
         if isfield(optional,'scale')
            scale=optional.scale;
         end
         if isfield(optional,'line_color')
            line_color=optional.line_color;
         end
         if isfield(optional,'data')
            data=optional.data;
         end
         if isfield(optional,'format')
            format=optional.format;
         end
         if isfield(optional,'alpha_range')
            alpha_range=optional.alpha_range
            [Y,start_temp]=min(abs(rad_ptrn.alpha-alpha_range(1)*...
               ones(size(rad_ptrn.alpha))));
            [Y,end_temp]=min(abs(rad_ptrn.alpha-alpha_range(2)*...
               ones(size(rad_ptrn.alpha))));
            rad_ptrn.alpha=rad_ptrn.alpha([start_temp:end_temp]);
            rad_ptrn.Voc=rad_ptrn.Voc([start_temp:end_temp],:);
         end
         if isfield(optional,'ki_range')
            ki_range=optional.ki_range;
            if ki_range(1)==ki_range(2)
               ki_range(2)=ki_range(2)+pi;
               [Y,start_temp]=min(abs(rad_ptrn.ki-ki_range(1)*...
                  ones(size(rad_ptrn.ki))));
            [Y,end_temp]=min(abs(rad_ptrn.ki-ki_range(2)*...
               ones(size(rad_ptrn.ki))));
            rad_ptrn.ki=ki_range(1);
            rad_ptrn_start=rad_ptrn.Voc(:,[start_temp]);
            rad_ptrn_end=rad_ptrn.Voc(:,[end_temp]);
            rad_ptrn.Voc=[rad_ptrn_end([length(rad_ptrn_end):-1:1]);...
               rad_ptrn_start];
            rad_ptrn.alpha=...
               [-1*rad_ptrn.alpha([length(rad_ptrn.alpha):-1:1])...
               ,rad_ptrn.alpha];
         else

            [Y,start_temp]=min(abs(rad_ptrn.ki-ki_range(1)*...
               ones(size(rad_ptrn.ki))));
            [Y,end_temp]=min(abs(rad_ptrn.ki-ki_range(2)*...
               ones(size(rad_ptrn.ki))));
            rad_ptrn.ki=rad_ptrn.ki([start_temp:end_temp]);
            rad_ptrn.Voc=rad_ptrn.Voc(:,[start_temp:end_temp]);
         end
      end

   end

   pwr=(abs(rad_ptrn.Voc)).^2;

   max_value=max(max(pwr));
   max_temp=max_value;

   if strcmp(data,'norm')
      max_temp=1;
   elseif strcmp(class(data),'double')
      if strcmp(scale,'linear')
         max_temp=data;
      else
         max_temp=10^(data/10);
      end
   end

   pwr=max_temp*pwr/max_value;

   if strcmp(scale,'dB')
      pwr=10*log10(pwr);
   end

   if length(rad_ptrn.alpha)==1
      hold on
      plot(rad_ptrn.ki/pi,pwr,line_color)
      title(['alpha = ',num2str(rad_ptrn.alpha(1))]);
      xlabel('chi (pi)')
      ylabel(['power (',scale,')'])
      rad_ptrn_plot.pwr=pwr;
```

```
         rad_ptrn_plot.alpha=rad_ptrn.alpha;
         rad_ptrn_plot.ki=rad_ptrn.ki;
         hold off
      elseif length(rad_ptrn.ki)==1
         hold on
         plot(rad_ptrn.alpha*180/pi,pwr,line_color)
         title(['alpha = ',num2str(rad_ptrn.ki(1))]);
         xlabel('alpha (degrees)')
         ylabel(['power (',scale,')'])
         rad_ptrn_plot.pwr=pwr;
         rad_ptrn_plot.alpha=rad_ptrn.alpha;
         rad_ptrn_plot.ki=rad_ptrn.ki;
         hold off
      else
         switch (format)
            case ('polar')
               for k=1:length(rad_ptrn.alpha)
                  for l=1:length(rad_ptrn.ki)
                     x(k,l)=rad_ptrn.alpha(k)*cos(rad_ptrn.ki(l));
                     y(k,l)=rad_ptrn.alpha(k)*sin(rad_ptrn.ki(l));
                  end
               end
               surf(x,y,pwr)
               title(['alpha is from ',num2str(rad_ptrn.alpha(1)/pi,2),'-'...
                  num2str(rad_ptrn.alpha(length(rad_ptrn.alpha))/pi,2),...
                  ' (pi)'])
               xlabel('x')
               ylabel('y')
               zlabel(['power (',scale,')'])
               axis('square')
               rad_ptrn_plot.pwr=pwr;
               rad_ptrn_plot.x=x;
               rad_ptrn_plot.y=y;
               rad_ptrn_plot.alpha_range=[rad_ptrn.alpha(1),...
                  rad_ptrn.alpha(length(rad_ptrn.alpha))];
            case('rect')
               surf(rad_ptrn.ki/pi,rad_ptrn.alpha/pi,pwr);
               xlabel('chi (pi)')
               ylabel('alpha (pi)')
               zlabel(['power (',scale,')'])
               rad_ptrn_plot.pwr=pwr;
               rad_ptrn_plot.alpha=rad_ptrn.alpha;
               rad_ptrn_plot.ki=rad_ptrn.ki;
         end
      end
      rad_ptrn_plot.create='plot_radiation';
end
```

plot_source.m

```
function plot_source(source,optional)

%this function plots a 3d picture of the source in figure 4
%optional.fig_num default is 4
%optional.radius changes the size of source symbol radius
%        default is 0.25
%optional.distance changes the distance the far field source symbols
%        are plotted default is 6

fig_num=4;
radius=0.25;
if nargin==2
   if isfield(optional,'fig_num')
      fig_num=optional.fig_num;
   end
   if isfield(optional,'radius')
      radius=optional.radius;
   end
   if isfield(optional,'distance')
      source.pos=(optional.distance/6)*source.pos;
   end
end

whitebg('k')
figure(fig_num)
view(3)

switch(source.type)
   case('nearfield')
      drawpolygon(source.base,source.pos,radius,20,'m');
   case('farfield')
      drawpolygon(source.base,source.pos,radius,6,'m');
end

hold on
plot3(source.pos(1),-1*source.pos(3),source.pos(2),'rx');
axis('equal')
hold off
grid on
```

plot_source_pos.m

```
function plot_source_pos(channel,name,offset)

%this function plots the signals on the radiation plattern plots
%the numbering is signals,source farfield 'x' and nearfield 'o'
%'name is the name of the source

figure(3)
```

```
whitebg('w')
for l=1:length(channel)
   source_l=channel{l};
   name=num2str(l,3);
   hold on
   switch (source_l.type)
      case('farfield')
         angle_l=source_l.angle;
         x=angle_l(1)*cos(angle_l(2));
         y=angle_l(1)*sin(angle_l(2));
         Htemp=plot3(x,y,0,'kx');
         set(Htemp,'MarkerSize',10);
         if nargin>1
            text(x+offset(1),y+offset(2),0,name)
         else
            text(x,y,0,[num2str(l,3)]);
         end
      case('nearfield')
         pos_l=source_l.pos
         alpha=atan2(sqrt(pos_l(1)^2+pos_l(2)^2),pos_l(3));
         ki=atan2(pos_l(2),pos_l(1))
         if ki<0
            ki=ki+2*pi;
         end
         x=angle_l(alpha)*cos(angle_l(ki));
         y=angle_l(alpha)*sin(angle_l(ki));
         Htemp=plot3(x,y,0,'ko');
         set(Htemp,'MarkerSize',10);
         if nargin>1
            text(x+offset(1),y+offset(2),0,name)
         else
            text(x,y,0,[num2str(l,3)]);
         end
   end
   hold off
end
```

ptrn_grid.m

```
function ptrn_grid(fig_num,angle_max,spacing,labels)

figure(fig_num);
angles=[spacing:spacing:angle_max];
theta=[0:pi/20:2*pi];

for k=1:length(angles)
   hold on
   for l=1:length(theta)
      x(l)=angles(k)*cos(theta(l));
      y(l)=angles(k)*sin(theta(l));
      z(l)=1;
   end
   plot3(x,y,z,'k:')
   if strcmp(labels,'x')
      Htext=text(angles(k),0.1,1,[num2str(angles(k)*180/pi ,2),'^o']);
   else
      Htext=text(0.0,angles(k),1,[num2str(angles(k)*180/pi ,2),'^o']);
   end
   set(Htext,'Color',[0,0,0]);
   hold off
end
hold on
plot3([0,angle_max],[0,0],[1,1],'b:')
plot3([0,0],[0,angle_max],[1,1],'b:')
plot3([0,-angle_max],[0,0],[1,1],'b:')
plot3([0,0],[0,-angle_max],[1,1],'b:')
hold off
```

rad_pattern.m

```
function [rad_ptrn]=rad_pattern(lens,detector,alpha_range,ki_range,optional)

%this function outputs the receive Voc at a detector
%for a source with unit input.  'rad_ptrn' contains fields 'Voc', 'alpha'
%and 'ki'.
%alpha_range is a vector that contains[alpha_start,alpha_end,alpha_N]
%ki_range is a vector that contains[ki_start,ki_end,ki_N]
%optional.polar_source use polarization for source coupling
%                    default is 'yes'
%optional.polar_feed use polarization for feed coupling
%                    default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                    default is 'yes'
%optional.pathloss_feed use pathloss for feed coupling
%                    default is 'yes'

optional_focus.polar_source='yes';
optional_focus.polar_feed='yes';
optional_focus.pathloss_source='yes';
optional_focus.pathloss_feed='yes';

%initalize default farfield source
middle=round(lens.N_c/2);
[nfd_antenna]=pull_antenna(lens,middle,'nonfeed');
[le_nfd,ro_nfd]=feval(nfd_antenna.name,0,0,nfd_antenna.polar);
J_sph_nfd=xyz2sph(0,0);
J_gnfd=nfd_antenna.local;
ro_source=J_gnfd'*J_sph_nfd'*ro_nfd;
```

```
polar=xyz2sph(0,0)*ro_source;
%

if nargin==5
   if isfield(optional,'polar_source')
      optional_focus.polar_source=optional.polar_source;
   end
   if isfield(optional,'polar_feed')
      optional_focus.polar_feed=optional.polar_feed;
   end
   if isfield(optional,'pathloss_source')
      optional_focus.pathloss_source=optional.pathloss_source;
   end
   if isfield(optional,'pathloss_feed')
      optional_focus.pathloss_feed=optional.pathloss_feed;
   end
end


if ki_range(3)==1
   ki=(1);
   alpha=[-alpha_range(2):(2*alpha_range(2))/(alpha_range(3)-1):...
      alpha_range(2)];
else
   alpha=[alpha_range(1):(alpha_range(2)-alpha_range(1))/(alpha_range(3)-1):...
      alpha_range(2)];
   ki=[ki_range(1):(ki_range(2)-ki_range(1))/(ki_range(3)-1):ki_range(2)];
end

for k=1:alpha_range(3)
   comments(['Radiation Pattern ',num2str(k/alpha_range(3),2)]);
   for l=1:ki_range(3)
      if alpha(k)==0 & ki(l)~=0
         Voc(k,l)=Voc(1,1);
      else
         [source_kl]=build_far(alpha(k),ki(l),0,1,polar);
         [Voc(k,l)]=lens_focus(lens,source_kl,detector,optional_focus);
      end
   end
end

rad_ptrn.Voc=Voc;
rad_ptrn.alpha=alpha;
rad_ptrn.ki=ki;
rad_ptrn.create='rad_pattern';
```

rad_ptrn_array.m

```
function [rad_ptrn]=rad_ptrn_array(array,weights,alpha_range,ki_range,optional)

%this function outputs the received weighted sum of Voc at each array element
%for a source with unit input.  'rad_ptrn' contains fields 'Voc', 'alpha'
%and 'ki'.
%weigts is the weigted sum of the Voc's in the array and is a column vector
%y=W'*X
%alpha_range is a vector that contains[alpha_start,alpha_end,alpha_N]
%ki_range is a vector that contains[ki_start,ki_end,ki_N]
%optional.polar_source use polarization for source coupling
%                     default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                     default is 'yes'

optional_focus.polar_source='yes';
optional_focus.pathloss_source='yes';

%initalize default farfield source
middle=round(array.N_c/2);
[nfd_antenna]=pull_antenna(array,middle,'nonfeed');
[le_nfd,ro_nfd]=feval(nfd_antenna.name,0,0,nfd_antenna.polar);
J_sph_nfd=xyz2sph(0,0);
J_gnfd=nfd_antenna.local;
ro_source=J_gnfd'*J_sph_nfd'*ro_nfd;
polar=xyz2sph(0,0)*ro_source;
%

if nargin==5
   if isfield(optional,'polar_source')
      optional_focus.polar_source=optional.polar_source;
   end
   if isfield(optional,'pathloss_source')
      optional_focus.pathloss_source=optional.pathloss_source;
   end
end


alpha=[alpha_range(1):(alpha_range(2)-alpha_range(1))/(alpha_range(3)-1):...
      alpha_range(2)];
ki=[ki_range(1):(ki_range(2)-ki_range(1))/(ki_range(3)-1):ki_range(2)];

for k=1:alpha_range(3)
   comments(['Radiation Pattern ',num2str(k/alpha_range(3),2)]);
   for l=1:ki_range(3)
      if alpha(k)==0 & ki(l)~=0
         Voc(k,l)=Voc(1,1);
      else
         [source_kl]=build_far(alpha(k),ki(l),0,1,polar);
         [Voc_kl]=array_focus(array,source_kl,optional_focus);
         Voc(k,l)=weights'*Voc_kl;
      end
   end
end

rad_ptrn.Voc=Voc;
rad_ptrn.alpha=alpha;
rad_ptrn.ki=ki;
rad_ptrn.create='rad_pattern';
```

QOL.m

```
function [lens]=QOL(shape,dimension,antenna,lensing,delay_0,theta_0,F,unitcellsize,latice)


%[lens]=QOL(shape,deminsion,antenna,lensing,delay_0,theta_0,F,unitcellsize,latice)
%Shape is a string argument that discribes the lens shape 'circle','rect'
%dimension is a vector that contains the important deminsions for the lens
%  circle [radius]
%  rect [width,height] width = x dimension and height = y dimension
%antenna is cell discribes the feed and non-feed side antenna types,
%polarization and orientation.  The polarization vector is for a model
%antenna in the oreintation for the theorical or measured radiation patterns.
%Orientation discribes the rotation of the QOL antennas from the model antenna.
%The information is listed as to cells in the order of {feed side,
%non-feedside}, where feed side is a sub cell containing {antenna,
%polarization,orientation}
% polarization} %lensing is a string that discribes the in how many dimensions the lens
%focuses in
%  ['2D'], ['1Dx'], ['1Dy']
%delay_0 is the base delay for the unitcell in the center of the lens
%unitcellsize defines the x/y dimensions for unitcell
%  [width,height] width = x  height = y
%latice is a cell of strings that discribes the how the unitcells are
%placed in the lens {pattern,boundary}
%  PATTERN
%  'triangle' uses a triangular lattice to fill the shape trying to maximise
%  the number of unit cells
%  'max'  uses a row structure with as many unitcells as possible in
%  each row
%  BOUNDARY
%  'best' is the tightest packing allowing for 1/4 of the unitcell
%  to extend outside the defined shape
%  'limit' is the tightest packing without letting the unitcells
%  extend outside the defined shape
%note: all dimensions are in wave length of the center frequency
%note: this code assumes for now that the lens is confined to a plane


%initialize lens
```

```
lens.device='lens';
lens.shape=shape;
lens.dimension=dimension;
lens.cell.width=unitcellsize(1);
lens.cell.height=unitcellsize(2);
lens.delay_0=delay_0;
lens.theta_0=theta_0;
lens.F=F;
lens.lensing=lensing;
switch shape
   case {'circle'}
      [lens]=packcircle(latice,lens);
   case {'rect'}
      [lens]=packrect(latice,lens);
end

lens.N_c=sum(lens.R);

%determine non-feed antenna points and delays

[lens]=fdside(lensing,lens);
[lens]=packant(lens,antenna);
lens.N_r=length(lens.R);
```

QOLTF.m

```
function [QOL_TF,mutual_TF]=QOLTF(lens,channel,imager,optional)

%this function creates a tranfer function for channel to imager
%mutual_TF is the transfer function produced by the mutual coupling currents
%optional.polar_source use polarization for source coupling
%                     default is 'yes'
%optional.polar_feed use polarization for feed coupling
%                     default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                     default is 'yes'
%optional.pathloss_feed use pathloss for feed coupling
%                     default is 'yes'
%optional.mutual.use ('yes', 'no','max') calculates the image do to mutual
%                  coupling.  'yes' uses the estimated magnitude and phase
%                  of the coupled currents and 'max' uses just the magnitude
%                  default is 'no'
%optional.mutual.extra is an open field for use by name_mutual.m mutual
%coupling calculations functions

mutual.use='no';
optional_focus.polar_source='yes';
optional_focus.polar_feed='yes';
optional_focus.pathloss_source='yes';
optional_focus.pathloss_feed='yes';

if nargin == 4
   if isfield(optional,'polar_source')
         optional_focus.polar_source=optional.polar_source;
   end
   if isfield(optional,'polar_feed')
      optional_focus.polar_feed=optional.polar_feed;
   end
   if isfield(optional,'pathloss_source')
      optional_focus.pathloss_source=optional.pathloss_source;
   end
   if isfield(optional,'pathloss_feed')
      optional_focus.pathloss_feed=optional.pathloss_feed;
   end
   if isfield(optional,'mutual')
      mutual=optional.mutual;
   end
end



for k=1:length(channel)
   for l=1:length(imager)
      %comments(['QOLTF ',num2str(((k-1)*length(channel)+l)/(length(channel)*...
      %  length(imager)),3)]);

      source=channel{k};

      detector=imager{l};
      [QOL_TF(l,k),Iin_fd_k]=lens_focus(lens,source,detector,optional_focus);

      if ~strcmp(mutual.use,'no')
         [mutual_Z]=get_mutual_Z(lens,mutual);
         switch(mutual.use)
            case {'yes'}
               Iin_fd_mutual_k=mutual_Z*Iin_fd_k;
            case {'max'}
               Iin_fd_mutual_k=abs(mutual_Z)*abs(Iin_fd_k);
         end
         [mutual_TF(l,k)]=mutual_focus(lens,Iin_fd_mutual_k,detector,...
            optional_focus);
      end
   end
end
```

Numerical Model Support Code

array_focus.m

```
function [Voc]=array_focus(array,source,optional)

%this fucntion calculates the open circuit voltage at each element of an array
%Voc is a column vector
%optional.pathloss_source defaults is 'yes'
%optional.polar_source defaults is 'yes'

Voc=0;
optional_source.pathloss='yes';
optional_source.polar='yes';

if nargin==3
   if isfield(optional,'pathloss_source')
      optional_source.pathloss=optional.pathloss_source;
   end
   if isfield(optional,'polar_source')
      optional_source.polar=optional.polar_source;
   end

end

for k=1:array.N_c
   [nfd_antenna]=pull_antenna(array,k,'nonfeed');

   switch source.type
      case ('farfield')
         [Voc(k,1)]=couple_farfield(source,nfd_antenna,optional_source);
      case ('nearfield')
         [Voc(k,1)]=couple(source,nfd_antenna,source.Iin,optional_source);
   end

end
```

build_ant.m

```
function [antenna]=build_ant(name,pos,a_z,theta_r,polar)

%this function creates an antenna element
%pos in global cooridaint
%name is antenna name
%a_z is z unit vector for base corridants (row)
```

```
%theta_r is local coordinat rotation
%polar is local sphereical corridaint polarization

antenna.pos={pos};
antenna.name={name};
antenna.base={get_base(a_z)};
antenna.theta_r={theta_r};
antenna.local={rotate_base(antenna.base{1},antenna.theta_r{1})};
antenna.polar={polar};
```

build_det.m

```
function [detector]=build_det(name,pos,theta_r,g,cell_g,width,height,delay,polar)

%This function makes a detector that point to the center of the lens

a_z=(-1*pos')/sqrt(pos'*pos);
[detector.cell]=build_ant(name,pos,a_z,theta_r,polar);
detector.g=g;
detector.device='detector';
detector.N_c=1;
detector.cell.g={cell_g};
detector.cell.width=width;
detector.cell.height=height;
detector.cell.delay={delay};
```

build_far.m

```
function [source_far]=build_far(alpha,ki,theta_r,field,polar)

%this function makes a far field source receive at angle 'alpha'
%and 'ki'.  Theta_ r is the angle of rotation of local corridiants
%polar is a length 3 column vector discribing the local polarization
%in sphereical corrdiants.

source_far.device='source';
source_far.type='farfield';
source_far.field=field;
source_far.angle=[alpha,ki];
a_z_b=[sin(alpha)*cos(ki),sin(alpha)*sin(ki),cos(alpha)];
source_far.base=get_base(a_z_b);
source_far.theta_r=theta_r;
source_far.local=rotate_base(source_far.base,theta_r);
source_far.polar=polar;
source_far.pos=-6*a_z_b';
```

build_signal.m

```
function [signal]=build_signal(channel,constell,w_RF,T_data,user)

%this function builds a signal

signal.channel=channel;
signal.constell=constell;
signal.w_RF=w_RF;
signal.T_data=T_data;
signal.user=user;
```

comments.m

```
function [Hfig]=comments(text_input)

%this function displays the text in a figure

figure(10)
Hfig=figure(10);
%set(Hfig,'Position',[550,500,300,100]);
menubar=get(Hfig,'Menubar');
switch(menubar)
   case('figure')
      set(Hfig,'Menubar','none');
      axis([-1,1,-1,1,-1,1])
      text(-2,0,text_input);
      Haxis=gca;
      set(Haxis,'Visible','off');
      Htext=get(Haxis,'Children');
      set(Htext,'FontSize',22);
   case('none')
      Haxis=gca;
      Htext=get(Haxis,'Children');
      set(Htext,'String',text_input);
end
```

couple.m

```
function [Voc]=couple(transmit,receive,Iin,optional)

%function [Voc]=couple(transmit,receive,Iin,optional)
%this function calculates the open circuit voltage genterated in the
%'receive' antenna by a 'transmit' antenna.
%optional.pathloss default is 'yes'
```

```
%optional.polar default is 'yes'

polar='yes';
optional_Ei.pathloss='yes';

if nargin==4
   if isfield(optional,'polar')
      polar=optional.polar;
   end
   if isfield(optional,'pathloss')
      optional_Ei.pathloss=optional.pathloss;
   end
end
   [Ei_mag,ro_t_l]=get_Ei(transmit,receive,Iin,optional_Ei);

[theta_rec,phi_rec]=get_direction(receive,transmit);
[le_r,ro_r_l]=feval(receive.name,theta_rec,phi_rec,receive.polar);
switch polar
   case ('yes')
      Voc=le_r*Ei_mag*ro_r_l'*ro_t_l;
   case ('no')
      Voc=le_r*Ei_mag;
end
```

couple.m

```
function [Voc]=couple_farfield(source,receive,optional)

%this function calculates the coupling between a farfield source and
% a non-feed side antenna element

polar='yes';

if nargin==3
   if isfield(optional,'polar')
      polar=optional.polar;
   end
end

alpha=source.angle(1);
ki=source.angle(2);
a_rad=[-1*sin(alpha)*cos(ki);-1*sin(alpha)*sin(ki);-1*cos(alpha)];
delay=-dot(a_rad,receive.pos);

Ei_mag=source.field*exp(-j*2*pi*delay);

source.pos=receive.pos+a_rad;
[theta_tran,phi_tran]=get_direction(source,receive);
[theta_rec,phi_rec]=get_direction(receive,source);
```

```
J_sph_tran=xyz2sph(theta_tran,phi_tran);
J_sph_rec=xyz2sph(theta_rec,phi_rec);
J_gt=source.local;
J_gr=receive.local;

ro_t_l=J_sph_rec*J_gr*J_gt'*J_sph_tran'*source.polar;
[le_r,ro_r_l]=feval(receive.name,theta_rec,phi_rec,receive.polar);

switch polar
   case ('yes')
      Voc=le_r*Ei_mag*ro_r_l'*ro_t_l;
   case ('no')
      Voc=le_r*Ei_mag;
end
```

couple_farfield.m

```
function drawball(radius)

%this function draws a hemisphere of radius 'radius'


figure(7)
view(3)

theta=[0:pi/30:pi/2];
phi=[0:pi/10:2*pi];

for k=1:length(theta)
   for l=1:length(phi)
      ball(k,l)=radius*cos(theta(k));
      x(k,l)=radius*sin(theta(k))*cos(phi(l));
      y(k,l)=radius*sin(theta(k))*sin(phi(l));
   end
end
surf(x,y,ball)

xlabel('x')
ylabel('y')
zlabel('z')

axis('equal')
```

drawpolygon.m

```
function drawpolygon(local,pos,radius,sides,color)

%drawpolygon(local,pos,radius,sides,color)
%this is a private function
%this function draws a polygon aperture with local coordinates system 'local'
% and position 'pos' in global coordinates

hold on

theta=[0:2*pi/sides:2*pi];
point=[];
for k=1:length(theta)
   x=radius*cos(theta(k));
   y=radius*sin(theta(k));
   point=[point,local'*[x;y;0]+pos];
end

x=point(1,:);
y=point(2,:);
z=point(3,:);
plot3(x,-1*z,y,color);
hold off
```

drawrect.m

```
function drawrect(local,pos,height,width,color)

%drawrect(local,pos,height,width,color)
%this is a private function
%this function draws a rectangle aperture with local coordinates system 'local'
% and position 'pos' in global coordinates

hold on

h=height;
w=width;

point1=local'*[.5*w;.5*h;0]+ pos;
point2=local'*[-.5*w;.5*h;0]+ pos;
point3=local'*[-.5*w;-.5*h;0]+ pos;
point4=local'*[.5*w;-.5*h;0]+ pos;
point5=local'*[.5*w;.5*h;0]+ pos;

x=[point1(1),point2(1),point3(1),point4(1),point5(1)];
y=[point1(2),point2(2),point3(2),point4(2),point5(2)];
z=[point1(3),point2(3),point3(3),point4(3),point5(3)];
plot3(x,-1*z,y,color);
hold off
```

drawunit.m

```
function drawunit(local,pos,color,optional)

%drawunit(local,pos)
%this function plots the local unit vectors at 1/4 length
%optional.font_size allows the change in font size
%        default is 10
%optional.length  length of unit vector
%        default is 0.25

font_size=10;
length_luv=0.25;

if nargin==4
   if isfield(optional,'font_size')
      font_size=optional.font_size;
   end
   if isfield(optional,'length')
      length_luv=optional.length;
   end
end

hold on
xpoint=local'*length_luv*[1; 0; 0] + pos;
ypoint=local'*length_luv*[0; 1; 0] + pos;
zpoint=local'*length_luv*[0; 0; 1] + pos;

plot3([pos(1),xpoint(1)],-1*[pos(3),xpoint(3)],[pos(2),xpoint(2)],color)
plot3([pos(1),ypoint(1)],-1*[pos(3),ypoint(3)],[pos(2),ypoint(2)],color)
plot3([pos(1),zpoint(1)],-1*[pos(3),zpoint(3)],[pos(2),zpoint(2)],color)

[H]=text(xpoint(1),-1*xpoint(3),xpoint(2),'x');
set(H,'FontSize',font_size)
[H]=text(ypoint(1),-1*ypoint(3),ypoint(2),'y');
set(H,'FontSize',font_size)
[H]=text(zpoint(1),-1*zpoint(3),zpoint(2),'z');
set(H,'FontSize',font_size)

hold off
```

fdside.m

```
function [lens]=fdside(lensing,lens)

%[lens]=fdside{lensing,lens)
%This is a private function that calculates the feed side antennas and
%delay in radians base on a paper by McGrath
```

```
%initalize variables
lens.cell.fd.pos=lens.cell.nfd.pos;
lens.cell.delay=[];
cell_rad_old=0;

for m=1:lens.N_c
   if lens.cell.nfd.pos{m}(1)==0 & lens.cell.nfd.pos{m}(2) ==0
      x_temp=0;
      y_temp=0;
   else
      cell_rad=sqrt(lens.cell.nfd.pos{m}'*lens.cell.nfd.pos{m});
      temp_rad=cell_rad*sqrt((lens.F^2-(cell_rad^2)*sin(lens.theta_0)^2)...
         /(lens.F^2-cell_rad^2));
      switch lensing
         case {'1Dx'}
            x_temp=(temp_rad/cell_rad)*lens.cell.nfd.pos{m}(1);
            y_temp=lens.cell.nfd.pos{m}(2);
            temp_rad=x_temp;
         case {'1Dy'}
            y_temp=(temp_rad/cell_rad)*lens.cell.nfd.pos{m}(2);
            x_temp=lens.cell.nfd.pos{m}(1);
            temp_rad=y_temp;
         case {'2D'}
            x_temp=(temp_rad/cell_rad)*lens.cell.nfd.pos{m}(1);
            y_temp=(temp_rad/cell_rad)*lens.cell.nfd.pos{m}(2);
      end
   end

   lens.cell.fd.pos{m}(1)=x_temp;
   lens.cell.fd.pos{m}(2)=y_temp;
   %calculate delays
   temp=lens.F+lens.delay_0-0.5*sqrt(lens.F^2+temp_rad^2-2*temp_rad*lens.F ...
      *sin(lens.theta_0));
   temp2=-0.5*sqrt(lens.F^2+temp_rad^2+2*temp_rad*lens.F*sin(lens.theta_0));

   lens.cell.delay=[lens.cell.delay,temp+temp2];

   if imag(temp+temp2)>.01
      disp('bad lens design, theta_0 or F is too small')
   end

   %find max radius
   if cell_rad>cell_rad_old
      cell_rad_old=cell_rad;
   end
end

lens.cell.g=ones(1,lens.N_c);
lens.r_max=cell_rad_old;
```

get_Ei.m

```
function  [Ei_mag,ro_t_l]=get_Ei(transmit,receive,Iin,optional)

%function  [Ei_mag,ro_t_l]=get_Ei(transmit,receive,Iin,optional)
%this function calcualtes the incident electric field on a recieve antenna
%in the local cooridainats of the receive antenna keeping polarization separte.
%It assumes unit input current and the output is a vector of the electric
%field in the local spherical cooridants [r,theta,phi].  There is an
%optional input that allows ignoring 1/r loss. optional.pathloss default
%is 'yes'.
%'receive' and 'transmit' contain the fields [name,polar,base,local,pos,theta_r]

pathloss='yes';
if nargin==4
   if isfield(optional,'pathloss')
      pathloss=optional.pathloss;
   end
end

Rt=transmit.pos;
Rr=receive.pos;
r=Rr-Rt;
r_mag=sqrt(r'*r);

[theta_tran,phi_tran]=get_direction(transmit,receive);
[theta_rec,phi_rec]=get_direction(receive,transmit);

[le_t,ro_t]=feval(transmit.name,theta_tran,phi_tran,transmit.polar);

J_sph_tran=xyz2sph(theta_tran,phi_tran);
J_sph_rec=xyz2sph(theta_rec,phi_rec);
J_gt=transmit.local;
J_gr=receive.local;

ro_t_l=J_sph_rec*J_gr*J_gt'*J_sph_tran'*ro_t;
switch pathloss
   case('no')
      Ei_mag=377*((-j*2*pi)/(4*pi))*le_t*exp(-j*2*pi*r_mag)*Iin;
   case('yes')
      Ei_mag=377*((-j*2*pi)/(4*pi*r_mag))*le_t*exp(-j*2*pi*r_mag)*Iin;
end
```

get_base.m

```
function [base]=get_base(a_z_b)

%[base]=get_base(a_z_b)
%this function gets to the base corridant system for an antenna with normal
%a_z_b with a_x_b normal a_y and in positive x direction

a_x=[1 0 0];
a_y=[0 1 0];
a_z=[0 0 1];

x_b=cross(a_y,a_z_b);

if x_b*a_x' < 0
   x_b=-1*x_b;
end

a_x_b=x_b/sqrt(x_b*x_b');
a_y_b=cross(a_z_b,a_x_b);

base=[a_x_b;a_y_b;a_z_b];
```

get_direction.m

```
function [theta,phi]=get_direction(transmit,receive)

%function [theta,phi]=get_direction(transmit,receive)
%this function calculates the look direction in the local cooridants of transmit
% antenna 'transmit' as it radiates to a receive antenna 'receive'
%'receive' and 'transmit' contain the fields [name,polar,base,local,pos,theta_r]

Rt=transmit.pos;
Rr=receive.pos;
r=Rr-Rt;

r_l=transmit.local*r;
theta=atan2(sqrt(r_l(1)^2+r_l(2)^2),r_l(3));

if theta <0
   theta = theta + 2*pi;
end

phi=atan2(r_l(2),r_l(1));

if phi <0
   phi=phi+2*pi;
end
```

```
%define phi=0 hen theta is 0 or pi
if theta < 10e-4 | abs(theta-pi) < 10e-4
   phi=0;
end
```

get_focal.m

```
function [focal_rad]=get_focal(lens, theta)

%this function calculates the focal distance for a 'lens' at angle theta
%this focal distance is really only known for circle antennas

theta=abs(theta);
phi=asin(lens.r_max/(lens.F));
temp=lens.F*(1+0.5*((sin(phi)^2*sin(theta)^2)/((1-sec(phi)) ...
   *(1+sin(phi)*sin(theta)))));
focal_rad=sec(lens.theta_0)*temp;
```

get_mutual_Z.m

```
function [mutual_Z]=get_mutual_Z(lens,mutual)


%this function create the coupling matrix between feed side antennas
%mutual.extra is an open field for uses with in mutual coupling
%calculating functions

mutual_Z=zeros(lens.N_c,lens.N_c);
for k=1:lens.N_c
   for l=1:lens.N_c
      if k==l
         mutual_couple(k,l)=0;
      else
         r_21_g=lens.cell.fd.pos{k}-lens.cell.fd.pos{l};
         r_21_l=lens.cell.fd.local{k}*r_21_g;
         temp=lens.cell.fd.name{k};
         space=findstr('_',temp);
         func_name=[temp([1:space-1]),'_mutual'];
         [Z_21]=feval(func_name,r_21_l,mutual.extra);
         mutual_Z(k,l)=Z_21;
      end
   end
end
```

lens_focus.m

```
function [Voc,Iin_fd]=lens_focus(lens,source,detector,optional)
```

```
%this fucntion calculates the open circuit voltage for a detectot
%do to a source focus through a lens.
%optional.pathloss defaults is 'yes'
%optional.polar defaults is 'yes'
%Iin_fd is the input current to each feed side antena for use in mutual
%coupling calculations

Voc=0;
optional_source.pathloss='yes';
optional_source.polar='yes';
optional_feed.pathloss='yes';
optional_feed.polar='yes';

if nargin==4
   if isfield(optional,'pathloss_feed')
      optional_feed.pathloss=optional.pathloss_feed;
   end
   if isfield(optional,'pathloss_source')
      optional_source.pathloss=optional.pathloss_source;
   end
   if isfield(optional,'polar_feed')
      optional_feed.polar=optional.polar_feed;
   end
   if isfield(optional,'polar_source')
      optional_source.polar=optional.polar_source;
   end

end

for k=1:lens.N_c
   [nfd_antenna]=pull_antenna(lens,k,'nonfeed');

   switch source.type
      case ('farfield')
         [Voc_nfd]=couple_farfield(source,nfd_antenna,optional_source);
      case ('nearfield')
         [Voc_nfd]=couple(source,nfd_antenna,source.Iin,optional_source);
   end

   Iin_fd(k,1)=(Voc_nfd/2)*lens.cell.g(k)*exp(-j*2*pi*lens.cell.delay(k));
   [fd_antenna]=pull_antenna(lens,k,'feed');

   for l=1:detector.N_c
      [det_antenna]=pull_antenna(detector,l);
      [Voc_l]=couple(fd_antenna,det_antenna,Iin_fd(k),optional_feed);
      Voc=Voc+Voc_l*detector.cell.g{l}*exp(-j*2*pi*detector.cell.delay{l})*...
         detector.g;
   end
end
```

mutual_focus.m

```
function [Voc]=mutual_focus(lens,Iin_fd_mutual,detector,optional)

%this funcition calculates the open circuit voltage at a detector
%do to mutual coupling currents
%optional.polar_feed 'yes','no' to use polarization between feed and lens
%                    default is 'yes'
%optional.pathloss_feed 'yes','no' to use pathloss between feed and lens
%                    default is 'yes'

Voc=0;
optional_feed.polar='yes';
optional_feed.pathloss='yes';

if nargin==4
   if isfield(optional,'polar_feed')
      optional_feed.polar=optional.polar_feed;
   end

   if isfield(optional,'pathloss_feed')
      optional_feed.pathloss=optional.pathloss_feed;
   end
end

for k=1:lens.N_c
   for l=1:detector.N_c
      [fd_antenna]=pull_antenna(lens,k,'feed');
      [det_antenna]=pull_antenna(detector,l);
      [Voc_l]=couple(fd_antenna,det_antenna,Iin_fd_mutual(k),...
         optional_feed);
      Voc=Voc+Voc_l*detector.cell.g{l}*detector.g*...
         exp(-j*2*pi*detector.cell.delay{l});
   end
end
```

omni_antenna.m

```
function [le_t,ro_t]=omni_antenna(theta,phi,polar)


%function [le_t,ro_t]=omni_antenna(theta,phi,polar)
%this function returns the magnitude of the effective length and polarization
%for a omni directional in local cooridants at angels (theta,phi)
%The structure 'polar' contains the polarization of the omin directional
%antenna in local xyz coordiants the default is x-polarized.

if isempty(polar);
```

```
   polar=[0;1;0];
end

J0=sqrt(1/(377*pi));
J_sph=xyz2sph(theta,phi);
if abs(theta-pi/2)> 10e-3
   ro_t=[0 0 0;0 1 0;0 0 1]*J_sph*polar;
   ro_t=ro_t/sqrt(ro_t'*ro_t) ;
   le_t=J0;
else
   ro_t=[0;0;0];
   le_t=0;
end
```

omni_mutual.m

```
function [Z_21]=omni_mutual(r_21,extra)

%this function calculates the complex impedance between to omni antennas
%extra contains the ratio of free space wave length  to effective
%wave length in material

%update with papers

Z_21_mag=1/(r_21'*r_21);
Z_21_phs=exp(-j*2*pi*sqrt(r_21'*r_21)*extra)
```

packant.m

```
function [lens]=packant(lens,antenna)

%[lens]=packant(lens,antenna)
%This function orients the antennas in the lens.  It assumes the antenna
%normals to point away from the lens.

a_x=[1 0 0];
a_y=[0 1 0];
a_z=[0 0 1];

for m=1:lens.N_c
   lens.cell.nfd.name{m}=antenna.nfd.name;
   lens.cell.nfd.polar{m}=antenna.nfd.polar;
   lens.cell.nfd.theta_r{m}=antenna.nfd.theta_r;
   lens.cell.nfd.base{m}=get_base([0 0 -1]);
   lens.cell.nfd.local{m}=rotate_base(lens.cell.nfd.base{m},antenna.nfd.theta_r);
   lens.cell.fd.name{m}=antenna.fd.name;
   lens.cell.fd.polar{m}=antenna.fd.polar;
   lens.cell.fd.theta_r{m}=antenna.fd.theta_r;
   lens.cell.fd.base{m}=get_base([0 0 1]);
   lens.cell.fd.local{m}=rotate_base(lens.cell.fd.base{m},antenna.fd.theta_r);

end
```

packant2.m

```
function [array]=packant2(array,antenna)

%[array]=packant2(array,antenna)
%This function orients the antennas in the array.  It assumes the antenna
%normals to point away from the array.

a_x=[1 0 0];
a_y=[0 1 0];
a_z=[0 0 1];

for m=1:array.N_c
   array.cell.nfd.name{m}=antenna.nfd.name;
   array.cell.nfd.polar{m}=antenna.nfd.polar;
   array.cell.nfd.theta_r{m}=antenna.nfd.theta_r;
   array.cell.nfd.base{m}=get_base([0 0 -1]);
   array.cell.nfd.local{m}=rotate_base(array.cell.nfd.base{m},...
      antenna.nfd.theta_r);
end
```

packcircle.m

```
function [lens]=packcircle(latice,lens)



%function [lens]=packcircle(dimensionunitcellsize,latice)
%this is a private function that packs the unitcells into a circle
%based on latice conditions

%initialize variables
rad=lens.dimension(1);
lens.R=[];
lens.cell.nfd.pos=[];
x_store=[];
y_store=[];
z_store=[];

switch latice{1}
   case {'max'}

      switch latice{2}
         case {'best'}
            lens.N_r=round(2*rad/lens.cell.height);
         case {'limit'}
            lens.N_r=floor(2*rad/lens.cell.height);
      end

      temp=[1:lens.N_r];
      temp2=((lens.N_r/2)+0.5)*ones(1,lens.N_r);
      temp3=lens.cell.height*(temp-temp2);
      y_temp=temp3([lens.N_r:-1:1]);

      for m=1:lens.N_r;
         row_length=2*sqrt((rad)^2-(y_temp(m))^2);
         switch char(latice(2))
            case {'best'}
               N_cell=round(row_length/lens.cell.width);
            case {'limit'}
               N_cell=floor(row_length/lens.cell.width);
         end

         temp=[1:N_cell];
         temp2=((N_cell/2)+0.5)*ones(1,N_cell);
         x_temp=lens.cell.width*(temp-temp2);

         lens.R=[lens.R,N_cell];
         x_store=[x_store,x_temp];
         y_store=[y_store,y_temp(m)*ones(1,N_cell)];
         z_store=[z_store,y_temp(m)*zeros(1,N_cell)];

      end

      for m=1:length(x_store)
         lens.cell.nfd.pos{m}=[x_store(m);y_store(m);z_store(m)];
      end


   case {'triangle'}

      switch latice{2}
         case {'best'}
            lens.N_r=round(2*rad/lens.cell.height);
         case {'limit'}
            lens.N_r=floor(2*rad/lens.cell.height);
      end

      temp=[1:lens.N_r];
      temp2=((lens.N_r/2)+0.5)*ones(1,lens.N_r);
      temp3=lens.cell.height*(temp-temp2);
      y_temp=temp3([lens.N_r:-1:1]);

      %find offset at row one
      mid_row=round(lens.N_r/2);
      row_length=2*sqrt((rad)^2-y_temp(mid_row)^2);

      switch latice{2}
         case {'best'}
            N_cell=round(row_length/lens.cell.width);
         case {'limit'}
            N_cell=floor(row_length/lens.cell.width);
      end

      if rem(N_cell,2)==0
         mid_offset=0.5;
      else
         mid_offset=0;
      end

      offset=mod((mid_offset+(mid_row-1)*0.5),1);

      for m=1:lens.N_r;
         row_length=2*sqrt((rad)^2-(y_temp(m))^2);

         switch latice{2}
            case {'best'}
               N_cell1=floor(((row_length/2)+(-offset+0.5)*lens.cell.width) ...
                  /lens.cell.width);

               if rem(((row_length/2)+(-offset+0.5)*lens.cell.width), ...
                  lens.cell.width)> 0.75*lens.cell.width

                  N_cell1=N_cell1+1;
               end

               N_cell2=floor(((row_length/2)+(offset-0.5)*lens.cell.width) ...
                  /lens.cell.width);

               if rem(((row_length/2)+(offset-0.5)*lens.cell.width), ...
                  lens.cell.width)> 0.75*lens.cell.width

                  N_cell2=N_cell2+1;
               end

               N_cell=N_cell1+N_cell2;
```

```
            case {'limit'}
               N_cell1=floor(((row_length/2)+(-offset+0.5)*lens.cell.width) ...
                  /lens.cell.width);
               N_cell2=floor(((row_length/2)+(offset-0.5)*lens.cell.width) ...
                  /lens.cell.width);
               N_cell=N_cell1+N_cell2;
         end


         temp=[1:N_cell];
         temp2=((N_cell2+1 )-offset)*ones(1,N_cell);
         x_temp=lens.cell.width*(temp-temp2);

         lens.R=[lens.R,N_cell];
         x_store=[x_store,x_temp];
         y_store=[y_store,y_temp(m)*ones(1,N_cell)];
         z_store=[z_store,y_temp(m)*zeros(1,N_cell)];

         offset=mod(offset+0.5,1);
      end
      for m=1:length(x_store)
         lens.cell.nfd.pos{m}=[x_store(m);y_store(m);z_store(m)];
      end
end
```

packrect.m

```
function [lens]=packrect(latice,lens)



% [lens]=packrect(dimension,unitcellsize,latice)
%this is a private function that packs the unitcells into a rectangle
%based on latice conditions

%initalize variables
width=lens.dimension(1);
height=lens.dimension(2);
lens.R=[];
x_store=[];
y_store=[];
z_store=[];

switch latice{1}

   case {'max'}

      switch latice{2}
      case {'best'}
         lens.N_r=round(height/lens.cell.height);
      case {'limit'}
         lens.N_r=floor(height/lens.cell.height);
   end

   temp=[1:lens.N_r];
   temp2=((lens.N_r/2)+0.5)*ones(1,lens.N_r);
   temp3=lens.cell.height*(temp-temp2);
   y_temp=temp3([lens.N_r:-1:1]);

   for m=1:lens.N_r;
      switch latice{2}
         case {'best'}
            N_cell=round(width/lens.cell.width);
         case {'limit'}
            N_cell=floor(width/lens.cell.width);
      end

      temp=[1:N_cell];
      temp2=((N_cell/2)+0.5)*ones(1,N_cell);
      x_temp=lens.cell.width*(temp-temp2);

      lens.R=[lens.R,N_cell];
      x_store=[x_store,x_temp];
      y_store=[y_store,y_temp(m)*ones(1,N_cell)];
      z_store=[z_store,y_temp(m)*zeros(1,N_cell)];

   end

   for m=1:length(x_store)
      lens.cell.nfd.pos{m}=[x_store(m);y_store(m);z_store(m)];
   end



case {'triangle'}

   switch latice{2}
      case {'best'}
         lens.N_r=round(height/lens.cell.height);
      case {'limit'}
         lens.N_r=floor(height/lens.cell.height);
   end

   temp=[1:lens.N_r];
   temp2=((lens.N_r/2)+0.5)*ones(1,lens.N_r);
   temp3=lens.cell.height*(temp-temp2);
   y_temp=temp3([lens.N_r:-1:1]);

   %find offset at row one
```

```
mid_row=round(lens.N_r/2);
switch latice{2}
   case {'best'}
      N_cell=round(width/lens.cell.width);
   case {'limit'}
      N_cell=floor(width/lens.cell.width);
end

if rem(N_cell,2)==0
   mid_offset=0.5;
else
   mid_offset=0;
end

offset=mod((mid_offset+(mid_row-1)*0.5),1);

for m=1:lens.N_r;

   switch latice{2}
      case {'best'}
         N_cell1=floor(((width/2)+(-offset+0.5)*lens.cell.width) ...
            /lens.cell.width);

         if rem(((width/2)+(-offset+0.5)*lens.cell.width), ...
            lens.cell.width)> 0.75*lens.cell.width

            N_cell1=N_cell1+1;
         end

         N_cell2=floor(((width/2)+(offset-0.5)*lens.cell.width) ...
               /lens.cell.width);

         if rem(((width/2)+(offset-0.5)*lens.cell.width), ...
            lens.cell.width)> 0.75*lens.cell.width

            N_cell2=N_cell2+1;
         end
         N_cell=N_cell1+N_cell2;
      case {'limit'}
         N_cell1=floor(((width/2)+(-offset+0.5)*lens.cell.width) ...
            /lens.cell.width);
         N_cell2=floor(((width/2)+(offset-0.5)*lens.cell.width) ...
            /lens.cell.width);
         N_cell=N_cell1+N_cell2;
   end


   temp=[1:N_cell];
   temp2=((N_cell2+1)-offset)*ones(1,N_cell);
   x_temp=lens.cell.width*(temp-temp2);
```

```
         lens.R=[lens.R,N_cell];
         x_store=[x_store,x_temp];
         y_store=[y_store,y_temp(m)*ones(1,N_cell)];
         z_store=[z_store,y_temp(m)*zeros(1,N_cell)];
         offset=mod(offset+0.5,1);
      end

      for m=1:length(x_store)
         lens.cell.nfd.pos{m}=[x_store(m);y_store(m);z_store(m)];
      end
end
```

patch_antenna.m

```
function [le_t,ro_t]=patch_antenna(theta,phi,polar)


%function [le_t,ro_t]=patch_antenna(theta,phi,polar)
%this function returns the magnitude of the effective length and polarization
%for a patch in local cooridants at angels (theta,phi).  The structure
%'polar' contains additional informationi specific to this antenna type.
%polar.Le   is the effective length of patch antenna
%polar.We   is the effective width of patch antenna
%polar.h    is the effective heigth of patch antenna
%polar.crosspole  is the cross polar ratio in dB  default is infinite
%polar.delta   is the phase shift between the two polarizations (determines
%             RHP LHP)

global Globalcal_patch

needcal='no';
if ~isempty(Globalcal_patch)
   cal=Globalcal_patch;
   verify='yes';
elseif exist('patch_antenna_cal.mat','file')
   disp('reading patch cal from file')
   load('patch_antenna_cal');
   Globalcal_patch=cal;
   verify='yes';
else
   needcal='yes'
   verify='no';
end

if strcmp(verify,'yes')
   if cal.polar.Le~=polar.Le | cal.polar.We~=polar.We | cal.polar.h~=polar.h
      needcal='yes';
   else
      if isfield(cal.polar,'crosspole') & isfield(polar,'crosspole')
         if cal.polar.crosspole~=polar.crosspole | cal.polar.delta~=polar.delta
            needcal='yes';
         end
      elseif isfield(cal.polar,'crosspole') & ~isfield(polar,'crosspole')
         needcal='yes';
      elseif ~isfield(cal.polar,'crosspole') & isfield(polar,'crosspole')
         needcal='yes';
      end
   end
end

switch(needcal)
   case('yes')

      disp('calibrating patch antenna')

      Prad=0;

      theta_step=pi/60;
      phi_step=pi/20;
      theta_temp=[0:theta_step:pi/2];
      phi_temp=[0:phi_step:2*pi];

      L_mag=[];
      flux=0;
      for k=1:length(theta_temp)
         for l=1:length(phi_temp)
            L_temp=patch_antenna_L(theta_temp(k),phi_temp(l),polar);
            L_mag(k,l)=L_temp'*L_temp;
         end
      end

      for k=1:length(theta_temp)-1
         for l=1:length(phi_temp)-1
            L_mag_avg=sum([L_mag(k,l),L_mag(k+1,l),L_mag(k,l+1),...
               L_mag(k+1,l+1)])/4;
            area=sin((theta_temp(k)+theta_temp(k+1))/2)*theta_step*phi_step;
            flux=flux+L_mag_avg*area;
         end
      end

      M0=1/sqrt(((2*pi)^2/(16*377*pi^2))*flux);

      cal.M0=M0;
      cal.polar=polar;

      Globalcal_patch=cal;
      save /home/sydney/vian/matlab/toolbox/QOlens/working/patch_antenna_cal cal
   case('no')

      M0=cal.M0;
end

L=patch_antenna_L(theta,phi,polar);
field=(-M0/377)*cross([1;0;0],L);

if field'*field < 10e-9
   le_t=0;
   ro_t=[0;0;0];
else
   le_t=sqrt(field'*field);
   ro_t=field/le_t;
end
```

patch_antenna_L.m

```
function [L]=patch_antenna_L(theta,phi,polar)


%this function calculates the results of the surface integral of the
%magnetics currents.
%'polar' contains additional informationi specific to this antenna type.
%polar.Le   is the effective length of patch antenna
%polar.We   is the effective width of patch antenna
%polar.h    is the effective heigth of patch antenna
%polar.crosspole  is the cross polar ratio in dB  default is infinite
%polar.delta   is the phase shift between the two polarizations (determines
%              RHP LHP)
%note i think i want the polar to contain fields (polar,crosspole,length)


Z=2*pi*polar.h*cos(theta);
X=2*pi*(polar.We/2)*sin(theta)*cos(phi);
AF=2*cos(2*pi*(polar.Le/2)*(sin(theta)*sin(phi)));

if Z==0
   temp1=1;
else
   temp1=sin(Z)/Z;
end

if X==0
   temp2=1;
else
   temp2=sin(X)/X;
end
```

```
L_theta=cos(theta)*cos(phi)*temp1*temp2*AF;
L_phi=-sin(phi)*temp1*temp2*AF;
L=[0;L_theta;L_phi];

if isfield(polar,'crosspole')
   CR=10^(polar.crosspole/10);

   if isfield(polar,'delta')
      delta=polar.delta;
   else
      delta=0;
   end

   Z=2*pi*polar.h*cos(theta);
   Y=2*pi*(polar.Le/2)*sin(theta)*sin(phi);
   AF=2*cos(2*pi*(polar.We/2)*(sin(theta)*cos(phi)));

   if Z==0
      temp1=1;
   else
      temp1=sin(Z)/Z;
   end

   if Y==0
      temp2=1;
   else
      temp2=sin(Y)/Y;
   end

   L_theta=cos(theta)*sin(phi)*temp1*temp2*AF;
   L_phi=cos(phi)*temp1*temp2*AF;
   L_temp=[0;L_theta;L_phi]*exp(j*delta);

   A=1/(1+CR);
   B=1-A;
   L=B*L+A*L_temp;

end

if theta>pi/2
   L=[0;0;0];
end
```

patch_mutual.m

```
function [Z_21]=patch_mutual(r_21,extra)

%this function calculates the complex impedance between to patch antennas
%extra.lambda_eff  contains the ratio of free space wave length  to effective
```

```
%wave length in material
%extra.ro contain the scale constant for the magnitude coupling

%update with papers

theta=atan2(sqrt(r_21(1)^2+r_21(2)^2),r_21(3));

if theta <0
   theta = theta + 2*pi;
end

phi=atan2(r_21(2),r_21(1));

if phi <0
   phi=phi+2*pi;
end

%define phi=0 hen theta is 0 or pi
if theta < 10e-4 | abs(theta-pi) < 10e-4
   phi=0;
end

r_21_mag=sqrt(r_21'*r_21);
Z_21_mag=(extra.ro/r_21_mag)^(1+sin(phi));
Z_21_phs=exp(-j*2*pi*r_21_mag*extra.lambda_eff);
Z_21=Z_21_mag*Z_21_phs;
```

pull_antenna.m

```
function [antenna]=pull_antenna(device,antenna_num,lens_side)

%this function pulls out one antenna from a device, the lens_side
%is only needed if the device is a lens and not a detector

switch device.device
   case('array')
      antenna.name=device.cell.nfd.name{antenna_num};
      antenna.polar=device.cell.nfd.polar{antenna_num};
      antenna.base=device.cell.nfd.base{antenna_num};
      antenna.local=device.cell.nfd.local{antenna_num};
      antenna.pos=device.cell.nfd.pos{antenna_num};
      antenna.theta_r=device.cell.nfd.theta_r{antenna_num};

   case('lens')
      switch lens_side
         case('nonfeed')
            antenna.name=device.cell.nfd.name{antenna_num};
            antenna.polar=device.cell.nfd.polar{antenna_num};
            antenna.base=device.cell.nfd.base{antenna_num};
            antenna.local=device.cell.nfd.local{antenna_num};
            antenna.pos=device.cell.nfd.pos{antenna_num};
            antenna.theta_r=device.cell.nfd.theta_r{antenna_num};
         case('feed')
            antenna.name=device.cell.fd.name{antenna_num};
            antenna.polar=device.cell.fd.polar{antenna_num};
            antenna.base=device.cell.fd.base{antenna_num};
            antenna.local=device.cell.fd.local{antenna_num};
            antenna.pos=device.cell.fd.pos{antenna_num};
            antenna.theta_r=device.cell.fd.theta_r{antenna_num};
      end
   case('detector')
      antenna.name=device.cell.name{antenna_num};
      antenna.polar=device.cell.polar{antenna_num};
      antenna.base=device.cell.base{antenna_num};
      antenna.local=device.cell.local{antenna_num};
      antenna.pos=device.cell.pos{antenna_num};
      antenna.theta_r=device.cell.theta_r{antenna_num};
end
```

rotate_base.m

```
function [local]=rotate_base(base,theta_r)

%[local]=rotate_base(base,theta_r)
%this function rotates the base coordinat system through an angle 'theta_r'
%in the base coordiants system xy plane

rotate=[cos(theta_r) sin(theta_r) 0; -sin(theta_r) cos(theta_r) 0; 0 0 1];
local=rotate*base;
```

semi_antenna.m

```
function [le_t,ro_t]=semi_antenna(theta,phi,polar)


%function [le_t,ro_t]=semi_antenna(theta,phi,polar)
%this function returns the magnitude of the effective length and polarization
%for a semi directional in local cooridants at angels (theta,phi)
%The structure 'polar' contains the polarization of the omin directional
%antenna in local xyz coordiants the default is x-polarized.

if isempty(polar);
   polar=[0;1;0];
```

```
end
J0=sqrt(2/(377*pi));
J_sph=xyz2sph(theta,phi);
if theta < (pi/2-10e-3)
   ro_t=[0 0 0;0 1 0;0 0 1]*J_sph*polar;
   ro_t=ro_t/sqrt(ro_t'*ro_t) ;
   le_t=J0;
else
   ro_t=[0;0;0];
   le_t=0;
end
```

semi_mutual.m

```
function [Z_21]=semi_mutual(r_21,extra)

%this function calculates the complex impedance between to semi antennas
%extra contains the ratio of free space wave length  to effective
%wave length in material

%update with papers

Z_21_mag=1/(r_21'*r_21);
Z_21_phs=exp(-j*2*pi*sqrt(r_21'*r_21)*extra)
```

xyz2sph.m

```
function [J_sph]=xyz2sph(theta,phi)

%function [J_sph]=xyz2sph(theta,phi)
%this function creates an Jacobian to convert cartisian vectors to
%spherical vectors

J_sph=[sin(theta)*cos(phi) sin(theta)*cos(phi) cos(theta); ...
       cos(theta)*cos(phi) cos(theta)*sin(phi) -1*sin(theta); ...
       -1*sin(phi)         cos(phi)            0 ];
```

LMS Code

adapt_rad_ptrn.m

```
function [rad_ptrn]=adapt_rad_ptrn(lens,imager,weight,alpha_range,ki_range,optional)
```

```
%this function outputs the receive Voc at a detector defined by the imager
%and weights for a source with unit input.
%'rad_ptrn' contains fields 'Voc', 'alpha' and 'ki'.
%alpha_range is a vector that contains[alpha_start,alpha_end,alpha_N]
%ki_range is a vector that contains[ki_start,ki_end,ki_N]
%optional.polar_source use polarization for source coupling
%                    default is 'yes'
%optional.polar_feed use polarization for feed coupling
%                    default is 'yes'
%optional.pathloss_source use pathloss for source coupling
%                    default is 'yes'
%optional.pathloss_feed use pathloss for feed coupling
%                    default is 'yes'
optional_rad.polar_source='yes';
optional_rad.polar_feed='yes';
optional_rad.pathloss_source='yes';
optional_rad.pathloss_feed='yes';
if nargin==6
   optional_rad=optional;
end

for m=1:length(imager)
   m/length(imager)
   if weight(m)~=0
      detector_m=imager{m};
      [rad_ptrn_m]=rad_pattern(lens,detector_m,alpha_range,ki_range...
         ,optional_rad);
      if m==1
         rad_ptrn=rad_ptrn_m;
         rad_ptrn.Voc=weight(m)*rad_ptrn_m.Voc;
      else
         rad_ptrn.Voc=rad_ptrn.Voc+weight(m)'*rad_ptrn_m.Voc;
      end
   end
end
```

cal_W_est .m

```
function [W_est]=cal_W_est(R_est,P_est,mask)

mask_pos=[];
for k=1:length(mask)
   if mask(k)==1
      mask_pos=[mask_pos,k];
   end
end

R_reduce=R_est(mask_pos,mask_pos);
```

```
R_reduce_inv=inv(R_reduce);
R_sub_inv=zeros(length(mask));
R_sub_inv(mask_pos,mask_pos)=R_reduce_inv;
W_est=R_sub_inv*P_est;
```

csf.m

```
function [signal]=csf(signal,field)

%this function changes the signal field
%field is a vector for each source in the channel


channel_temp=signal.channel;
for k=1:length(field)
   source_temp=channel_temp{k};
   source_temp.field=field(k);
   channel_temp{k}=source_temp;
end
signal.channel=channel_temp;
```

drawvar.m

```
function drawvar(pos,radius,sides,color)

%drawpolygon(local,pos,radius,sides,color)
%this is a private function
%this function draws a polygon aperture with local coordinates system 'local'
% and position 'pos' in global coordinates

hold on

theta=[0:2*pi/sides:2*pi];
point=[];
for k=1:length(theta)
   x=radius*cos(theta(k));
   y=radius*sin(theta(k));
   point=[point,[x;y;0]+pos];
end

x=point(1,:);
y=point(2,:);
z=point(3,:);
plot3(x,y,z,color);
hold off
```

get_data.m

```
function [data_new]=get_data(constell)

%this function randomly detemines the next data sample for a given
%constellation and symbol period.  All constellations have peak power
%of unity.
% support for 'QPSK' 'BPSK' 'carrier'

switch(constell)
   case('QPSK')
      symbols=[1,j,-1,-j];
   case('BPSK')
      symbols=[1, -1]
   case('carrier')
      symbols=[1, 1];
end

data_new=symbols(1+floor(length(symbols)*rand(1,1)));
```

get_data_pwr.m

```
function [data_pwr]=get_data_pwr(constell)

%this function calculates the data pwr
% support for 'QPSK' 'BPSK' 'carrier'

switch(constell)
   case('QPSK')
      symbols=[1,j,-1,-j];
   case('BPSK')
      symbols=[1, -1]
   case('carrier')
      symbols=[1, 1];
end

data_pwr=symbols*symbols'/length(symbols);
```

plot_error.m

```
function plot_error(sub_non,window)

font_size=10;
colors=['y','m','c','r','g','b','y'];
[m,n]=size(sub_non.QOL.e);
```

```
for l=1:m
   for k=1:window-1
      e_temp(l,k)=sum(sub_non.QOL.e(l,[1:k]));
   end
   for k=window:n
      e_temp(l,k)=sum(sub_non.QOL.e(l,[k-window+1:k]));
   end
end

for k=1:10:m
   hold on
   plot(abs(e_temp(k,:)),[colors(mod(k,7)+1),'-'])
   [H_temp]=text(n,e_temp(n),num2str(k,3));
   set(H_temp,'FontSize',font_size)
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('error')
```

plot_eye.m

```
function plot_eye(sub_non,item,type,pass,N_samples)

%this function plots the y, and estimated mean and variance in a eye pattern
%for the item (QOL or array) with pass (for array it is one)
%type is for normal of opt

clf
N_y=length(sub_non.array.y);
if strcmp(item,'array')
   if strcmp(type,'normal')
      hold on
      plot(sub_non.array.y([N_y-N_samples+1:N_y]),'b.')
      hold off
      for k=1:length(sub_non.array.y_SNR.y_symbol.mean)
      hold on
         plot(sub_non.array.y_SNR.y_symbol.mean(k),'ro')
         drawvar([real(sub_non.array.y_SNR.y_symbol.mean(k));...
            ;imag(sub_non.array.y_SNR.y_symbol.mean(k));0],...
            sqrt(sub_non.array.y_SNR.y_symbol.var(k)),20,'r')
      hold off
      end
   end
   if strcmp(type,'opt')
      hold on
      plot(sub_non.array.y_opt([N_y-N_samples+1:N_y]),'b.')
      hold off
      for k=1:length(sub_non.array.y_SNR.y_symbol_opt.mean)
      hold on
         plot(sub_non.array.y_SNR.y_symbol_opt.mean(k),'ro')
         drawpolygon([1 0 0;0 0 -1;0 1 0],...
            [real(sub_non.array.y_SNR.y_symbol_opt.mean(k));...
            ;0;imag(sub_non.array.y_SNR.y_symbol_opt.mean(k))],...
            sqrt(sub_non.array.y_SNR.y_symbol_opt.var(k)),20,'r')
      hold off
      end
   end
end
if strcmp(item,'QOL')
   if strcmp(type,'normal')
      hold on
      plot(sub_non.QOL.y(pass,[N_y-N_samples+1:N_y]),'b.')
      hold off
      for k=1:length(sub_non.QOL.y_SNR.y_symbol.mean(:,pass))
      hold on
         plot(sub_non.QOL.y_SNR.y_symbol.mean(k,pass),'ro')
         drawpolygon([1 0 0;0 0 -1;0 1 0],...
            [real(sub_non.QOL.y_SNR.y_symbol.mean(k,pass));...
            ;0;imag(sub_non.QOL.y_SNR.y_symbol.mean(k,pass))],...
            sqrt(sub_non.QOL.y_SNR.y_symbol.var(k,pass)),20,'r')
      hold off
      end
   end
   if strcmp(type,'opt')
      hold on
      plot(sub_non.QOL.y_opt(pass,[N_y-N_samples+1:N_y]),'b.')
      hold off
      for k=1:length(sub_non.QOL.y_SNR.y_symbol_opt.mean(:,pass))
      hold on
         plot(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,pass),'ro')
         drawpolygon([1 0 0;0 0 -1;0 1 0],...
            [real(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,pass));...
            ;0;imag(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,pass))],...
            sqrt(sub_non.QOL.y_SNR.y_symbol_opt.var(k,pass)),20,'r')
      hold off
      end
   end
end
axis('equal')
%axis('square')
```

plot_eye_para.m

```
function plot_eye_para(sub_non,type,color)

%this function plots the y, and estimated mean and variance in a eye pattern
%for the item (QOL or QOL) with pass (for QOL it is one)
```

```
%type is for normal of opt

[m,n]=size(sub_non.QOL.y_SNR.y_symbol.mean);
if strcmp(type,'normal')
   for l=[[1:10],n]
      for k=1:m
      hold on
         plot(sub_non.QOL.y_SNR.y_symbol.mean(k,l),[color,'o'])
         drawpolygon([1 0 0;0 0 -1;0 1 0],...
            [real(sub_non.QOL.y_SNR.y_symbol.mean(k,l));...
            ;0;imag(sub_non.QOL.y_SNR.y_symbol.mean(k,l))],...
            sqrt(sub_non.QOL.y_SNR.y_symbol.var(k,l)),20,color)
      hold off
      end
   end
end
if strcmp(type,'opt')
   for l=1:n
      for k=1:m
      hold on
         plot(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,l),[color,'o'])
         drawpolygon([1 0 0;0 0 -1;0 1 0],...
            [real(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,l));...
            ;0;imag(sub_non.QOL.y_SNR.y_symbol_opt.mean(k,l))],...
            sqrt(sub_non.QOL.y_SNR.y_symbol_opt.var(k,l)),20,color)
      hold off
      end
   end
end
axis('equal')
```

plot_signals_pos.m

```
function plot_signals_pos(signals)

%this function plots the signals on the radiation plattern plots
%the numbering is signals,source

figure(3)
for k=1:length(signals)
   signal_k=signals{k};
   channel=signal_k.channel
   for l=1:length(channel)
      source_kl=channel{l};

      hold on
      switch (source_kl.type)
         case('farfield')
            angle_kl=source_kl.angle;
            x=angle_kl(1)*cos(angle_kl(2));
            y=angle_kl(1)*sin(angle_kl(2));
            plot3(x,y,0,'bx')
            text(x,y,0,[num2str(k,2),',num2str(l,2)]);
         case('nearfield')
            pos_kl=source_kl.pos
            alpha=atan2(sqrt(pos_kl(1)^2+pos_kl(2)^2),pos_kl(3));
            ki=atan2(pos_kl(2),pos_kl(1));
            if ki<0
               ki=ki+2*pi;
            end
            x=angle_kl(alpha)*cos(angle_kl(ki));
            y=angle_kl(alpha)*sin(angle_kl(ki));
            plot3(x,y,0,'bo')
            text(x,y,0,[num2str(k,2),',num2str(l,2)]);
      end
      hold off
   end
end
```

plot_W_est_QOL.m

```
function plot_W_est_QOL(sub_non)
N_imager=121
for k=[1:11]
   mask=zeros(N_imager,1);
   mask(sub_non.QOL.order([1:k]))=ones(k,1);
   R_est=sub_non.QOL.R_est{k};
   P_est=sub_non.QOL.P_est(:,k);
   W_est(:,k)=cal_W_est(R_est,P_est,mask);
end
plot_weights(W_est);
```

plot_weight_noise.m

```
function plot_weigh_noise(sub_nons)

font_size=10;
colors=['y','m','c','r','g','b','y'];

figure(1)
clf
for k=1:length(sub_nons)
   hold on
   sub_non=sub_nons{k};
   plot(10*log10(sub_non.QOL.W_noise.mean),[colors(mod(k,7)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(weight noise mean)')

figure(2)
clf
for k=1:length(sub_nons)
   hold on
   sub_non=sub_nons{k};
   plot(10*log10(sub_non.QOL.W_noise.var),[colors(mod(k,7)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(weight noise variance)')
```

plot_weight_noise_cal.m

```
function plot_weigh_noise_cal(sub_non)

font_size=10;
colors=['y','m','c','r','g','b','y'];

[m,n]=size(sub_non.QOL.W_noise.mean);
figure(1)
clf
for k=1:m
   hold on
   plot(10*log10(sub_non.QOL.W_noise.mean(k,:)),[colors(mod(k,6)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(weight noise mean)')

figure(2)
clf
for k=1:m
   hold on
   plot(10*log10(sub_non.QOL.W_noise.var(k,:)),[colors(mod(k,6)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(weight noise variance)')

figure(3)
```

```
clf
for k=1:m
   hold on
   plot(10*log10(abs(sub_non.QOL.error_min(k,:))),[colors(mod(k,6)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(|Minimum Error|)')

figure(4)
clf
for k=1:m
   hold on
   plot(10*log10(abs(sub_non.QOL.mu(k,:))),[colors(mod(k,6)+1),'-'])
   hold off
end
axis('square')
xlabel('number of signals')
ylabel('10log_{10}(mu)')
```

plot_weights.m

```
function plot_weights(weights,optional)

%this function plots the weight tracks
%optional.font_size  default is 10
%optional.figure  default is 8

font_size=10;
fig=8;

if nargin==2
   if isfield(optional,'font_size')
      font_size=optional.font_size;
   end
   if isfield(optional,'figure')
      fig=optional.figure;
   end
end

figure(fig)
clf

colors=['y','m','c','r','g','b','y'];
[m,n]=size(weights);

for k=1:m
   hold on
   plot(real(weights(k,:)),imag(weights(k,:)),[colors(mod(k,7)+1),'.'])
   plot(real(weights(k,:)),imag(weights(k,:)),[colors(mod(k,7)+1),':'])
   hold off
end

for k=1:m
   hold on
   plot(real(weights(k,n)),imag(weights(k,n)),'wo')
   [H_temp]=text(real(weights(k,n)),imag(weights(k,n)),num2str(k,3));
   set(H_temp,'FontSize',font_size)
   hold off
end
axis('equal')
xlabel('Re\{W_k\}')
ylabel('Im\{W_k\}')
```

slicer.m

```
function [data_guess]=get_data(sample,constell)

%this function performs a minium distance guess of the output data
% support for 'QPSK' 'BPSK' 'carrier'

switch(constell)
   case('QPSK')
      symbols=[1,j,-1,-j];
      if angle(sample) <= pi/4 & angle(sample) >= -pi/4
         data_guess=1;
      elseif angle(sample) > pi/4 & angle(sample) < 3*pi/4
         data_guess=j;
      elseif angle(sample) < -pi/4 & angle(sample) > -3*pi/4
         data_guess=-j;
      else
         data_guess=-1;
      end
   case('BPSK')
      symbols=[1, -1]
      if real(sample) > 0
         data_guess=1;
      else
         data_guess=-1;
      end
   case('carrier')
      symbols=[1, 1];
      data_guess=1;
end

data_new=symbols(1+floor(length(symbols)*rand(1,1)));
```

sub_LMS_array.m

```
function [output]=sub_LMS_array(arrayTF_signal,noise_TF,N_samples,signals,time_step,noise_pwr

%this function does the LMS subset processing on array
%the output contians the weights, y, data, for the system

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initized variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_s=length(signals);
[N_array,dumb]=size(noise_TF);
data=zeros(N_s,1);
y=0;
y_opt=0;
W=zeros(N_array,2);
W_opt=zeros(N_array,1);
P_est=zeros(N_array,1);
R_est=zeros(N_array);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%find desire signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
desire_signal=0;
for k=1:N_s
   signal_k=signals{k};
   if strcmp(signal_k.user,'desire')
      desire_signal=k;
   end
end

if desire_signal==0
   disp('no desire user')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate R,P, Wopt and mu
%assume all signals are uncorrelated
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R=zeros(N_array,N_array);
for k=1:N_s
   signal_k=signals{k};
   TF_signal_k=arrayTF_signal{k};
   [m_k,n_k]=size(TF_signal_k);
   R=R+get_data_pwr(signal_k.constell)*(TF_signal_k*ones(n_k,1))*...
      (TF_signal_k*ones(n_k,1))';
   if k==desire_signal
      P=get_data_pwr(signal_k.constell)*(TF_signal_k*ones(n_k,1));
   end
end
if noise_pwr.array~=0
   R=R+noise_pwr.array*eye(N_array);
end
W_opt=inv(R)*P;
mu=.05/trace(R);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



for l=2:N_samples
   %comments(['array LMS  ', num2str(l/N_samples,3)]);

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %get data
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   time_now=(l-1)*time_step;
   time_then=(l-2)*time_step;
   for k=1:N_s
      signal_k=signals{k};
      if time_then<=floor(time_now/signal_k.T_data)*signal_k.T_data...
         & time_now > floor(time_now/signal_k.T_data)*signal_k.T_data
         data(k,l)=get_data(signal_k.constell);
      else
         data(k,l)=data(k,l-1);
      end
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %modulation/demodulation
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   signal_temp=signals{desire_signal};
   w_IF=signal_temp.w_RF;
   d=zeros(N_array,1);
   for k=1:N_s
      signal_k=signals{k};

      TF_signal_k=arrayTF_signal{k};
      [m,n]=size(TF_signal_k);
      d=d+TF_signal_k*data(k,l)*exp(signal_k.w_RF-w_IF)*...
         ones(n,1);

      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      %estimate P
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      if k==desire_signal
         P_est=P_est + abs(data(k,l))^2*(TF_signal_k)*...
            exp(signal_k.w_RF-w_IF)*ones(n,1)/N_samples;
      end
      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %noise
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   [m,n]=size(noise_TF);
   noise_temp=sqrt(noise_pwr.array)*(randn(n,1)+j*randn(n,1))/sqrt(2);
   noise=noise_TF*noise_temp;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %LMS
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   X=d+noise;
   y(l)=W(:,l)'*X;
   y_opt(l)=W_opt'*X;
   e_l=data(desire_signal,l)-y(l);
   e(l)=e_l;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate R
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   R_est=R_est+X*X'/N_samples;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   W(:,l+1)=W(:,l)+2*mu*e_l'*X;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
output.data=data;
output.e=e;
output.y=y;
output.y_opt=y_opt;
output.W_opt=W_opt;
output.W=W;
output.R=R;
output.R_est=R_est;
output.P=P;
output.P_est=P_est;
```

sub_LMS_QOL.m

```
function [output]=sub_LMS_QOL(QOLTF_signal,noise_TF,N_imager,N_samples,signals,time_step,nois

%this function does the LMS subset processing on QOL give the mask
%mu is .05/trace[R]
%the output contians the weights, y, data, for the system
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%initized variables
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N_s=length(signals);
data=zeros(N_s,1);
y=0;
y_opt=0;
W=zeros(N_imager,2);
W_opt=zeros(N_imager,1);
P_est=zeros(N_imager,1);
R_est=zeros(N_imager);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%find desire signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
desire_signal=0;
for k=1:N_s
   signal_k=signals{k};

   if strcmp(signal_k.user,'desire')
      desire_signal=k;
   end
end

if desire_signal==0
   disp('no desire user')
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate R,P, W_opt and mu
%assume all signals are uncorrelated
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
R=zeros(N_imager,N_imager);
for k=1:N_s
   signal_k=signals{k};
   TF_signal_k=QOLTF_signal{k};
   [m_k,n_k]=size(TF_signal_k);
   R=R+get_data_pwr(signal_k.constell)*(TF_signal_k*ones(n_k,1))*...
      (TF_signal_k*ones(n_k,1))';
   if k==desire_signal
      P=get_data_pwr(signal_k.constell)*(TF_signal_k*ones(n_k,1));
   end
end
if noise_pwr.lens~=0
   R=R+noise_pwr.lens*noise_TF_QOL.lens*noise_TF_QOL.lens';
end
if noise_pwr.detector~=0
   for k=1:N_imager
```

```
      noise_TF_k=noise_TF_QOL.det{k};
      R_temp(k,k)=noise_pwr.detector*sum(abs(noise_TF_k).^2);
   end
   R=R+R_temp;
end
if noise_pwr.imager~=0
   R=R+noise_pwr.imager*eye(N_imager);
end
R_sub=diag(mask).'*R*diag(mask);
mu=.05/trace(R_sub);

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %calculating W_opt
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   mask_pos=[];
   for k=1:length(mask)
      if mask(k)==1
         mask_pos=[mask_pos,k];
      end
   end
   R_reduce=R_sub(mask_pos,mask_pos);
   R_reduce_inv=inv(R_reduce);
   R_sub_inv=zeros(length(mask));
   R_sub_inv(mask_pos,mask_pos)=R_reduce_inv;
   W_opt=R_sub_inv*P;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for l=2:N_samples
%  comments(['QOL  LMS   ', num2str(l/N_samples,3)]);

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %get data
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   time_now=(l-1)*time_step;
   time_then=(l-2)*time_step;
   for k=1:N_s
      signal_k=signals{k};
      if time_then<=floor(time_now/signal_k.T_data)*signal_k.T_data...
         & time_now > floor(time_now/signal_k.T_data)*signal_k.T_data
         data(k,l)=get_data(signal_k.constell);
      else
         data(k,l)=data(k,l-1);
      end
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%modulation/demodulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
signal_temp=signals{desire_signal};
w_IF=signal_temp.w_RF;
d=zeros(N_imager,1);
for k=1:N_s
   signal_k=signals{k};

   TF_signal_k=QOLTF_signal{k};
   [m,n]=size(TF_signal_k);
   d=d+TF_signal_k*data(k,l)*exp(signal_k.w_RF-w_IF)*...
      ones(n,1);

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %save desire signal on max detector
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %if sum(mask)==1
   %  if k==desire_signal
   %     output.d(l)= mask'*TF_signal_k*data(k,l)*...
   %        exp(signal_k.w_RF-w_IF)*ones(n,1);
   %  end
   %end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate P
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%  if k==desire_signal
%     P_est=P_est + abs(data(k,l))^2*(mask.*TF_signal_k)*...
%        exp(signal_k.w_RF-w_IF)*ones(n,1)/N_samples;
%  end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%noise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noise=zeros(N_imager,1);
if noise_pwr.lens~=0
   noise_TF_temp=noise_TF.lens;
   [m,n]=size(noise_TF_temp);
   noise_temp=sqrt(noise_pwr.lens)*(randn(n,1)+j*randn(n,1))/sqrt(2);
   noise=noisel+noise_TF_temp*noise_temp;
end

if noise_pwr.detector~=0
   for k=1:length(imager)
      noise_TF_temp=noise_TF.det{k};
      [m,n]=size(noise_TF_temp);
```

```
         noise_temp=sqrt(noise_pwr.detector)*(randn(n,1)+j*randn(n,1))/sqrt(2);
         noise_QOL_temp(k,1)=noise_TF_temp*noise_temp;
      end
   noise=noise+noise_QOL_temp;
   end

   if noise_pwr.imager~=0
      noise_TF_temp=noise_TF.imager;
      [m,n]=size(noise_TF_temp);
      noise_temp=sqrt(noise_pwr.imager)*(randn(n,1)+j*randn(n,1))/sqrt(2);
      noise=noise+noise_TF_temp*noise_temp;
   end

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %noise on max detector
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %output.noise(l)=mask'*noise;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %LMS
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   X=(d.*mask)+(noise.*mask);
   y(l)=(W(:,l).*mask)'*(X.*mask);
   y_opt(l)=(W_opt.*mask)'*(X.*mask);
   e_l=data(desire_signal,l)-y(l);
   e(l)=e_l;

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate R
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %R_est=R_est+X*X'/N_samples;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   W(:,l+1)=W(:,l)+2*mu*e_l'*X;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
output.data=data;
output.e=e;
output.y=y;
output.y_opt=y_opt;
output.W_opt=W_opt;
output.W=W;
output.R=R;
output.R_est=R_est;
output.P=P;
output.P_est=P_est;
```

W_est_SNR.m

```
function [SNR]=W_est_SNR(output,constell,desire_signal,N_sample,TF)

%this function estimates the SNR of y in two different ways
%BER and variance from data
%N_sample is the last N_sample of y to use for the estimation


if strcmp(constell,'QPSK')
   symbol=[1,-1,j,-j];
elseif strcmp(constell,'BPSK')
   symbol=[1,-1];
elseif strcmp(constell,'carrier')
   symbol=[1,1];
end
symbol_start=ones(length(symbol));
symbol_start_opt=ones(length(symbol));

err=0;
err_opt=0;
N_y=length(output.y);
for k=(N_y-N_sample+1):N_y
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate y_signal
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   y_s_est(k+N_sample-N_y)=output.W_est'*output.data(desire_signal,k)*TF;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate BER
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   data_guess=slicer(output.y(k),constell);
   if data_guess~=output.data(desire_signal,k)
      err=err+1;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate eye diagram mean and variance
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for l=1:length(symbol)
      if output.data(desire_signal,k)==symbol(l)
         y_symbol(l,symbol_start(l))=output.y(k);
         symbol_start(l)=symbol_start(l)+1;
      end
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   noise(k+N_sample-N_y)=output.y(k)-output.data(desire_signal,k);
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%estimate noise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y_n_est=output.y([N_y-N_sample+1:N_y])-y_s_est;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate BER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.BER=err/N_sample;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate SNR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.y_SNR_est=(var(y_s_est)+mean(y_s_est)*mean(y_s_est)')/...
   (var(y_n_est)+mean(y_n_est)*mean(y_n_est)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate SNR data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.data=get_data_pwr(constell)/(abs(mean(noise))^2+var(noise));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate y mean and variance on symbols
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for l=1:length(symbol)
   temp=y_symbol(l,[1:symbol_start(l)-1]);
   SNR.y_symbol.mean(l,1)=mean(temp);
   SNR.y_symbol.var(l,1)=var(temp);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

weight_noise.m

```
function [W_noise]=wieght_noise(output,N_sample,N_W)

[m,n]=size(output.W);
n=n-1;
W_var=var(output.W(:,[n-N_sample+1:n]).').';
W_noise.mean=sum(W_var)/N_W;
if N_W~=1
   W_noise.var=sum(W_var'*W_var)/(N_W-1);
else
   W_noise.var=0;
```

```
end
```

win_stats.m

```
function [stats]=win_stats(data,window)

%this function calculates the mean and variance for a windowed set of data
%out of the data vector.  The window moves from the start of the data set
%to the end of the data set changing in size at the beginning.  the last
%'window' samples for each point is used. if data is
%a matrix, then the rows are operated on.

[m,n]=size(data);
for k=1:m
   for l=1:n
      if l-window < 0
         start=1;
      else
         start=l-window+1;
      end
      stats.mean(k,l)=mean(data(k,[start:l]));
      stats.var(k,l)=var(data(k,[start:l]));
   end
end
```

y_SNR.m

```
function [SNR]=y_SNR(output,constell,desire_signal,N_sample,TF)

%this function estimates the SNR of y in two different ways
%BER and variance from data
%N_sample is the last N_sample of y to use for the estimation

[m,n]=size(output.W);
n=n-1;
W.mean=mean(output.W(:,[n-N_sample+1:n]).').';
SNR.W_mean=W.mean;

if strcmp(constell,'QPSK')
   symbol=[1,-1,j,-j];
elseif strcmp(constell,'BPSK')
   symbol=[1,-1];
elseif strcmp(constell,'carrier')
   symbol=[1,1];
end
```

```
symbol_start=ones(length(symbol));
symbol_start_opt=ones(length(symbol));

err=0;
err_opt=0;
N_y=length(output.y);
for k=(N_y-N_sample+1):N_y
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate y_signal
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   y_s_opt(k+N_sample-N_y)=output.W_opt'*output.data(desire_signal,k)*TF;
   y_s_est(k+N_sample-N_y)=W.mean'*output.data(desire_signal,k)*TF;
   y_s(k+N_sample-N_y)=output.W_opt'*output.data(desire_signal,k)*TF;
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate BER
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   data_guess=slicer(output.y(k),constell);
   if data_guess~=output.data(desire_signal,k)
      err=err+1;
   end
   data_guess=slicer(output.y_opt(k),constell);
   if data_guess~=output.data(desire_signal,k)
      err_opt=err_opt+1;
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %estimate eye diagram mean and variance
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for l=1:length(symbol)
      if output.data(desire_signal,k)==symbol(l)
         y_symbol(l,symbol_start(l))=output.y(k);
         symbol_start(l)=symbol_start(l)+1;
         y_symbol_opt(l,symbol_start(l))=output.y_opt(k);
         symbol_start_opt(l)=symbol_start_opt(l)+1;
      end
   end
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   noise(k+N_sample-N_y)=output.y(k)-output.data(desire_signal,k);
   noise_opt(k+N_sample-N_y)=output.y_opt(k)-output.data(desire_signal,k);
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%estimate noise
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
y_n_opt=output.y_opt([N_y-N_sample+1:N_y])-y_s_opt;
y_n_est=output.y([N_y-N_sample+1:N_y])-y_s_est;
y_n=output.y([N_y-N_sample+1:N_y])-y_s;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate BER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.BER_opt=err_opt/N_sample;
SNR.BER=err/N_sample;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate SNR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.y_SNR=(var(y_s)+mean(y_s)*mean(y_s)')/(var(y_n)+mean(y_n)*mean(y_n)');
SNR.y_SNR_opt=(var(y_s_opt)+mean(y_s_opt)*mean(y_s_opt)')/...
   (var(y_n_opt)+mean(y_n_opt)*mean(y_n_opt)');
SNR.y_SNR_est=(var(y_s_est)+mean(y_s_est)*mean(y_s_est)')/...
   (var(y_n_est)+mean(y_n_est)*mean(y_n_est)');
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate SNR data
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SNR.data=get_data_pwr(constell)/(abs(mean(noise))^2+var(noise));
SNR.data_opt=get_data_pwr(constell)/(abs(mean(noise_opt))^2+var(noise_opt));
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%calculate y mean and variance on symbols
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for l=1:length(symbol)
   temp=y_symbol(l,[1:symbol_start(l)-1]);
   SNR.y_symbol.mean(l,1)=mean(temp);
   SNR.y_symbol.var(l,1)=var(temp);
   temp=y_symbol_opt(l,[1:symbol_start_opt(l)-1]);
   SNR.y_symbol_opt.mean(l,1)=mean(temp);
   SNR.y_symbol_opt.var(l,1)=var(temp);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

y_SNR_wiener.m

```
function [y_SNR]=y_SNR_wiener(output,TF,desire_signal,N_s)

[m,n]=size(output.W);
n=n-1;
W.mean=mean(output.W(:,[n-N_s+1:n]).').';

for k=n-N_s+1:n
   y_s(-n+N_s+k)=W.mean'*TF*output.data(desire_signal,k);
```

```
end

y_n=output.y([n-N_s+1:n])-y_s;

y_s_pwr=var(y_s)+mean(y_s)*mean(y_s)';
y_n_pwr=var(y_n)+mean(y_n)*mean(y_n)';

y_SNR=y_s_pwr/y_n_pwr;
```