A CONFORMAL TAGUCHI OPTIMIZED E-PATCH ANTENNA

By

C. M. GARDNER

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

ELECTRICAL ENGINEERING

2010

ABSTRACT

A CONFORMAL TAGUCHI OPTIMIZED E-PATCH ANTENNA

By

Chad Michael Gardner

The performance of a cavity-backed E-patch antenna placed conformal to a cylindrical conducting surface is explored numerically using the finite element boundary integral (FE-BI) method. From using rigorous full-wave simulations based on the finite element method, the effects of curvature are assessed for such antenna characteristics as bandwidth and impedance. The necessary changes to the patch geometry to maintain an acceptable impedance match and a desired bandwidth for rectangular cavity-backed E-patch antennas are determined. Cylindrically conformal cavity-backed E-patch antennas are simulated and perform well with many different cylinder radii. A experimental cylindrical conformal cavity-backed E-patch antenna was built to verify simulations and preformed well at the L1 and L2 frequencies. Radiation patterns of both experimental and simulated cylindrically conformal cavity-back E-patch antennas are analyzed and perform well.

Taguchi's method of optimization is used to optimize $|S_{11}|$ of rectangular cavity-backed E-patch antennas. Using Matlab and rigorous full-wave simulations based on the finite element method in combination, the effects of having different fitness functions, level differences, and input parameters in Taguchi's method of optimization are explored and determined.

To my family and friends.

ACKNOWLEDGMENTS

Dr. Edward Rothwell, Dr. Leo Kempel, Dr. Shanker Balasurbamium, Dr. Prem Chahal, Air-force Research Labs, Michael Varney, Gordon Shetler, Melissa Sisk, Jose Hejase, Raoul Ouedraogo, and the entire EM research group at Michigan State University.

PREFACE

O Ye, who in some pretty little boat,
Eager to listen, have been following
Behind my ship, that singing sails along,

Turn back to look again upon your shores;
Do not put out to sea, lest peradventure,
In losing me, you might yourselves be lost.

The sea I sail has never yet been passed;
Minerva breathes, and pilots me Apollo,
And Muses nine point out to me the Bears

Ye other few who have the neck uplifted
Betimes to th' bread of Angels upon which
One Liveth here and grows not sated by it

Well may you launch upon the deep salt-sea
Your vessel, Keeping still my wake before you
Upon the water that grows smooth again.

The divine comedy, In Pardiso Canto II Verses 1-15. Colonial Press, 1895.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 E-patch Background

E-patch antennas were introduced in [1] as a novel way to increase the bandwidth* of conventional rectangular patch antennas. Unlike some wider bandwidth patch antennas and their configurations†, the E-patch is a single uniform patch that is easily constructed. Figure 1.1 depicts the topology of the antenna. The patch dimensions are denoted by L, W, h and the antenna is normal to the patch surface fed by a coaxial probe at position $(W - X_f, Y_f)$. To increase the antenna bandwidth, two parallel slots are incorporated and positioned in vertical symmetry with respect to the feed point. Due to the patch resembling the letter "E", the name E-patch antenna is used. To maintain vertical symmetry about the top and bottom of the patch, Y_f must be $\frac{1}{2}L$. The placement of the probe at X_f can affect the properties of the antenna and is an essential parameter; however, it is not the focus of discussion for this thesis. The choice of feeding the antenna at $(W - X_f, Y_f)$, was chosen primarily to excite Mode 2 of the E-patch antenna (discussed by the authors in [1]). Mode 1 and Mode 2 excitation is indicated using the black dotted lines in Figure 1.2. The operation of

*Bandwidth is defined as the frequency span between -10 dB values of $|S_{11}|$.

†Some additional patch antennas are discussed by researchers in [4], [5] and [6].

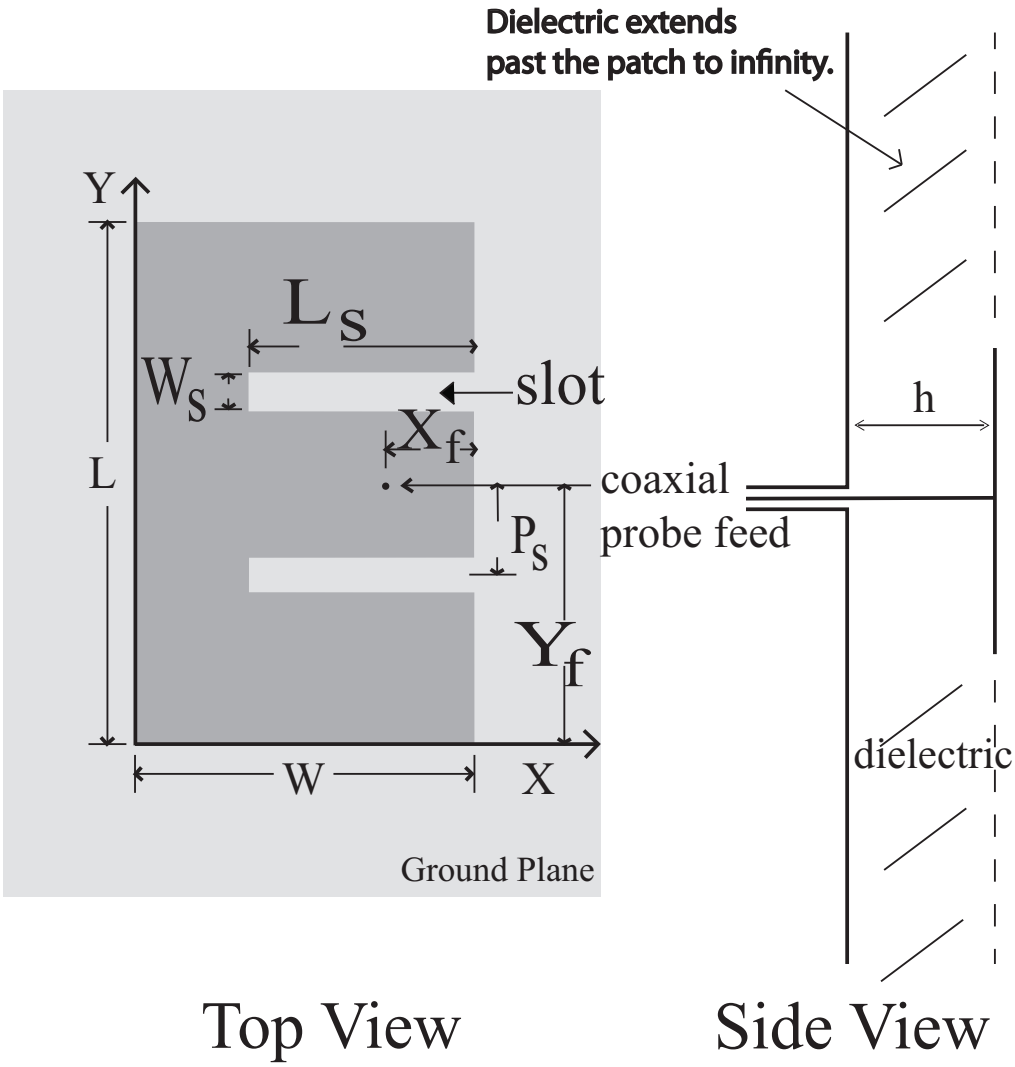

Figure 1.1: E-patch topology from [1].

the antenna using Mode 2 excitation is explained later. The slot length L_S, slot width W_S, slot placement P_S, and dielectric height h are crucial to control the bandwidth of the antenna. The slots of the E-patch antenna allow it to resonant at two frequencies, and the bandwidth primarily is determined by the separation of those two frequencies.

Figure 1.3 is from [1] and shows two parallel circuits containing capacitors and inductors. The parallel circuits of Figure 1.3 can be used to represent the transmission line resonant circuit model for E-patch antennas. The resonance frequency of a parallel LC circuit is

$$f = \frac{1}{2\pi\sqrt{LC}} \tag{1.1}$$

and can explain the existence of the two resonance frequencies. Assuming mode two excitation, currents flow from the probe to the top and bottom edges of the patch. Due to the slots, some currents travel a longer distance to reach the extreme edges of the patch. This added distance that the current travels is modeled as a added inductance ΔL_S in series with L_S. This added inductance forces the antenna to resonate at a lower frequency. Current paths which aren't detoured by the slots do not have the additional inductance ΔL_S, and thus the antenna resonates at a higher frequency because the inductance is lower in (1.1).

Controlling the inductance of the slots determined by P_S, W_S, and L_S specifies where the two resonant frequencies of the antenna will be. Refer to Figure 1.4 for a sample simulated E-patch frequency response. The dimensions for corresponding E-patch are given in [1] and are in Table 1.1. Figure 1.4 was simulated with the Method of Moments (MoM) simulator, Sonnet. The two resonance points of the antenna are clearly visible at 2.3 GHz and 2.82 GHz.

Another parameter that affects the frequency response of the E-patch antenna is the dielectric height, h. The resonance frequency for a cavity-model of a rectangular

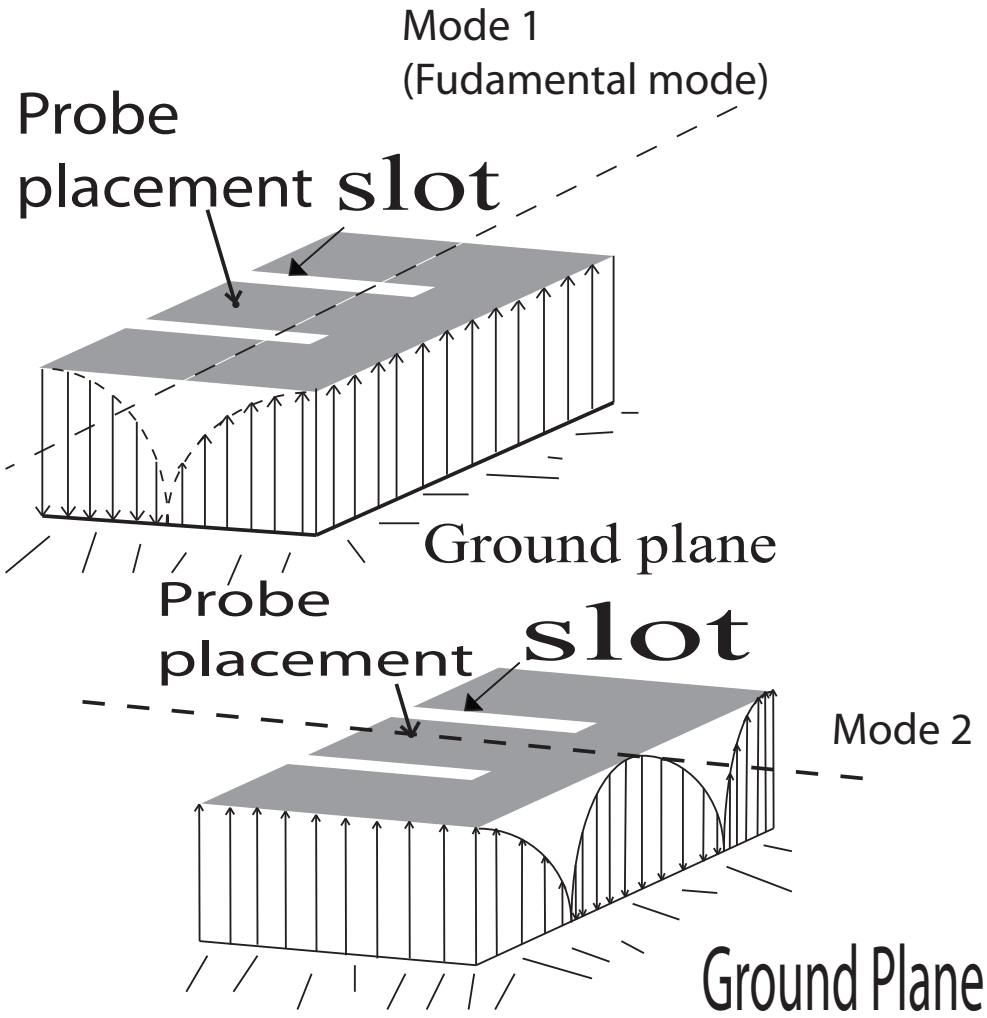

Figure 1.2: Field configurations (modes) of a E-patch antenna.

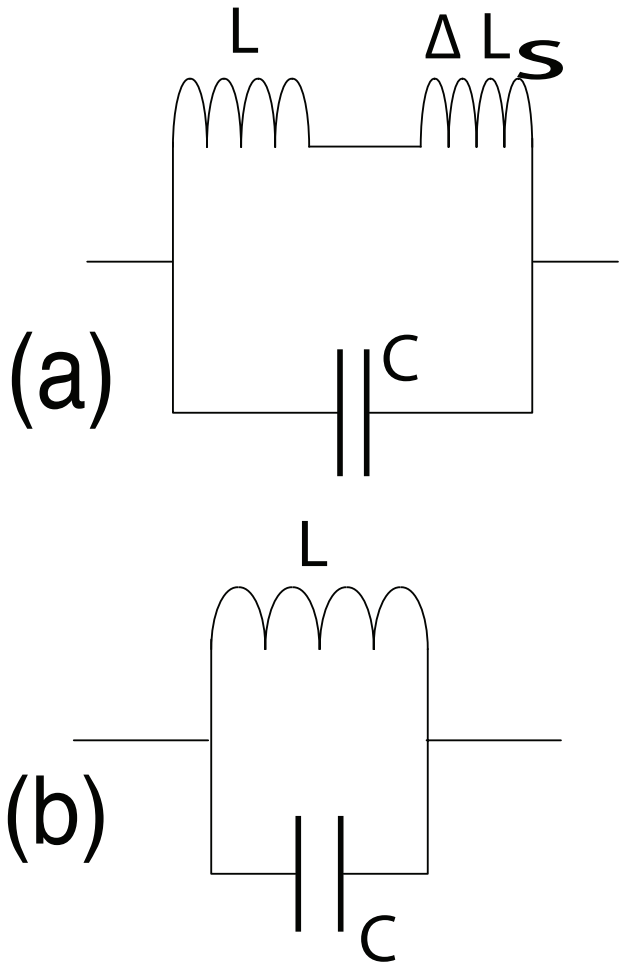

Figure 1.3: (a) An LC circuit representing the transmission line model for the lower resonance frequency of an E-patch antenna. (b) An LC circuit representing the transmission line model for the higher resonant frequency of an E-patch antenna.

patch antenna is

$$(f_r)mnp = \frac{1}{2\pi\sqrt{\mu\epsilon}}\sqrt{\left(\frac{m\pi}{h}\right)^2 + \left(\frac{n\pi}{L}\right)^2 + \left(\frac{p\pi}{W}\right)^2}. \tag{1.2}$$

The values p, n, and m are the mode numbers for the cavity in the x, y, and z directions respectively. The dielectric height of the antennas explored in this thesis have insignificant contribution to equation (1.2) frequency of resonance since it is sufficiently less then a wavelength so that the dominant mode has $m = 0$; however, the dielectric h has a significant affect on the bandwidth of the antenna. The authors in [7] explored how the bandwidth of a rectangular patch can increase up to 20% by varying the height of patch antenna lying on top of a dielectric slab. The dielectric material used for exploration in this thesis is air. Air substrates have the advantage that the dielectric height is simple to vary, and as such is a viable solution to increase the bandwidth of E-patch antennas. In addition, the bandwidth of a low-order microstrip patch mode varies as $\frac{1}{\sqrt{\varepsilon_{\text{reff}}}}$ when $\varepsilon_{\text{reff}}$ is given by (2.1). Hence, $\varepsilon_r = 1$ will yield the largest bandwidth.

Table 1.1: Dimensions for an E-patch example given in [1].

E-patch Dimension	Value in mm
L	70
W	45
X_f	7
Y_f	35
L_s	35
W_s	4
P_s	9
h	11

Other papers, such as [8], [9], [10], and [11], manipulate the E-patch topology to obtain better antenna properties such as bandwidth, radiation pattern and S-parameters. In particular, [11] gives empirical formulas for optimal (broadband op-

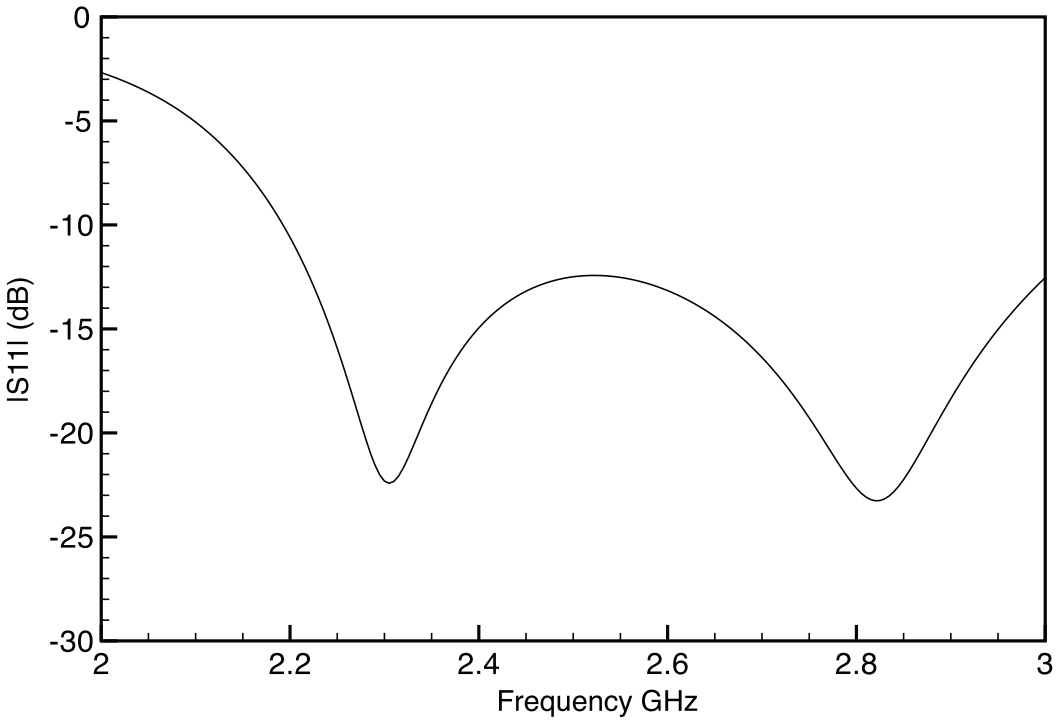

Figure 1.4: $|S_{11}|$ vs frequency plot of a E-patch antenna found using Sonnet.

eration) E-patch dimensions when a infinite ground plane is implemented. These empirical formulas are

$$L_s = .39564\lambda_0,\ W_s = .026336\lambda_0,\ P_s = .051134\lambda_0$$
$$h = .054557\lambda_0,\ X_f = .025488\lambda_0,\ Y_f = .25\lambda_0. \tag{1.3}$$

The symbol λ_0 refers to the center wavelength. The center wavelength is defined as the wavelength in-between the two resonate wavelengths of the E-patch antenna. For example, the center wavelength between 169.49 mm and 265.49 mm is 217.49 mm. In [11] the authors reported that equations (1.3) yielded approximately a 40% increase in bandwidth compared to a rectangular patch, an improvement of about 11% over that given in [1]. For many E-patch antenna designs the equations (1.3) provide a simple method for improving the frequency response of the antenna.

In [12] a Particle Swarm Optimization (PSO) was used to further increase the

bandwidth of the E-patch from 40% to 55%. In [9] the authors used capacitive input probes to improve the input impedance of the antenna. The researchers in [13] implemented a Genetic Algorithm (GA) to optimize the radiation pattern of the E-patch antenna. Reference [10] documents methods to reduce the size of the E-patch while maintaing working functionality at the two frequencies of interest. None of the authors in [1], [4], [5], [11], [14], [12], [9], [13], and [10] explored how the antenna would behave on a cylindrical curved surface, nor did they address how a cavity-backed E-patch antenna would function. Airborne applications could benefit from exploring the functionality and feasibility of a cavity-backed and cylindrically-conformal E-patch. A goal of this thesis is to present simulated and experimental data of the antenna properties of a cavity-backed and cylindrically-conformal E-patch antenna.

1.2 Background on Taguchi's Method of Optimization

Taguchi's Method of optimization was developed by Dr. Genchi Taguchi as a way of using statistics to design and improve quality in manufactured goods [15]. It is a fractional factorial approach to optimization. Instead of exhausting all possible combinations of parameters, a smaller number of the parameter combinations is used to sample the entire exhaustive set. This fraction of possibilities achieves a comparable outcome to the full factorial approach. In order to use Taguchi's Method the concept of Orthogonal Arrays (OAs) needs to be understood. OAs were introduced in the 1940s in a series of seminal papers [16], [17], and [18]. OAs provide a convenient and orderly way to utilize the fractional factorial approach to optimization. An Orthogonal Array is defined in Definition 1 [19].

Definition 1. *Let* S *be a set of* s *symbols or levels. A matrix* A*, commonly called an array, of* N *rows and* k *columns with entries from* S *is said to be an orthogonal array*

with s *levels and strength* t (0≤t≤k) *if in every* N x t *subarray of* A, *each* t-*tuple based on* S *appears exactly the same number of times as any other* t-*tuple.*

OAs are typically represented by the notation $OA(N, k, s, t)$. In Table 1.2 is an example of an OA with nine rows N, four columns k, three levels s, and a strength t of two. Each parameter in this OA is selected from three numbers, e.g. s = (0,1,2). Thus Table 1.2 is a 3-level OA. For example, If one forms a sub-array of any two (strength $t = 2$) columns, one can see 9 possible combinations of rows: (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2). Each of these nine combinations appears exactly the same times as in a row. If Table 1.2 were used in a optimization problem, the four columns would represent four parameters that need optimization. Each level number (0,1,2) corresponds to a variation in one of the parameters that is being optimized. Every row corresponds to a combination of varying level values that affect optimization. For example, the second row indicates that columns two and three take on the value of level two; column one takes on the value of level one, and column four takes on the value of level three. If one were to count all the possible combinations of

Table 1.2: An *Orthogonal Array OA(9,4,3,2)*

0	0	0	0
0	1	1	2
0	2	2	1
1	0	1	1
1	1	2	0
1	2	0	2
2	0	2	2
2	1	0	1
2	2	1	0

a problem with four parameters that had three different possibilities each, it would be 3^4 or 81 different combinations. Using Taguchi's method and OAs, only nine different experiments are carried out. Due to this being a fraction of the total combinations, Taguchi's Method has much to offer in terms of reducing the time and cost required

to run simulations and obtain results. Chapter 3 discusses Taguchi's method and use of OAs in more detail.

In [20] the authors discuss application of Taguchi's Method to electromagenetics (EM). The results and methods of [20] are expanded on in [3] and [21]. The researchers in [20] and [21] address how to use Taguchi's method for optimizing radiation patterns, filter designs, and antenna factors in various EM problems. They do not discuss explicitly the application to patch antennas. However, the researchers do provide ample groundwork to create a Taguchi based optimizer for optimizing $|S_{11}|$ of an antenna. Another goal of this thesis is to investigate Taguchi's Method and apply it to optimizing the E-patch antenna in both cavity-backed and cylindrically-conformal implementations.

Chapter 2

Conformal E-patch Antenna

2.1 Infinite ground plane simulations

As discussed in Chapter 1, other authors have looked into improving the E-patch antenna parameters by using a variety of different methods. Most of these researchers used infinite ground plane simulations for the E-patch. In order to better understand how the E-patch antenna can be simulated, it would prove useful to try to replicate some results in [1] and use the formulas in [11] to create a E-patch.

The MoM simulator Sonnet was used as a comparison simulation tool for the simulations appearing in [1]. It is important to note that Sonnet uses a infinite grounded slab for simulating antenna properties. Refer to Appendix A for step-by-step directions of how to use Sonnet for E-patch antenna simulations. One will note that the cell size is 1 mm by 1 mm in the Sonnet setup for this work. When Sonnet incorporates E-patch antenna parameters, the dimensions of the metallic patches are thus rounded to the nearest mm. Figure 2.1 shows the Sonnet simulation of the second E-patch in [1]. The dimensions for the antenna used to generate Figure 2.1 are given in Table 2.1. In [1] the authors found that an antenna constructed using the values in Table 2.1 resonated in their simulation at 2.12 GHz and 2.66 GHz. It is

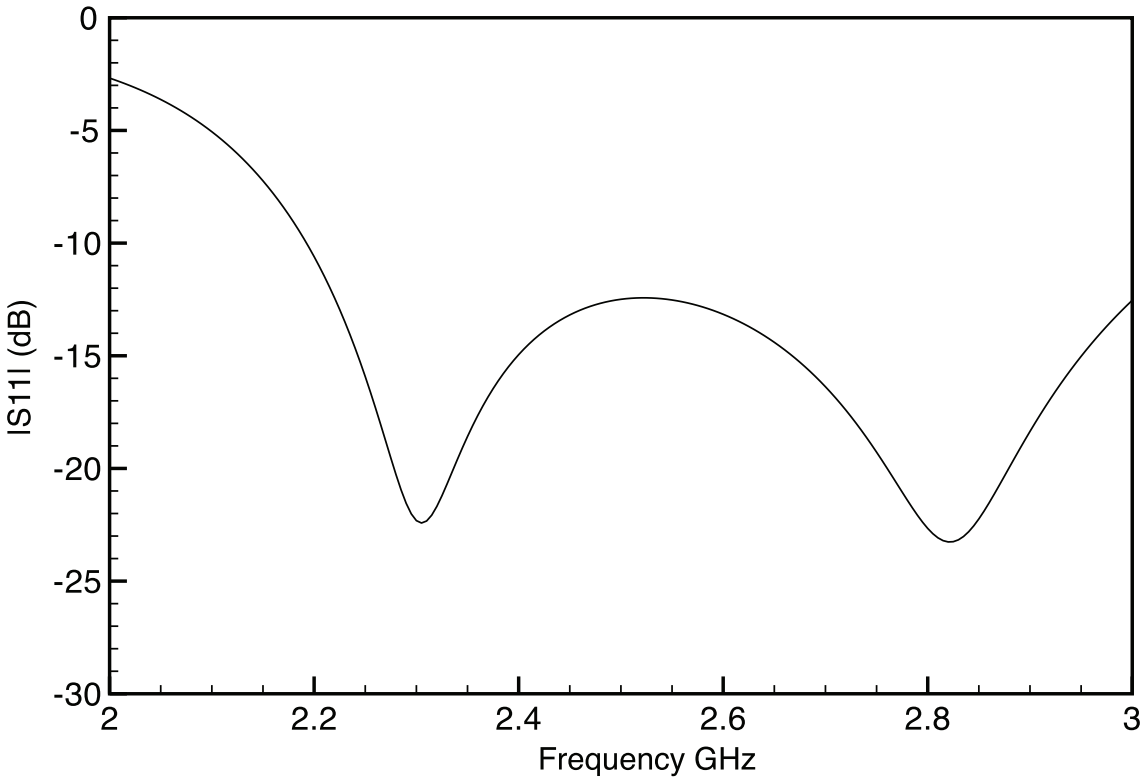

Figure 2.1: Sonnet simulation frequency response of the second E-patch in [1] ($L = 70$ mm, $W = 45$ mm). Dimensions provided in Table 2.1.

clear in Figure 2.1 that the results are slightly different than in [1]. In Figure 2.1 the resonant points are 2.30 GHz and 2.81 GHz; this is a difference of a bit less than 200 MHz from the results in [1]. Figure 2.2 shows the Sonnet simulation of the third E-patch in [1]. The dimensions for the antenna used to generate Figure 2.2 are in Table 2.2. In [1] the authors found that the antenna constructed using Table 2.2 values resonated in their simulation at 1.8 GHz and 2.3 GHz. Figure 2.2 shows the Sonnet simulated resonant points are approximately 2 GHz and 2.6 GHz, a difference again of about 200-300 MHz. With a 200 MHz shift in frequency noted, one can extrapolate how an E-patch antenna designed in Sonnet would performed compared to E-patches simulated in [1]. The 200 MHz shift in frequency maybe due to a different resolution cell size between Sonnet and the MoM simulator in [1].

The L1 (1.57542 GHz) and L2 (1.2276 GHz) frequencies are used to communicate

Table 2.1: Geometry values for the second E-patch example given in [1].

E-patch Dimension	Value in mm
L	70
W	45
X_f	7
Y_f	35
L_s	35
W_s	4
P_s	9
h	11

Global Positioning System (GPS) signals. This thesis investigates the performance of the E-patch antenna designed to operate at the frequencies at and between L1 and L2. The authors in [1] primarily investigated the performance of an E-patch in the S-band range of frequencies. An L-band E-patch antenna is designed starting from [1] with the antenna dimensions modified so that the antenna resonates at L1 and L2. Refer to Table 2.2 and Table 2.3. Table 2.3 values were generated by scaling according to the first resonant frequency taking Table 2.2 values and multiplying them by the ratio $\frac{1.8}{1.227} \approx 1.4$. Since given the same substrate and electrical thickness, the resonant frequencies are primarily determined by the relevant geometric dimensions. Figure 2.3 shows a Sonnet simulation of a designed L-band E-patch using the values of Table 2.3.

It is clear that $|S_{11}|$ in Figure 2.3 is not the desired -10 dB or less at the L1 and L2 frequencies. Instead of manipulating the dimensions of the antenna by trial and error, one could use the empirical equations (1.3) in hopes of improving $|S_{11}|$. If one uses 214 mm (1.4 GHz frequency) as λ_O, the E-patch dimensions become those in Table 2.4. The L and W values were determined using [22]

$$\varepsilon_{\text{reff}} = \frac{\varepsilon_r + 1}{2} + \frac{\varepsilon_r - 1}{2}\left[1 + 12\frac{h}{W}\right]^{-1/2} \tag{2.1}$$

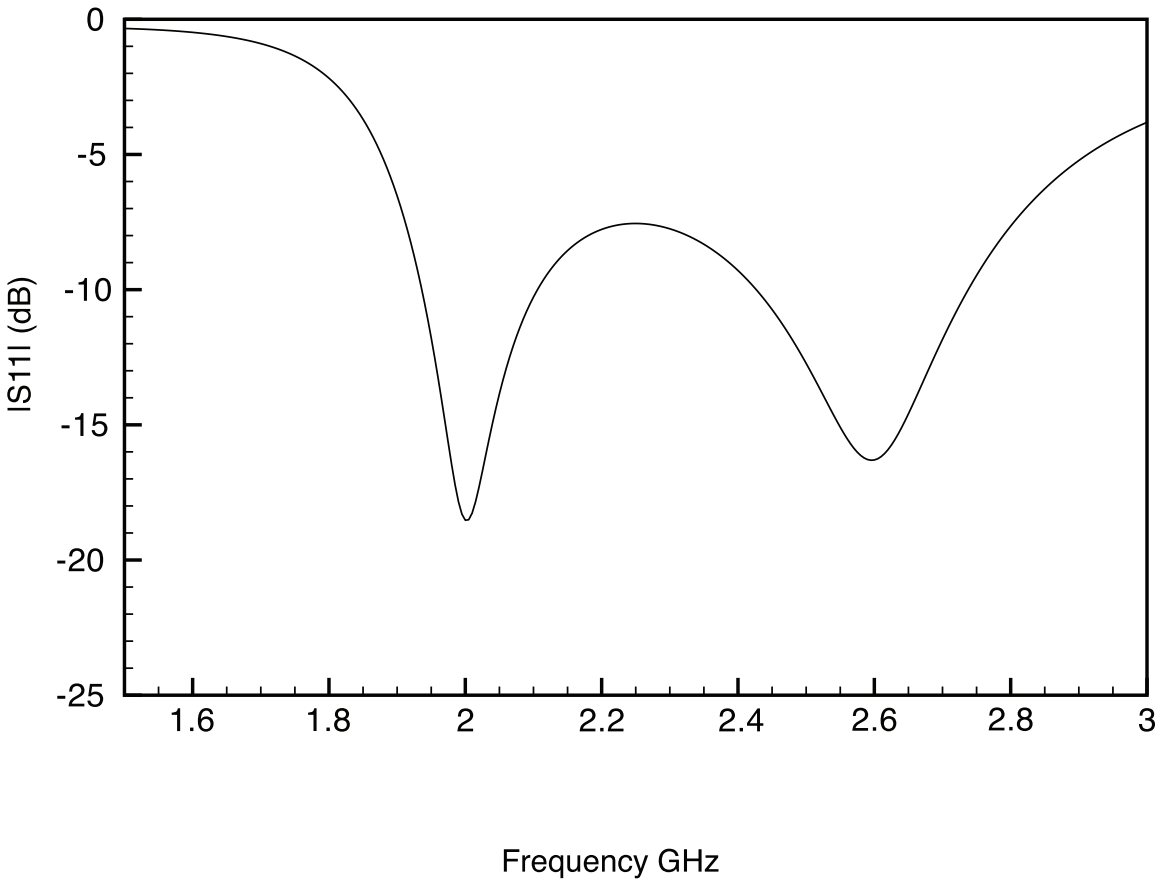

Figure 2.2: Sonnet simulation frequency response of the third E-patch in [1] ($L = 70$ mm, $W = 50$ mm). Dimensions provided in Table 2.2.

Table 2.2: Geometry values for the third E-patch example given in [1].

E-patch Dimension	Value in mm
L	70
W	50
X_f	6
Y_f	35
L_s	40
W_s	6
P_s	10
h	15

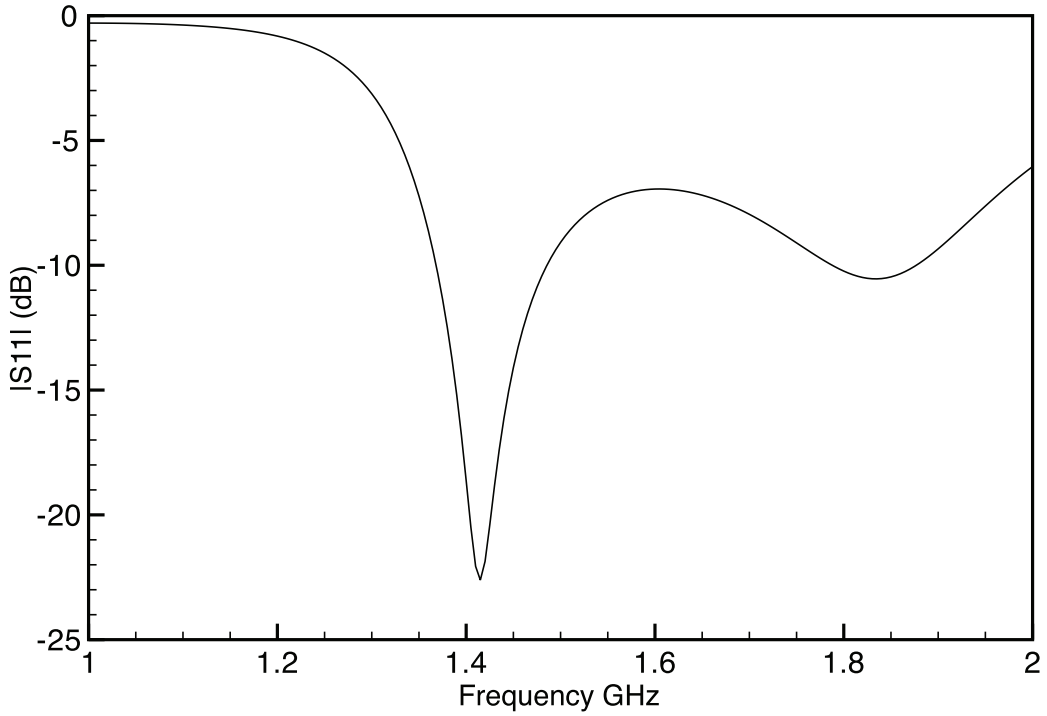

Figure 2.3: Sonnet simulation frequency response of a first designed L-band E-patch ($L = 98$ mm, $W = 75$ mm). Dimensions provided in Table 2.3.

Table 2.3: Geometry values for the 40% enlarged E-patch dimensions.

E-patch Dimension	Value in mm
L	98
W	75
X_f	8.4
Y_f	49
L_s	56
W_s	8.4
P_s	14
h	21

$$\Delta L = 0.412h\frac{\left(\varepsilon_{\text{reff}} + .3\right)\left(\frac{W}{h} + 0.264\right)}{\left(\varepsilon_{\text{reff}} - 0.258\right)\left(\frac{W}{h} + .8\right)} \tag{2.2}$$

$$W = \frac{\nu_o}{2f_r}\sqrt{\frac{2}{\varepsilon_r + 1}}$$

$$L = \frac{1}{2f_r\sqrt{\varepsilon_{\text{reff}}}\sqrt{\mu_o\epsilon_o}} - 2\Delta L \tag{2.3}$$

where ϵ_r is the relative permittivity of the dielectric, μ_o is the permeability of free space, ϵ_o is permittivity of free space, h is the height of the dielectric, f_r is the center frequency between the resonance points of the antenna, and ν_o is the speed of light in a vacuum. Equations (2.1) - (2.3) are the typical equations used to design a patch antenna with a microstrip line feed. Figure 2.4 shows the Sonnet simulation of the E-patch designed using equations (1.3) and (2.1) - (2.3). The dimensions for the antenna used to generate Figure 2.4 are given in Table 2.4*. Figure 2.5 shows the Sonnet simulation of the E-patch designed using equations 1.3 and the online calculator found at [2]. The dimensions of the antenna used to generate Figure 2.5 are given in Table 2.5. The L and W values in Table 2.5 were chosen from [2]. The probe placement for E-patch antennas is typically near the edge of the patch. To provide a better input impedance match, the online calculator in [2] can estimate the radiation edge impedance of a rectangular patch antenna. It was approximated that a 50 Ω radiation edge impedance would yield a better input impedance match to a E-patch antenna (assuming 50 Ω input impedance).

*In Sonnet the values from Table 2.4 were rounded to the nearest mm except h, which was rounded down to the nearest mm.

Table 2.4: Geometry values for E-patch dimensions generated from equations (1.3) and (2.1) - (2.3).

E-patch Dimension	Value in mm
L	107.14
W	91.1731
X_f	5.4617
Y_f	53.57
L_s	84.7800
W_s	5.6434
P_s	10.9573
h	11.6908

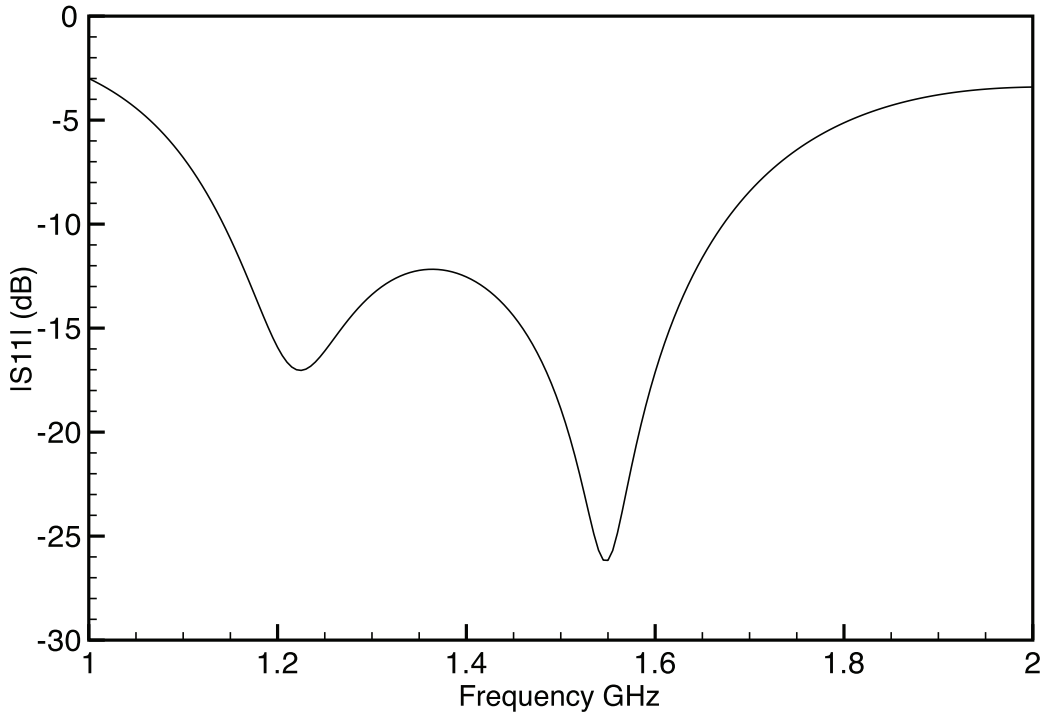

Figure 2.4: Sonnet simulation frequency response of the second designed L-band E-patch ($L = 107.14$ mm, $W = 91.1731$ mm). Dimensions provided in Table 2.4.

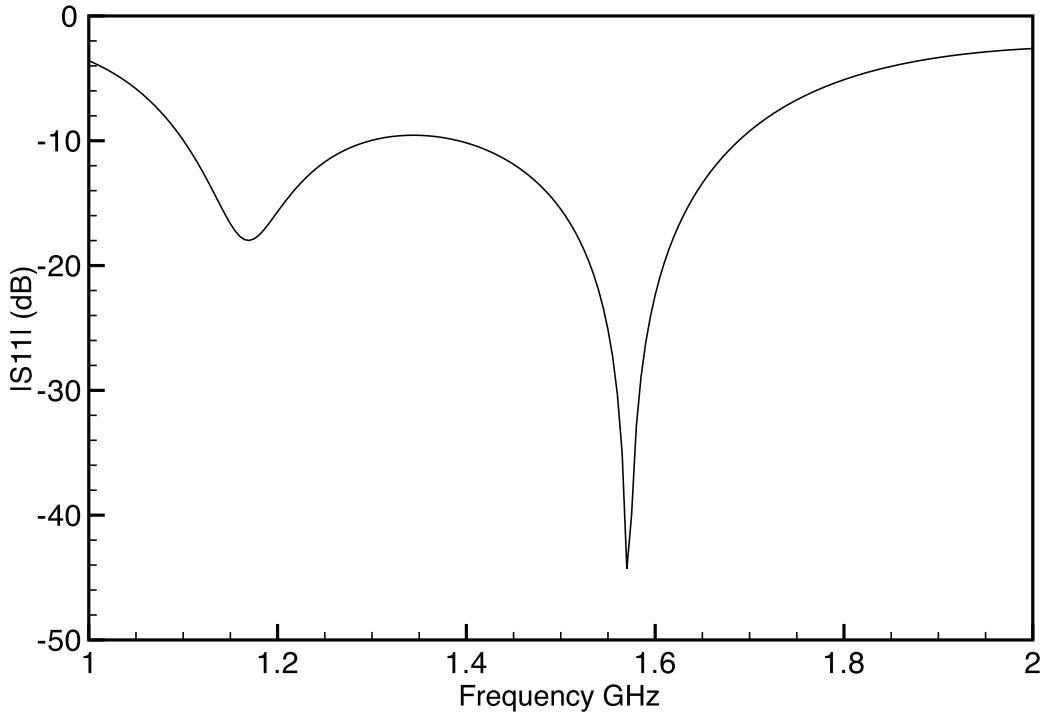

Figure 2.5: Sonnet simulation frequency response of the third designed L-band E-patch ($L = 142$ mm, $W = 90$ mm). Dimensions provided in Table 2.5.

Table 2.5: Geometry values for E-patch dimensions generated from equations (1.3) and [2].

E-patch Dimension	Value in mm
L	142
W	90
X_f	5
Y_f	71
L_s	85
W_s	6
P_s	11
h	11

2.2 K-brick simulations

K-brick is a finite-element boundary-integral (FE-BI) program developed by Professor Leo Kempel at Michigan State University. K-brick simulates antenna characteristics (radiation pattern, $|S_{11}|$, TM modes, TE modes, etc.) for rectangular-cavity-backed patch antennas. K-brick serves (in this thesis) as a comparison program to HFSS, a general purpose finite element method simulation program (discussed in section 2.3). K-brick is a compiled Fortran program run within a Unix operating system. Geometry and operating parameters are specified using an input text file (.txt) which must be edited each time the program is executed. Appendix D contains a sample input text file for K-brick and a start-up guide. Figure 2.6 is a depiction of a simulated rectangular cavity-backed E-patch antenna in K-brick. Dimensions Cx, and Cy are the length and width of the cavity and h is the depth of the cavity. Figure 2.7 shows the top view of the cavity backed E-patch antenna with associated dimension labels[†]. As discussed in [23], the procedure of formulating the FE-BI begins with a weak form of the vector wave equation. Quoting from [23],

> "The FE-BI formulation begins with the weak form of the vector wave equation followed by specification of appropriate vector shape functions and dyadic Greens function. The resulting FE-BI equations are then used to solve for the total electric fields within the cavity and on the aperture."

In [23], the authors discussed only circular cavity-backed patch antennas. K-brick was written after the publication; however, it is functionally similar to programs developed by Prof Jian-Ming Jin and others. K-brick adheres to the same principles used and discussed in [23], namely the formulation of the FE-BI equations in rectangular coordinates.

[†]It should be noted that K-brick uses the center of the cavity when referencing the probe placement. In other words, the probe is placed irrespective of the cell size of the cavity. This is discussed in Appendix D.

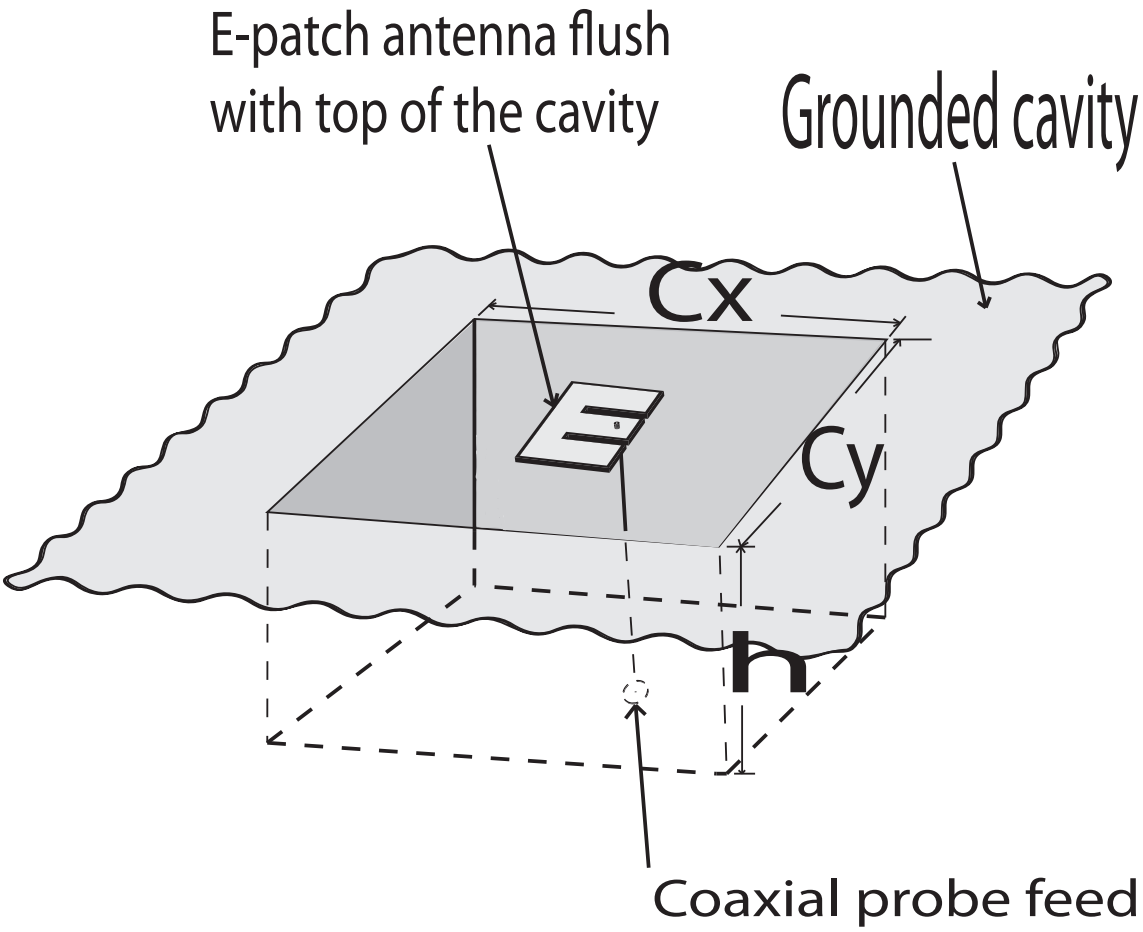

Figure 2.6: A typical rectangular cavity-backed E-patch antenna analyzed in K-brick. Depicted cavity dimensions are not to scale and typically $h \ll C_x$, $h \ll C_y$.

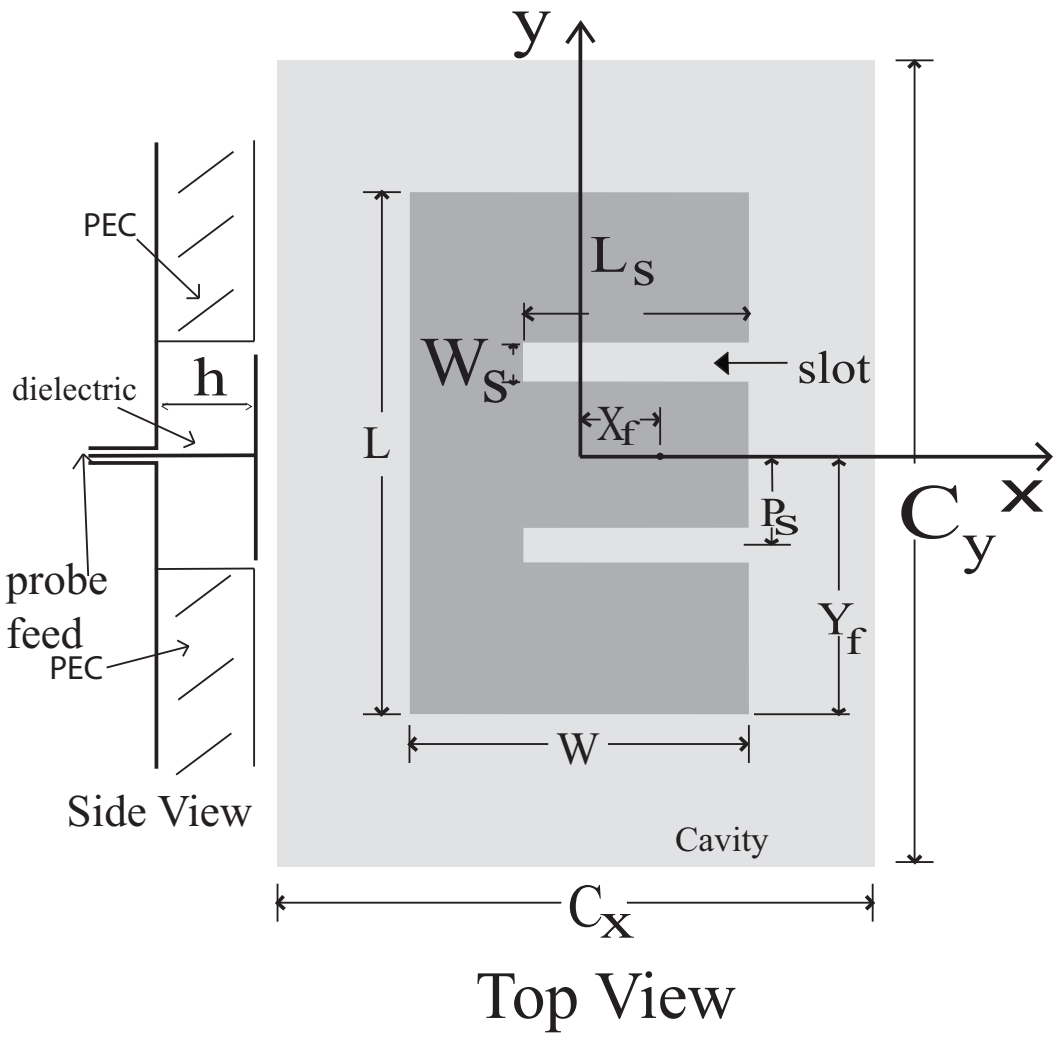

Figure 2.7: Top and side view of an E-patch in a cavity.

As a starting example for simulating an E-patch antenna in a cavity, a simple rectangular patch antenna is simulated in K-brick. Figure 2.8 shows the top view topology of a rectangular patch with associated dimension labels backed by a rectangular. Table 2.6 contains the geometry values for the simulated rectangular cavity-backed patch. Figure 2.9 depicts the K-brick simulated frequency response of the cavity-backed rectangular patch antenna in a cavity. The response of the antenna show in Figure 2.9 is what is excepted of a single rectangular cavity-backed patch antenna.

In section 2.1 a simulated E-patch with the lowest $|S_{11}|$ at the L1 and L2 frequencies was found in a Sonnet Simulation (no Taguchi method was used); refer back to Figure 2.5 and Table 2.5. Figure 2.10 shows the frequency response of a antenna with Table 2.7 values simulated in K-brick. Comparing Figure 2.10 and Figure 2.5 the simulated frequency response was not the same. Figure 2.10 shows that the simulated $|S_{11}|$ is not -10 dB and below for L1 or L2 frequencies. The large difference in the $|S_{11}|$ of the Sonnet simulation and K-brick simulation is due to the presence of the cavity in the K-brick simulation.

Table 2.6: Dimensions of a cavity-backed rectangular patch antenna. Values were generated from a Taguchi optimization.

Dimension	Value in mm
L	60
W	89
X_f	2.1
Y_f	30
C_y	200
C_x	200
H	8.6

In the pursuit of a value of $|S_{11}|$ of -10 dB or below at the L1 and L2 frequencies, a Matlab graphical user interface (GUI) was written to allow the user to specify a range of E-patch dimensions to vary. Matlab generates an input file (K-brick .txt

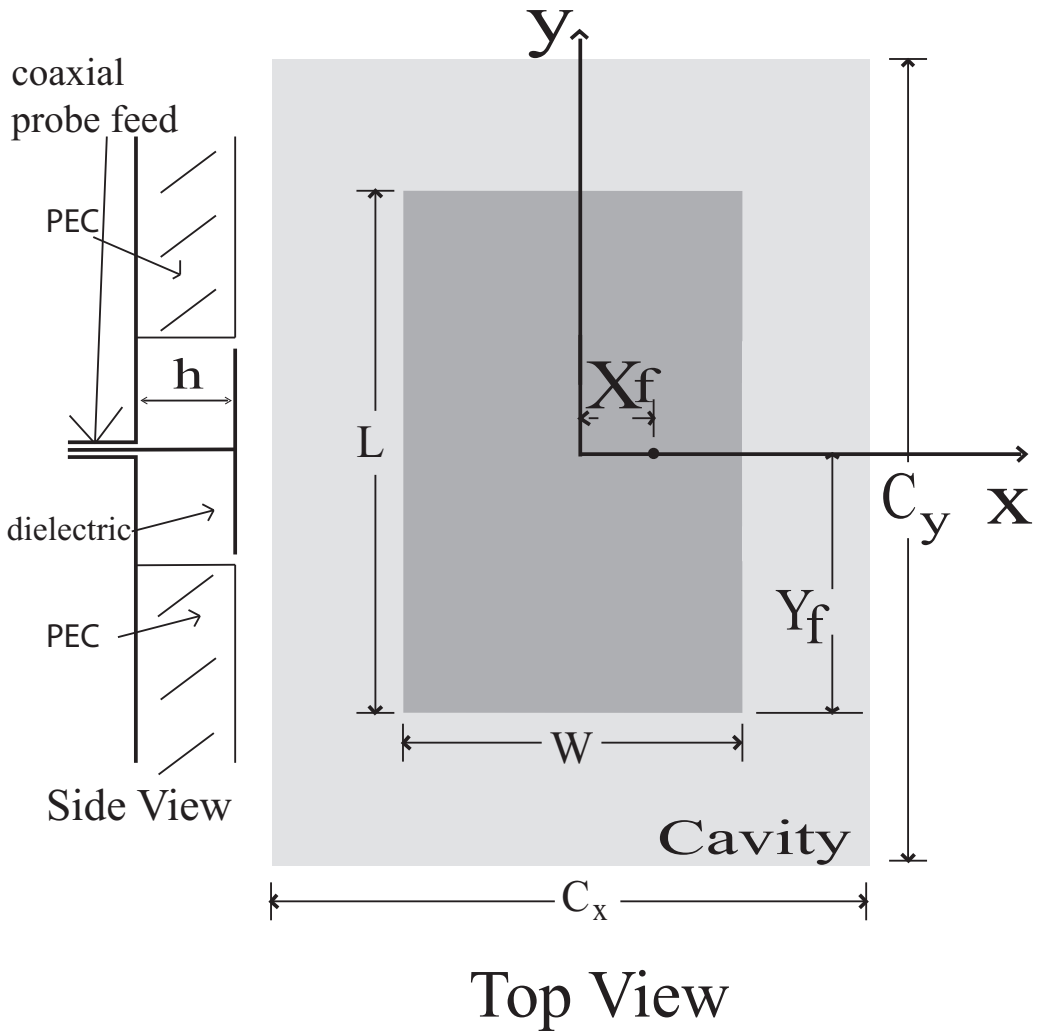

Figure 2.8: Top and side view of a rectangular patch backed by a rectangular cavity.

Table 2.7: Dimensions of a cavity-backed E-patch antenna.

Dimension	Value in mm
L	142
W	90
X_f	5
Y_f	71
L_s	85
W_s	6
P_s	11
h	11
C_x	200
C_y	200

file) for each set of E-patch dimensions, executes K-brick, and places the results in seven variables h, L_S, W_S, L, X_f, W, and P_S; Y_f is always placed along the vertical symmetry plane of the antenna. At the end of all of the programs run, the desired output parameters (typically return loss) are written to an output text file. Figure 2.11 shows a screen capture of the GUI. Any or all of these variables may be varied by specifying starting and ending values, and a step size. K-brick is executed for all combinations of the parameters that are varied. Appendix F has the Matlab code associated with this program. After a series of parameter runs, a simulated E-patch antenna frequency response was found that has a -10 dB for $|S_{11}|$ at the L1 and L2 frequencies. The simulated frequency response of that E-patch is shown in Figure 2.12 and the antenna dimensions are provided in Table 2.8. However, this result is still not optimal, as the $|S_{11}|$ between the frequencies 1.227 and 1.575 raises above -10 dB. In section 3.3 a Taguchi based optimizer is used in conjunction with K-brick to further improve the simulated frequency response of the cavity-backed E-patch antenna.

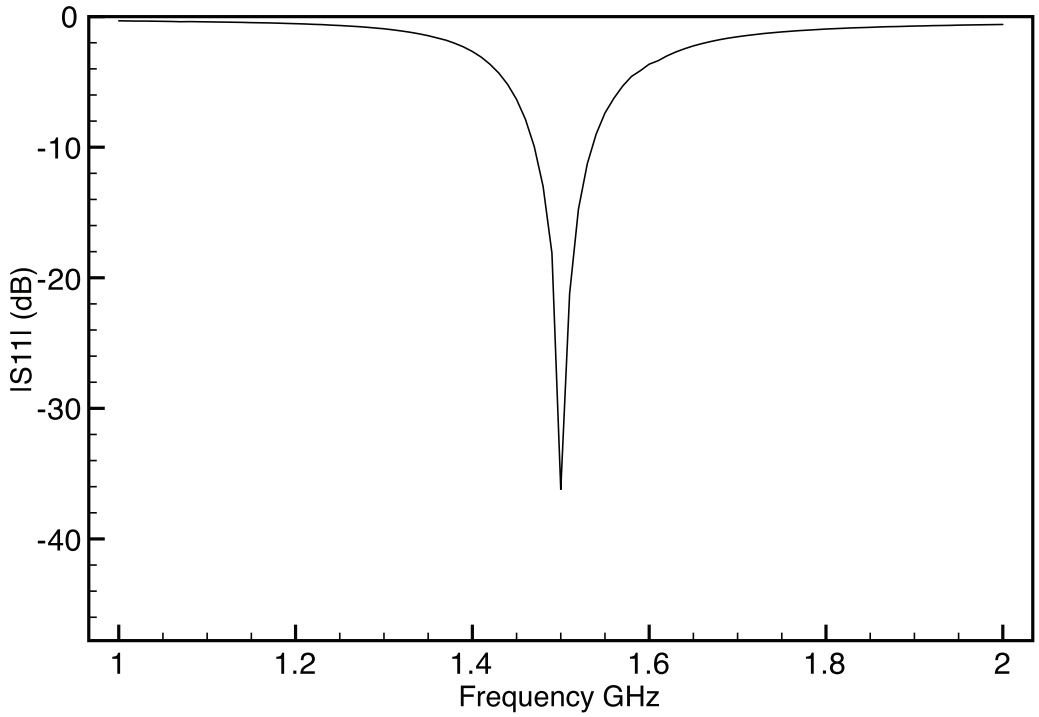

Figure 2.9: K-brick simulation frequency response of the cavity-backed rectangular patch. Table 2.6 contains the dimension values.

Table 2.8: Dimensions of a cavity-backed E-patch antenna optimized using the GUI Matlab K-brick interface.

Dimension	Value in mm
L	101
W	88
X_f	6
Y_f	50.5
L_s	75
W_s	5
P_s	9
h	13
C_x	200
C_y	200

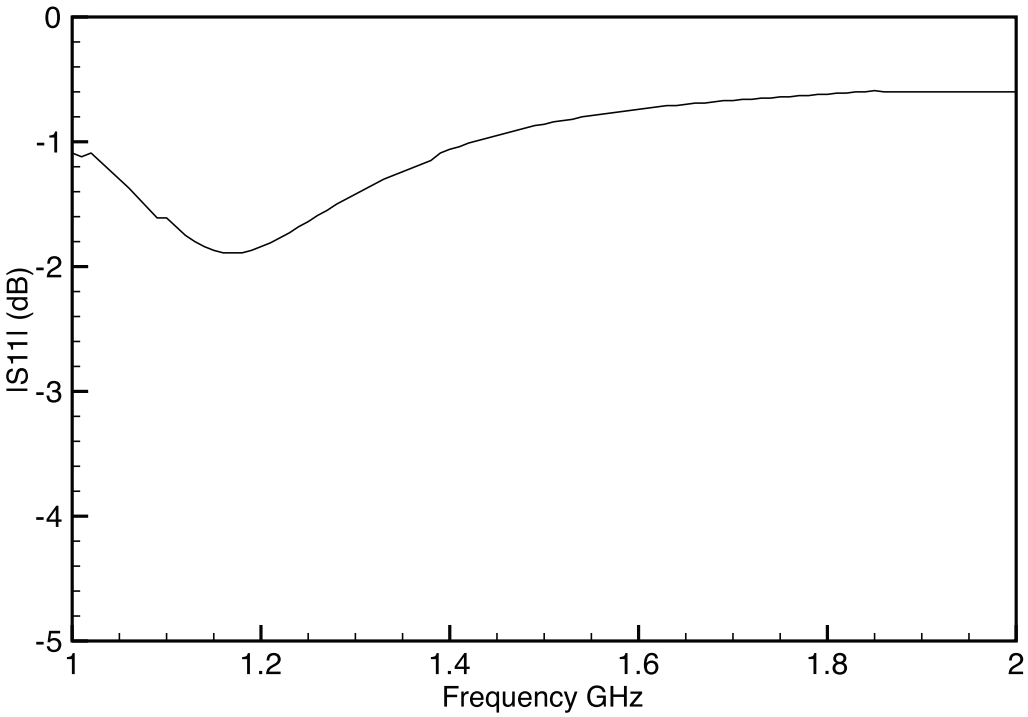

Figure 2.10: K-brick simulation frequency response of the E-patch antenna with geometry values given in Table 2.7.

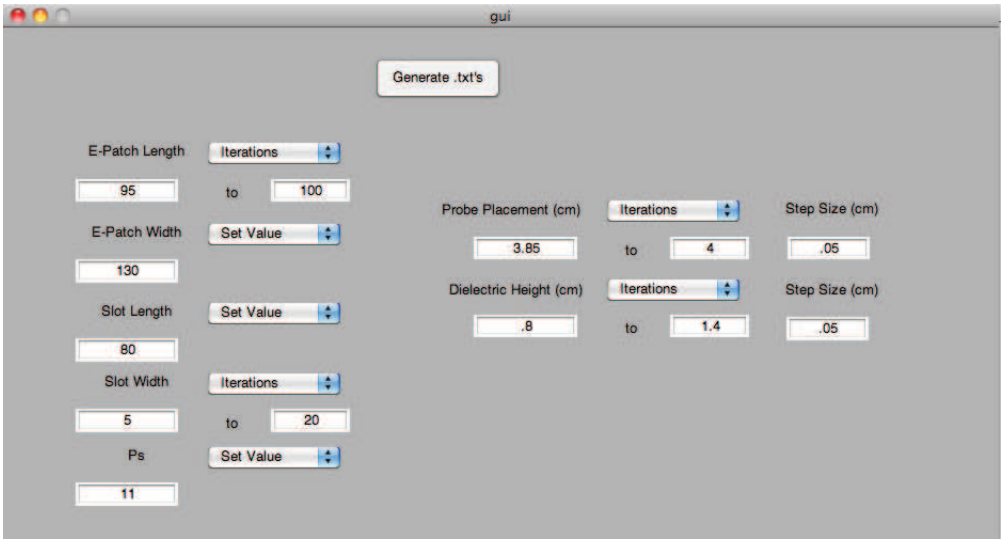

Figure 2.11: Matlab GUI interface for varying the geometry of an E-patch antenna. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.

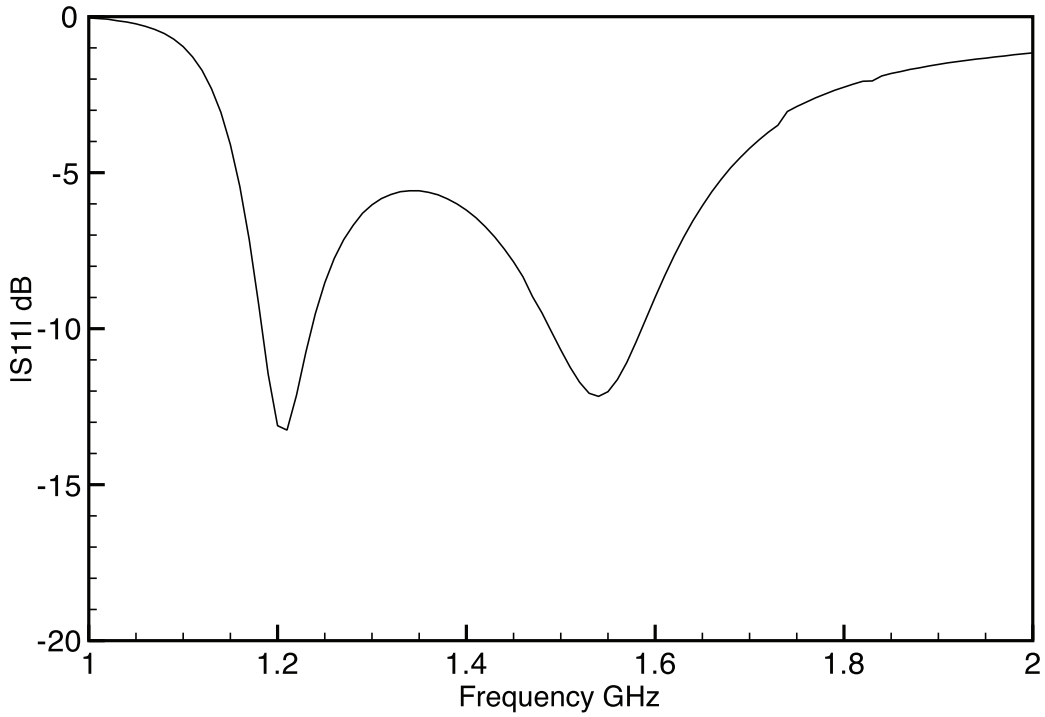

Figure 2.12: K-brick simulation frequency response of an E-patch antenna with geometry values given in Table 2.8.

2.3 HFSS flat bottom cavity simulations

To begin the study of conforming the E-patch antenna to a cylinder, it would prove useful to approximate how the curvature varies according to different cylinder radii. Consider Figure 2.13, which shows a simple top down view of a cylinder with a triangle inscribed inside of the circle. From Figure 2.13 the deflection d can be estimated. Using Pythagoras' theorem,

$$a^2 + (\frac{w}{2})^2 = (a+d)^2. \tag{2.4}$$

Then expanding out the squared terms gives

$$a^2 + \frac{w^2}{4} = a^2 + 2ad + d^2. \tag{2.5}$$

Collecting terms and putting them on the left side of the equal sign gives

$$d^2 + 2ad - \frac{w^2}{4} = 0. \tag{2.6}$$

Using the quadratic formula then gives

$$d = -a + \frac{\sqrt{4a^2 + w^2}}{2}. \tag{2.7}$$

Simplifying and bringing an a outside the square root produces

$$d = -a + a\sqrt{1 + \frac{w^2}{4a^2}}. \tag{2.8}$$

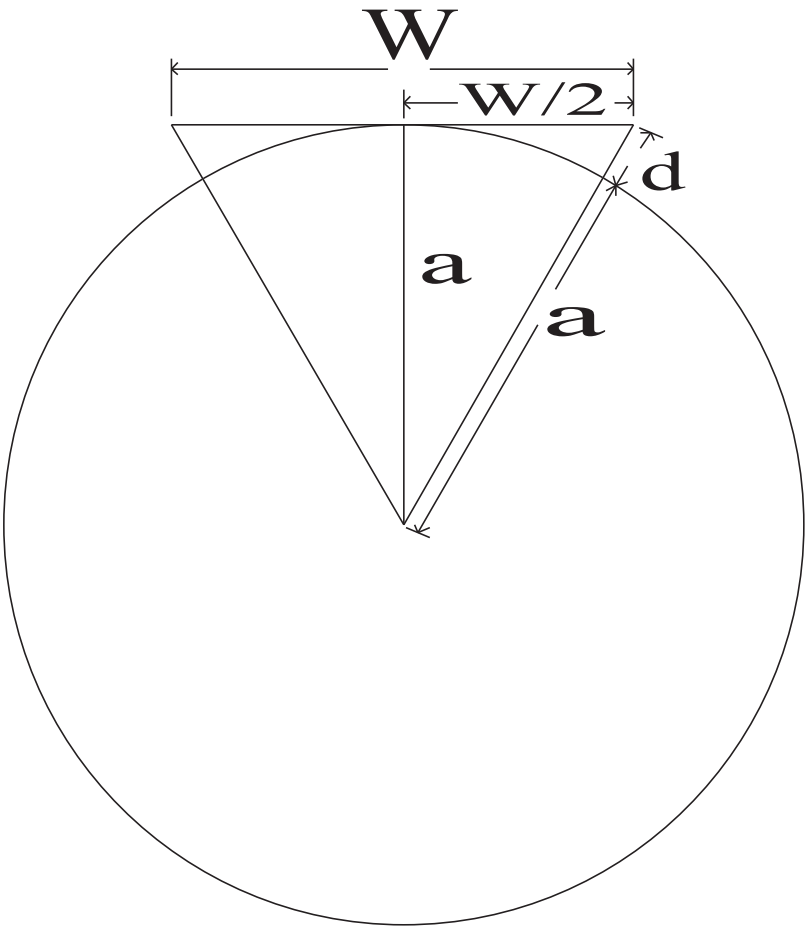

Figure 2.13: Curvature diagram.

Now, assuming that $\left(\frac{w}{2a}\right)^2 \ll 1$ and using the first two terms in the binomial expansion of $\sqrt{1+x} \approx 1 + \frac{x}{2}$, (2.8) becomes,

$$d \approx -a + a\left(1 + \frac{1}{2}\left(\frac{w}{2a}\right)^2\right). \tag{2.9}$$

Simplifying (2.9) then gives

$$d \approx \frac{1}{8}\frac{w^2}{a}. \tag{2.10}$$

Table 2.9 assumes that the cavity dimensions (C_x, C_y, and h) do not change and the radius a is changing. From Table 2.9 one can note how the curvature of a rectangular cavity-backed antenna depends on cylinder radius for a flat bottom cavity.

Table 2.9: Curvature deflection from using (2.10).

Radius of Cylinder a in cm	Deflection d in cm
2000	.025
200	.25
100	.5
70	.714
50	1
25	2

High Frequency Structure Simulator (HFSS) was used to simulate the cylindrically conformal cavity-backed E-patch antenna. HFSS provides E and H-fields, currents, S-parameters and near and far radiated field simulation results. Table 2.10 contains the settings used in the HFSS simulations. Using the standard cylindrical coordinate system where $\rho = \sqrt{x^2 + y^2}$, $\phi = tan^{-1}\frac{y}{x}$ and $z = z$ (x, y, and z are variables in the rectangular coordinate system), one can develop a topology for the cylindrically conformal cavity-backed E-patch antenna. Figure 2.14 shows a conceptual depiction of a cavity-backed E-patch conformed to a cylinder. Figure 2.15 is a screen capture of an cavity-backed E-patch conformed to a cylinder's surface in HFSS. In order to analyze how the curvature of the cylinder changes the antenna's performance, as compared to the planar cavity-backed case, the E-patch dimensions need to stay consistent between both cases. Therefore, the width dimensions of the E-patch (W and L_S) will be specified using ϕ, at a distance ρ from the z axis. Further, the arc length along the circumference of the cylinder is

$$g = \rho\phi. \tag{2.11}$$

The variable g in (2.11) is the E-patch width dimension W or L_S. The length dimensions of the E-patch (L, W_S, and P_S) will be along the z axis. To keep the cavity in the same configuration as the rectangular cavity-backed case, the dielectric h will

be specified along ρ, C_x will be specified by a ϕ and C_y will be specified along the z. The probe feed (X_f, Y_f) will be placed consistent with that of the E-patch. In other words X_f will be an arc length adhering to (2.11) where $g = X_f$ and Y_f will be specified along the z axis. In all HFSS simulations, the height of the cylinder is fixed at 100 cm and the cavity dimensions, C_x and C_y, are both 200 mm.

Figure 2.16 shows the simulated frequency response of the cylindrically conformal (100 cm radius) cavity-backed E-patch antenna. The E-patch antenna in Figure 2.16 has the same dimension values as the rectangular cavity case in Table 3.12. At a radius of 100 cm, d in (2.10) is 0.5 cm. Figure 2.17 shows the simulated frequency response of the cylindrically conformal (70 cm radius) cavity-backed E-patch antenna. The E-patch antenna in Figure 2.17 has the same dimension values as the rectangular cavity case in Table 3.12. At a radius of 70 cm, d in (2.10) is .714 cm. In Figure 2.18 the simulated frequency response of the cylindrically conformal (50 cm radius) cavity-backed E-patch antenna is shown. The E-patch antenna in Figure 2.18 has the same dimension values as the rectangular cavity case in Table 3.12. At a radius of 50 cm, d in (2.10) is 1 cm.

Table 2.10: HFSS settings for the cylindrically conformal cavity-backed E-patch antenna.

HFSS setting	HFSS setting value(s)
Frequency of solving	1.6 GHz
Adaptive passes	7
Maximum Delta S	.01
Sweep Type	Fast
Frequency Set up	linear step 1 to 2 GHz step size .02
Initial mesh options	do lambda refinement .3
Solution Options	First order
Excitation	Waveport at the coaxial probe feed
Port Renormalization	50 Ω
Deembed setting	checked -35 mm

Figure 2.19 contains all frequency response data in Figures 2.16-2.18 and Figure

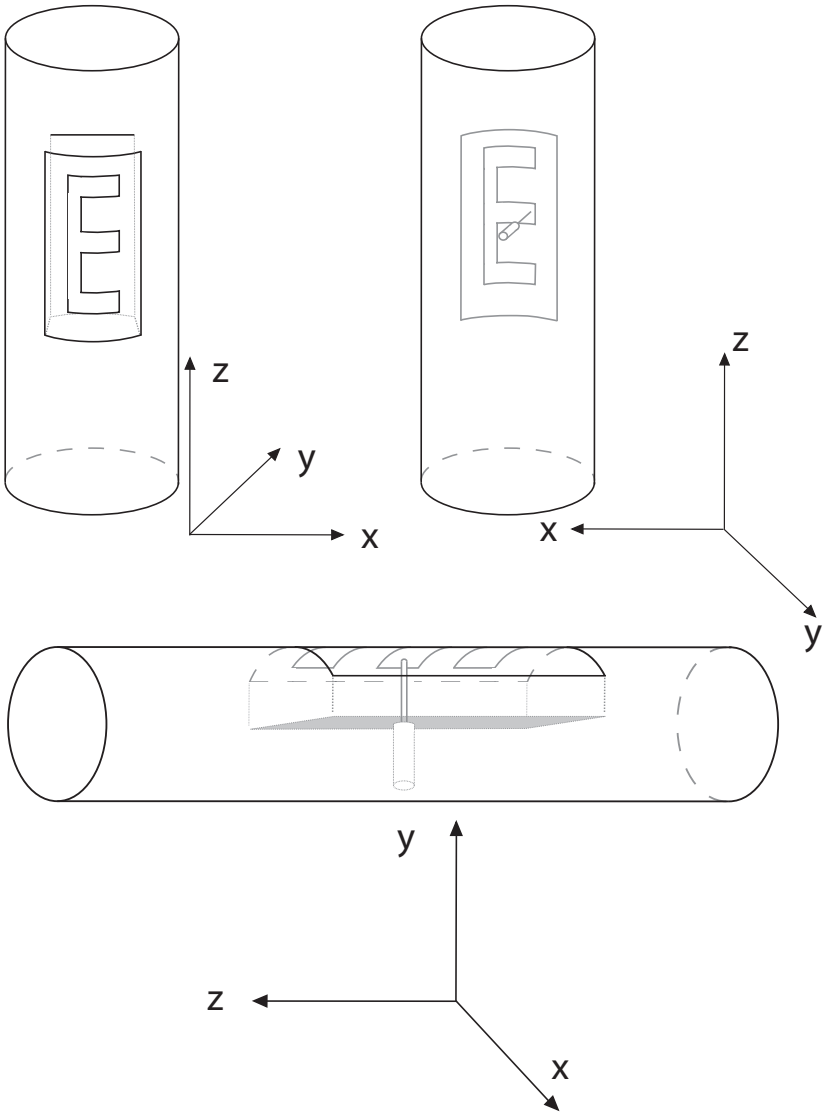

Figure 2.14: A conceptual depiction of an cavity-backed E-patch conformed to a cylinder. Depicted cavity and E-patch dimensions are not to scale.

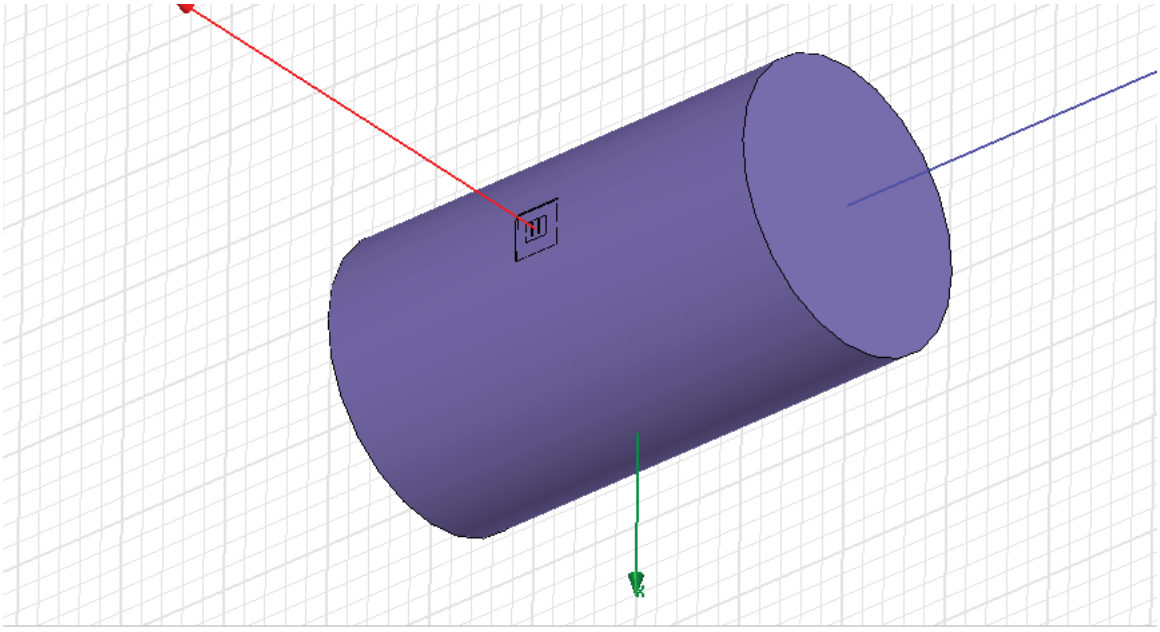

Figure 2.15: HFSS screen capture of an cylindrically conformal cavity backed E-patch antenna.

3.10. Using Figure 2.19 to compare frequency responses of different E-patch antennas on different radii of cylinders, a trend is noted for the flat bottom cavity. As the radius ρ is increased, $|S_{11}|$ decreases for the lower resonant frequency and increases for the higher resonance frequency. In addition the higher resonance frequency has shifted upwards in frequency. These differences in $|S_{11}|$ versus frequency might be due to the mode two resonance point being altered by curvature of the cylinder. With the exception of the 50 cm radius, at the L1 and L2 frequencies $|S_{11}|$ is less then -10 dB. For the 50 cm radius cylinder case, $|S_{11}|$ at the L2 frequency is approximately -9 dB, and is less than -10 dB at the L1 frequency.

2.4 HFSS cylindrically conformal cavity simulations

Refer to Figure 2.20 and Figure 2.21 for depictions of the E-patch antenna and associated cavity conformed to the surface of a cylinder. In [24] the authors discussed conforming a rectangular patch antenna to the surface of a cylinder. The authors in

Table 2.11: Dimensions and probe placement of a cylindrically conformal cavity-backed E-patch antenna.

Dimensions	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
ρ	1000

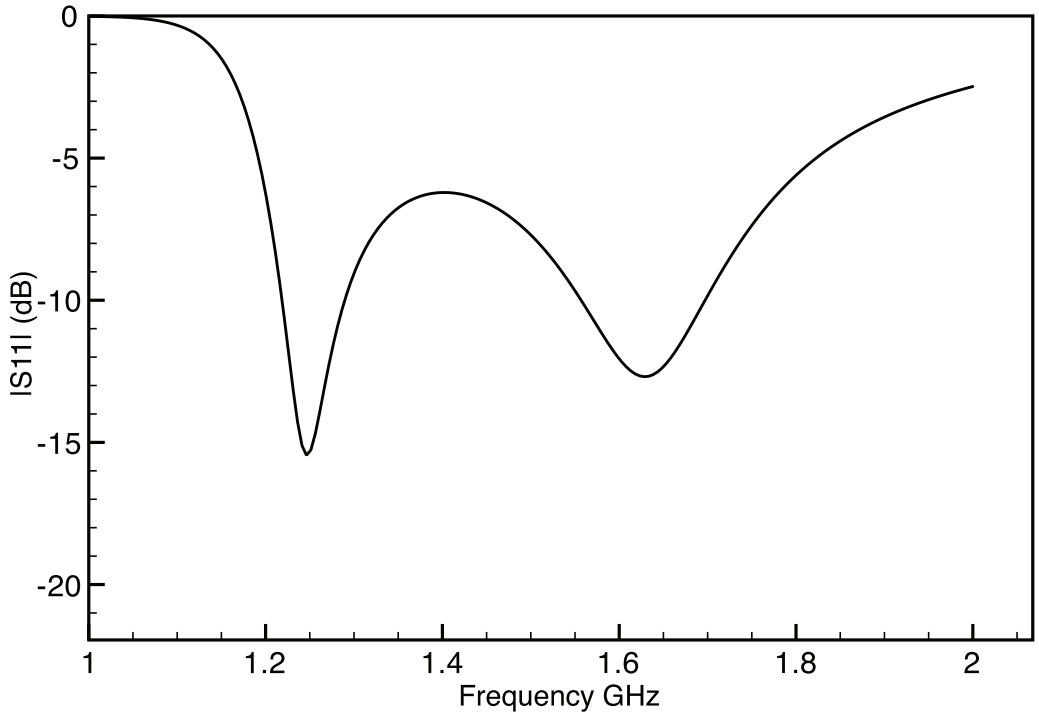

Figure 2.16: Simulated frequency response of an cylindrically conformal cavity-backed E-patch antenna ($\rho = 100\,\text{cm}$). Dimensions provided in Table 2.11.

Table 2.12: Dimensions and probe placement of a cylindrically conformal cavity-backed E-patch antenna.

Dimensions	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
ρ	700

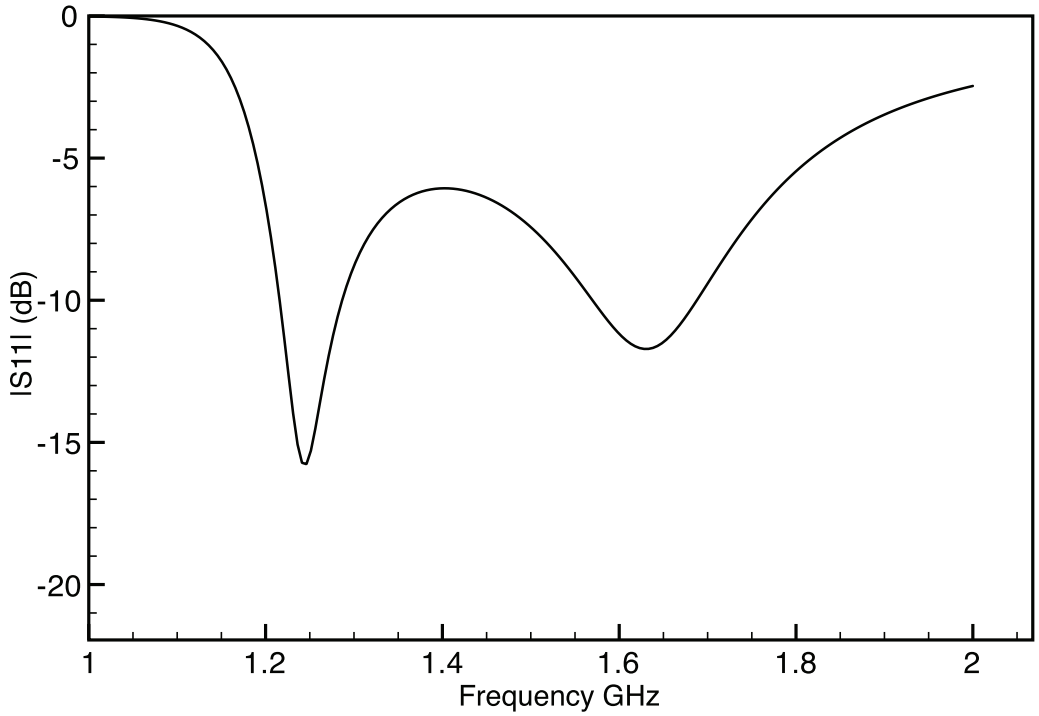

Figure 2.17: Simulated frequency response of an cylindrically conformal cavity-backed E-patch antenna($\rho = 70\,\text{cm}$). Dimensions provided in Table 2.12.

Table 2.13: Dimensions and probe placement of a cylindrically conformal cavity-backed E-patch antenna.

Dimensions	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
ρ	500

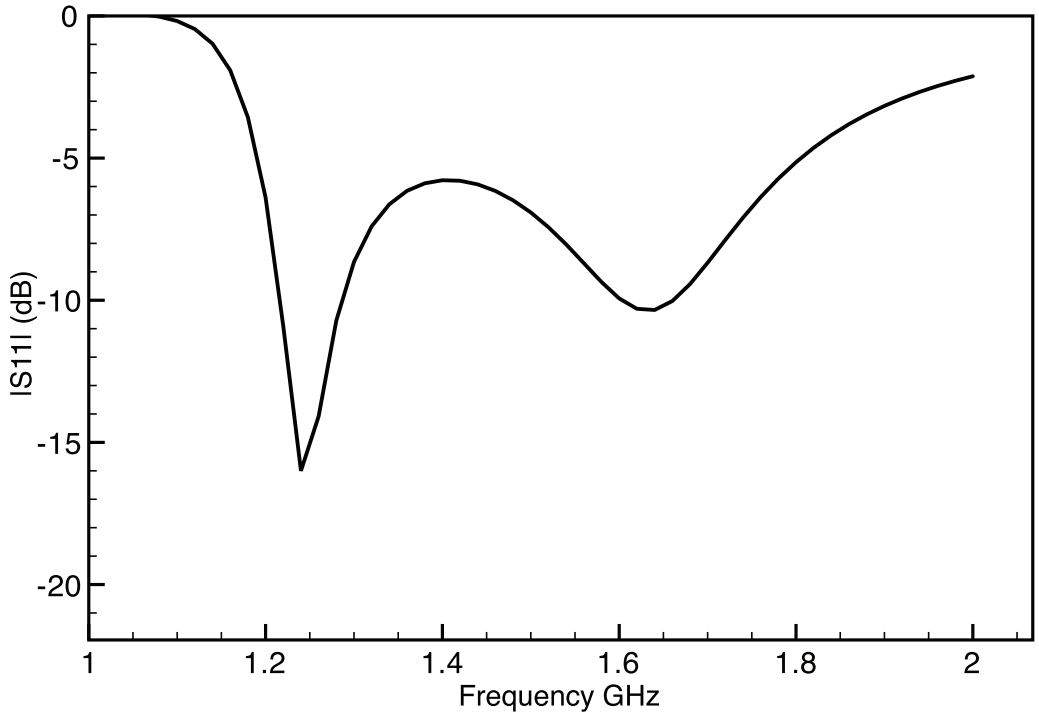

Figure 2.18: Simulated frequency response of an cylindrically conformal cavity-backed E-patch antenna ($\rho = 50\,\text{cm}$). Dimensions provided in Table 2.13.

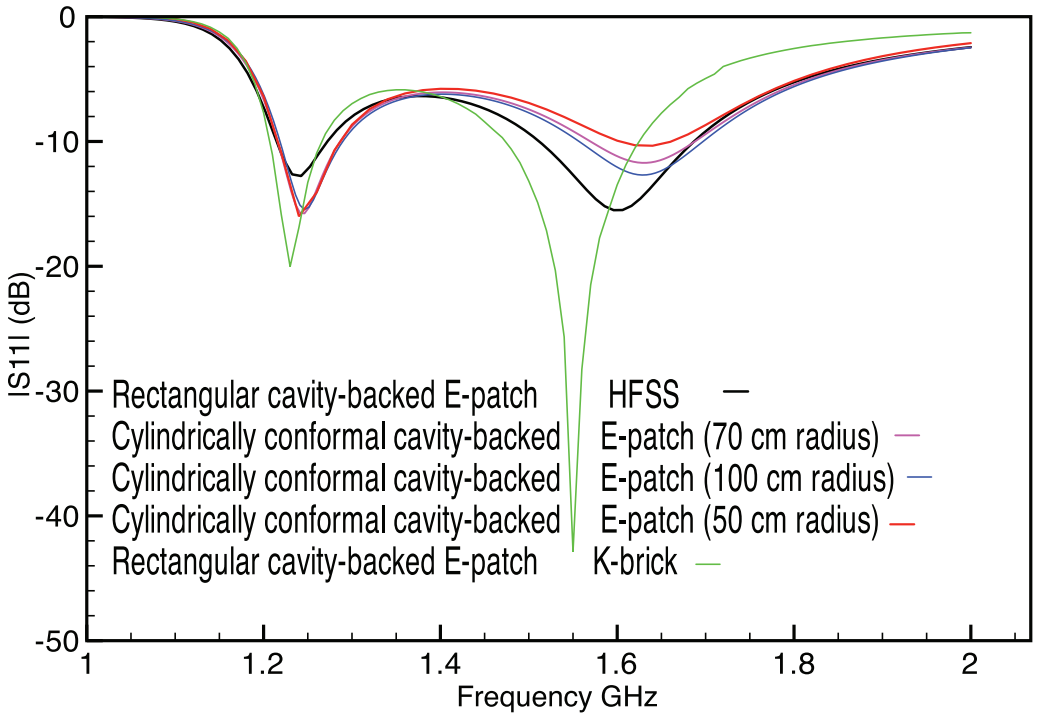

Figure 2.19: Simulated frequency responses of cylindrically conformal and rectangular cavity-backed E-patch antennas. Dimensions provided in Tables 2.11 - 2.13 and 3.12.

[24] did not discuss cavity-backed scenarios and assumed a finite dielectric beneath the surface of the patch with a free space air dielectric surrounding the patch and corresponding grounded cylinder. However, the authors in [24]did provide data for insight into the effect of curvature on patch antennas. The effect of conforming a patch antenna to a cylinder was to produce a broader radiation pattern and larger bandwidth for the TM_{01} mode. As discussed in section 2.3, HFSS provides E and H-fields, currents, S-parameters and near and far radiated field simulation results. HFSS is used to provide $|S_{11}|$ for the cylindrically conformal cavity-backed E-patch antenna for the conformed cavity case. Figures 2.22 and 2.23 show some screen captures of the simulations from HFSS.

The HFSS settings for the simulations involving the cylindrically conformal cavity-backed E-patch antenna are listed in Table 2.14. All simulations were done in a driven set-up.

Table 2.14: HFSS settings for the cylindrically conformal cavity-backed E-patch antenna.

HFSS setting	HFSS setting value(s)
Frequency of solving	1.6 GHz
Adaptive passes	7
Maximum Delta S	.01
Sweep Type	Fast
Frequency Set up	linear step 1 to 2 GHz step size .02
Initial mesh options	do lambda refinement .3
Solution Options	First order
Excitation	Waveport at the coaxial probe feed
Port Renormalization	50 Ω
Deembed setting	checked -35 mm

Figure 2.24 shows the simulated frequency response of the cylindrically conformal (100 cm radius) cavity-backed E-patch antenna. The E-patch antenna in Figure 2.24 has the same dimension values as the rectangular cavity case in Table 3.12. Figure 2.25 shows the simulated frequency response of the cylindrically conformal (70 cm

radius) cavity-backed E-patch antenna. The E-patch antenna in Figure 2.25 has the same dimension values as the rectangular cavity case in Table 3.12. In Figure 2.26 the simulated frequency response of the cylindrically conformal (50 cm radius) cavity-backed E-patch antenna is shown. The E-patch antenna in Figure 2.26 has the same dimension values as the rectangular cavity case in Table 3.12. In Figure 2.27 the simulated frequency response of the cylindrically conformal (25 cm radius) cavity-backed E-patch antenna is shown. The E-patch antenna in Figure 2.27 has the same dimension values as the rectangular cavity case in Table 3.12. In Figure 2.28 the simulated frequency response of the cylindrically conformal (15.4 cm radius) cavity-backed E-patch antenna is shown. The E-patch antenna in Figure 2.28 has the same dimension values as the rectangular cavity case in Table 3.12.

Table 2.15: Dimensions and probe placement of a cylindrically conformal cavity-backed E-patch antenna.

Dimensions	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
ρ	250

Figure 2.29 contains all frequency response data in Figures 2.24-2.28, and 4.11. Using Figure 2.29 to compare frequency responses of E-patch antennas on different radii of cylinders, a trend is noted. As the radius ρ is decreased, $|S_{11}|$ stays approximately the same for the lower resonant frequency and decreases for the higher resonance frequency. In addition the higher resonance frequency has shifted lower in

Table 2.16: Dimensions and probe placement of a cylindrically conformal cavity-backed E-patch antenna.

Dimensions	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
ρ	154

frequency as the radius decreases. Also, bandwidth enlarges as the radius decreases. Notably, the combined first and second resonance bandwidth for the 100 cm case is 245 MHz (-10 dB or lower at 1245 MHz to 1330 MHz and 1590 MHz to 1750 MHz) while the bandwidth for the 15.4 cm case is 320 MHz (-10 dB or lower at 1250 MHz to 1340 MHz and 1520 MHz to 1750 MHz). At 50, 25, and 15.4 cm radius, the L2 frequencies $|S_{11}|$ is less then -10 dB. The L1 frequency stays approximately -9 dB for all the cases. These differences in $|S_{11}|$ versus frequency might be due to the mode two resonance point being altered by curvature of the cylinder.

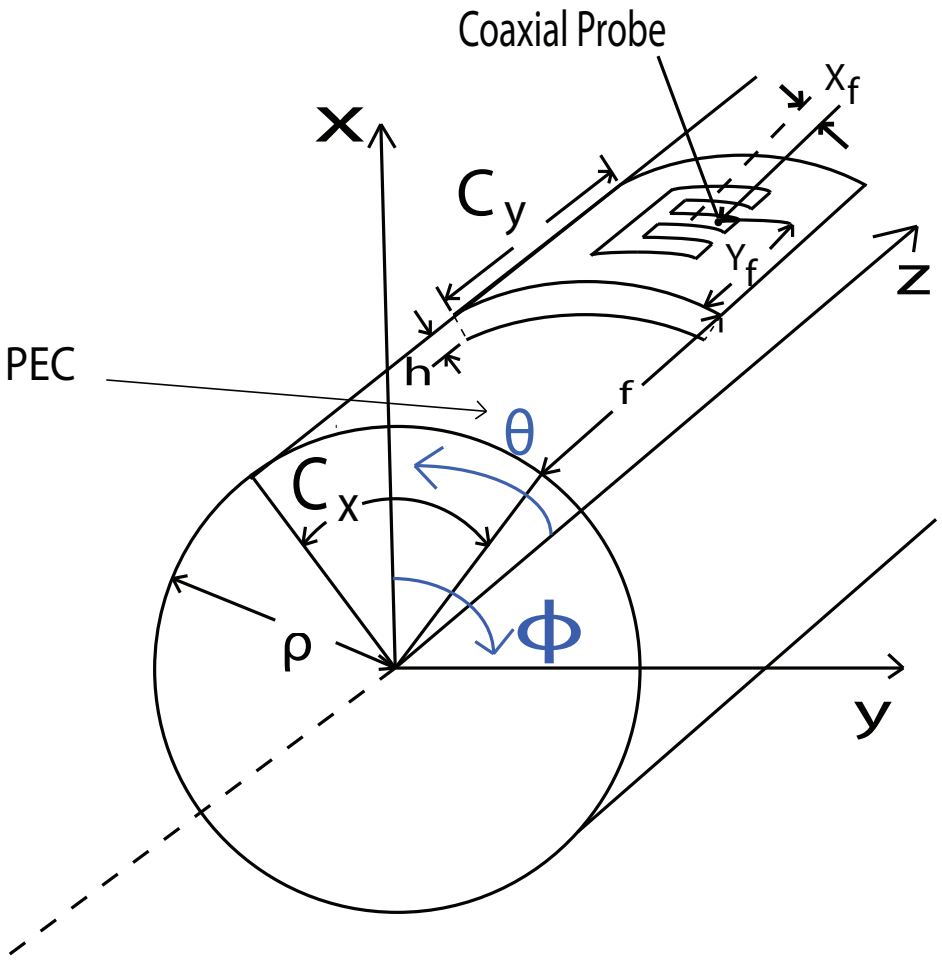

Figure 2.20: An angled-view conceptual depiction of the E-patch antenna conformed to the surface of a cylinder with a radius ρ.

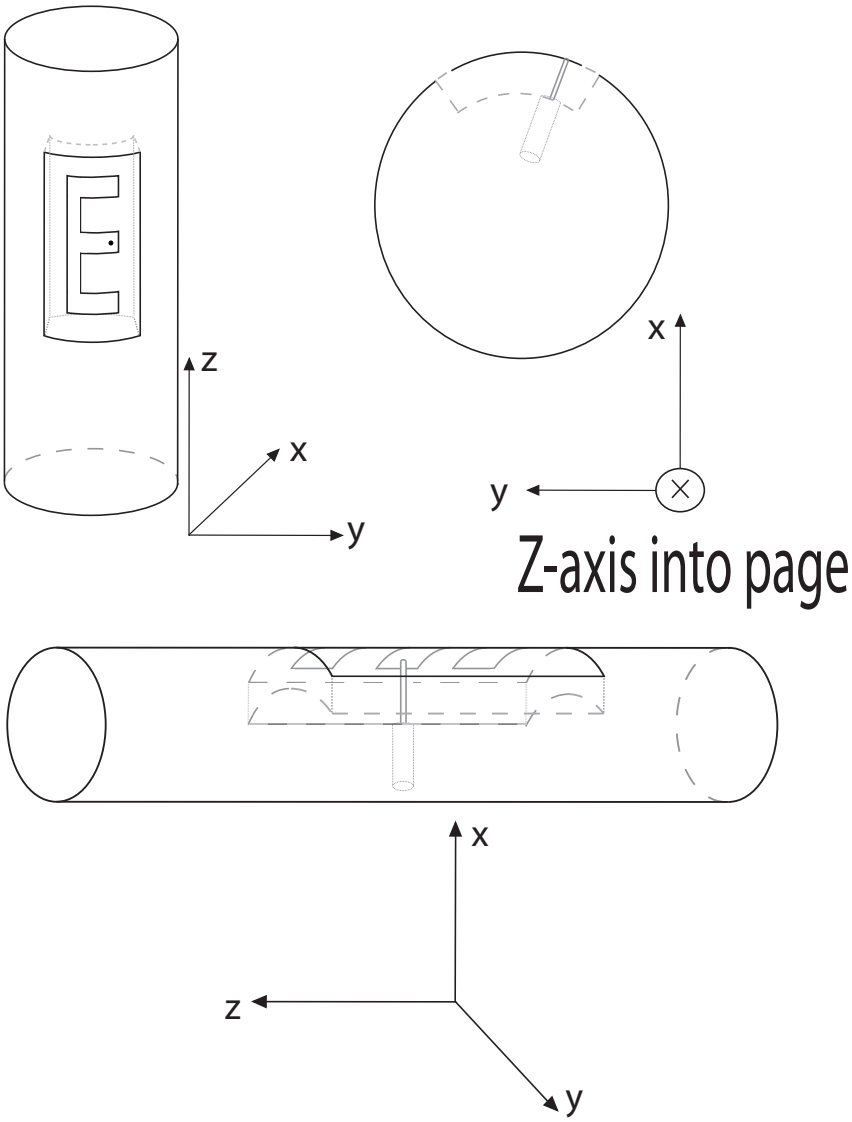

Figure 2.21: A 3-view depiction of the E-patch antenna conformed to the surface of cylinder.

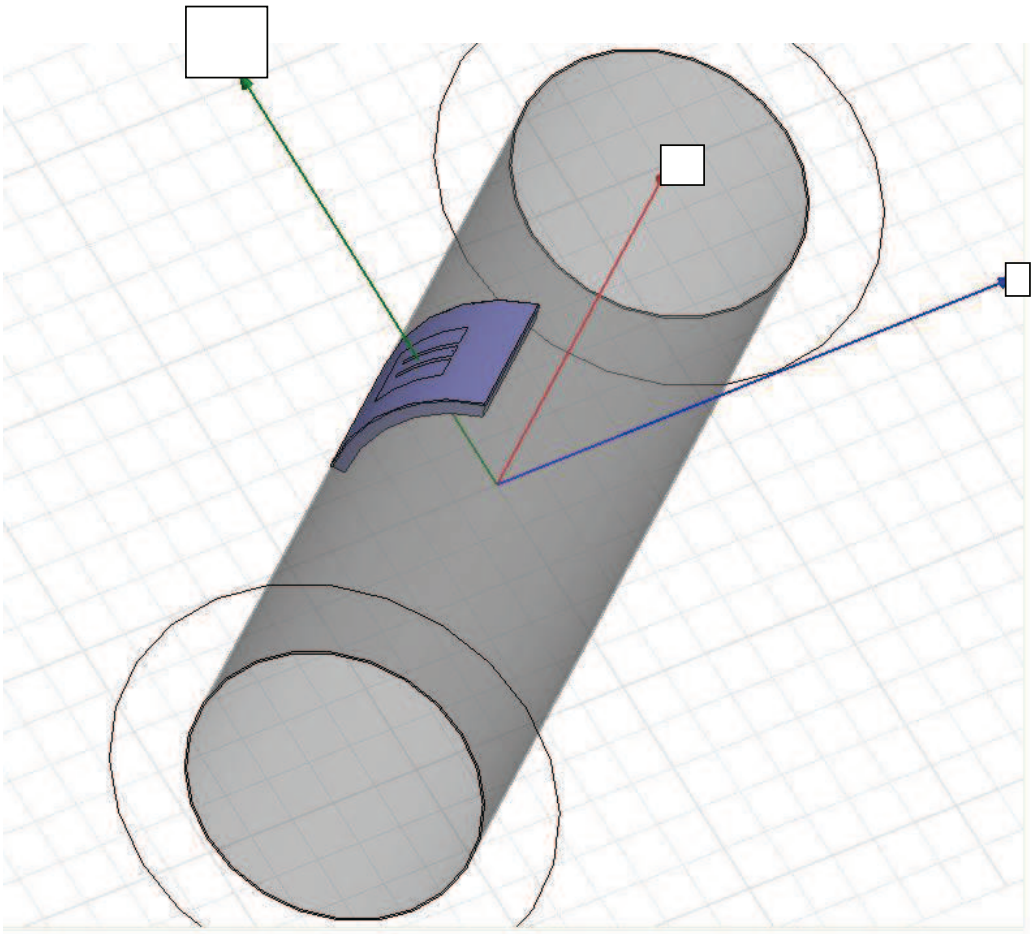

Figure 2.22: An screen capture of the cylindrically conformal cavity-backed E-patch antenna in HFSS.

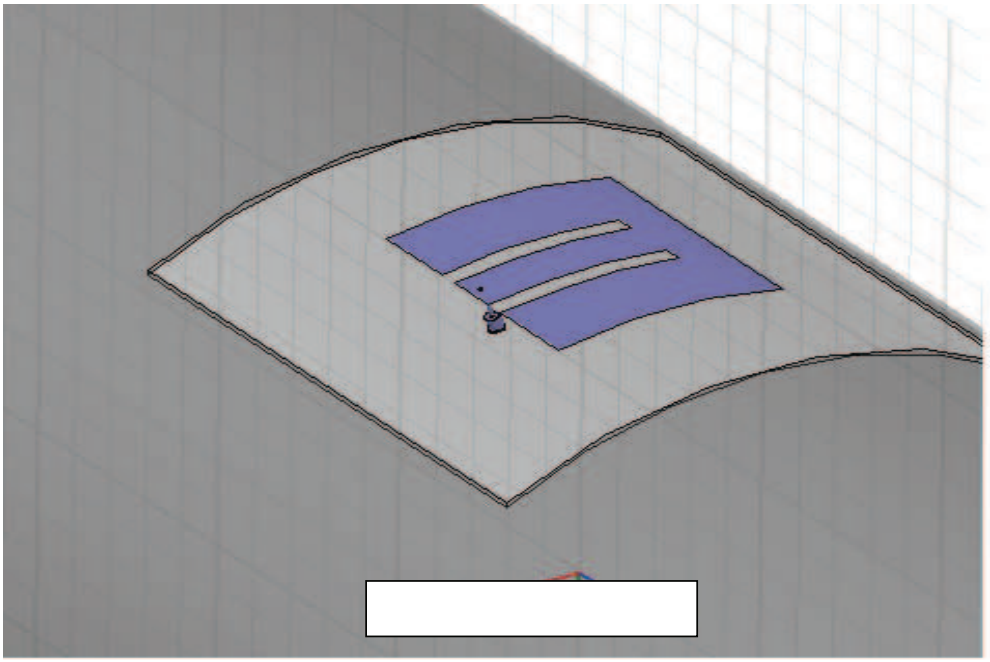

Figure 2.23: An closer view screen capture of the cylindrically conformal cavity-backed E-patch antenna in HFSS.

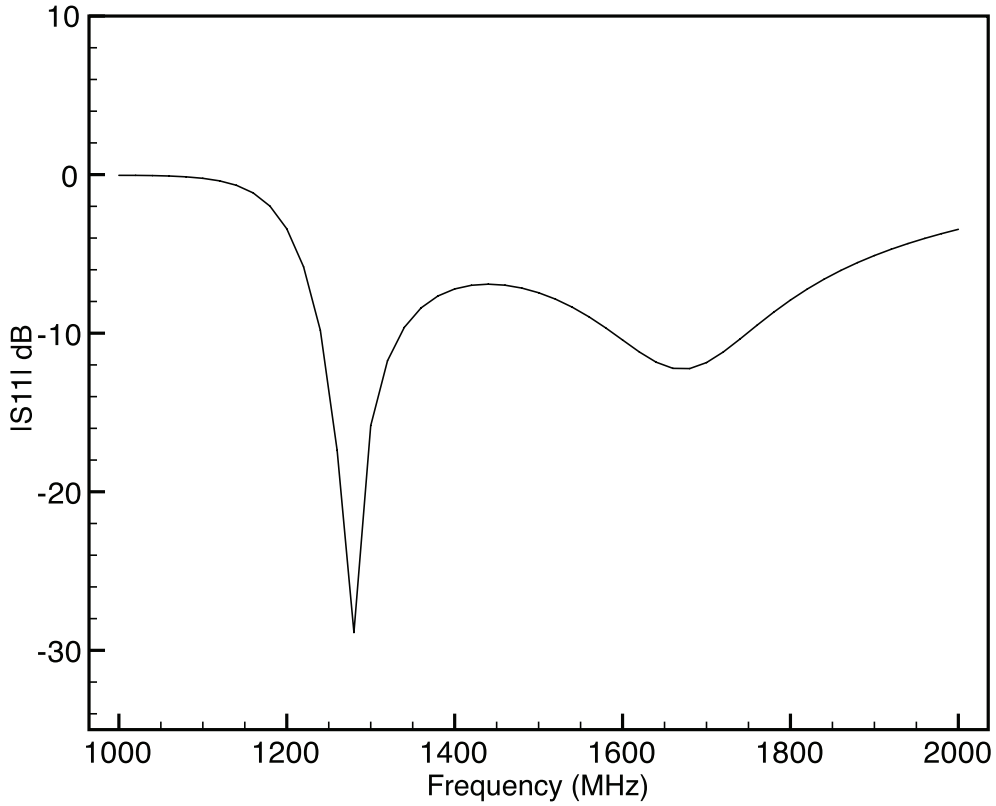

Figure 2.24: Simulated frequency response of a cylindrically conformal cavity-backed E-patch antenna ($\rho = 100\,\text{cm}$). Dimensions provided in Table 2.11.

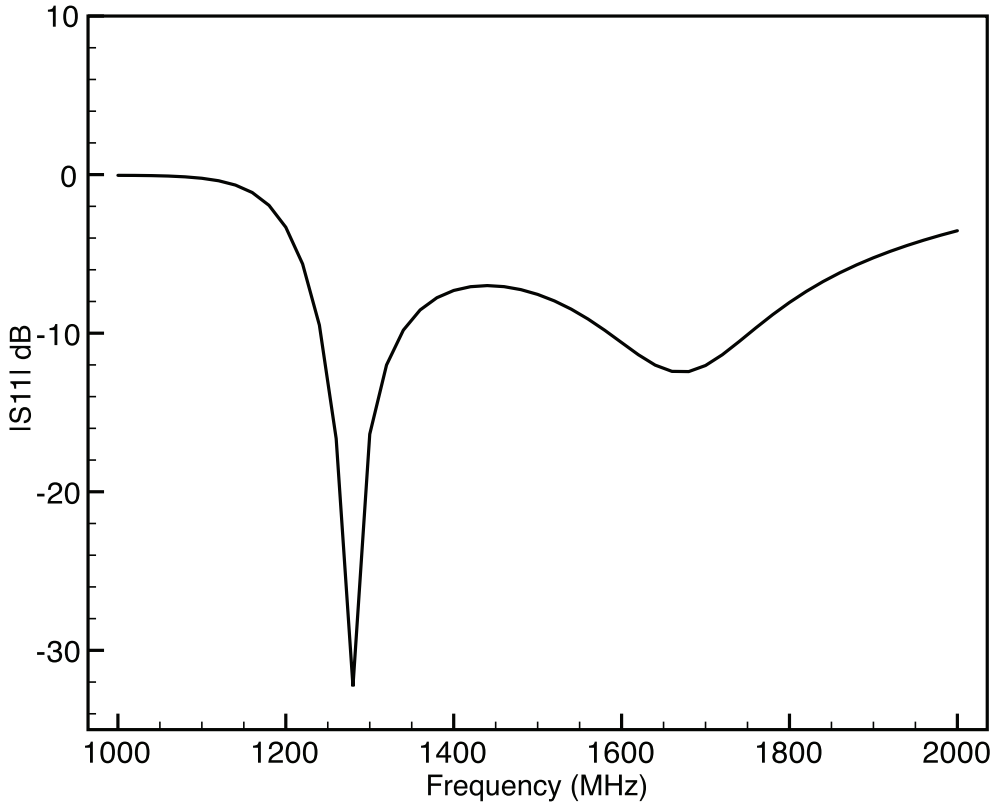

Figure 2.25: Simulated frequency response of a cylindrically conformal cavity-backed E-patch antenna($\rho = 70\,\text{cm}$). Dimensions provided in Table 2.12.

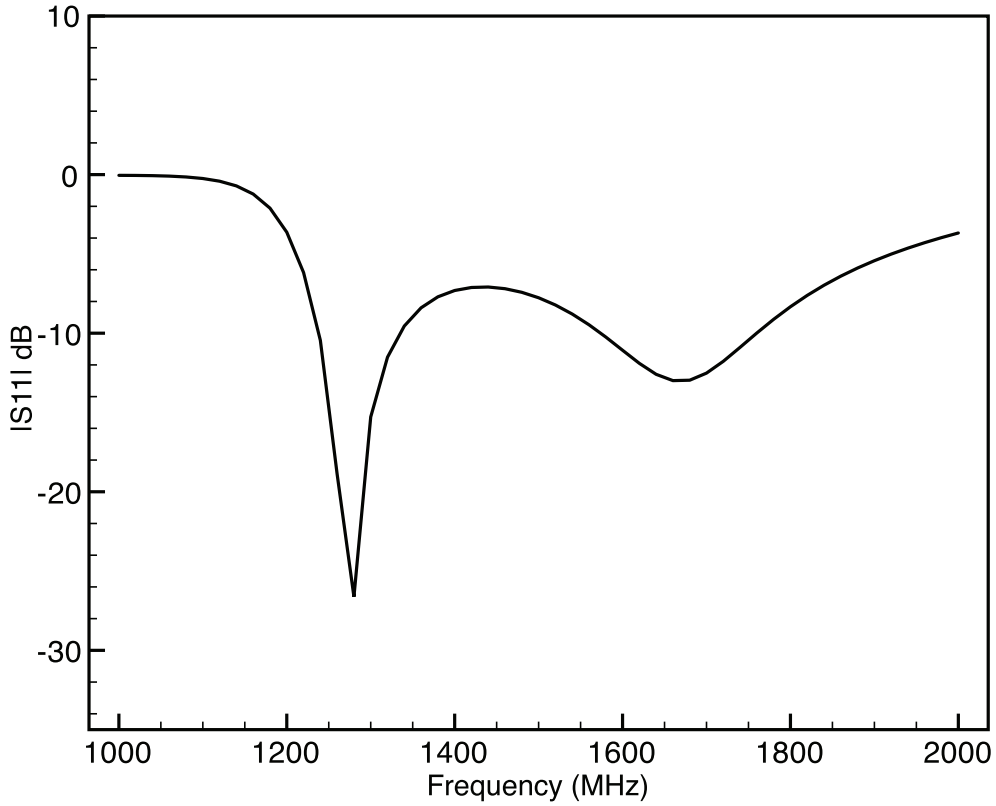

Figure 2.26: Simulated frequency response of a cylindrically conformal cavity-backed E-patch antenna ($\rho = 50\,\text{cm}$). Dimensions provided in Table 2.13.

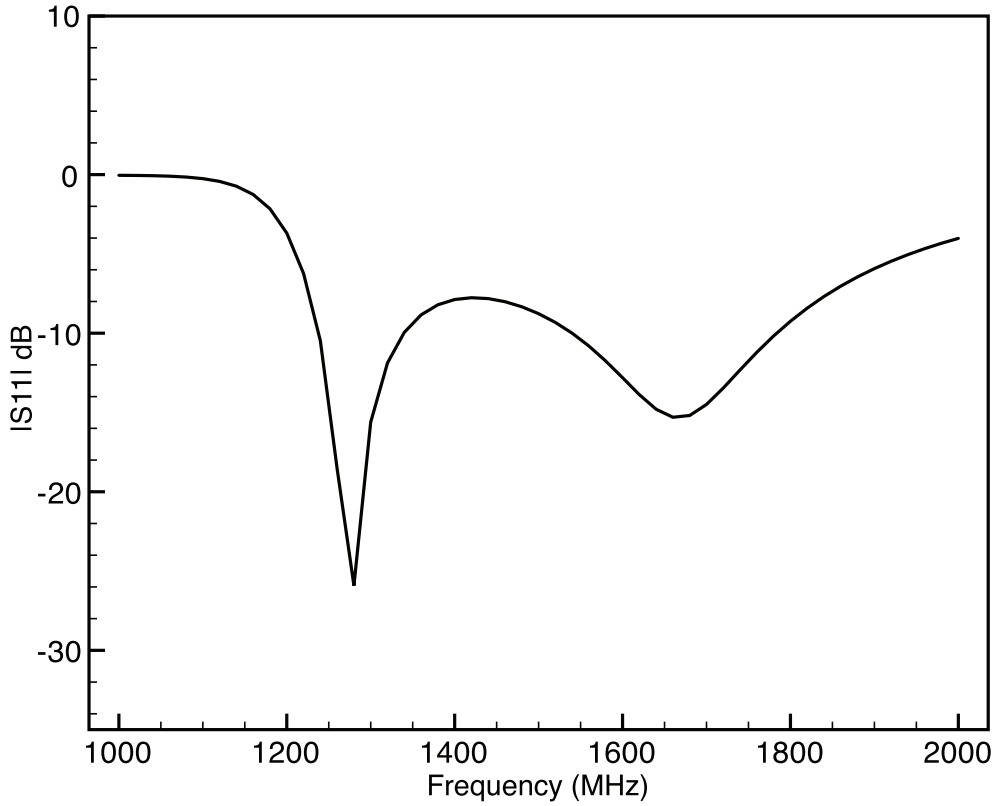

Figure 2.27: Simulated frequency response of a cylindrically conformal cavity-backed E-patch antenna ($\rho = 25\,\text{cm}$). Dimensions provided in Table 2.15.

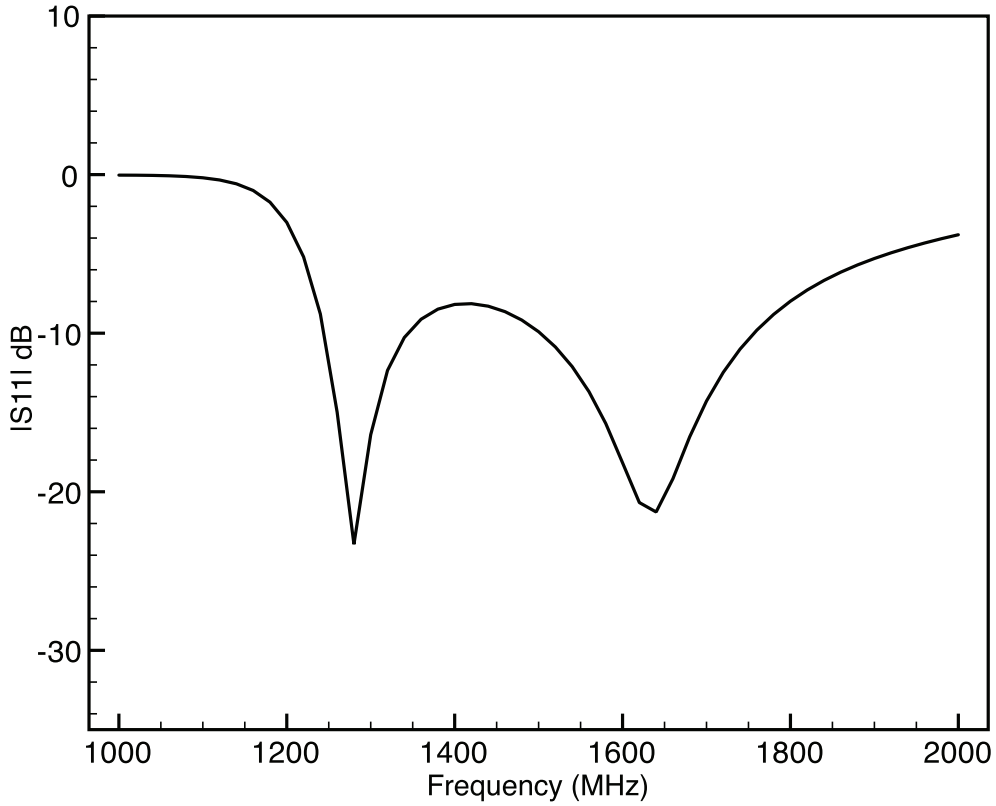

Figure 2.28: Simulated frequency response of a cylindrically conformal cavity-backed E-patch antenna ($\rho = 15.4\,\text{cm}$). Dimensions provided in Table 2.16.

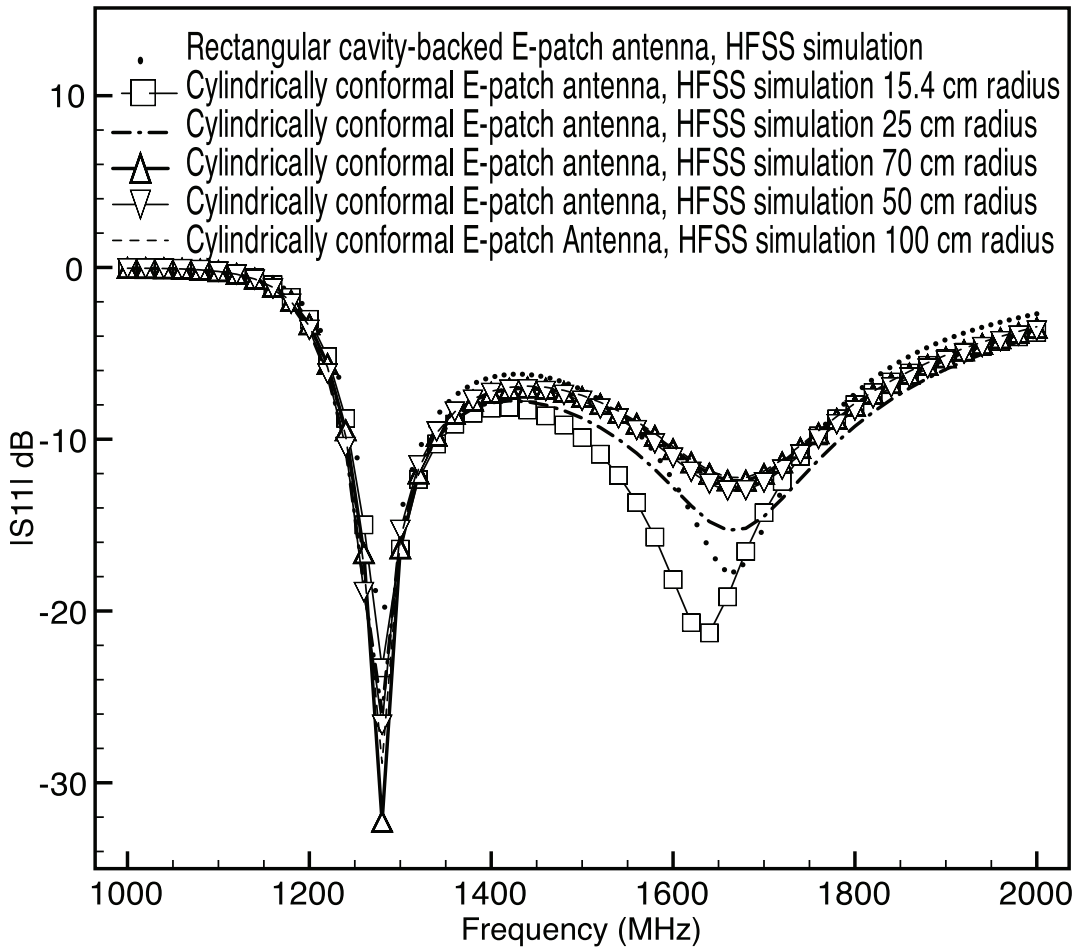

Figure 2.29: Simulated frequency responses of cylindrically conformal and rectangular cavity-backed E-patch antennas. Dimensions provided in Tables 2.11 - 2.13, 2.15, 2.16 and 3.12.

Chapter 3

Taguchi's Method of Optimization for E-patches

3.1 Orthangonal Arrays

As discussed in section 1.2, Taguchi's method involves using OAs to perform optimization. When using OAs for optimization problems two questions arise: Does an OA exist, and if so how can it be constructed? To determine the existence of an OA one must, for given values of k, s, and t, determine the minimum number of rows N so that an OA(N, k, s, t) exists. One would like to minimize N, as it reduces the number of experiments carried out. Rao's inequalities are established as a mathematical answer to the existence question. To minimize N, the parameters of OA(N, k, s, t) must satisfy the following Rao inequalities [14],

$$N \geq \sum_{i=0}^{u} \binom{k}{i} (s-1)^i, \text{ if } t = 2u,\ u > 0 \tag{3.1}$$

$$N \geq \sum_{i=0}^{u} \binom{k}{i} (s-1)^i + \binom{k-1}{u} (s-1)^{u+1}, \text{ if } t = 2u+1,\ u \geq 0. \tag{3.2}$$

More improvements on the Rao's inequalities can be found in [14] for different t strengths and s levels. Numerous techniques to construct OAs are discussed in [14] and some of these techniques are based on Galois fields and error-correcting codes. Galois fields are finite sets of elements where one can add, subtract, multiply and divide within the set. The virtue of the Galois fields is that it is possible to multiply and divide vectors with a specified number of components. Error-correcting codes are designed to correct errors in the transmission of data over noisy communication channels. These error-correcting codes can also be used to construct OAs. One may consult [14] for more information on error-correcting codes, Galois fields, and their use in construction of OAs.

An online database, [25], of OAs has been established by the authors in [14]. OAs with different parameters, levels, and strengths have been developed and archived in this online database, which can also be found in books related to OAs or Taguchi's Method. The online database contains the OAs used in this thesis. The authors in [20] discuss explicitly how to use Matlab code to produce OAs that adhere to the Rao inequalities. However, the authors in [20] only discuss constructing OAs with an arbitrary odd-level ($s = 3, 5, 7, ...$) and two-strength ($t = 2$).

OAs have many important properties that are associated with them. The ability of OAs to lessen the number of experiments in optimization problems is discussed in Section 1.2. Another property of OAs is that all possible combinations of the parameters of up to the strength t occur equally; this ensures a balanced and fair comparison of all interactions of levels (s). In Table 3.1 if one picks any level value ($s = (0, 1, 2)$) in a column, that value occurs exactly the same number of times as the other level values. For example, the level values zero, one, and two all occur six times in a column. This means that all possible level values are tested fairly within the OA. Further, if one looks at a combination of any two columns (strength $t = 2$) as a row, one sees exactly the same number of combinations. For example, two combinations

of (0,0), (0,1), (0,2), (1,0), (1,1), (1,2), (2,0), (2,1), (2,2) can be noted as a row in any two columns. To consider more interactions of parameters, one could increase the strength t, but then more rows (experiments) would need to be included to maintain the properties of the OA. The OAs in this thesis use a strength of two, as this seems sufficient for the problems considered.

Another practical property of OAs is discussed by the authors in [20] and is quoted here:

> "... property of OAs is that any $N \times k'$ subarray of an existing OA(N, k', s, t') is still an OA with a notation OA(N, k', s, t'), where $t' = \min(k', t)$. In other words, if one or more columns are deleted from an OA, the resulting array is still an OA but with a smaller number of parameters. For example, if we delete the last two columns in [Table 3.1], we can obtain an [OA$(18, 5, 3, 2)$]. This property is especially useful when selecting an OA from an existing OA database. If an OA with a certain number of columns (k') cannon be found in a database, one can choose an OA with a larger number of columns ($k > k'$) and manually delete the redundant ($k - k'$) columns to obtain the desired OA."

This property influences the choice of an OA for Taguchi optimization and is used later in section 3.2.

3.2 Taguchi's method and optimization example

Taguchi's method can be understood using an example optimization problem. In [26] the authors discuss finding the global minimum of the function

$$f(x, y) = x \sin(4x) + 1.1y \sin(2y), \ 0 \leq x \leq 10, \ 0 \leq y \leq 10. \tag{3.3}$$

Table 3.1: An *Orthogonal Array OA(18,7,3,2)*

0	0	0	0	0	0	0
1	1	1	1	1	1	0
2	2	2	2	2	2	0
0	0	1	2	1	2	0
1	1	2	0	2	0	0
2	2	0	1	0	1	0
0	1	0	2	2	1	1
1	2	1	0	0	2	1
2	0	2	1	1	0	1
0	2	2	0	1	1	1
1	0	0	1	2	2	1
2	1	1	2	0	0	1
0	1	2	1	0	2	2
1	2	0	2	1	0	2
2	0	1	0	2	1	2
0	2	1	1	2	0	2
1	0	2	2	0	1	2
2	1	0	0	1	2	2

This function has a global minimum of -18.5547 located at $x = 9.0390$, $y = 8.6682$. The difficulty in finding a global minimum for this function is that there are many local minima close to the global minimum. Figure 3.1 is a plot for the solution surface of (3.3). To find the global minimum, one may establish a procedure of operation for Taguchi's method. Consider Figure 3.2, which shows a flow chart for Taguchi's Method of optimization from [3]. As shown there, selecting an OA for the optimization problem is the first step. To select the proper OA, first determine the number of variables that needs to be optimized. For the optimization of (3.3), two variables are needed because x and y are both varying; thus $k = 2$. A strength of $t = 2$ is chosen by default because there are only two variables in the optimization. A suitable OA can be constructed from Table 3.2 (from the online database [25]) with the last two columns deleted. As discussed in Section 3.1 the deletion of columns is an acceptable practice as it does not invalidate the properties of an OA. The OA

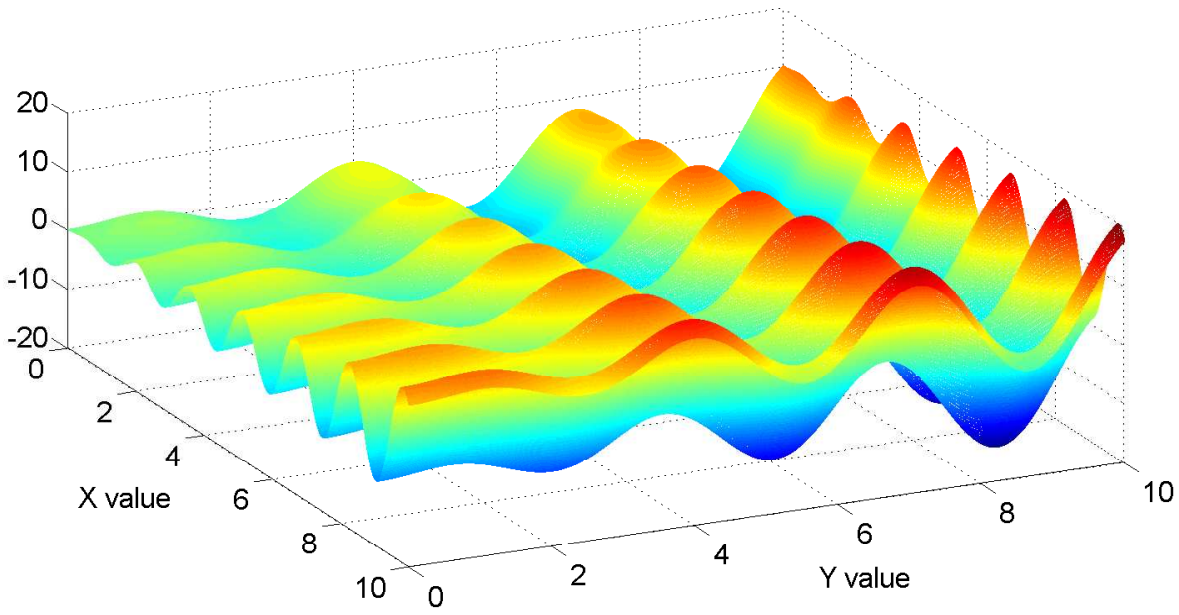

Figure 3.1: Three-dimensional solution surface of (3.3).

constructed from Table 3.2 becomes the OA in Table 3.3.

The next step is to select a fitness function. A fitness function is selected based on its ability to reach the optimization goal. The selection of the fitness function may be the most important factor in Taguchi's method. If one chooses a wrong fitness function, then an optimal solution may not be reached. The fitness function for (3.3) should take on its lowest value when (3.3) is at the global minimum. The designed fitness function for (3.3) is,

$$Fitness = \exp\left(x\sin(4x) + 1.1y\sin(2y)\right). \tag{3.4}$$

The role of the exponential function in the fitness of (3.4) will be explained later in this section. All one needs to know at this point is that a smaller fitness means a result closer to the optimum.

A range of numbers needs to be selected for the variables (input parameters) in

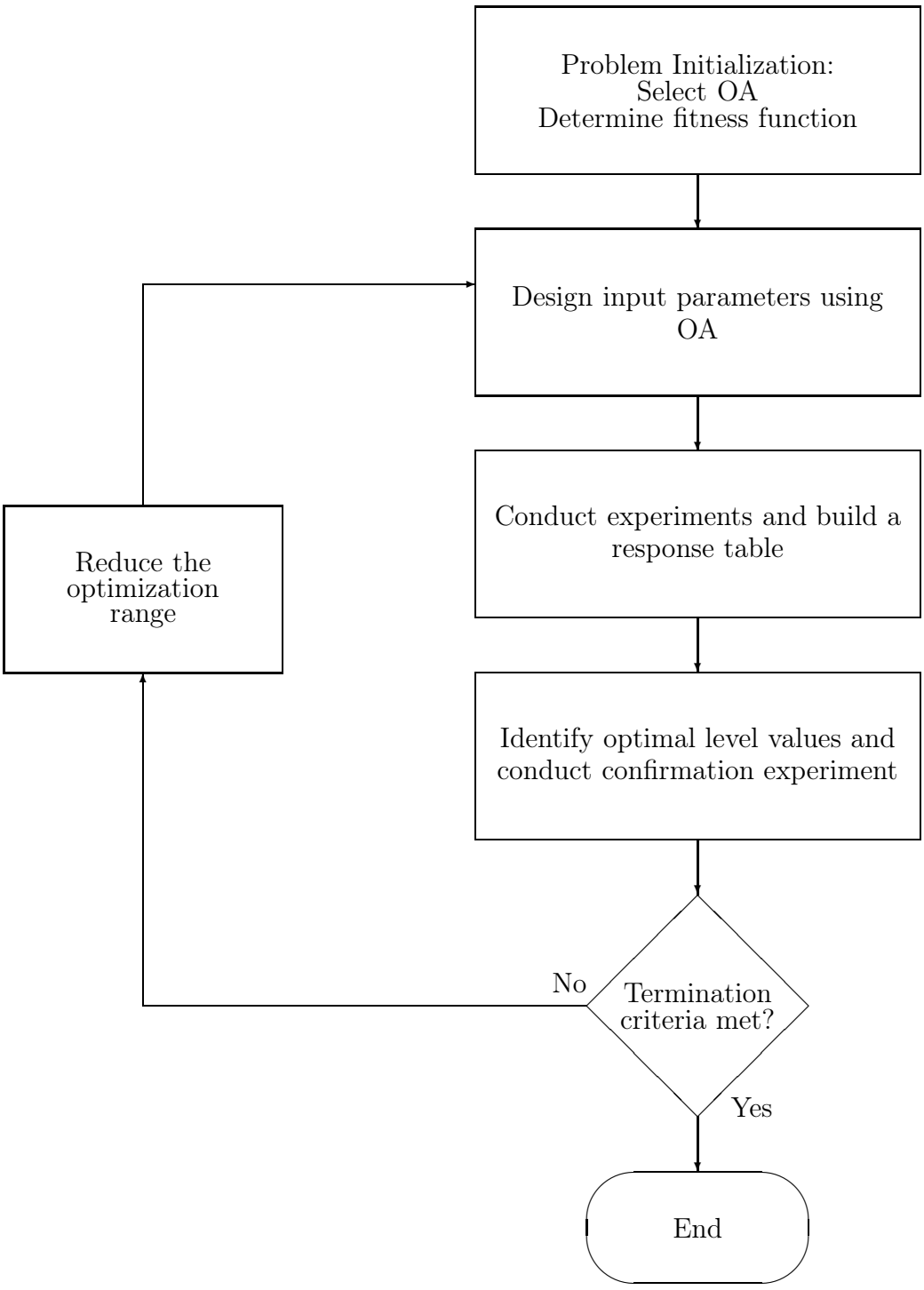

Figure 3.2: Flowchart of Taguchi's optimization method from [3].

the fitness function. When an OA is used in Taguchi's Method, the level values correspond to numbers in the input parameter range. The input range of (3.3) is between 0 and 10 for both x and y variables. To determine which input numbers to be used as level values in each iteration, a level difference (LD) needs to be calculated. The level difference is the amount of separation between level 2 and level 1 (also level 2 and level 3). The equation,

$$LD_1 = \frac{\text{max-min}}{\text{number of levels} + 1} = \frac{10 - 0}{3 + 1} = 2.5 \tag{3.5}$$

is used to calculate the first LD of the x variable of (3.3). Max and min are the maximum and minimum values of the input range, in this case 10 and 0 respectively. If the input range was instead 20 and -10 for the maximum and minimum values, then the value of LD would be 7.5. As Taguchi's Method is implemented, the LD changes depending on the iteration (discussed later in this section). For the first iteration of (3.3) optimization, the middle of the input range was selected and thus level 2 becomes 5. Table 3.4 is an OA with input values corresponding to the level values of the first iteration of (3.3) optimization. Level 1 corresponds to 2.5 $(5 - 2.5)$ and Level 3 becomes 7.5 $(5 + 2.5)$.

Now that the OA level values have input parameters associated with them, experiments can be conducted and a response table can be built. Conducting experiments means computing the fitness function with the level values for a given row in the OA. For example, the first experiment outcome is,

$$Fitness = \exp\left(2.5\sin(4 * 2.5) + 1.1 * 2.5\sin(2 * 2.5)\right) = .0184. \tag{3.6}$$

The seventh experiment outcome is,

$$Fitness = \exp\left(7.5\sin(4 * 7.5) + 1.1 * 2.5\sin(2 * 2.5)\right) = .000043303. \tag{3.7}$$

Table 3.5 shows the OA with the associated experiment outcome (fitness). Once all experiments have been conducted and fitnesses determined, the fitness values are converted to a signal-to-noise (S/N) ratio η. The S/N is determined by the following,

$$\eta = -20 \log_{10} (\text{Fitness}). \tag{3.8}$$

As (3.4) decreases, (3.8) results in a large S/N ratio. The role of the exponential function in (3.4) now becomes evident. If (3.3) were used as the fitness function, then the exponential function compensates for the logarithm in the S/N making the S/N proportional to the function value. The scaling is such that less than unity gives positive S/N values. A fitness function of

$$Fitness = 20 + (x \sin(4x) + 1.1y \sin(2y)). \tag{3.9}$$

could also be used as a fitness function for this optimization. However, it relies on knowing where the minimum point already is. The fitness function (3.9) yields the same results presented in this section. The S/N ratio for the first iteration (fitness function (3.3)) is included in Table 3.5.

Table 3.2: An Orthogonal Array OA(9,4,3,2)

0	0	0	0
0	1	1	2
0	2	2	1
1	0	1	1
1	1	2	0
1	2	0	2
2	0	2	2
2	1	0	1
2	2	1	0

When the S/N ratios are obtained, then a response table is built. A response

Table 3.3: An Orthogonal Array OA(9,2,3,2)

0	0
0	1
0	2
1	0
1	1
1	2
2	0
2	1
2	2

table is populated with the averaged S/N ratios for each parameter n and each level m using the following,

$$\bar{\eta}(m,n) = \frac{s}{N}\sum \eta_i. \tag{3.10}$$

Here N is the experiment (row) number, s is the total level value (3 in this case) and i is the iteration number. For example, the average S/N ratio for the 2nd column (y), 1st iteration, 3rd level is:

$$\bar{\eta}(3,2) = \frac{3}{9}\sum \eta_1 = \frac{1}{3}[(-34.7854) + (-86.2474) + (17.7658)] = -34.4224(dB). \tag{3.11}$$

The rest of the average S/N ratios are found in the response Table 3.6. After the response table has been built, the largest S/N ratio for each parameter is identified. Italicized in Table 3.6 are the largest S/N ratios. A confirmation experiment is performed using the combination of the optimal levels identified in the response table, in this case, level 3 for column 1 ($x = 7.5$) and level 2 for column 2 ($y = 5.0$). As mentioned in Section 1.2, Taguchi's method is a fractional factorial approach. Therefore, the optimal combination may not be included in Table 3.3. The confirmation experiment is therefore not a redundant experiment and the fitness value obtained from the confirmation experiment is regarded as the fitness values of the current i iteration. If the results of the current iteration do not meet the optimization criteria

Table 3.4: The Orthogonal Array OA(9,2,3,2) and level values in the first iteration of Equation (3.3) optimization.

2.5	2.5
2.5	5.0
2.5	7.5
5.0	2.5
5.0	5.0
5.0	7.5
7.5	2.5
7.5	5.0
7.5	7.5

or termination criteria (discussed later), then the process is repeated in the next iteration. The optimal level values of the current iteration are used as level 2 values in the next iteration. For example, after the first iteration of (3.3) optimization, 7.5 would be the new level 2 value for the 1st element (x variable) and 5.0 would be the level 2 value for the 2nd element (y variable). To reduce the optimization range for a converged result, the LD is multiplied by a reduction rate to obtain a new LD for the new iteration. An example reduction rate is rr^i, where i is the iteration. The larger rr is, the slower the convergence rate. Typically, rr is chosen to be between .9 and .5. For (3.3) optimization, the rr is chosen to be .9. After the first iteration in (3.3) optimization process, LD_2 would be

$$LD_2 = LD_1 * rr^i = 2.5 * (.9)^1 = 2.25. \tag{3.12}$$

If LD_i is large enough, and the level 2 of the next generation is close to the upper or lower input parameter bounds, then a danger exists that the corresponding level 1 or level 3 values may lay outside the input parameter bounds. A simple solution is to use an if-then statement at the start of the iteration. If the level value is below the minimum input parameter, then that level is set to the minimal value of the input parameter. Similarly, if the level value is above the maximum input parameter, then

that level is set to the maximum value of the input parameter.

Table 3.5: The Orthogonal Array OA(9,2,3,2), level values, S/N ratio, and experiment outcome in the first iteration of Equation (3.3) optimization.

Experiment (row)	Elements (columns)		Fitness	S/N ratio
1	2.5	2.5	.0184	34.7183
2	2.5	5.0	.0129	37.8025
3	2.5	7.5	54.8621	-34.7854
4	5.0	2.5	6.8736	-16.7437
5	5.0	5.0	4.8192	-13.6595
6	5.0	7.5	20529	-86.2474
7	7.5	2.5	.000043303	87.2696
8	7.5	5.0	.000030361	90.3537
9	7.5	7.5	.1293	17.7658

Table 3.6: Response table for the optimization of (3.3).

Level (row)	Averaged S/N ratio (columns)	
1	12.5784	35.0814
2	-38.8835	*38.1655*
3	*65.1297*	-34.4224

A termination criteria in the optimization procedure may be established by using the following condition

$$\frac{LD_i}{LD_1} < \text{converged value.} \tag{3.13}$$

Normally the converged value is set between 0.0001 and 0.01 depending on the problem. The iterative optimization process is stopped if the optimization criterion is met or if (3.13) is satisfied.

Appendix C shows the Matlab code used in the optimization of (3.3) and can be used as a guide for coding other optimization problems. Figure 3.5 shows the value of (3.3) versus the iteration number of Taguchi's method. Figure 3.3 and Figure 3.4 show the x and y variables versus the iteration number of Taguchi's method. The optimization goal (global minimum of (3.3)) was not met. The global minimum

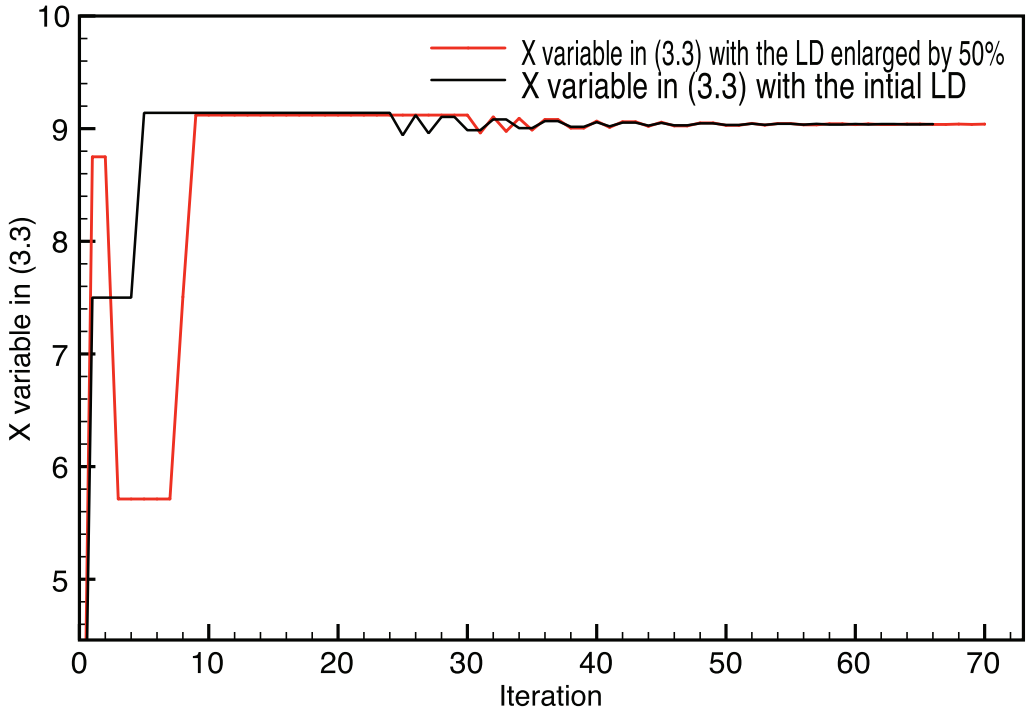

Figure 3.3: X value of (3.3) versus iteration number in Taguchi's Method.

of -18.5547 located at $x = 9.0390$, $y = 8.6682$ was not found after 68 iterations of Taguchi's method. The reason for not finding the global minimum is that LD_1 wasn't large enough to contain for the necessary x and y variables. If LD_1 is enlarged by 50% then the input optimization range is wider and contains the location of the global minimum. This global minimum is found after 71 iterations.

Techniques of decreasing the number of iterations to more quickly achieve the optimization result, such as increasing the level number, changing boundary treatments, and using a Gaussian reduce function, are discussed in [26]. The Rosenbrock function was also optimized using Taguchi's method. It is included in Appedix .

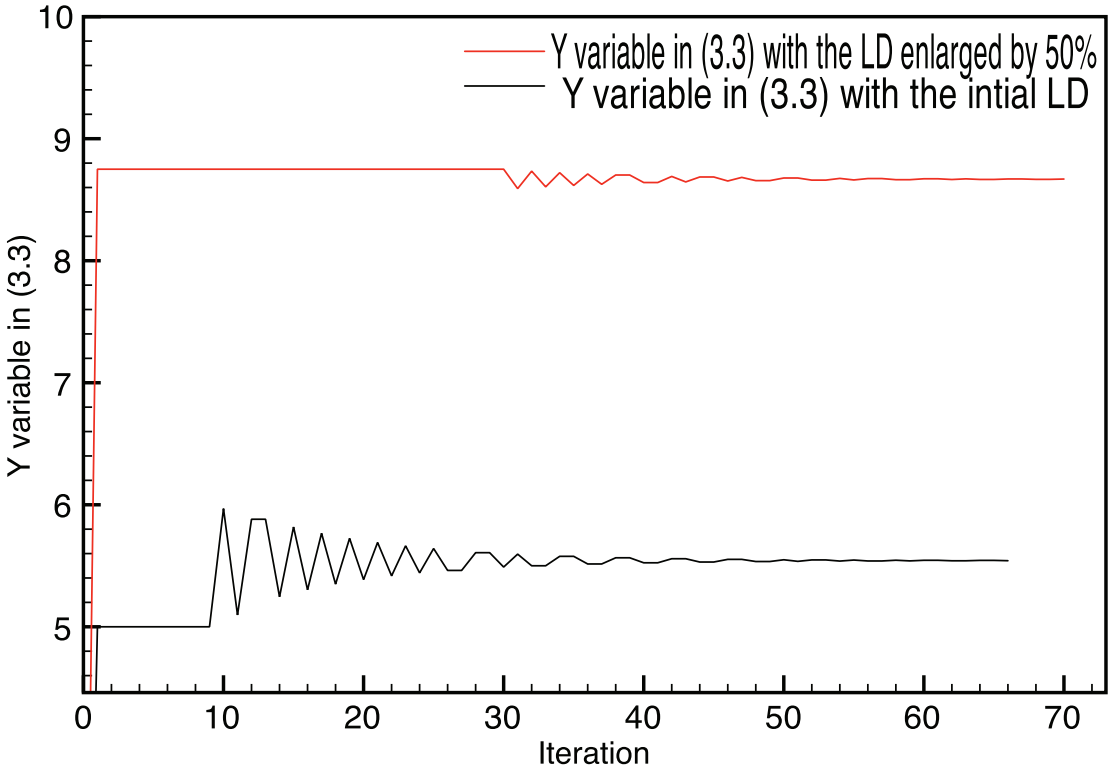

Figure 3.4: Y value of (3.3) versus iteration number in Taguchi's Method.

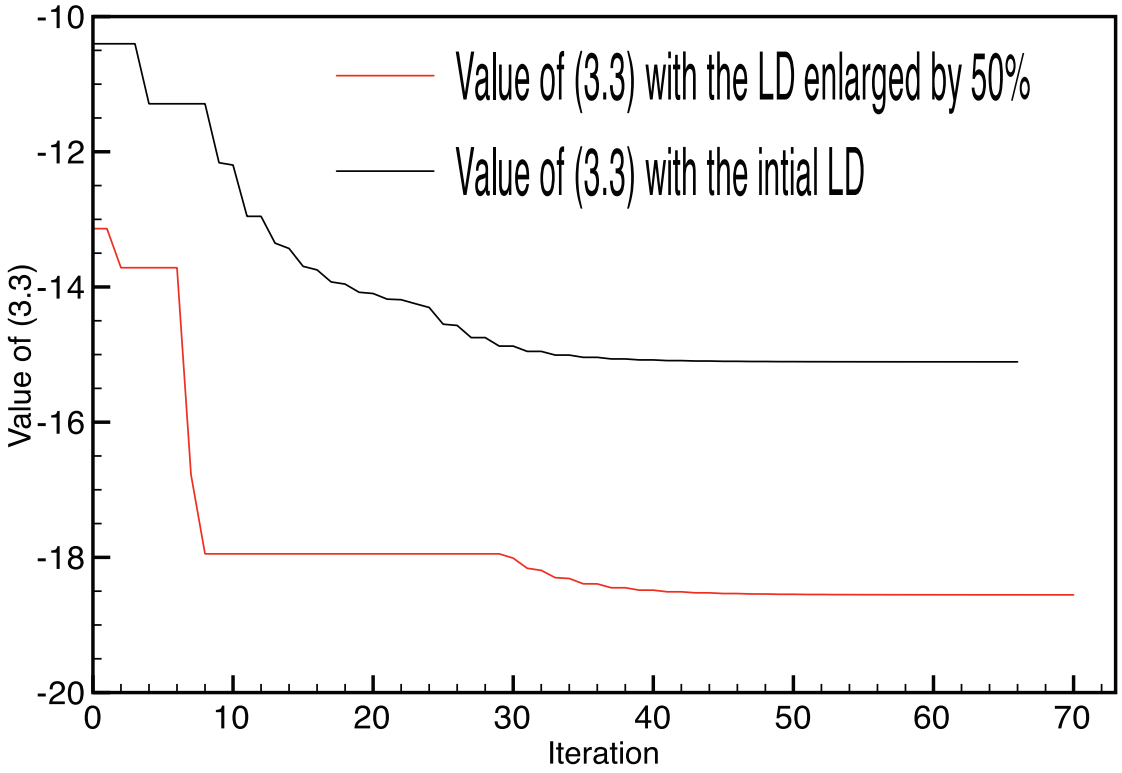

Figure 3.5: Value of (3.3) versus iteration number in Taguchi's Method.

3.3 Taguchi-K-brick-Matlab results

In section 3.2 a Taguchi based optimizer was described that locates a global minimum in a two variable function. Techniques discussed in section 3.1 and 3.2 provide introductory information (OA selection, fitness selection etc.) needed to create a Taguchi based optimizer for $|S_{11}|$ of a cavity-backed E-patch antenna.

Following the procedure shown in Figure 3.2 for developing a Taguchi optimizer, one of the first steps is determining a suitable OA. First, consider the number of parameters of a cavity-backed E-patch antenna to be optimized. For simplicity, the cavity's width C_x (x-direction) and length C_y (y-direction) are fixed at 20 cm x 20 cm. This is done so that the cell size in K-brick is 1mm x 1mm x h (and thus K-brick's calculation time is lessened considerably). The probe placement Y_f (in the y-direction) is always at the vertical symmetric middle of antenna (since the desired mode is symmetric about the y-direction of the probe placement); this eliminates the need for Y_f to be a parameter of optimization. The rest of the E-patch dimensions, h, L_s, W_s, L, X_f*, W, and P_s are the same dimensions as shown in Figure 2.7. The restriction of the dimensions of the cavity to 20 cm x 20 cm, and Y_f to the vertical symmetric middle of antenna, yields seven different dimensional parameters that affect the geometry of the E-patch, and thus the value of $|S_{11}|$.

An OA of seven columns is needed to include the effects of the seven parameters that determine $|S_{11}|$. An OA (27,13,3,2) (selected from the online database [25]), modified by deleting the last six columns, yields a suitable OA. As discussed in section 3.1, the deletion of columns or variables in an OA is an acceptable practice. Table 3.7 is the OA (27,13,3,2) and Table 3.8 is the OA (27,7,3,2). An OA (27,7,3,2) fits all the criteria for this optimization in that it contains seven columns or variables, each of the variables has a middle, higher, and lower value, (three levels) and it satisfies the

*Probe placement X_f is different then the previous E-patch designs in that that the probe placement is given from the center of the cavity instead of placed at $(W - X_f)$.

equations 3.1 and 3.2. As with all the Taguchi optimizations in this thesis, a strength of two is sufficient for the problems considered.

Table 3.7: An Orthogonal Array OA(27,13,3,2)

0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	2	2	0	1	2	1	0	0
2	0	2	2	2	1	1	0	2	1	2	0	0
0	1	0	1	1	1	2	2	0	1	2	1	0
1	1	1	2	2	0	1	2	1	0	0	1	0
2	1	2	0	0	2	0	2	2	2	1	1	0
0	2	0	2	2	2	1	1	0	2	1	2	0
1	2	1	0	0	1	0	1	1	1	2	2	0
2	2	2	1	1	0	2	1	2	0	0	2	0
0	0	1	0	1	1	1	2	2	0	1	2	1
1	0	2	1	2	0	0	2	0	2	2	2	1
2	0	0	2	0	2	2	2	1	1	0	2	1
0	1	1	1	2	2	0	1	2	1	0	0	1
1	1	2	2	0	1	2	1	0	0	1	0	1
2	1	0	0	1	0	1	1	1	2	2	0	1
0	2	1	2	0	0	2	0	2	2	2	1	1
1	2	2	0	1	2	1	0	0	1	0	1	1
2	2	0	1	2	1	0	0	1	0	1	1	1
0	0	2	0	2	2	2	1	1	0	2	1	2
1	0	0	1	0	1	1	1	2	2	0	1	2
2	0	1	2	1	0	0	1	0	1	1	1	2
0	1	2	1	0	0	1	0	1	1	1	2	2
1	1	0	2	1	2	0	0	2	0	2	2	2
2	1	1	0	2	1	2	0	0	2	0	2	2
0	2	2	2	1	1	0	2	1	2	0	0	2
1	2	0	0	2	0	2	2	2	1	1	0	2
2	2	1	1	0	2	1	2	0	0	2	0	2

The next step in designing a Taguchi based K-brick optimizer is the selection of a proper fitness function. Consider the equation

$$Fitness = |(db1 + 40)| + |(db2 + 40)|. \tag{3.14}$$

The variables $db1$ and $db2$ are the simulated $|S_{11}|$ in dB produced by K-brick at the

Table 3.8: An Orthogonal Array OA(7,13,3,2)

0	0	0	0	0	0	0
1	0	1	1	1	2	2
2	0	2	2	2	1	1
0	1	0	1	1	1	2
1	1	1	2	2	0	1
2	1	2	0	0	2	0
0	2	0	2	2	2	1
1	2	1	0	0	1	0
2	2	2	1	1	0	2
0	0	1	0	1	1	1
1	0	2	1	2	0	0
2	0	0	2	0	2	2
0	1	1	1	2	2	0
1	1	2	2	0	1	2
2	1	0	0	1	0	1
0	2	1	2	0	0	2
1	2	2	0	1	2	1
2	2	0	1	2	1	0
0	0	2	0	2	2	2
1	0	0	1	0	1	1
2	0	1	2	1	0	0
0	1	2	1	0	0	1
1	1	0	2	1	2	0
2	1	1	0	2	1	2
0	2	2	2	1	1	0
1	2	0	0	2	0	2
2	2	1	1	0	2	1

L1 and L2 frequencies, respectively. For optimization one would want $|S_{11}|$ to be a minimum at the L1 and L2 frequencies. The absolute value ensures equal weight in the fitness calculation. For example, if $db1$ was -50 and $db2$ was -3, this would not be optimal. Indeed, the fitness would be 47, whereas a $db1$ and $db2$ value of both -40 would yield 0. Obviously, the antenna is optimized when the fitness function is 0. The value of 40 was chosen as a sufficient number to add to an optimal dB response of $|S_{11}|$ at the L1 and L2 frequencies. Another fitness function considered was

$$Fitness = |(db1 + 20)| + |(db2 + 20)|. \tag{3.15}$$

Different but similar fitness functions yield different results as discussed later in this section.

In Table 3.9 is the input parameter selection of seven variables used in the optimization of (3.14). In Table 3.10 is the input parameter selection of the seven variables used in the optimization of (3.15). With the OA, fitness function, and input

Table 3.9: The input parameter ranges for E-Patch Taguchi optimization. Fitness function is (3.14).

E-patch dimension variable	Min Value in mm	Max Value in mm
W	60	90
L	85	110
L_s	40	89
W_s	1	10
P_s	5	25
h	8	20
X_f	0	70

parameters chosen, the Taguchi optimization can proceed. However, special attention needs to be given to undesirable E-patch geometries that may occur during optimization. For example, if L_s was 75 mm and W was 65 mm then the slots would cut through the patch and would yield three separate patch antennas. To exclude these

Table 3.10: The input parameter ranges for E-patch Taguchi optimization. Fitness function is (3.15).

E-patch dimension variable	Min Value in mm	Max Value in mm
W	60	90
L	85	110
L_s	40	89
W_s	1	7
P_s	8	15
h	8	16
X_f	0	70

geometries a simple if-then statement is used in the Matlab code. Another parameter to monitor is the X_f placement. If the probe is placed off the antenna, then the impedance result would not be practically useful. Once again during optimization, a simple if-then statement is used to check this condition.

The reduction rate (rr) used for all cases was .9. Note, however, that this value of reduction rate poses a problem with K-brick input. With the cell size restricted to 1 mm by 1 mm by h, only integer mm multiples of the dimensions parameters will be accepted into K-brick (otherwise K-brick crashes). To make sure K-brick will not crash, after the reduction rate has been applied to the current iteration of input parameter values, the E-patch dimensions are rounded to the nearest mm[†]. Refer to Appendix F for the Matlab code used in the K-brick Taguchi optimization. Due to the rounding of E-patch parameters during Taguchi optimization, there is potential for wasting computation time after the iteration number has grown significantly. Wasting of computation time by the rounding of input parameters is not fully addressed in this thesis and remains a future improvement to the Matlab code.

The $|S_{11}|$ at the L1 and L2 frequencies versus the numbered iteration in Taguchi's method determing using the fitness function (3.14) is shown in Figure 3.6. Figure 3.7

[†]E-patch geometry values h and X_f can be integer values as they are independent of the cell size of the cavity and therefore not rounded in the matlab code.

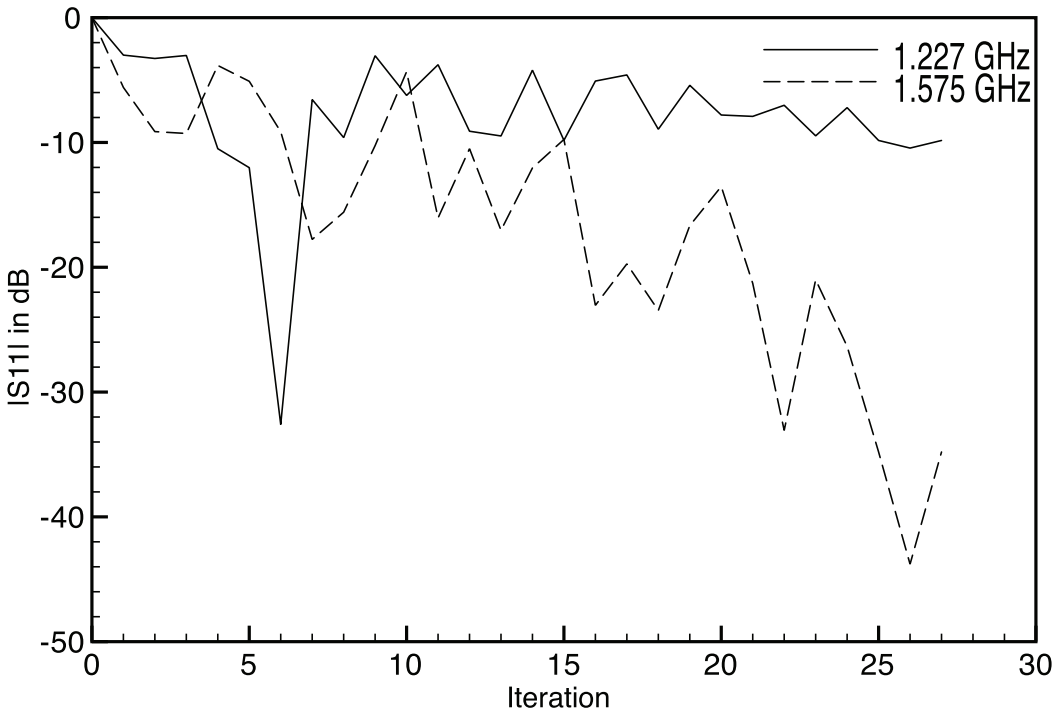

Figure 3.6: Simulated $|S_{11}|$ of an E-patch antenna versus iteration number in Taguchi's Method. Input parameter ranges are given in Table 3.9.

shows $|S_{11}|$ at the L1 and L2 frequencies versus the iteration number in Taguchi's method with a fitness function of (3.15). Comparing Figure 3.6 and Figure 3.7, it is seen that $|S_{11}|$ is different at all iteration numbers. The difference in $|S_{11}|$ is primarily because of the different fitness functions used. Comparing Figure 3.6 and Figure 3.7, $|S_{11}|$ seems to be greatly different between L1 and L2 when the fitness function (3.14) is used. This may imply that $|S_{11}|$ results from K-brick between 0 and -20 dB are more frequent than between 0 and -40 dB. In the fitness function optimizations using (3.15) and (3.14) the input range of parameters were slightly different, and could have also affected the values of $|S_{11}|$ for a simulated E-patch in K-brick. However, there is no conclusive evidence that the different input parameter ranges are a major factor in the different results shown in Figures 3.6 and 3.7. Another factor that may have effected both optimizations was the if-then statements used to correct for undesirable E-patch geometries. These if-then statements, however, executed under the same

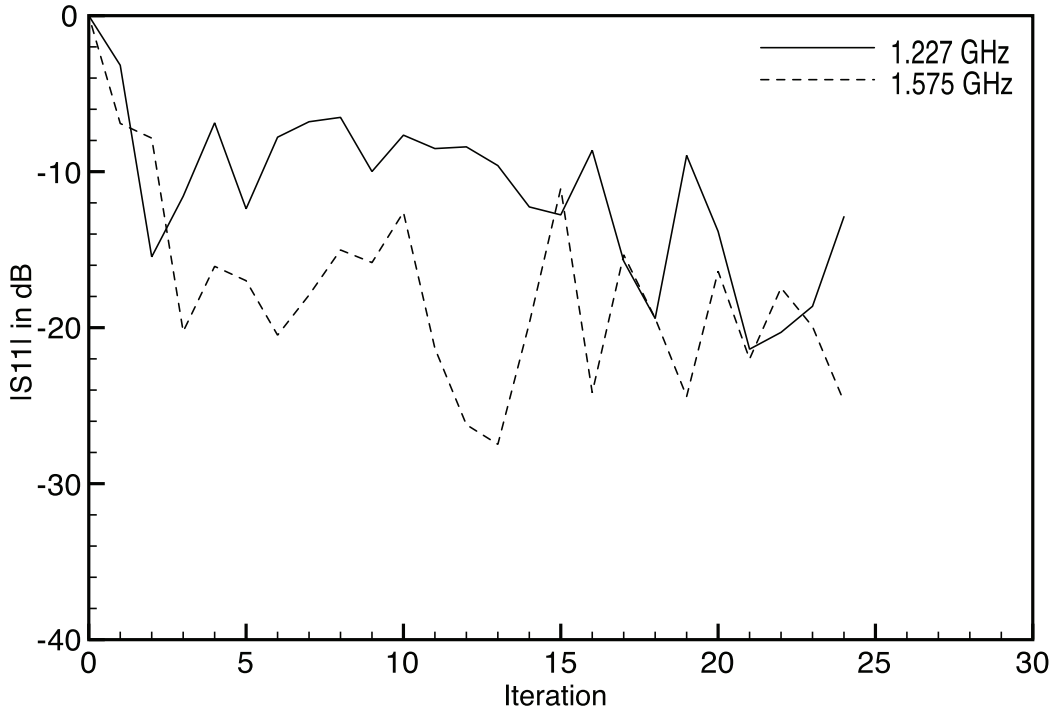

Figure 3.7: Simulated $|S_{11}|$ of an E-patch antenna versus iteration number in Taguchi's Method. Input parameter ranges are given in Table 3.10.

conditions for both optimizations possibly unaffecting the end optimization value.

The smallest value of $|S_{11}|$ in Figures 3.6 and 3.7 (iterations 26 and 21 respectively) have E-patch dimensions given in Tables 3.11 and 3.12 respectively. Figures 3.8 and 3.9 show $|S_{11}|$ versus frequency for E-patch antennas optimized in the Taguchi-K-brick program. In both Figures 3.8 and 3.9, a good result of -10 dB and below for the L1 and L2 frequencies is noted.

To provide a comparison simulation, HFSS was used to simulate the antenna with geometry values given in Table 3.12. Figure 3.10 shows $|S_{11}|$ versus frequency found by HFSS. From Figure 3.10 it is clear that the resonance points are the nearly the same. However, $|S_{11}|$ is about approximately 6 dB higher. This could be due to differences of probe modeling in HFSS and K-brick.

Refer to Appendix G for a matlab K-brick-Taguchi code that optimizes a simple cavity-backed rectangular patch antenna in K-brick.

Table 3.11: Dimensions of a cavity-backed E-patch antenna optimized by Taguchi-K-brick code. Input parameter ranges used to generate the geometry values are given in Table 3.9.

E-patch Dimension	Value in mm
L	96
W	81
X_f	31.7
Y_f	48
L_s	71
W_s	8
P_s	11
h	15
C_x	200
C_y	200

Table 3.12: Dimensions of a cavity-backed E-patch antenna optimized by Taguchi-K-brick code. Input parameter ranges used to generate the geometry values are given in Table 3.10.

E-patch Dimension	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200

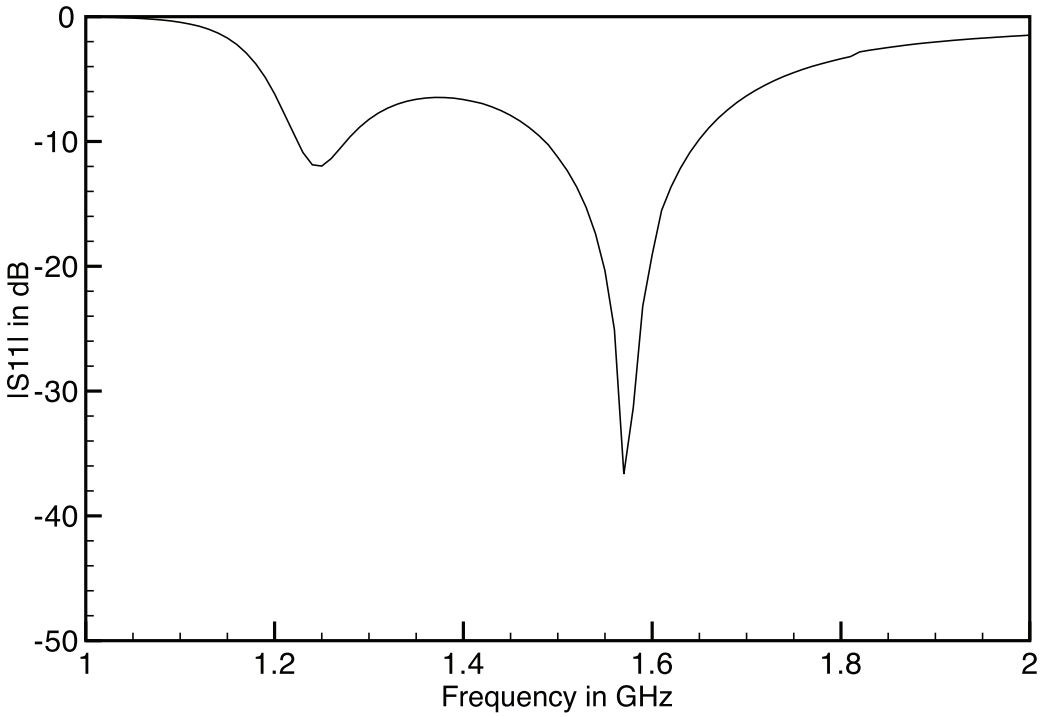

Figure 3.8: Simulated $|S_{11}|$ of an E-patch antenna optimized in K-brick.. Geometry values are given in Table 3.11.

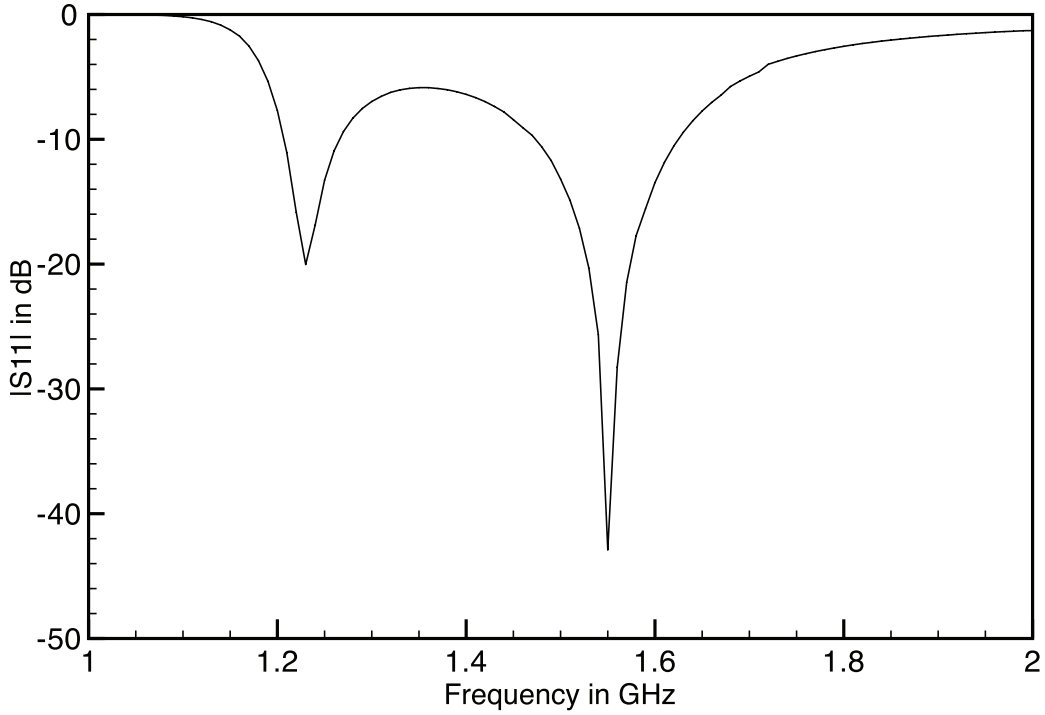

Figure 3.9: Simulated $|S_{11}|$ of an E-patch antenna optimized in K-brick. Geometry values are given in Table 3.12.

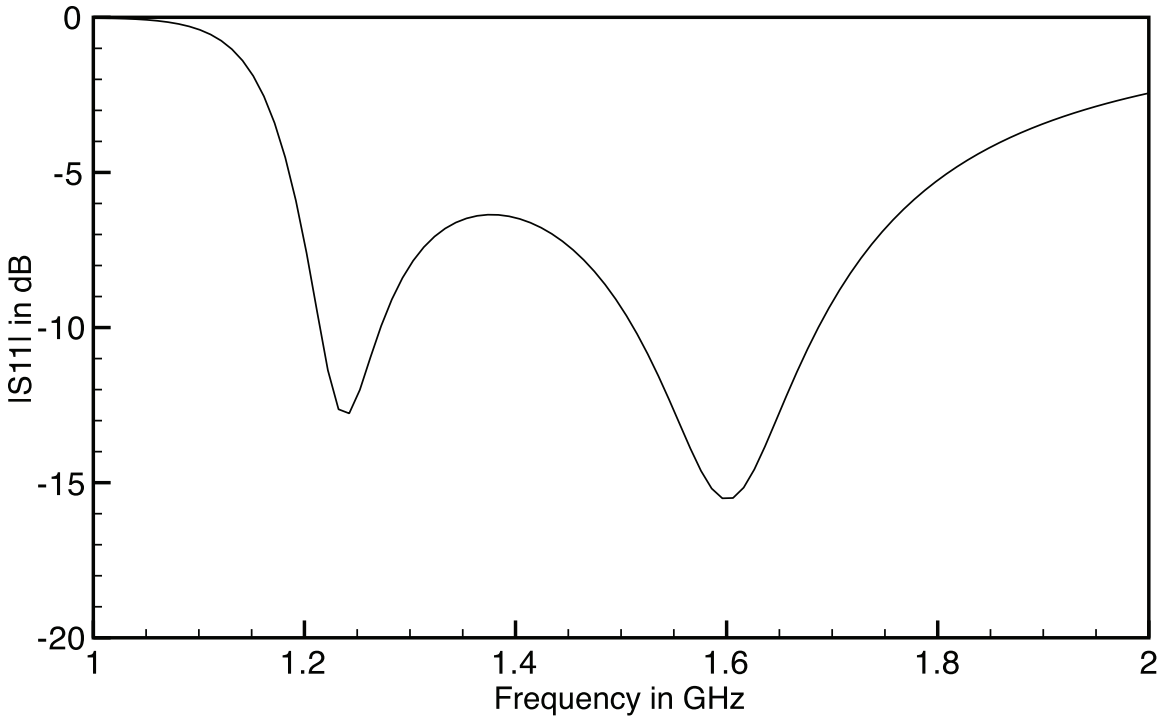

Figure 3.10: Simulated $|S_{11}|$ of an E-patch antenna optimized in HFSS. Geometry values given in Table 3.12.

Chapter 4

Experimental Results

4.1 Rectangular cavity-backed E-patch antenna

An experimental rectangular cavity-backed E-patch antenna was built to validate simulations done in HFSS. Figures 4.2 through 4.6 show detailed pictures of the experimental antenna. Figure 4.7 shows a top and side view depiction of the E-patch antenna with dimension variables. The ground plane, as shown in Figure 4.2, was constructed out of a corrugated rectangular cardboard box. The cardboard box length, width, and depth was 54 cm by 69 cm by 11.2 cm. A hole was cut in the center of the rectangular box in order to place the dielectric cavity. The hole's length and width dimensions are the same as those of the dielectric cavities (20 cm by 20 cm). The cavity and experimental antenna's dimensions are provided in Table 4.1. In order to have a good conductor for the ground plane in the experiment, aluminum tape was adhered to the surface of the rectangular cardboard box. The hole cut out for containment of the cavity has a small thickness in which copper tape was wrapped around; refer to Figure 4.5 for a picture of this thickness covered with aluminum tape. The dielectric cavity material was a rectangular Styrofoam piece cut to the dimensions of the cavity given in Table 4.1. To cut the Styrofoam to the proper

dimensions, a Styrofoam cutter was used; Figure 4.9 shows the Styrofoam cutting tool. Styrofoam's μ and ϵ material properties are nearly identical to a free-space air dielectric and for experimental construction purposes Styrofoam is more practical. The Styrofoam cavity's outer and bottom walls had copper tape adhered to them to provide a good ground plane conductor. The E-patch was constructed with four pieces of copper tape. The copper tape used to assemble the E-patch was cut to size using a standard exact-o knife. The E-patch dimensions are provided in Table 4.1. The probe feed, as shown in Figure 4.6, is a female SMA connector and is placed at the proper position (X_f, Y_f) as depicted in Figure 4.7. The non-threaded end of the female SMA connector has a wire soldered to it. The lead was cut to the dielectric height h. The non-threaded end of the female SMA connector (lead soldered and cut to h) pierces the copper tap adhered to the bottom surface of the cavity and through the copper tape conductor of the E-patch antenna. Once the lead end was through the dielectric cavity and the E-patch, it was soldered to the copper tape (E-patch copper tape).

Table 4.1: Dimensions of the experimental rectangular cavity-backed E-patch antenna.

E-patch and cavity dimension	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200

The experiment to measure $|S_{11}|$ versus frequency was conducted using a Hewlett Packard 8753D network analyzer. The power setting was 10 dBm, averaging set

to 4, IF Bandwidth set to 3700 KHz, the start and stop frequencies set to 1 and 2 GHz. The Hewlett Packard 8753D was calibrated for $|S_{11}|$ using the 3.5 mm SMA connector Agilent test calibration set 8753D. The experimental antenna was orientated perpendicular to the ground and rested on a 3 foot block of Styrofoam.

Radiation gain measurements of the E-patch antenna were conducted at 1300 MHz and 1640 MHz in the anechoic chamber located at the engineering building at Michigan State University. Unfortunately, the measurement setup used is not calibrated for gain measurements and a reference antenna with a known gain should be used to determine the true gain of the E-patch antenna. This is a fairly simple procedure. For instance, if the gain of the reference antenna is known to be G_{ref} and the uncalibrated measurement produces a gain value G_{ref_un} and if the uncalibrated measurement of the E-patch antenna gain is measured to be G_{e_un} then, the true gain of the E-patch antenna G_e is determined using

$$G_e = G_{ref} \frac{G_{e\_un}}{G_{ref\_un}}. \tag{4.1}$$

In our case, the reference antenna used is a broad band pyramidal horn* and its gain is evaluated using a two identical antenna gain measurement approach. In this approach, the ratio of the power received (P_r) to the power transmitted (P_t) between two identical antennas placed a distance R away from each other is used in conjunction with the Friis equation given in (2).

$$\frac{P_r}{P_t} = \left(\frac{\lambda}{4\pi R}\right)^2 G_{0t} G_{0r} \tag{4.2}$$

From equation (4.2), λ is the operating wavelength, G_{0t} and G_{0r} are the gains of the transmitting and receiving antenna respectively. Since the two antennas are identical,

*The antenna specifications are at `http://www.cobham.com/media/101070/H-1734%20Data%20Sheet.pdf`

G_{0t}=G_{0r} and equation (2) reduces to equation (3).

$$\left(\frac{4\pi R}{\lambda}\right)^2 \frac{P_r}{P_t} = G_{ref} \tag{4.3}$$

In our setup, both transmitting and receiving antennas are connected to a network analyzer, and the ratio $\frac{P_r}{P_t}$ is the transmission coefficient ($|S_{21}|^2$) measured by the network analyzer. Note that the network analyzer needs to be calibrated at the end of the cables connecting the two antennas in order to remove losses from the cables.

Figure 4.1 is a screen capture of the HFSS simulation used to compare with the experimental rectangular cavity-backed E-patch antenna. The HFSS settings are listed in Table 4.2. Figure 4.11 is the frequency response data for the simulation of the experimental antenna.

Table 4.2: HFSS settings for the rectangular cavity-backed E-patch antenna.

HFSS setting	HFSS setting value(s)
Frequency of solving	1.4 GHz
Adaptive passes	10
Maximum Delta S	.01
Sweep Type	Fast
Frequency Set up	linear step 1 to 2 GHz step size .01
Initial mesh options	do lambda refinement .15
Solution Options	First order iterative solver .001
Excitation	Waveport at the coaxial probe feed
Port Renormalization	50 Ω
Deembed setting	checked -15.6 mm

To understand gain measurements in HFSS realized gain needs to be defined. Realized gain in HFSS is

$$\text{realized gain} = 4\pi \frac{U}{P_{incident}}. \tag{4.4}$$

where U is the radiation intensity in watts per steradian in the direction specified,

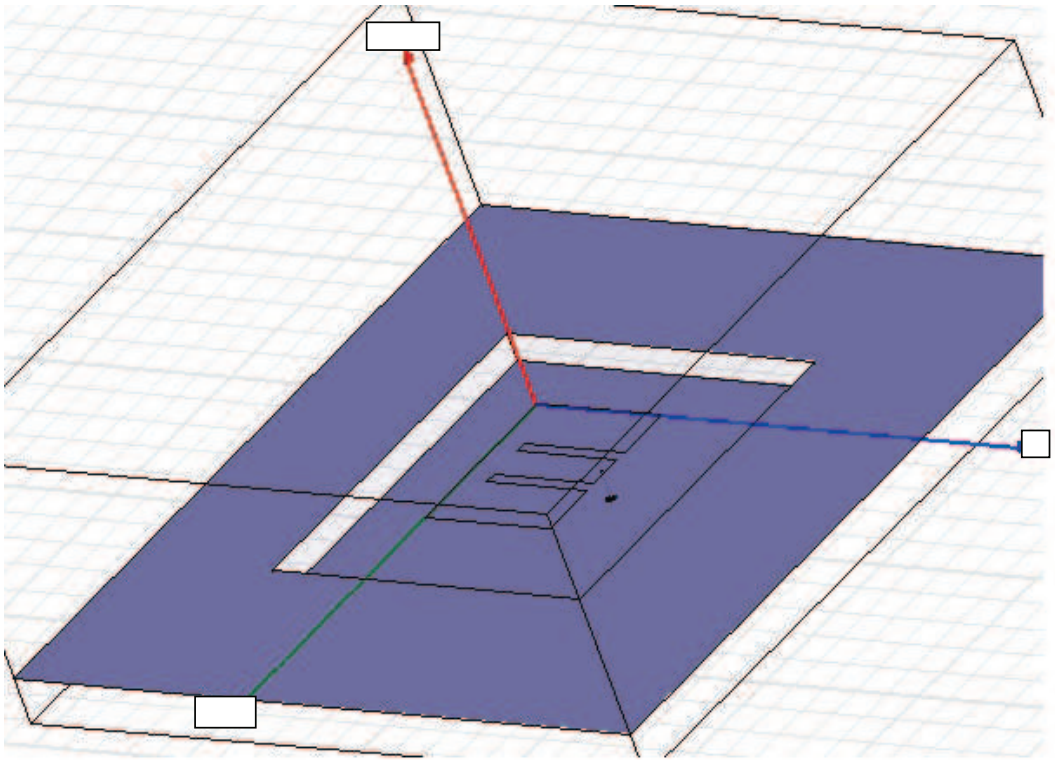

Figure 4.1: A screen capture of the rectangular cavity-backed E-patch antenna in HFSS.

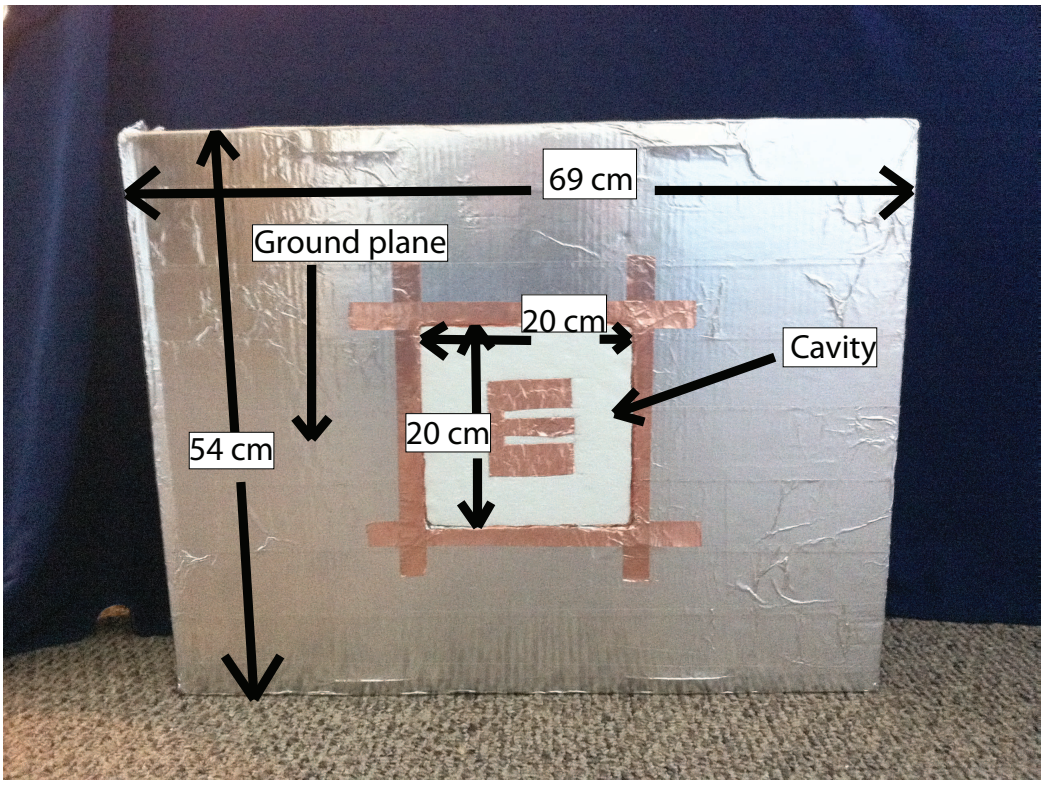

Figure 4.2: A front view of the rectangular cavity-backed E-patch experimental antenna.

and $P_{incident}$ is power associated with the conjugate match of the accepted power $P_{accepted}$. In HFSS $P_{accepted}$ is defined as,

$$P_{accepted} = Re \int_{A} \left(\vec{E} \times \vec{H}^{*} \right) . d\vec{S} \tag{4.5}$$

Where Re is the real part of a complex number, A is the union of all port boundaries in the model (including wave and lumped Ports, but excluding Floquet ports), $\vec{E}$ is the electric field and $\vec{H}^{*}$ is the conjugate of the magnetic field and $d\vec{S}$ is the local port-boundary unit normal directed out of the 3D HFSS model.

Figure 4.10 shows $|S_{11}|$ versus frequency measured on an Hewlett Packard 8753D network analyzer. Comparing Figure 4.10 with Figure 4.11 in Figure 4.12, $|S_{11}|$ is near in agreement with simulation and experiment. The bandwidth of the simulation

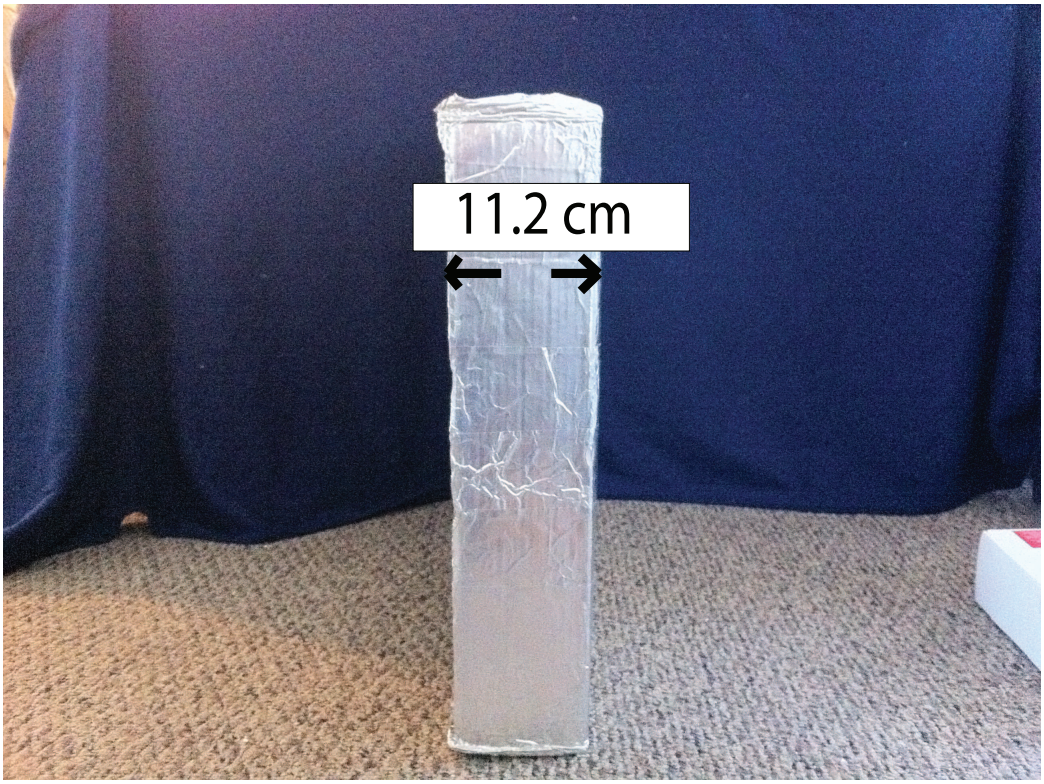

Figure 4.3: A side view of the rectangular cavity-backed E-patch experimental antenna.

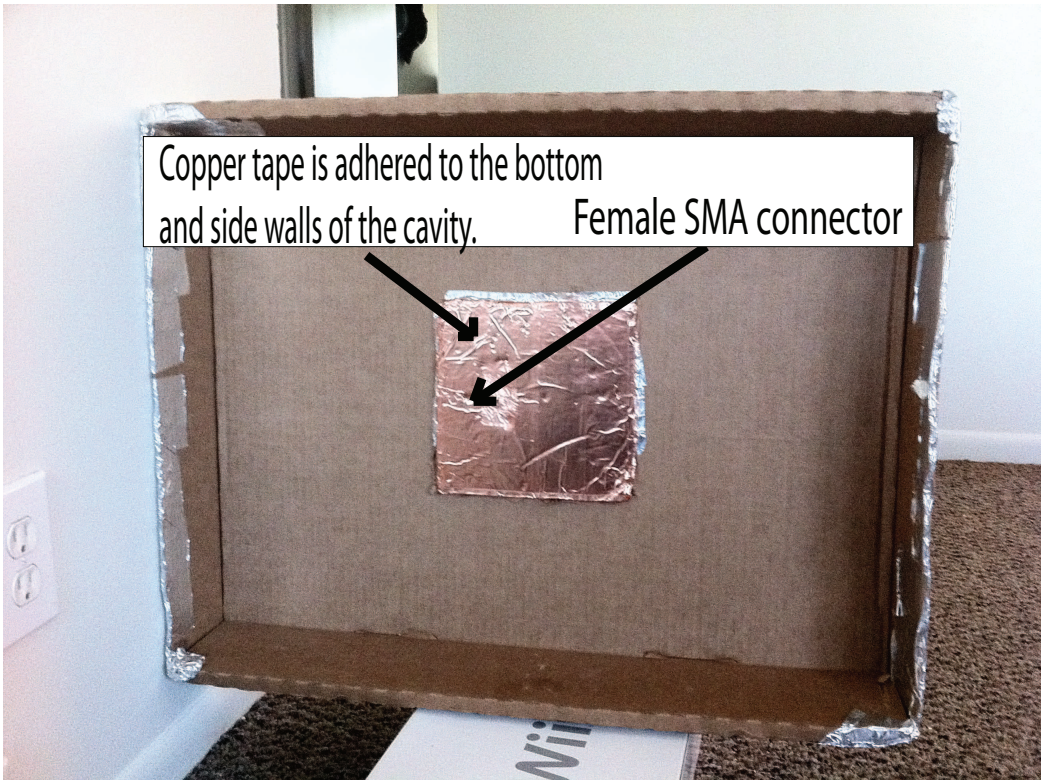

Figure 4.4: A back view of the rectangular cavity-backed E-patch experimental antenna.

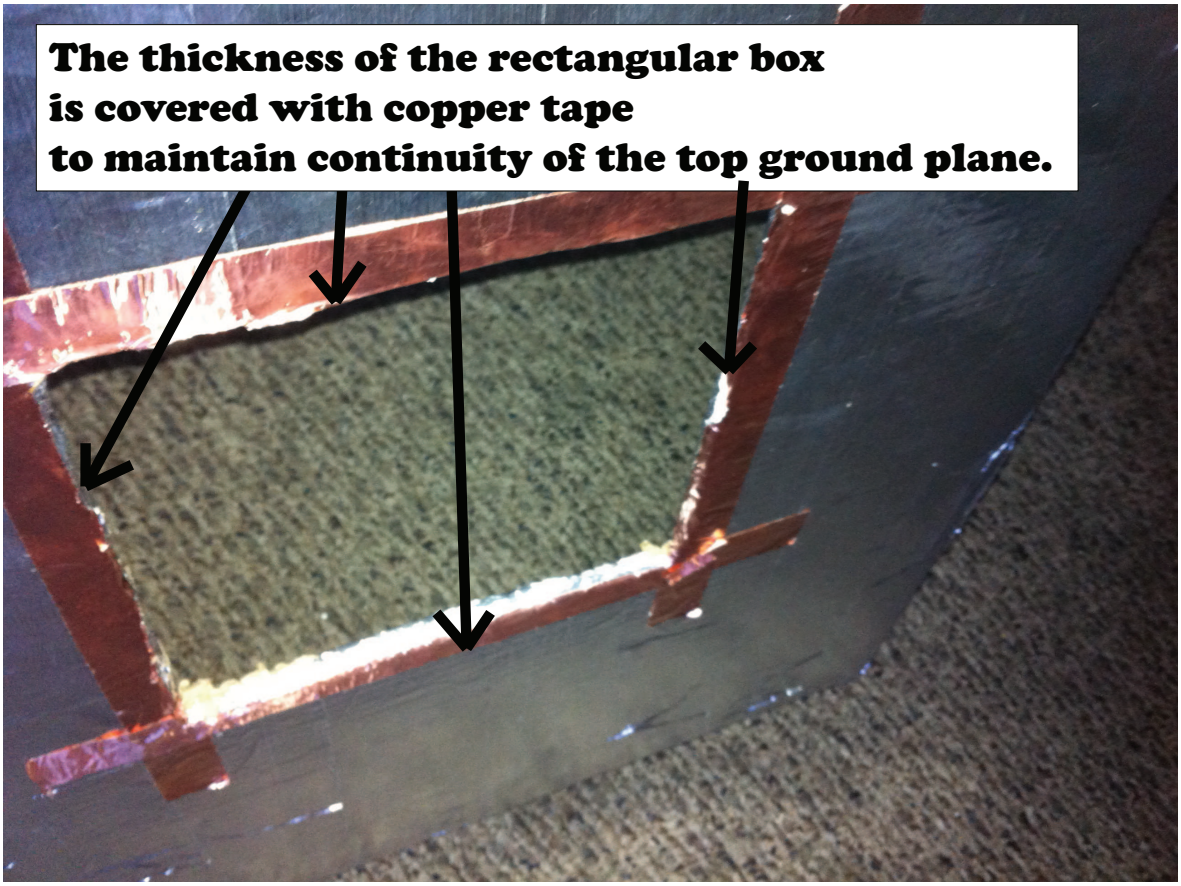

Figure 4.5: A picture of the copper tape covering up the thickness of the cardboard box top.

Figure 4.6: A close up picture of the feed point of the experimental cavity.

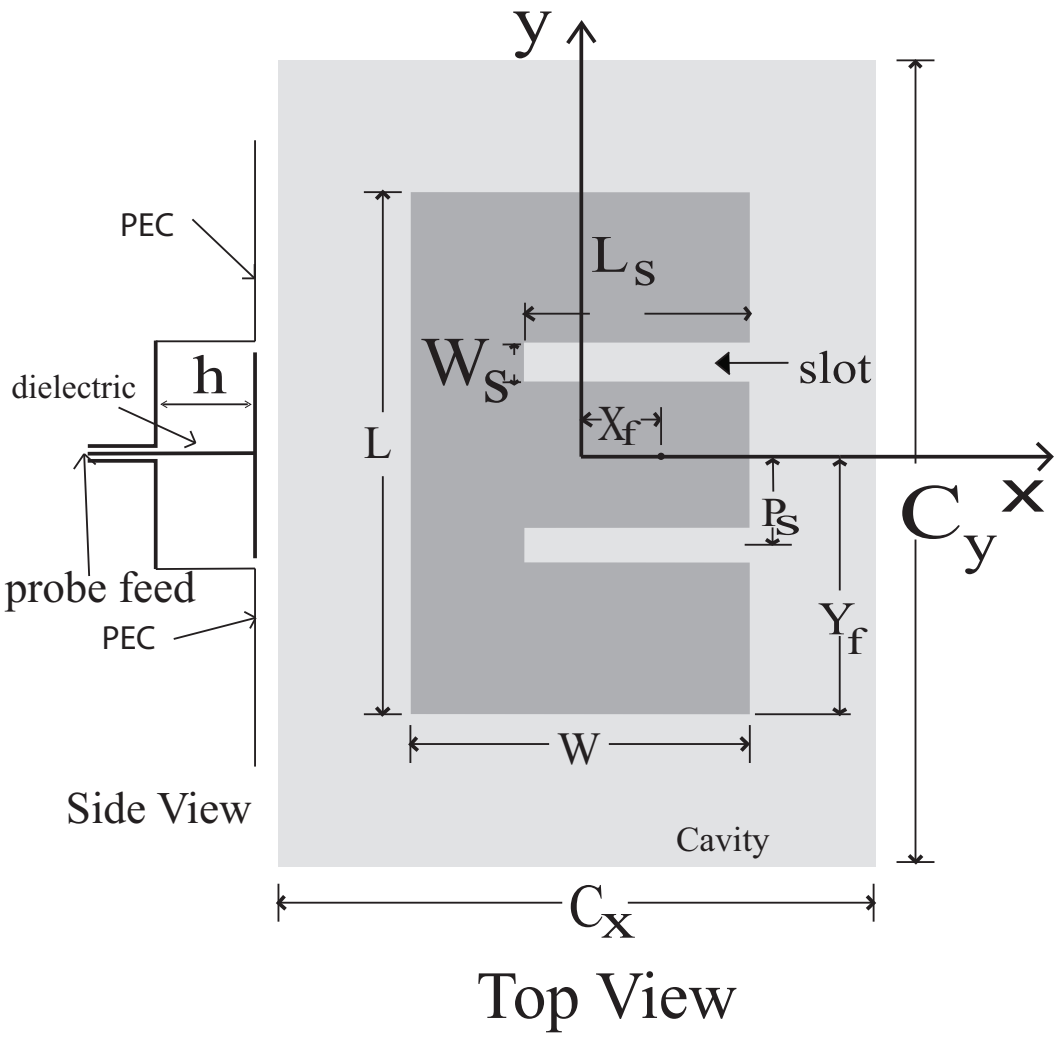

Figure 4.7: Top and side view depiction of an experimental E-patch in a cavity.

was approximately 243 MHz while the bandwidth of experimental antenna was 330 MHz. The return loss difference at both points of resonance (1.3 and 1.64 GHz) and bandwidth differences may be attributed to differences between the constructed antenna and the HFSS model.

Refer to Figure 4.8 for a graphical depiction of the coordinate system used to describe the radiation patterns in HFSS and for the experimental antenna. For the Z-X plane case, $\phi = 0^{\circ}$ and θ is varying from -180 to 180 degrees. -180 degrees to 0 degrees refers to negative x values and 0 to 180 degrees refers to positive x values. For the Z-Y plane case, $\phi = 90^{\circ}$ and θ is varying from -180 to 180 degrees. -180 degrees to 0 degrees refers to negative y values and 0 to 180 degrees refers to positive y values. Figure 4.13 and Figure 4.14 are HFSS simulated radiation patterns of the experimental antenna. The realized gain for the experiment is approximately 10 dB in the forward broadside direction of the antenna, indicating that the antenna transmitting capabilities in that direction are good. Figure 4.15 and 4.16 are the radiation gain measurements of the experimental antenna, and the gain was found using (4.1). The gain at 0 degrees in Figure 4.16 is about 10 dB and about 7 dB in Figure 4.15. Comparing Figure 4.13 and Figure 4.16 in Figure 4.17, the simulated results match reasonably well with the experiment. Differences could be attributed to human construction of the experimental antenna. Comparing Figure 4.14 and Figure 4.15 in Figure 4.18, the simulated results did not match well with the experiment. Differences could be attributed to human construction of the experimental antenna and HFSS's radiation pattern accuracy.

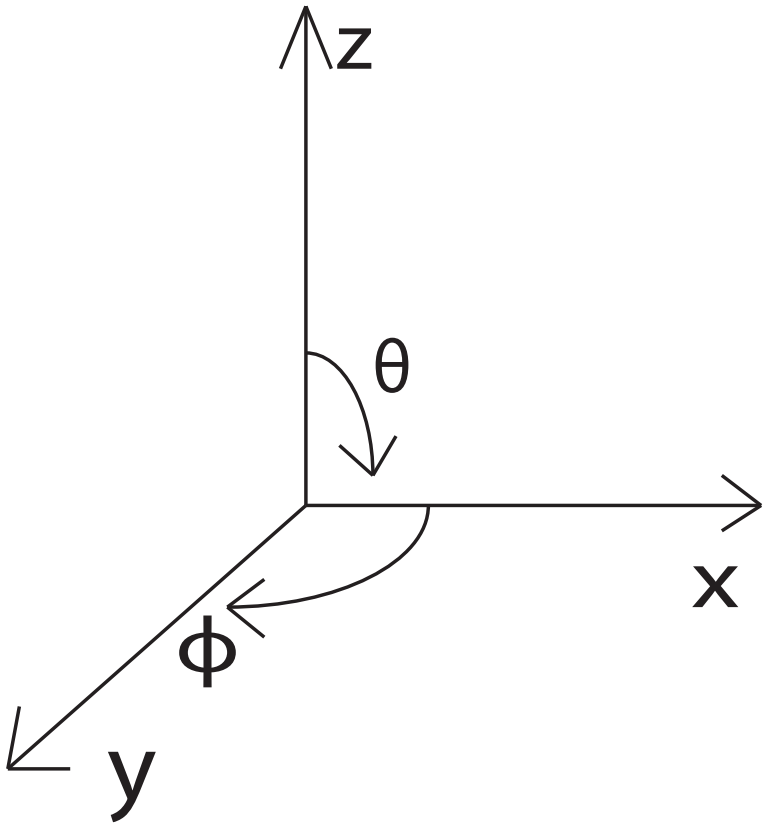

Figure 4.8: Coordinate system for the radiation patterns.

Figure 4.9: A picture of the Uchida 3500 Super Hot-wire Foam-Cutter Crafting Tool used to cut the cavity to the proper dimensions.

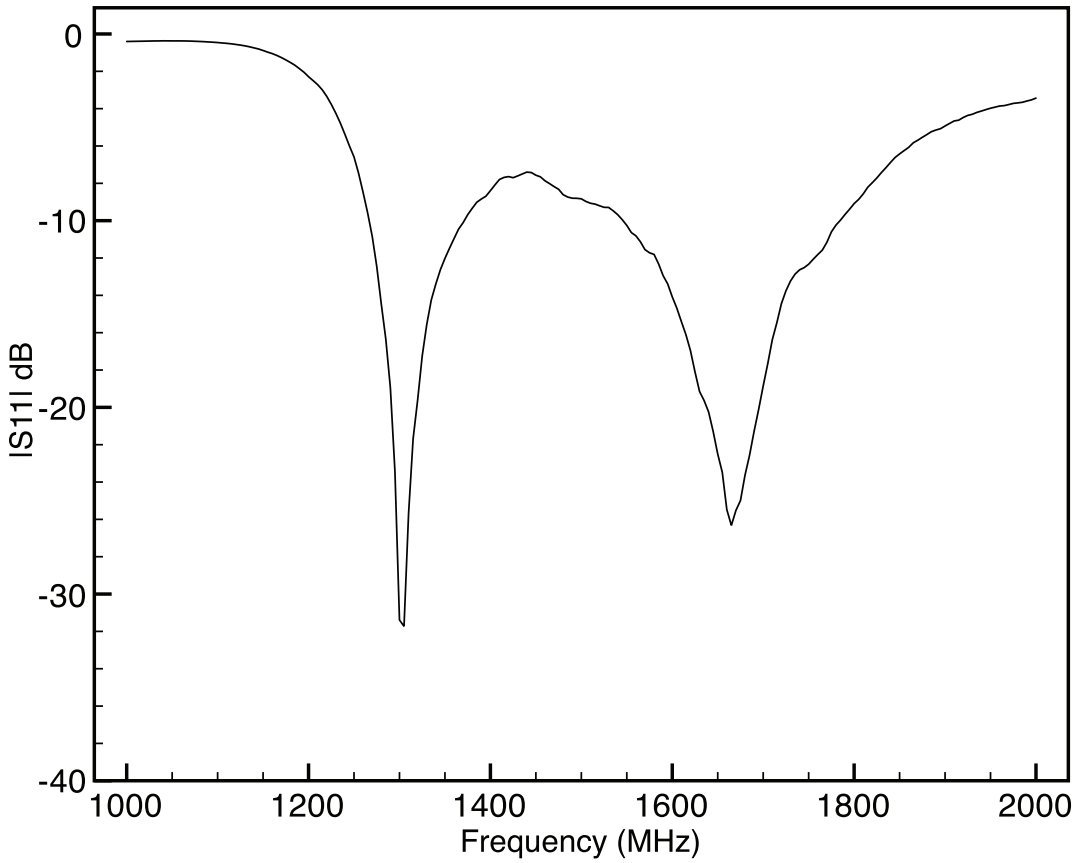

Figure 4.10: $|S_{11}|$ versus frequency measured on a Hewlett Packard 8753D network analyzer for experimental rectangular cavity-backed E-patch antenna. Dimensions provided in Table 4.1.

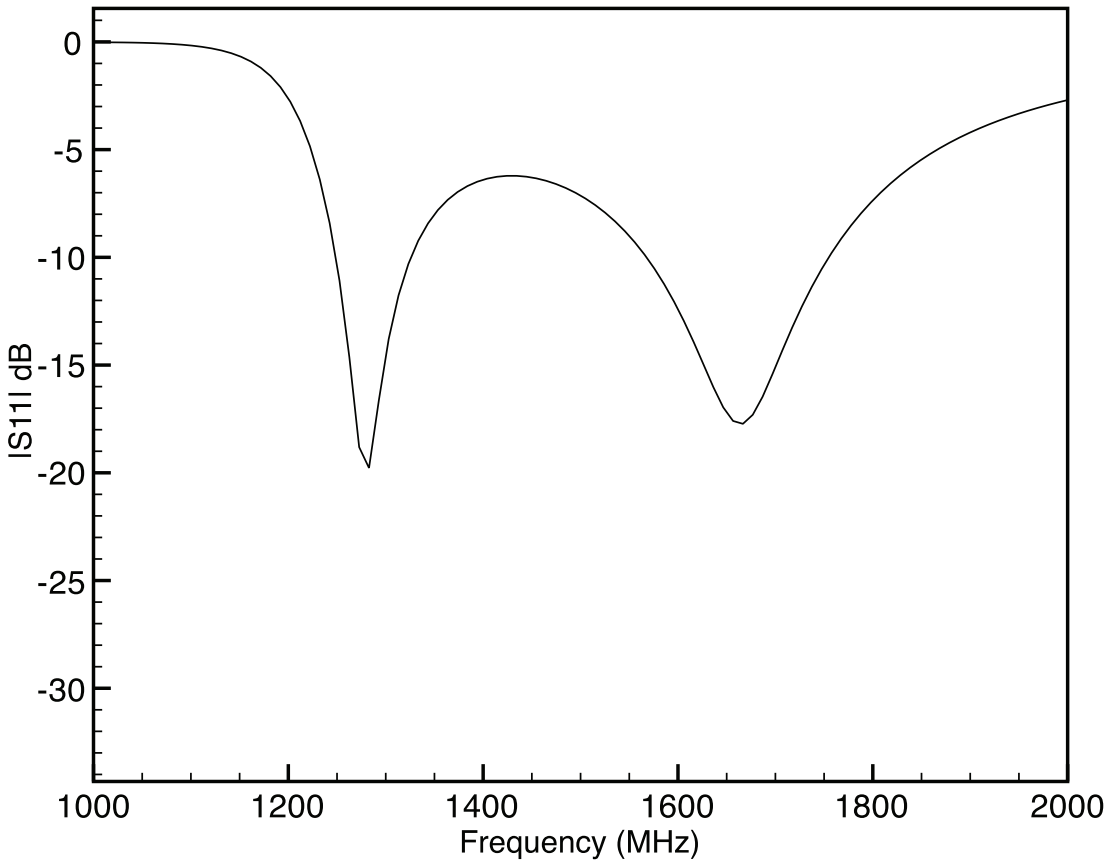

Figure 4.11: $|S_{11}|$ versus frequency of a simulated rectangular cavity-backed E-patch antenna. Dimensions provided in Table 4.1.

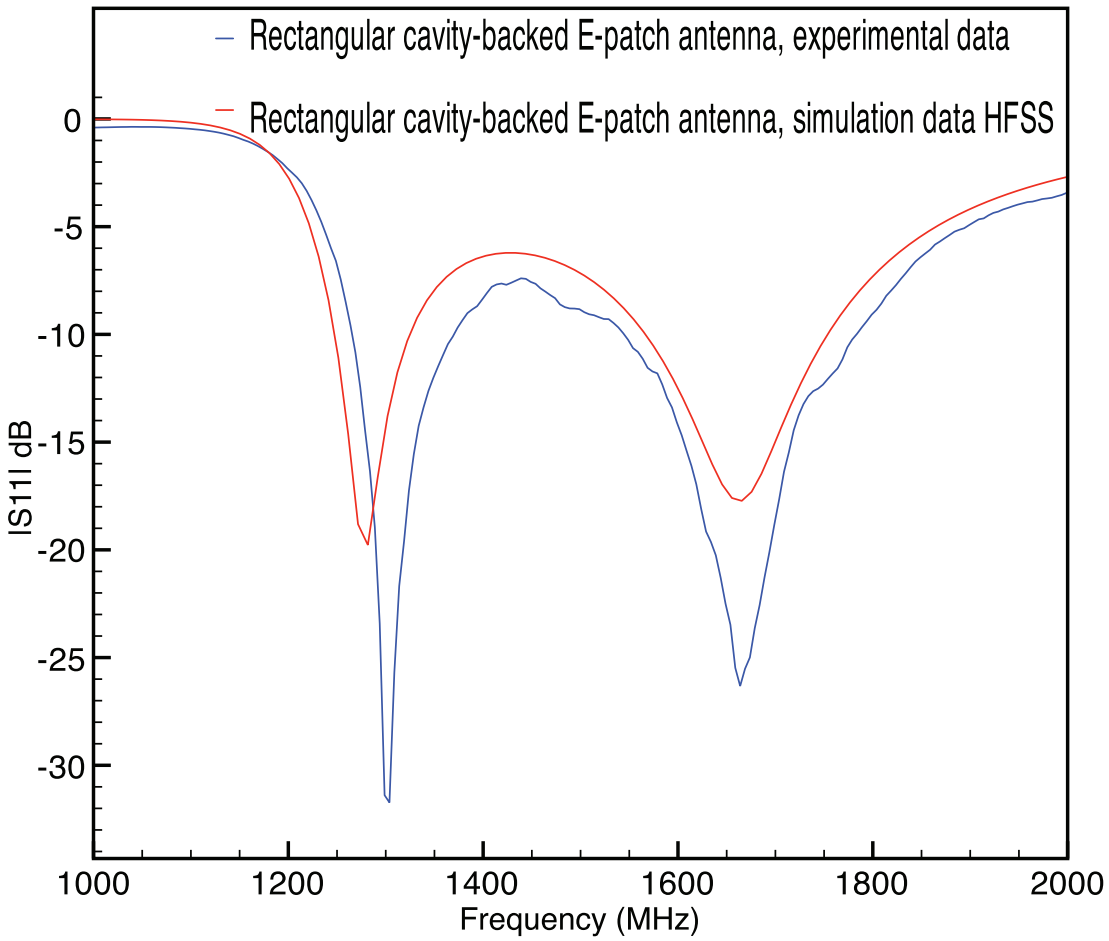

Figure 4.12: $|S_{11}|$ versus frequency for experimental and simulated rectangular cavity-backed E-patch antenna. Dimensions provided in Table 4.1.

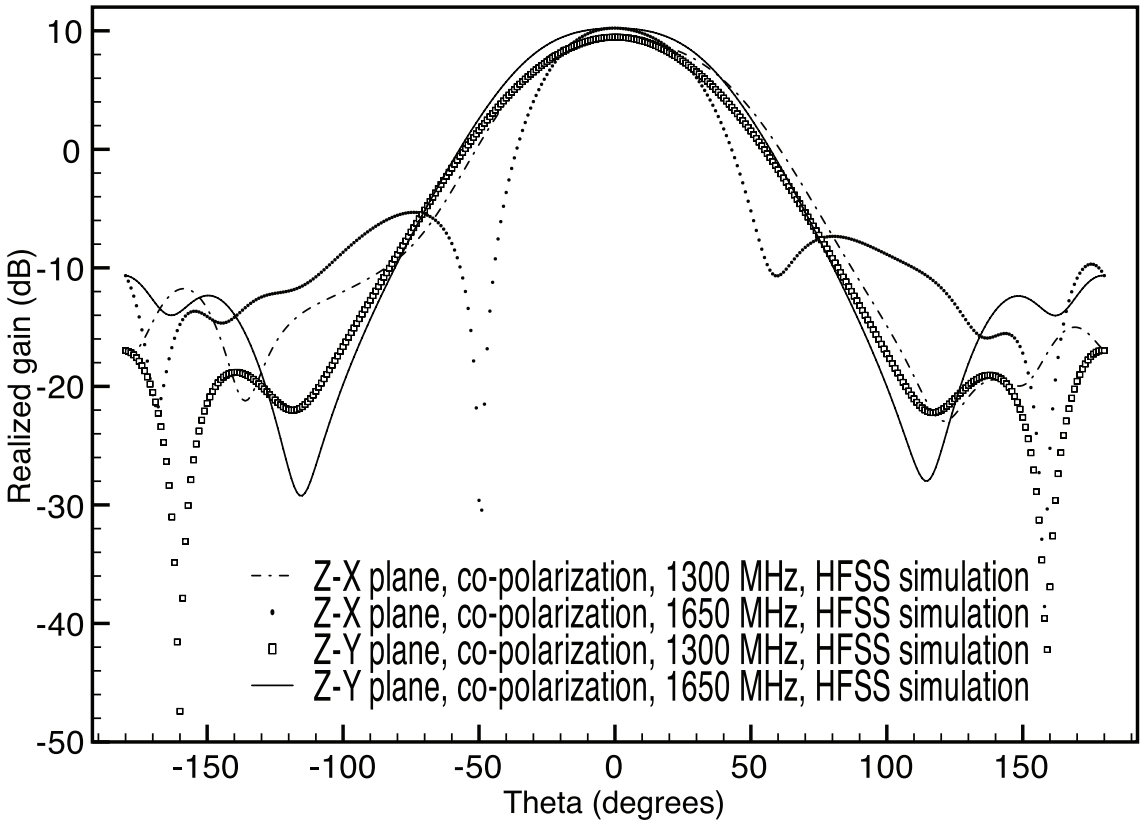

Figure 4.13: HFSS gain radiation pattern of the simulated cavity-backed E-patch antenna, co-polarization.

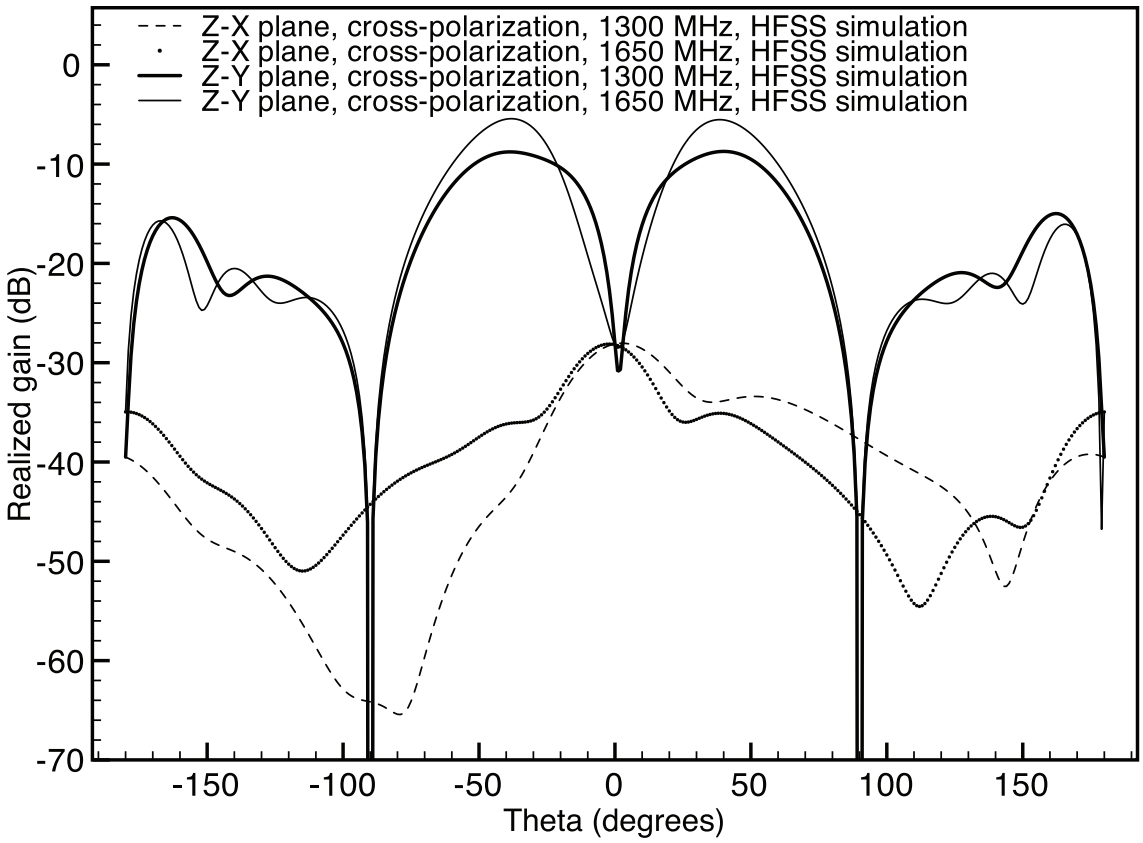

Figure 4.14: HFSS gain radiation pattern of the simulated cavity-backed E-patch antenna, cross-polarization.

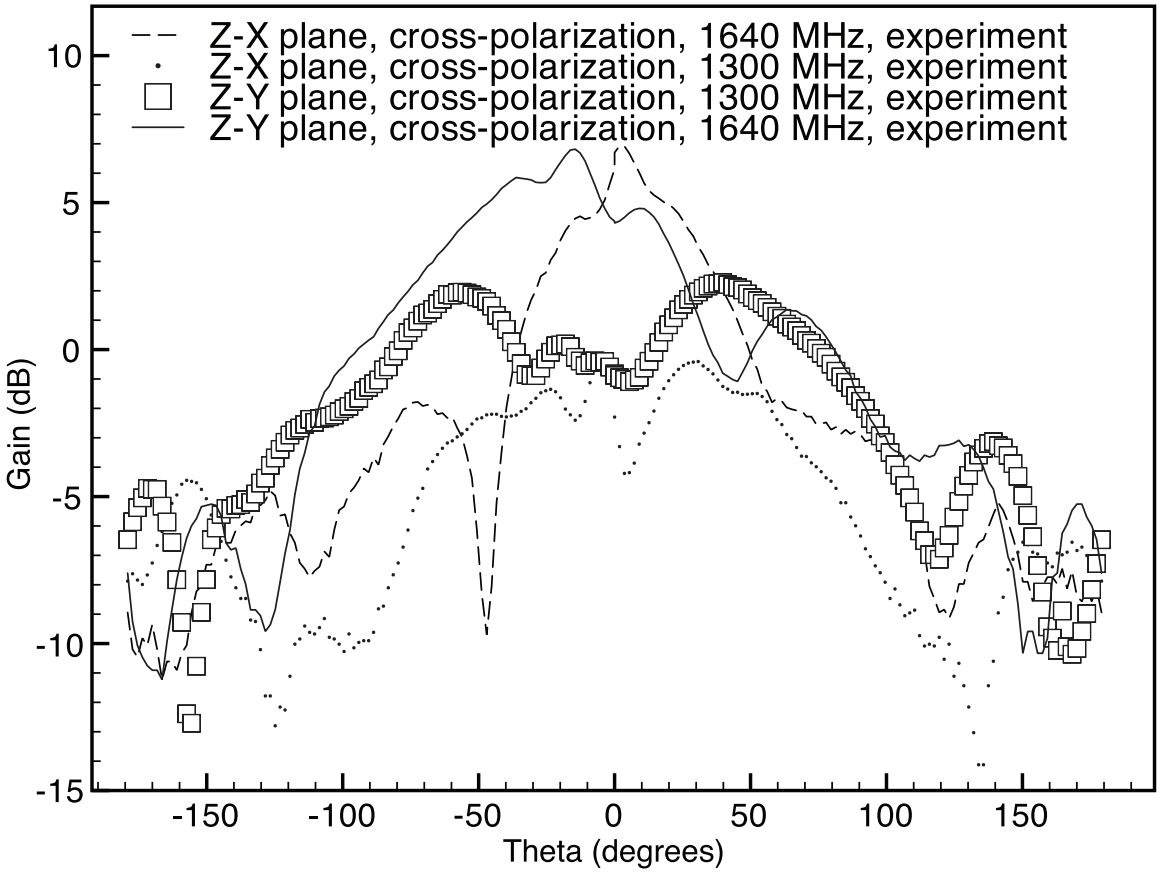

Figure 4.15: Gain radiation pattern of the experimental rectangular cavity-backed E-patch antenna, cross-polarization.

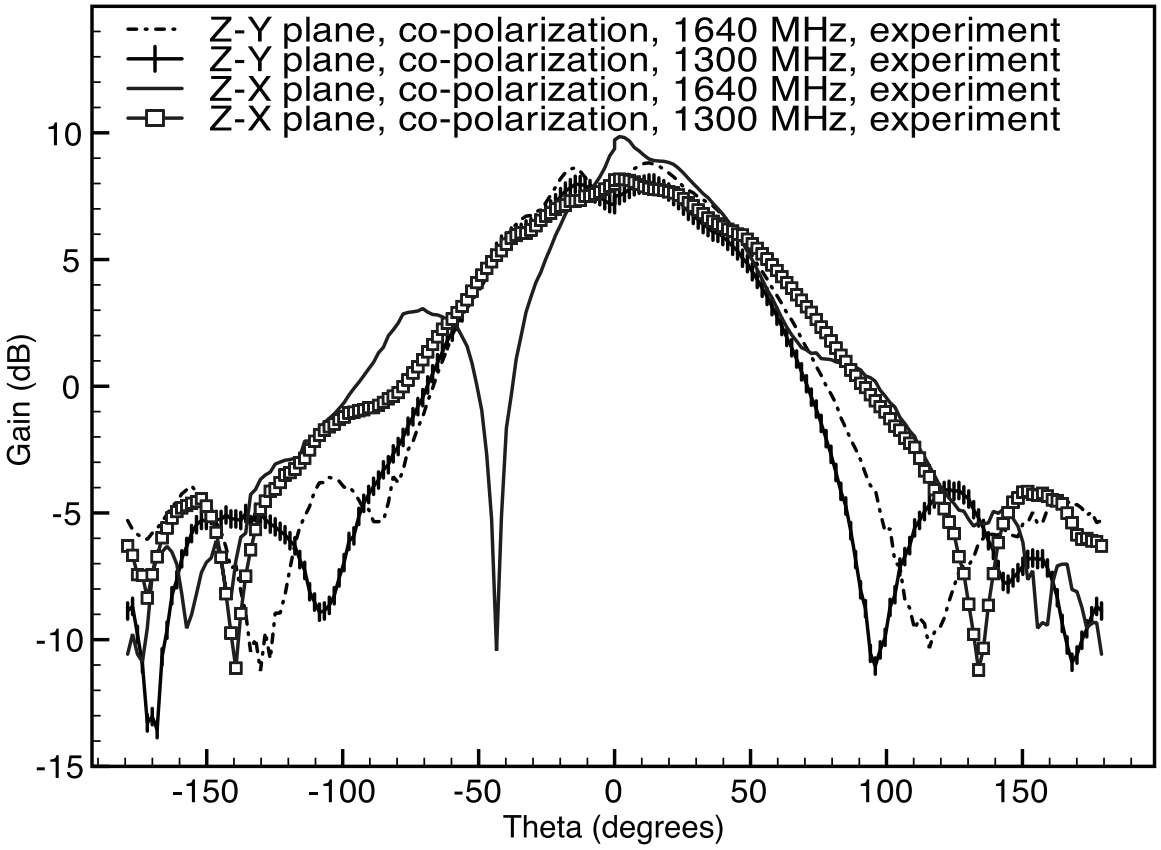

Figure 4.16: Gain radiation pattern of the experimental rectangular cavity-backed E-patch antenna, co-polarization.

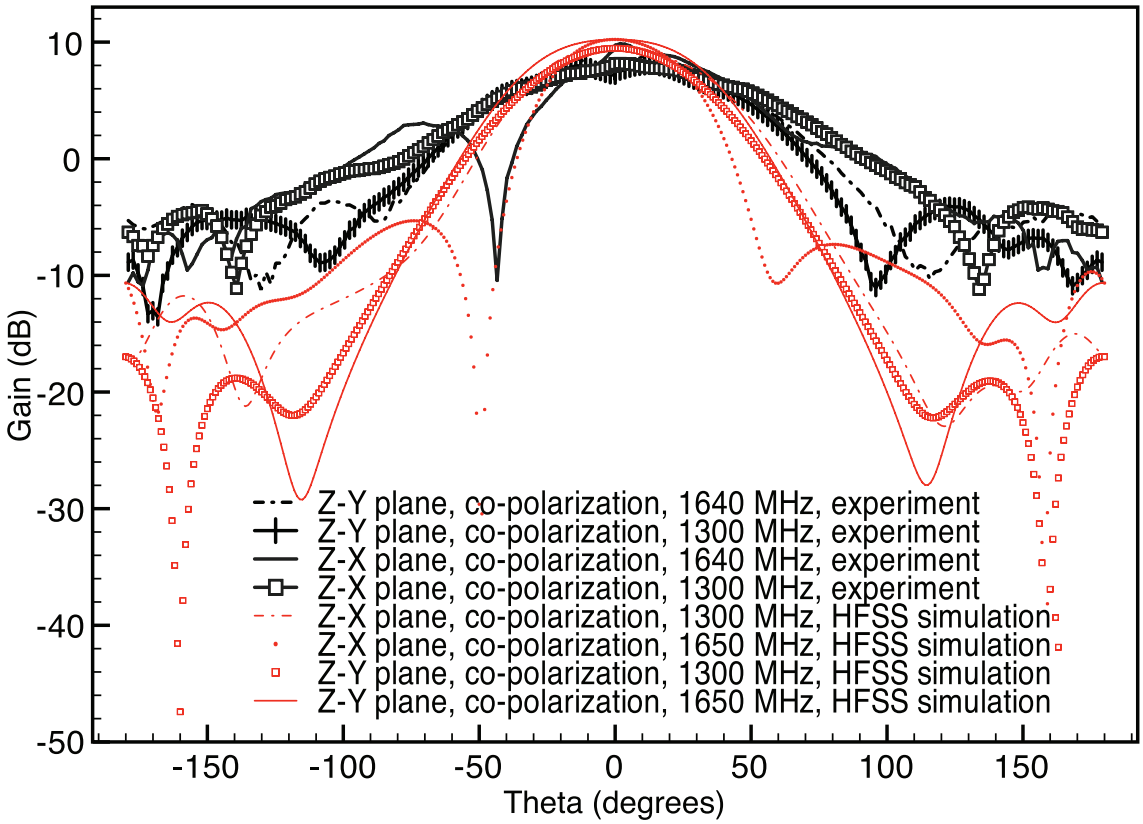

Figure 4.17: Gain radiation patterns of the experimental and simulated rectangular cavity-backed E-patch antenna, co-polarization.

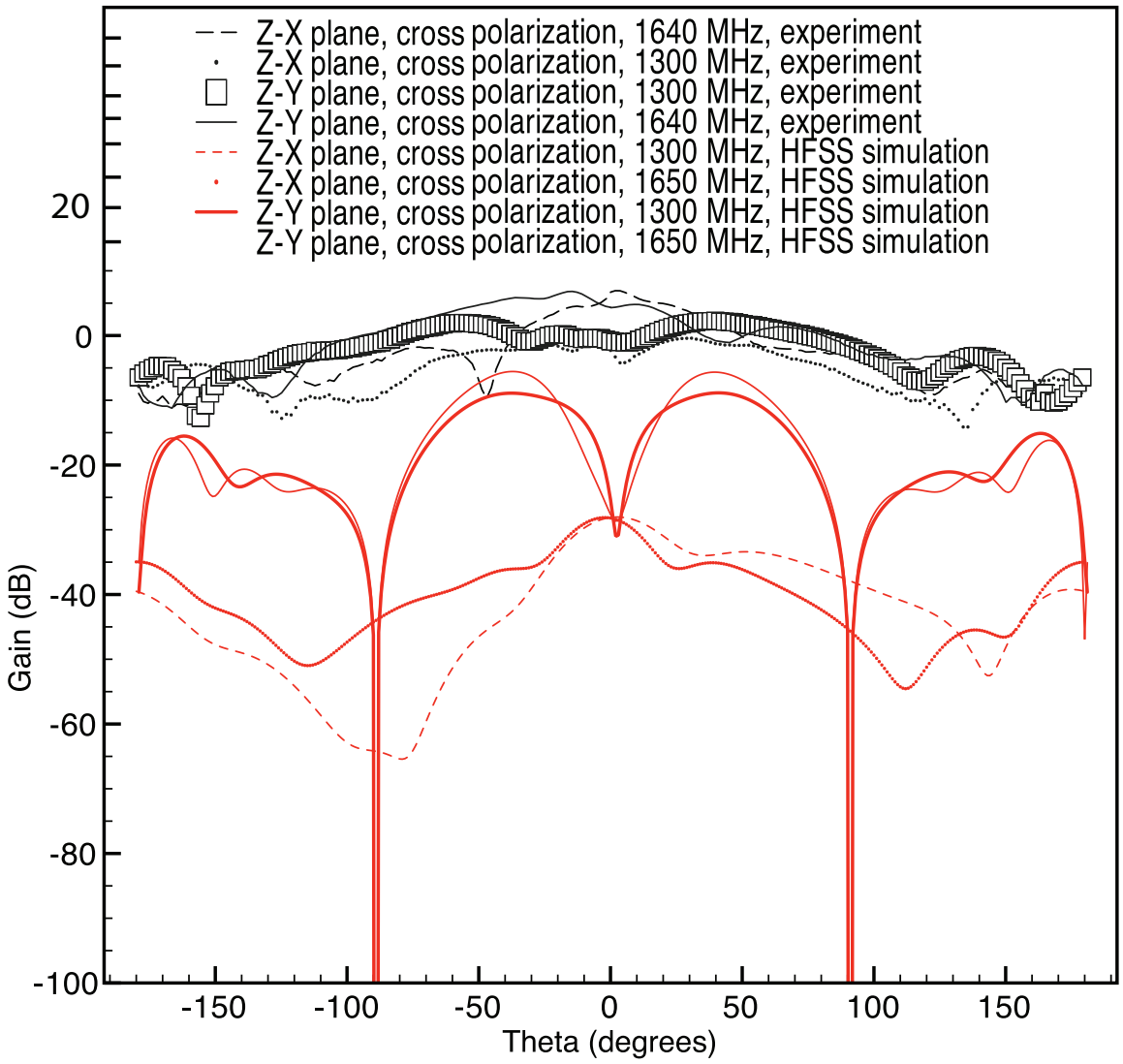

Figure 4.18: Gain radiation patterns of the experimental and simulated rectangular cavity-backed E-patch antenna, cross-polarization.

4.2 Cylindrical conformal cavity-backed E-patch antenna

A experimental cylindrical conformal cavity-backed E-patch antenna was built to compare and verify simulation results in HFSS. Comparing with results from the experimental rectangular cavity-backed E-patch antenna presented in section 4.1, the effects of curvature can be investigated.

Using the standard cylindrical coordinate system where $\rho = \sqrt{x^2 + y^2}$, $\phi = tan^{-1}\frac{y}{x}$ and $z = z$ (x, y, and z are variables in the rectangular coordinate system), one can develop a topology for the cylindrically conformal cavity-backed E-patch antenna. Figure 4.19 and Figure 4.20 show conceptual depictions of a experimental cavity-backed E-patch conformed to a cylinder. In order to analyze how the curvature of the cylinder changes the antenna's performance, as compared to the planar cavity-backed case, the E-patch dimensions need to stay consistent between both cases. Therefore, the width dimensions of the E-patch (W and L_S) will be specified using ϕ, at a distance ρ from the z axis. Further, the arc length along the circumference of the cylinder is

$$g = \rho\phi. \tag{4.6}$$

The variable g in (4.6) is the E-patch width dimension W or L_S. The length dimensions of the E-patch (L, W_S, and P_S) will be along the z axis. To keep the cavity in the same configuration as the rectangular cavity-backed case, the dielectric height h will be specified along ρ, C_x will be specified by a ϕ and C_y will be specified along the z. The probe feed (X_f, Y_f) will be placed consistent with that of the E-patch. In other words X_f will be an arc length according to (4.6) where $g = X_f$ and Y_f will be specified along the z axis.

Figures 4.21 through 4.25 show detailed pictures of the experimental antenna. As discussed earlier, Figures 4.19 and 4.20 show depictions of the cylindrically conformal

cavity-backed E-patch antenna. The ground plane, as shown in Figure 4.21, was constructed out of a cardboard cylinder. The cardboard cylinder's height along the z-axis was 121.92 cm and radius ρ was 15.24 cm. Further E-patch and cavity dimensions for the experiment are provided for Table 4.3. Figure 4.26 shows the cardboard cylinder as it was purchased from Home Depot. A hole was cut in the center of the cardboard cylinder in order to place the dielectric cavity. The hole's length and height dimensions are the same as those of the dielectric cavity's (C_x by C_y). In order to have a good conductor for the ground plane, aluminum tape was adhered to the surface of the cardboard cylinder. The hole cut out for containment of the cavity has a small thickness in which aluminum tape was wrapped around; this was done to maintain continuity of the top ground plane to the outer and bottom walls of the dielectric cavity. The dielectric cavity material is made out of two pieces of Styrofoam as shown in Figure 4.23. To cut the Styrofoam pieces to the proper dimensions, a Styrofoam cutter was used; Figure 4.9 shows the Styrofoam cutting tool. The Styrofoam's μ and ϵ material properties are nearly identical to an free-space air dielectric and for experimental construction purposes it is more practical to use Styrofoam than air. The Styrofoam cavity's outer and bottom walls had copper tape adhered to them to provide a good ground plane conductor. The E-patch was constructed with four pieces of copper tape. The copper tape used to assemble the E-patch was cut to size using a standard exact-o knife. The probe feed, as shown in Figure 4.24, is a female SMA connector and is placed at the position (X_f, Y_f) as depicted in Figure 4.19. The non-threaded end of the female SMA conductor has a wire lead soldered to it. The lead was cut to the dielectric height h. The non-threaded end of the female SMA connector (wire lead soldered and cut to the length h) pierces the copper tap adhered to the bottom surface of the cavity and is placed in between both pieces of the Styrofoam cavity through the copper tape conductor of the E-patch antenna. Once the lead end was through the dielectric cavity and the E-patch, it was

soldered to the copper tape (E-patch copper tape).

Table 4.3: Dimensions of a experimental cylindrical conformal cavity-backed E-patch antenna.

E-patch and Cavity Dimension	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	65
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
f	509.6
ρ	154

The experiment for the $|S_{11}|$ versus frequency was conducted using a Hewlett Packard 8753D network analyzer for measurement. The Hewlett Packard 8753D is located in the Engineering Research Center, at MSU. The power setting was 10 dbm, the averaging set to 4, the IF Bandwidth set to 3700 KHz, and the start and stop frequencies were set to 1 and 2 GHz. The Hewlett Packard 8753D was calibrated for $|S_{11}|$ using the Female SMA connector Agilent test calibration set 8753D. The experimental antenna was orientated perpendicular to the ground and rested on a 3 foot block of Styrofoam.

Figure 4.27 and Figure 4.28 are screen captures of the HFSS simulation used to compare with the experimental cylindrically conformal cavity-backed E-patch antenna. The HFSS settings are listed in Table 4.4. Figure 4.30 is the frequency response data for the simulation of the experimental cylindrically conformal cavity-backed E-patch antenna.

As discussed in the previous section, the radiation reference gain for the measured

Table 4.4: HFSS settings for the cylindrically conformal cavity-backed E-patch antenna.

HFSS setting	HFSS setting value(s)
Frequency of solving	1.6 GHz
Adaptive passes	20
Maximum Delta S	.005
Sweep Type	Fast
Frequency Set up	linear step 1 to 2 GHz step size .02
Initial mesh options	do lambda refinement .3
Solution Options	First order
Excitation	Waveport at the coaxial probe feed
Port Renormalization	50 Ω
Deembed setting	checked -18 mm

experimental E-patch antenna frequencies is determined using equations (4.3) - (4.1)[†]. Appendix H contains the matlab file used to convert the gain measurements in the anechoic chamber at Michigan State University engineering building to true gain measurements.

Figure 4.29 is $|S_{11}|$ versus frequency measured on a Hewlett Packard 8753D network analyzer. Comparing Figure 4.29 with Figure 4.30 in Figure 4.31, $|S_{11}|$ versus frequency is similar. In Figure 4.31 there are two distinct points of resonance. However, the points of resonance are in different positions and $|S_{11}|$ is lower in the simulated case. The bandwidth of the simulated antenna is 320 MHz (-10 dB or lower at 1250 MHz to 1340 MHz and 1520 MHz to 1750 MHz). The bandwidth of the experimental antenna is 321 MHz (-10 dB or lower at 1254 MHz to 1310 MHz and 1520 MHz to 1785 MHz). The differences in $|S_{11}|$ versus frequency in Figure 4.31 may be due to construction techniques and an inability of HFSS to simulate the E-patch antenna exactly.

In order to achieve -10 dB and below for $|S_{11}|$ at the L1 and L2 frequencies, another experimental antenna was built. Figure 4.33 is the $|S_{11}|$ versus frequency for

[†]The frequencies of interested tested in the cylindrical case were 1.3 GHz and 1.64 GHz.

a experimental cylindrically conformal cavity-backed E-patch antenna. Dimensions are provided in Table 4.5. The bandwidth of the experimental antenna in Figure 4.33 is 273 MHz (-10 dB or lower at 1195 MHz to 1253 MHz and 1500 MHz to 1715 MHz). At the L1 and L2 frequencies $|S_{11}|$ was measured as -12 dB and -20 dB respectively.

Table 4.5: Dimensions of a experimental cylindrical conformal cavity-backed E-patch antenna.

E-patch and Cavity Dimension	Value in mm
L	96
W	83
X_f	37.5
Y_f	48
L_s	67.5
W_s	7
P_s	13
h	15.6
C_x	200
C_y	200
f	509.6
ρ	154

Figure 4.32 shows the experimental $|S_{11}|$ versus frequency of the cylindrically conformal and rectangular cavity-backed E-patch antennas. The bandwidth of the experimental rectangular cavity-backed patch antenna is 330 MHz (-10 dB or lower at 1270 MHz to 1370 MHz and 1550 MHz to 1780 MHz). The bandwidth of the experimental cylindrically conformal cavity-backed E-patch antenna is 321 MHz (-10 dB or lower at 1254 MHz to 1310 MHz and 1520 MHz to 1785 MHz). The effect of curvature of the experimental antennas indicates a minor loss in bandwidth and a slightly lower shift in frequencies of the main points of resonance. However, the simulation of the experiments shown in Figures 4.31 and 4.12 seem to be good results verifying HFSS operation. The effect of curvature may have changed mode two's operation of the antenna and contributed to the loss in bandwidth and shift in resonant frequency. Comparing the simulation and experiment bandwidths, the

difference may be due to construction techniques of the experimental antennas and an inability of HFSS to simulate the E-patch antenna exactly.

Refer to the blue arrows in Figure 4.19 for the ϕ and θ values used to describe the radiation patterns in HFSS and for the experimental antenna. For the Z-X plane case, $\phi = 0^{\circ}$ and θ is varying from -180 to 180 degrees. -180 degrees to 0 degrees refers to negative x values and 0 to 180 degrees refers to positive x values. For the Z-Y plane case, $\phi = 90^{\circ}$ and θ is varying from -180 to 180 degrees. -180 degrees to 0 degrees refers to negative y values and 0 to 180 degrees refers to positive y values. For the X-Y plane case, $\theta = 90^{\circ}$ and ϕ is varying from -180 to 180 degrees.

Figures 4.34 - 4.39 are the radiation gain measurements of the experimental and simulated antenna. The realized gain in the simulation is approximately 10 dB in the forward broadside direction of the antenna indicating that the antenna transmitting and receiving capabilities in that direction are good. Comparing the simulated and experiment radiation patterns in Figures 4.34, 4.36, and 4.39 the simulated results are near in agreement with the experiment. The gain at 0 degrees in Figure 4.36 and Figure 4.34 is about 6.7 dB and about 4.5 dB in Figure 4.35 and Figure 4.37. Small differences could be attributed to construction issues of the experimental antenna. Comparing Figure 4.35, 4.37, and 4.38 the simulated results were not in agreement with the experiment. Differences could be attributed to human construction of the experimental antenna and how the antenna was mounted (non-air dielectric platform) for the radiation pattern measurement.

Comparing the experimental rectangular and cylindrically conformal cavity-backed E-patch antenna radiation patterns (Figures 4.15 and 4.16 and Figures 4.34, 4.35, 4.36 and 4.37), it can be noted that the radiation pattern of the cylindrically conformal case is broader then that of the rectangular case. The broadening of the radiation pattern might be due to creeping wave effects and subsequent diffraction occurring around the cylinder.

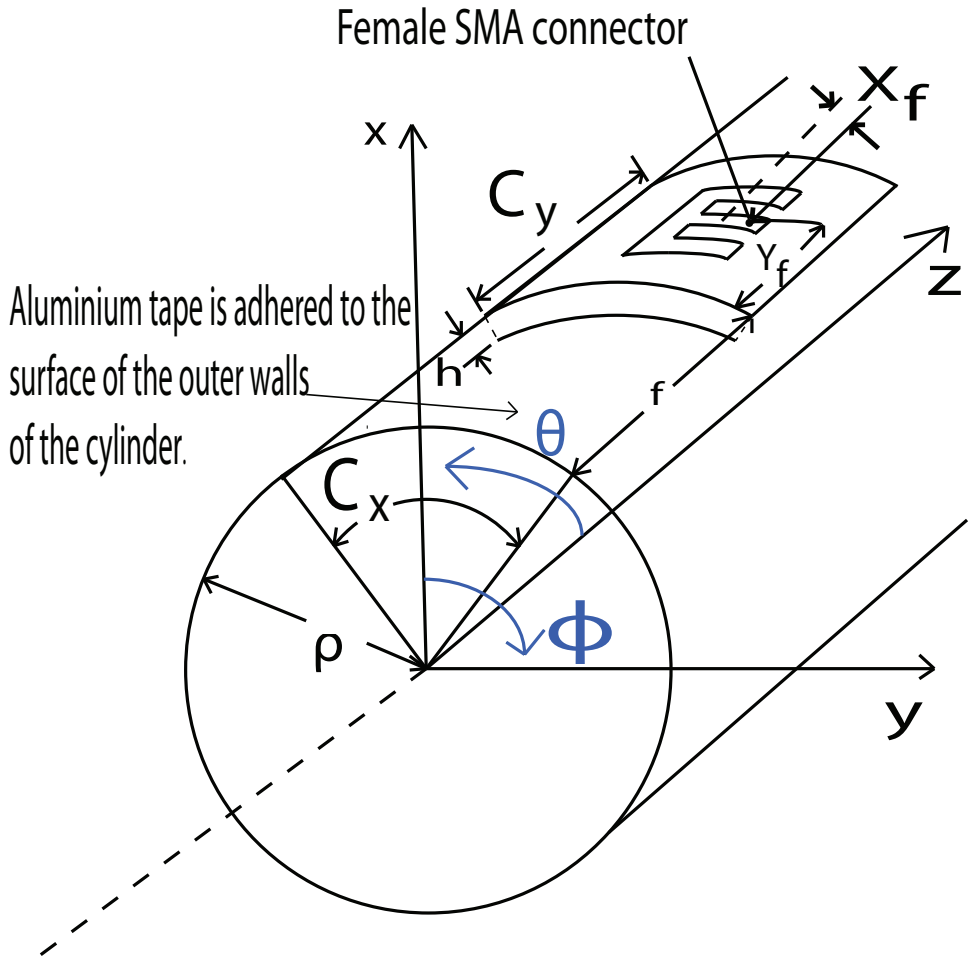

Figure 4.19: A angled-view conceptual depiction of the E-patch antenna conformed to the surface of a cylinder with a radius ρ.

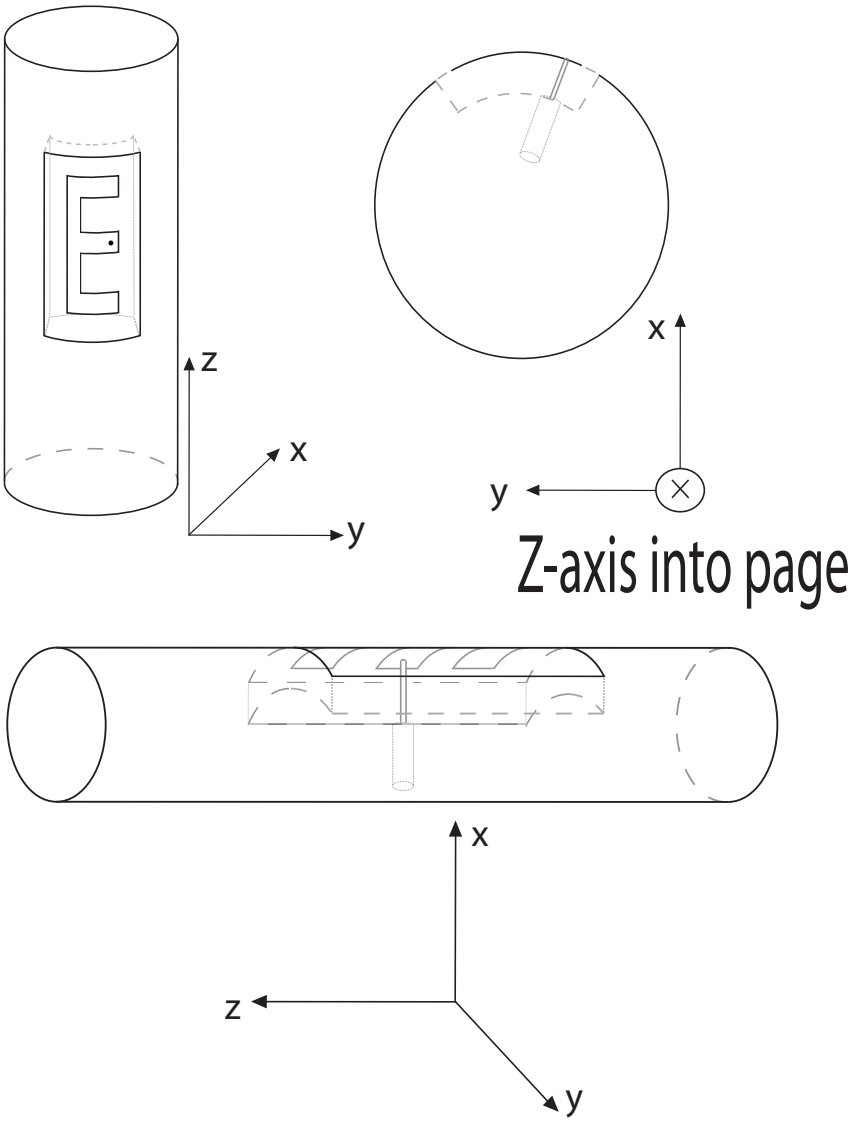

Figure 4.20: A 3-view depiction of the E-patch antenna conformed to the surface of cylinder.

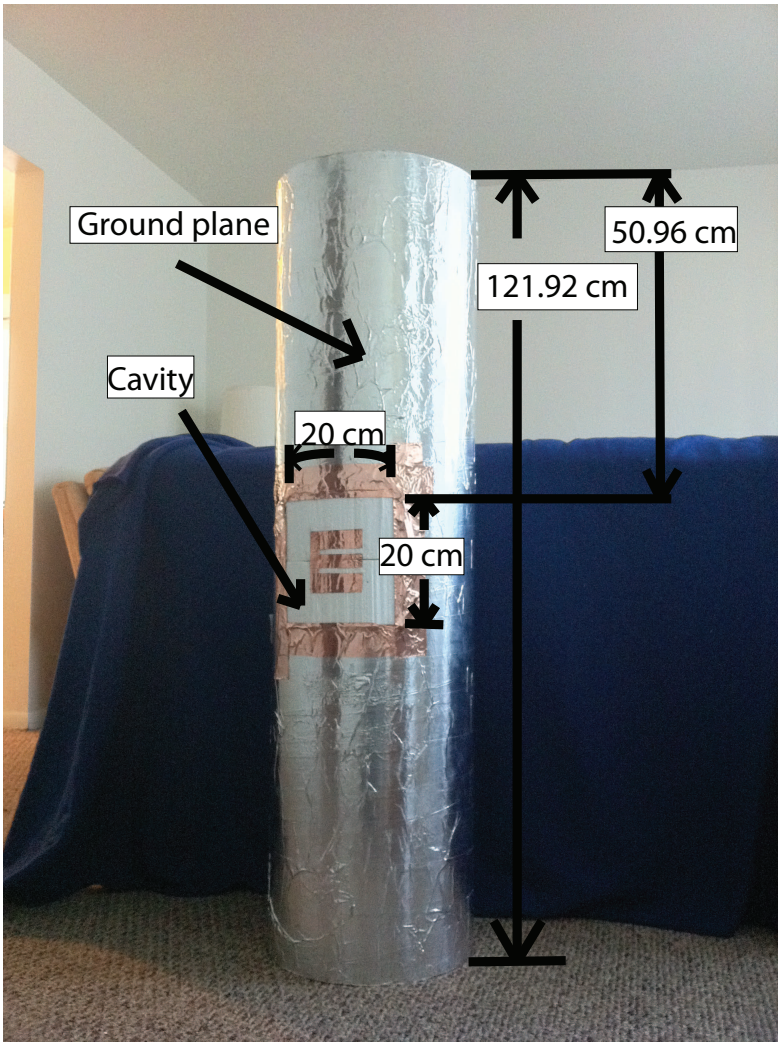

Figure 4.21: A front view of the cylindrically conformal cavity-backed E-patch experimental antenna.

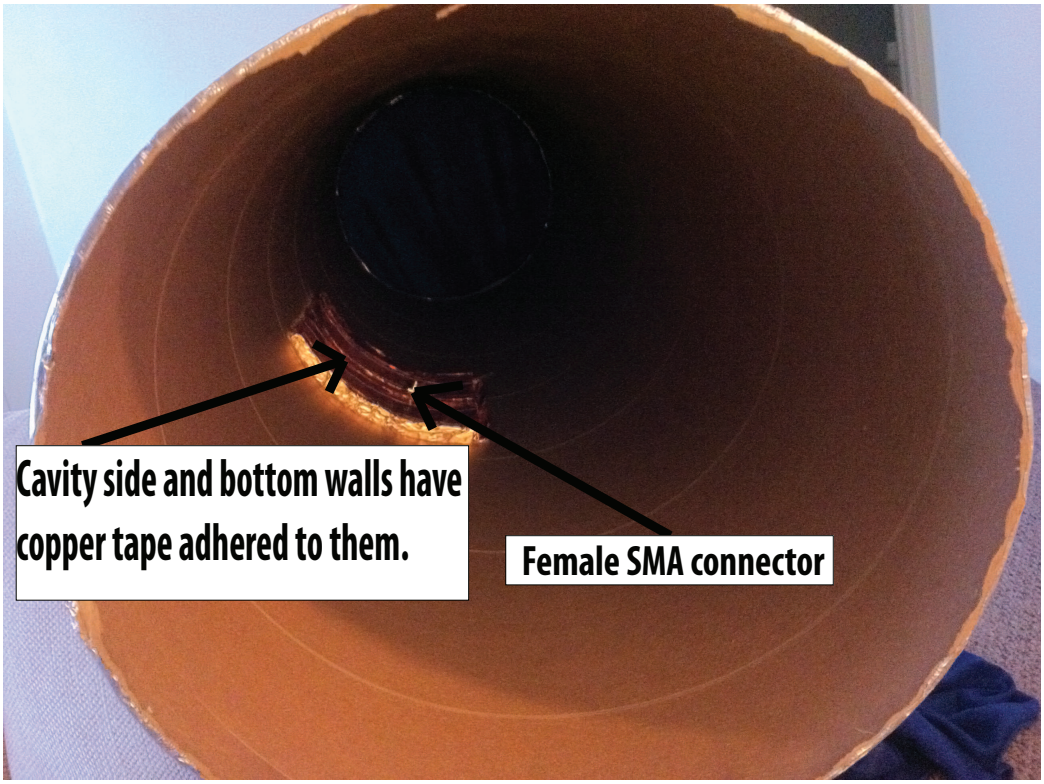

Figure 4.22: A top view of the cylindrically conformal cavity-backed E-patch experimental antenna.

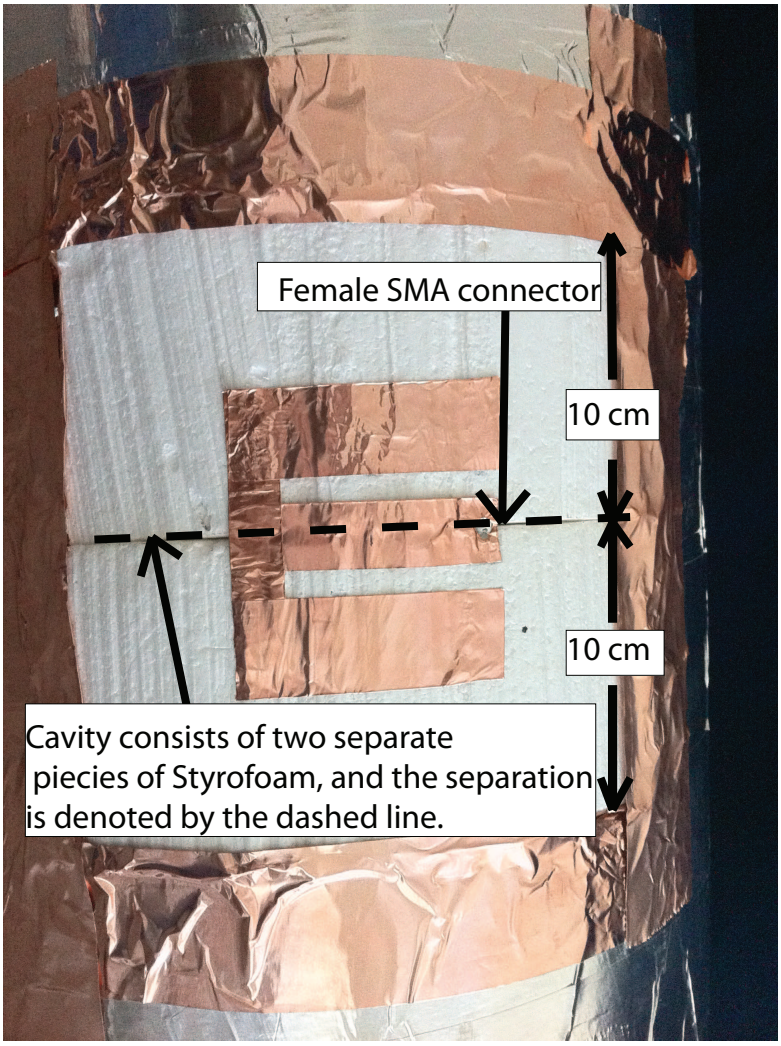

Figure 4.23: A close up picture of the front cavity E-patch experimental antenna.

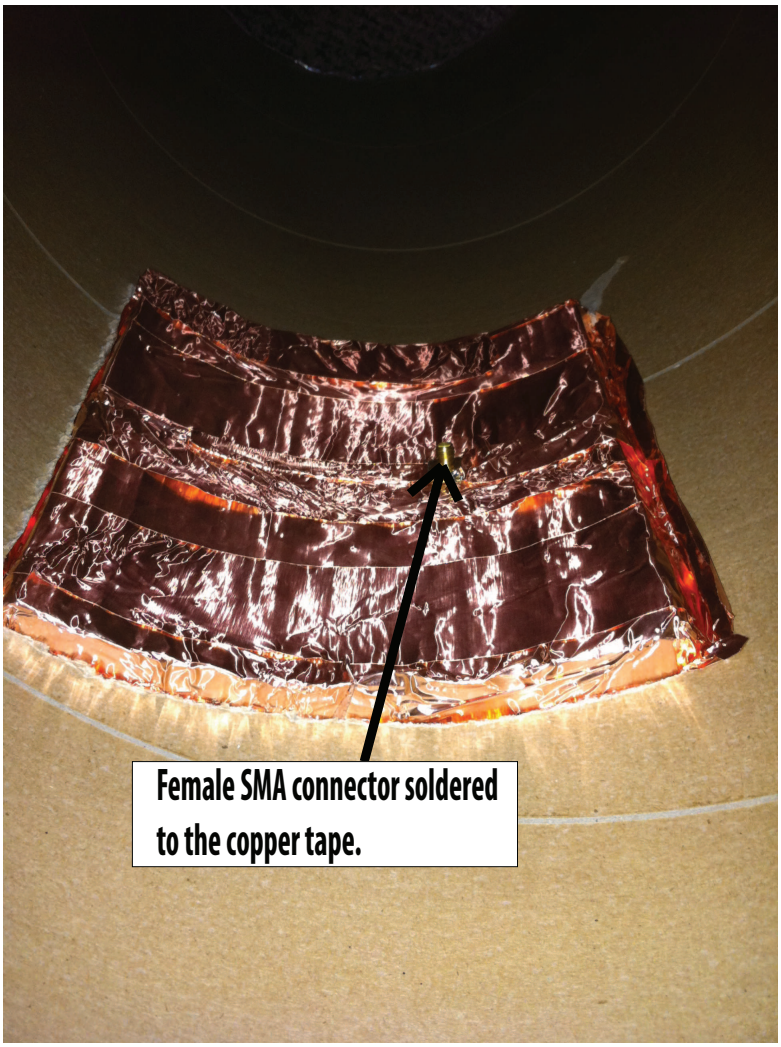

Figure 4.24: A close up picture of the feed point of the experimental cylindrically conformal cavity.

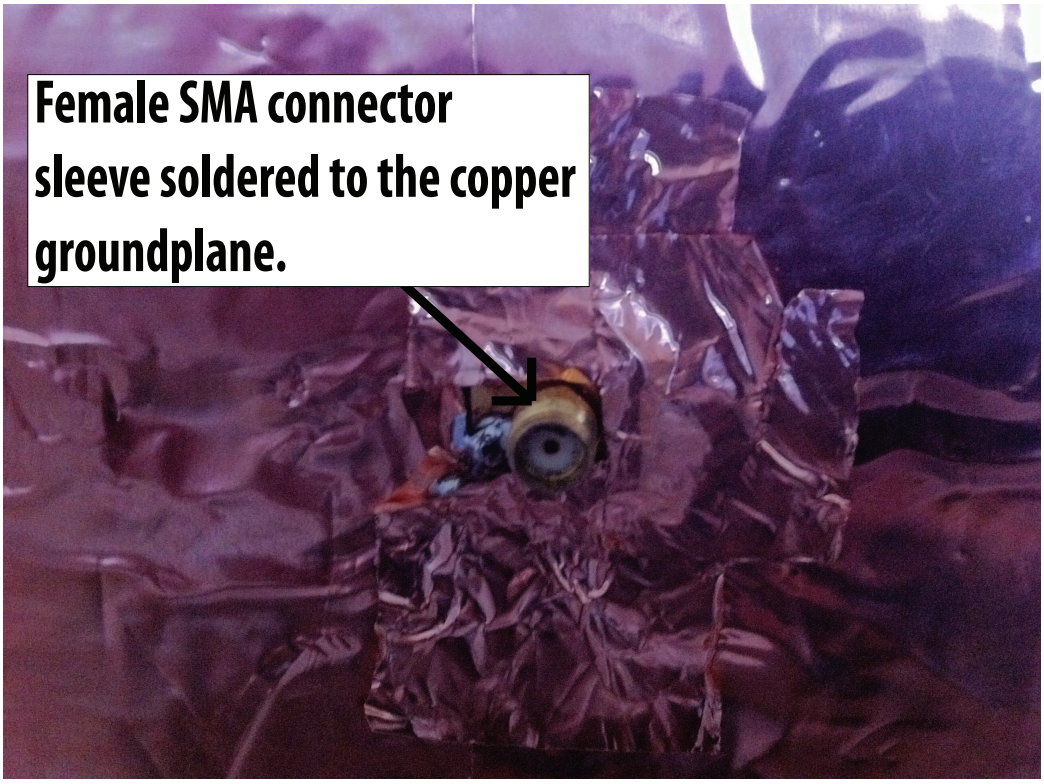

Figure 4.25: A closer up picture of the feed point of the experimental cylindrically conformal cavity.

Figure 4.26: A picture of the cylinder cardboard tube used to construct the cylindrically conformal cavity-backed E-patch antenna.

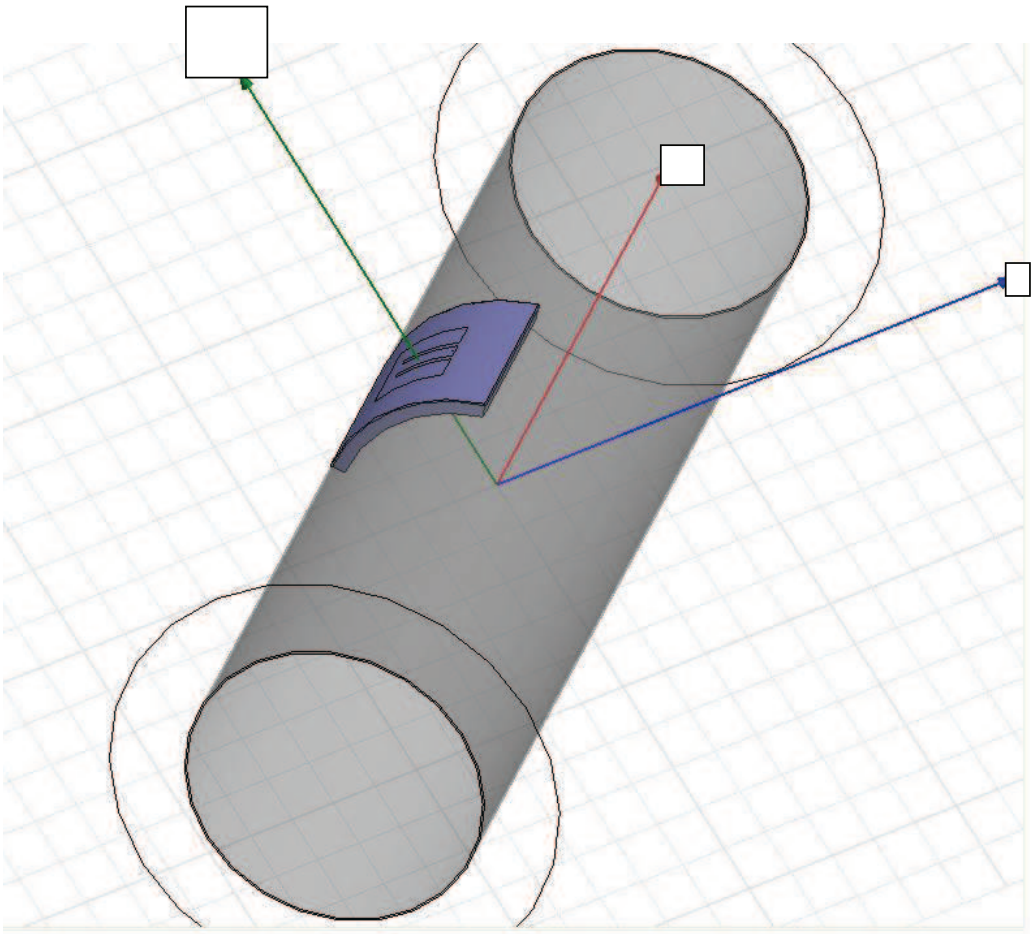

Figure 4.27: A screen capture of the cylindrically conformal cavity-backed E-patch antenna in HFSS.

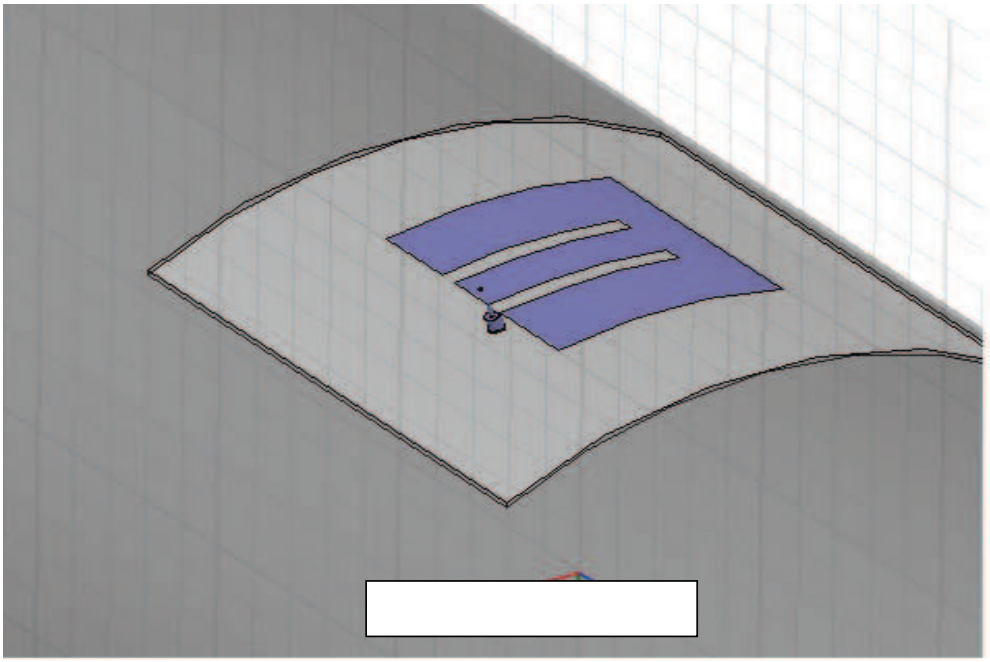

Figure 4.28: A closer view screen capture of the cylindrically conformal cavity-backed E-patch antenna in HFSS.

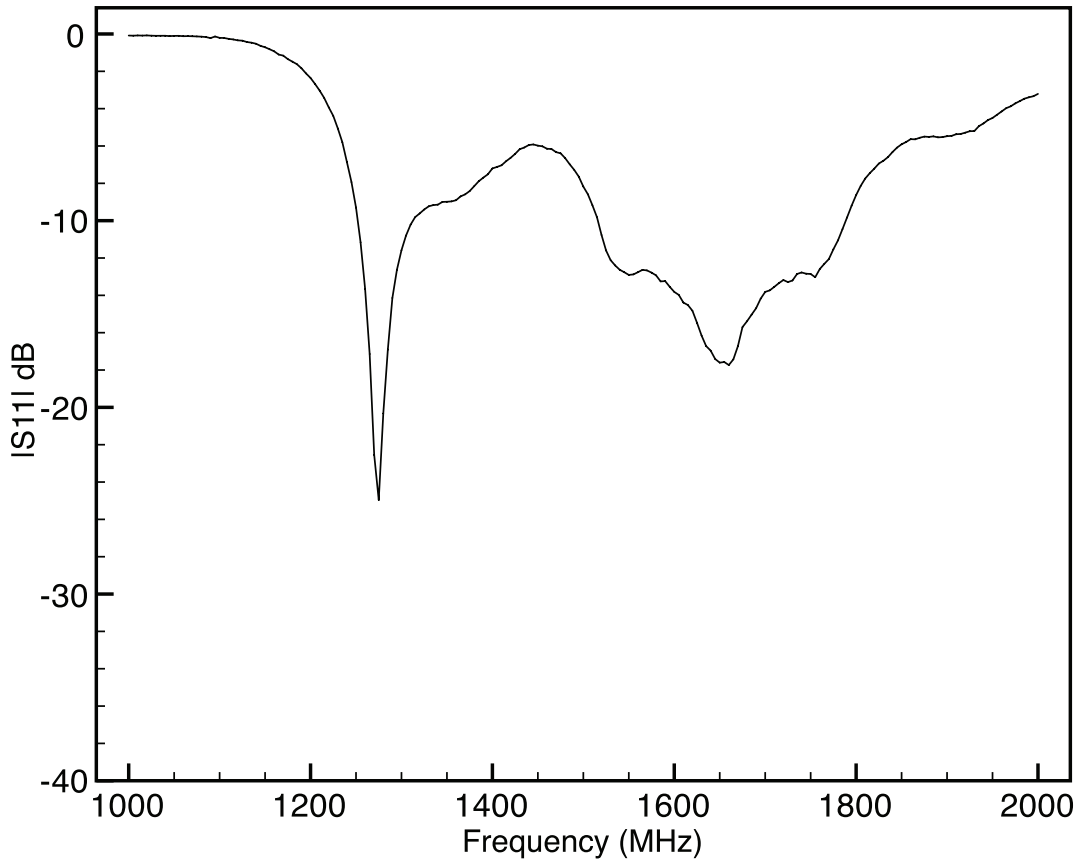

Figure 4.29: $|S_{11}|$ versus frequency measured on a Hewlett Packard 8753D network analyzer for experimental cylindrically conformal cavity-backed E-patch antenna. Dimensions provided in Table 4.1.

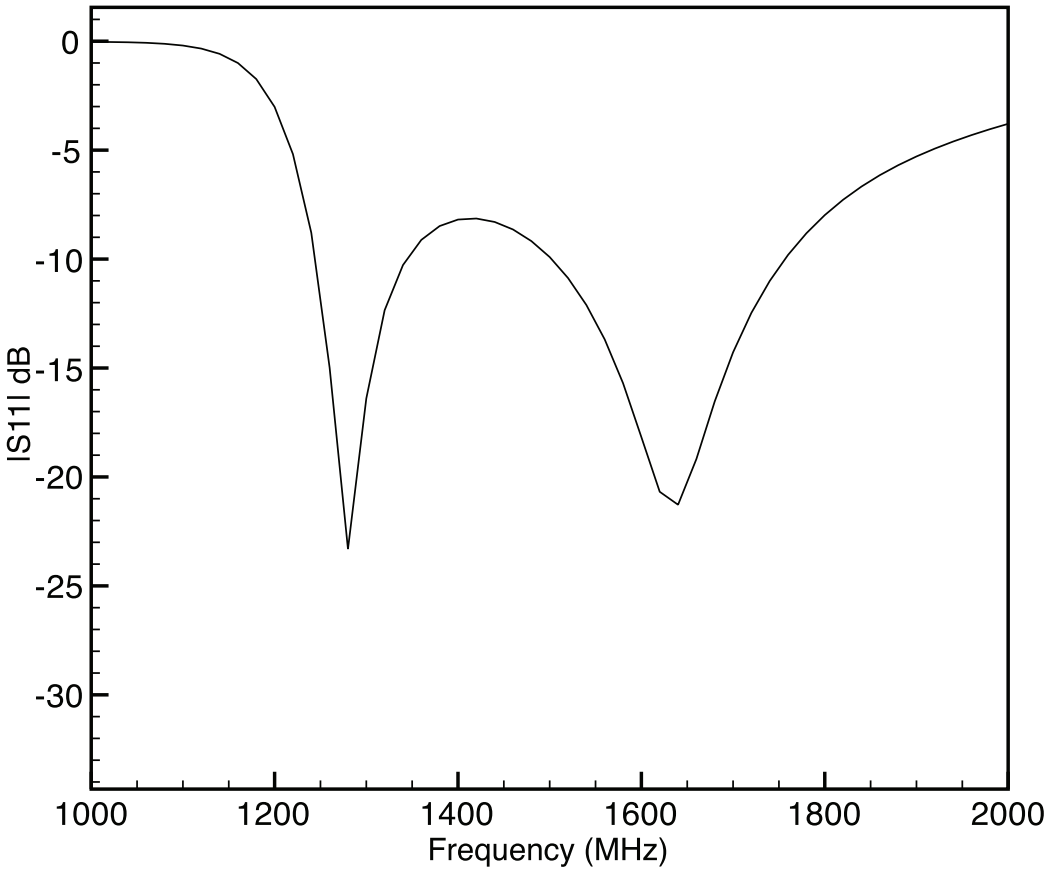

Figure 4.30: $|S_{11}|$ versus frequency of a simulated cylindrically conformal cavity-backed E-patch antenna. Dimensions provided in Table 4.3.

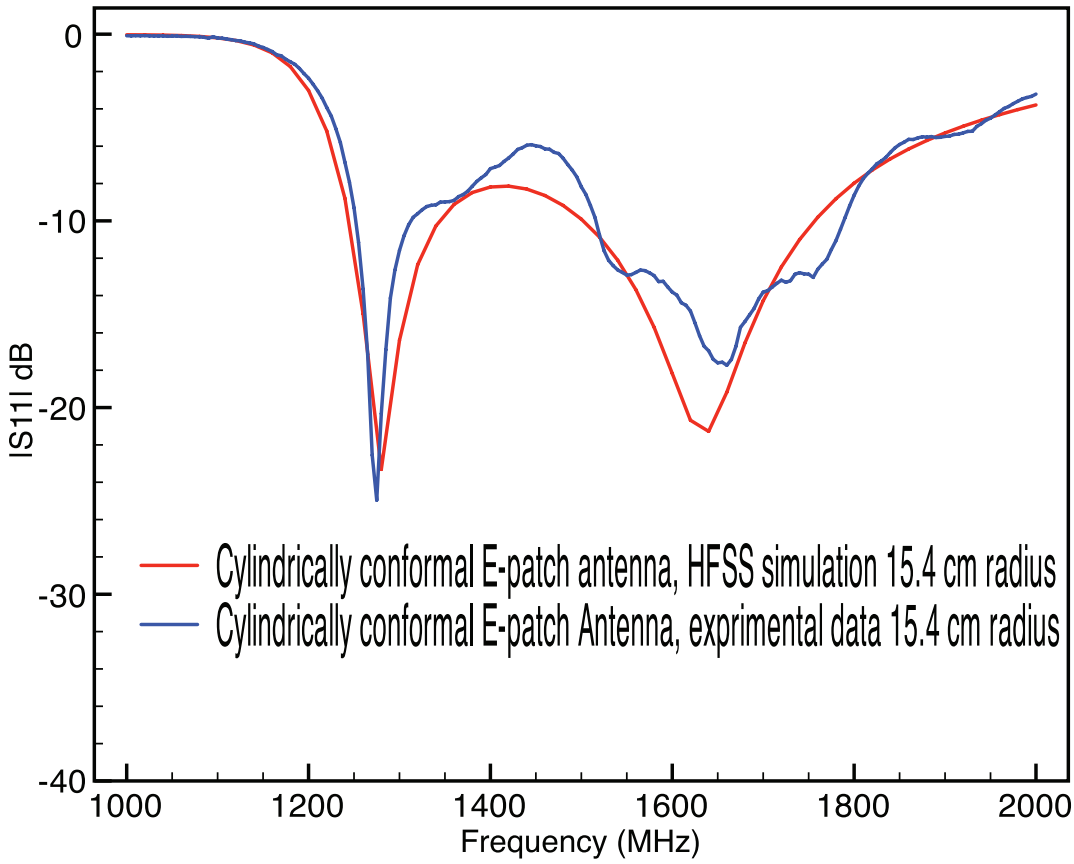

Figure 4.31: $|S_{11}|$ versus frequency for experimental and simulated cylindrically conformal cavity-backed E-patch antennas. Dimensions provided in Table 4.3.

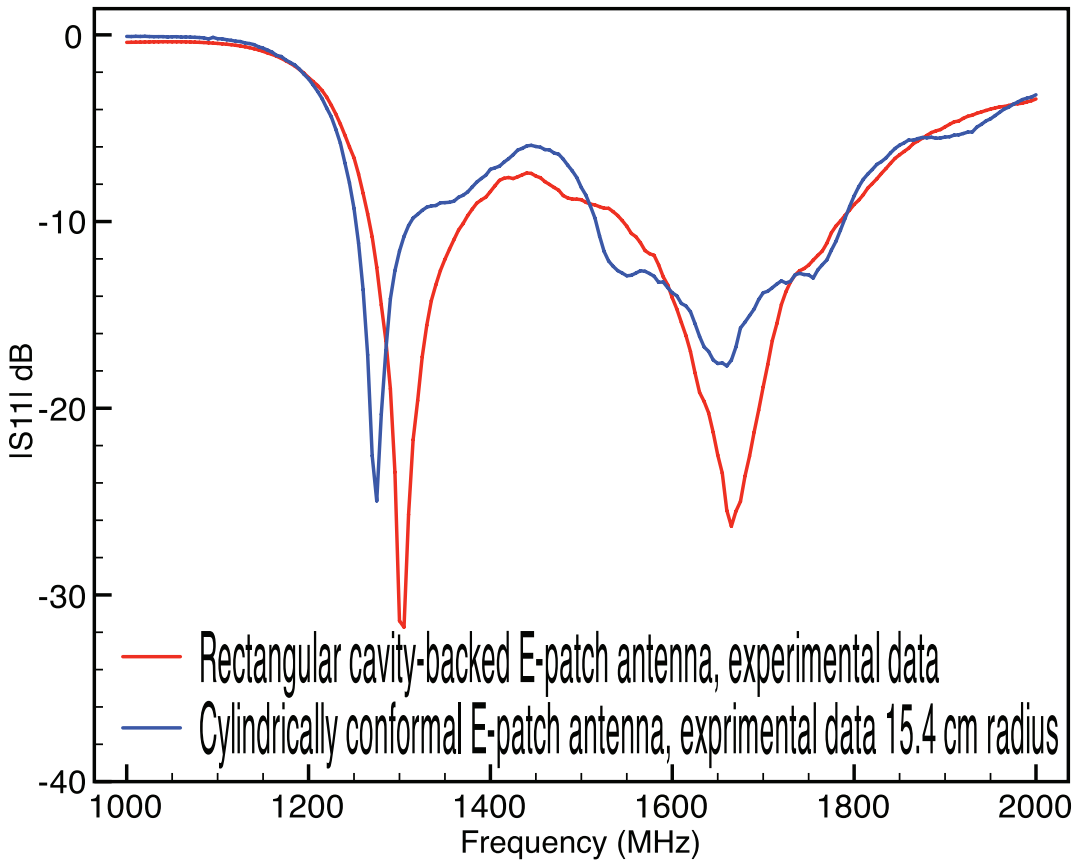

Figure 4.32: $|S_{11}|$ versus frequency for experimental cylindrically conformal and rectangular cavity-backed E-patch antennas. Dimensions provided in Table 4.3 and Table 4.1.

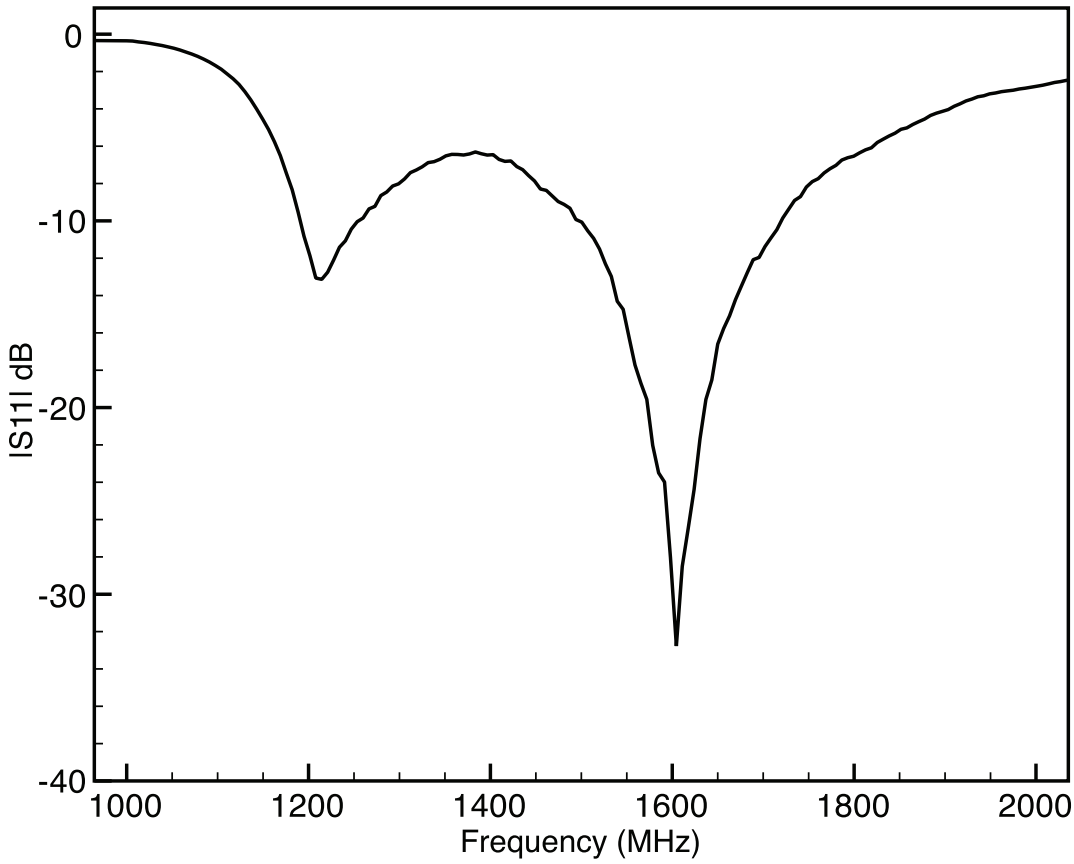

Figure 4.33: $|S_{11}|$ versus frequency for a cylindrically conformal cavity-backed E-patch antenna. Dimensions provided in Table 4.5.

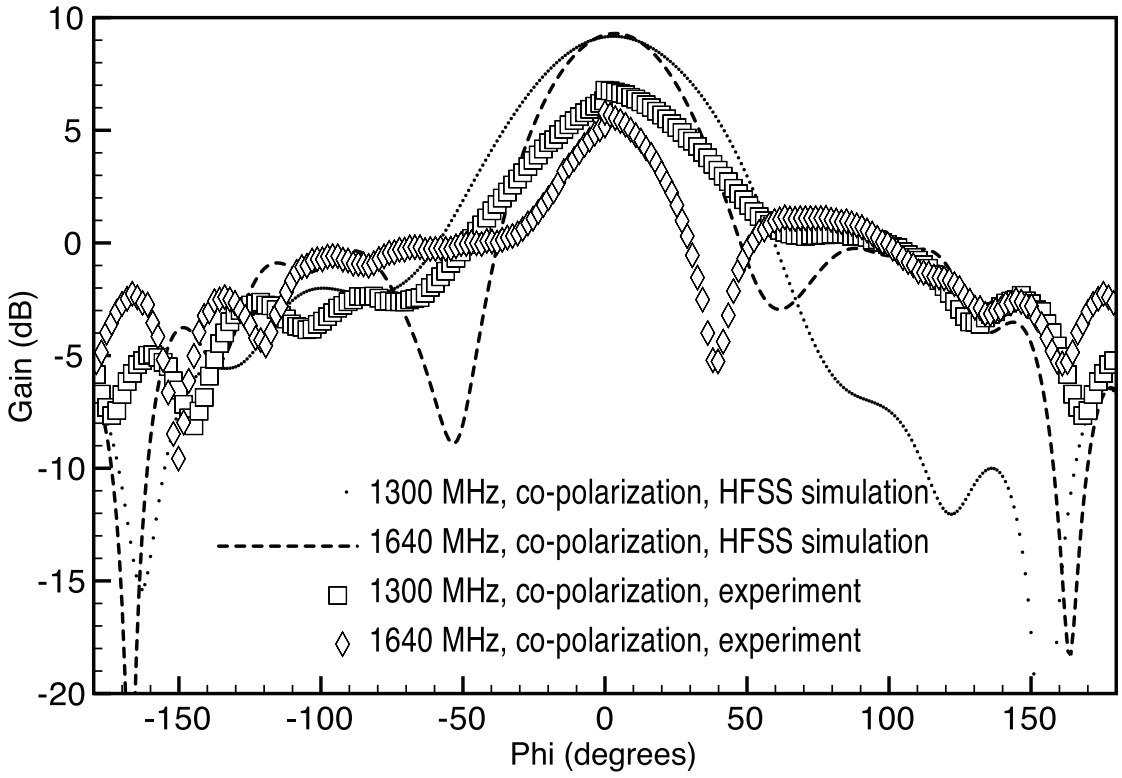

Figure 4.34: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna, X-Y plane and co-polarized. Dimensions provided in Table 4.3.

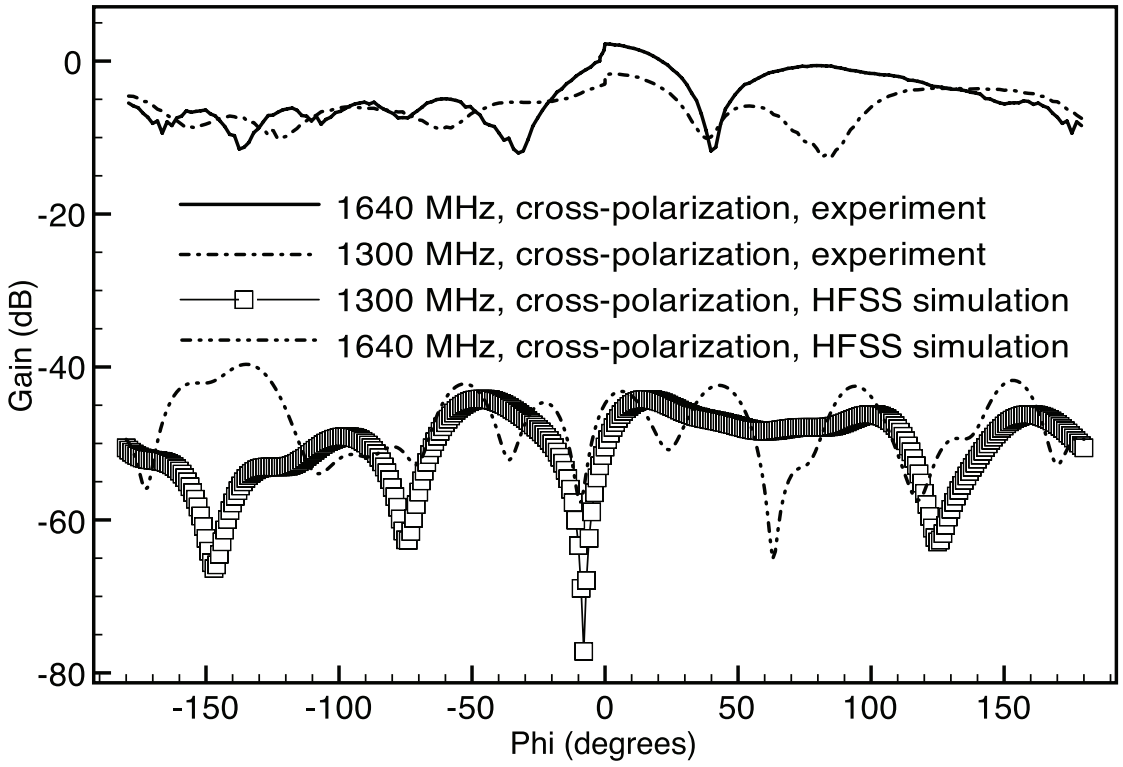

Figure 4.35: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna, X-Y plane cross-polarized. Dimensions provided in Table 4.3.

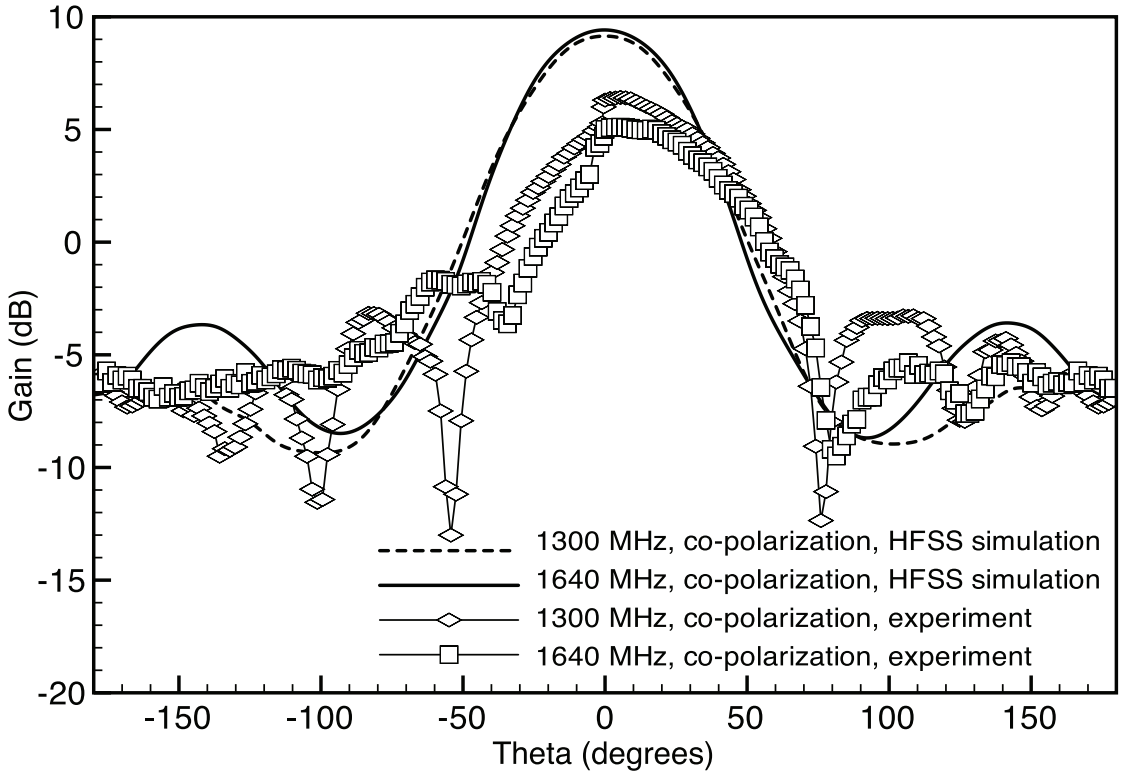

Figure 4.36: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna, Z-X plane and co-polarized. Dimensions provided in Table 4.3.

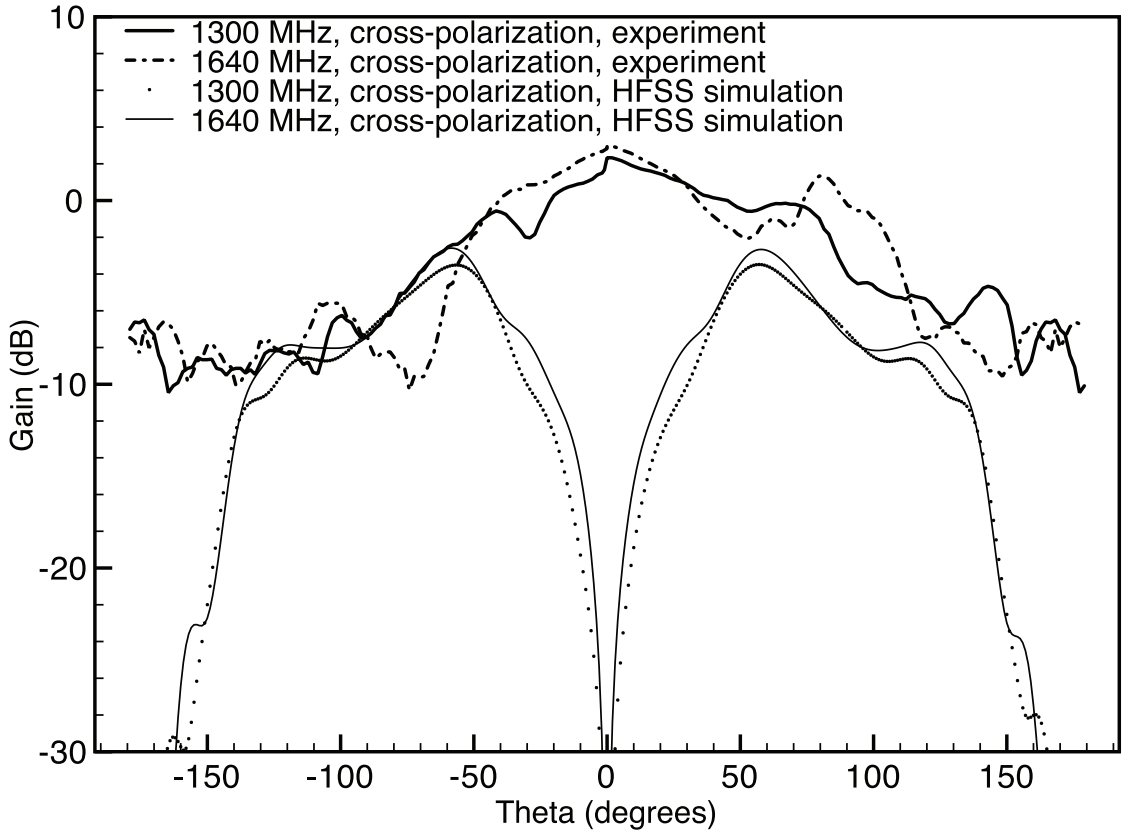

Figure 4.37: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna, Z-X plane cross-polarized. Dimensions provided in Table 4.3.

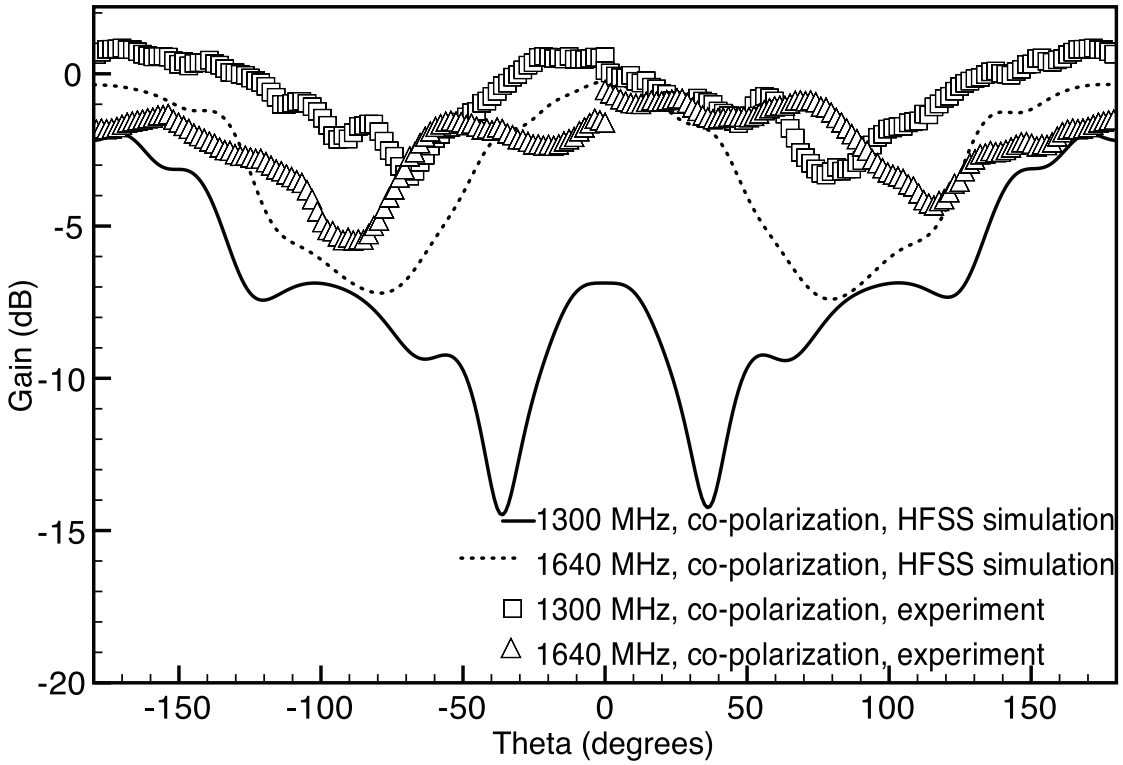

Figure 4.38: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna, Z-Y plane and co-polarized. Dimensions provided in Table 4.3.

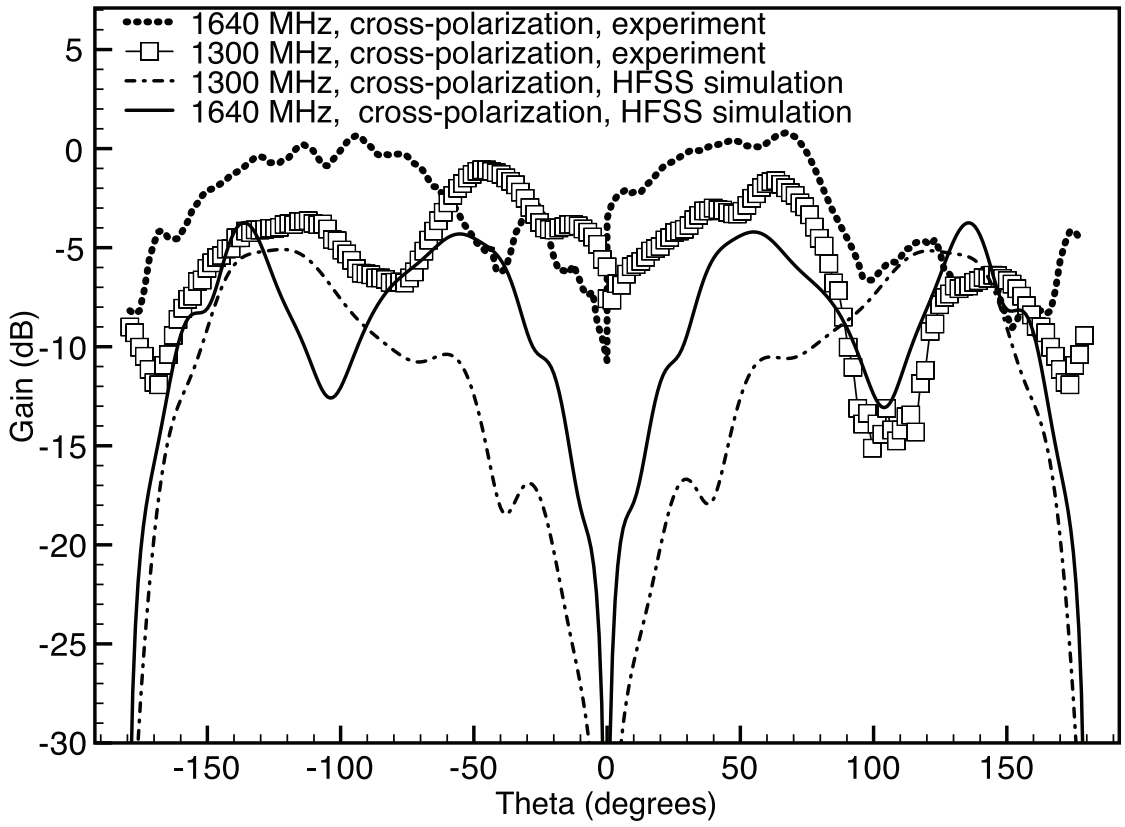

Figure 4.39: Gain radiation pattern of the experimental and simulated cylindrically conformal cavity-backed E-patch antenna,Z-Y plane cross-polarized. Dimensions provided in Table 4.3.

Chapter 5

Conclusion

Infinite ground plane simulations for E-patch antennas were thoroughly investigated at the L1 and L2 frequencies. The simulated $|S_{11}|$ for the E-patch antenna at the L1 and L2 frequencies performed well when the design equations (1.3) were used. The -10 dB bandwidth of the E-patch antenna in the infinite ground plane simulation extended between the L1 and L2 frequencies. However, when the same E-patch antenna, whose dimensions were determined from equations (1.3), was simulated with a rectangular cavity backing, it did not perform well.

In the pursuit of a value of $|S_{11}|$ of -10 dB or below at the L1 and L2 frequencies in the rectangular cavity-backed case, a Matlab graphical user interface (GUI) was written to allow the user to specify a range of E-patch dimensions to vary. Using K-brick and the Matlab GUI, acceptable values for $|S_{11}|$ at L1 and L2 frequencies were determined.

A cylindrically conformal cavity-backed E-patch antenna with a flat bottom cavity was simulated for multiple radii of cylinders. As the radius ρ of the cylinder was increased, $|S_{11}|$ decreased for the L1 frequency and increased for the L2 frequency. In addition the higher resonance frequency shifted upwards beyond the L2 frequency. With the exception of the 50 cm radius case, at the L1 and L2 frequencies $|S_{11}|$

was less then -10 dB. For the 50 cm radius cylinder case, $|S_{11}|$ was approximately -9 dB at the L2 frequency, and was less than -10 dB at the L1 frequency. For the flat bottom cavity case of the conformal E-patch antenna, it is unclear if the operation of the cylindrically conformal cavity-backed E-patch antenna would be effective and reliable at the L1 and L2 frequencies.

A cylindrically conformal cavity-backed E-patch antenna with a conformal cavity was simulated for multiple radii of cylinders. As the radius ρ decreased, $|S_{11}|$ stayed approximately the same for the lower resonance frequency and decreased for the higher resonance frequency. In addition the higher resonance frequency shifted lower in frequency as the radius decreased. The bandwidth does enlarge as the radius decreases. Notably, the bandwidth for the 100 cm simulated cylindrically conformal cavity-backed E-patch antenna case is 245 MHz (-10 dB or lower at 1245 MHz to 1330 MHz and 1590 MHz to 1750 MHz) while the bandwidth for the 15.4 cm case is 320 MHz (-10 dB or lower at 1250 MHz to 1340 MHz and 1520 MHz to 1750 MHz). At 50, 25, and 15.4 cm radius, $|S_{11}|$ at the L2 frequencies is less then -10 dB. The L1 frequency stays approximately -9 dB for all the cases. These differences in $|S_{11}|$ versus frequency might be due to the mode two resonance point being altered by curvature of the cylinder.

Two experimental E-patch antennas were built to verify simulations done in HFSS. The rectangular cavity-backed E-patch antenna simulation and experiment matched well. $|S_{11}|$ at frequencies of interest for the experimental rectangular cavity-backed E-patch antenna was -10 dB less than the HFSS simulation at the two resonant frequencies. The bandwidth of the experimental rectangular cavity-backed patch antenna was 330 MHz (-10 dB or lower at 1270 MHz to 1370 MHz and 1550 MHz to 1780 MHz). The bandwidth of the experimental cylindrically conformal cavity-backed E-patch antenna was 321 MHz (-10 dB or lower at 1254 MHz to 1310 MHz and 1520 MHz to 1785 MHz). The effect on curvature of the experimental antennas

indicates a loss in bandwidth and a shift in the main points of resonance to lower frequency. The curvature may have effected mode two's operation and contributed to the loss in bandwidth and shift in resonant frequency. Comparing the simulation and experiment bandwidths, the difference may be due to construction techniques of the experimental antennas and the lack of HFSS ability to simulate the E-patch antenna exactly in both cases.

An experimental antenna was built to achieve a -10 dB or lower value for $|S_{11}|$ at the L1 and L2 frequencies; that antenna's $|S_{11}|$ at the L1 and L2 frequency was acceptable with -12 dB at 1227 MHz and -20 dB at 1575 MHz.

The radiation pattern of the two experimental antennas was also investigated and simulated. The experimental radiation pattern of the cylindrically conformal antenna was broader than that of the rectangular case. The broadening of the radiation pattern might be due to creeping wave effects and subsequent diffraction occurring around the cylinder. In addition the broadside gain of the experimental cylindrically conformal cavity-backed E-patch antenna was found to be 8 dB, matching well with the simulated gain from HFSS of approximately 10 dB.

Future investigations into the cylindrically conformal cavity-backed E-patch antenna should include better techniques for construction and better modeling for the feed in HFSS. An implementation of Taguchi's method of optimization might improve $|S_{11}|$ at the L1 and L2 frequencies and the overall bandwidth of the cylindrically conformal cavity-backed E-patch antenna.

Taguchi's method of optimization for E-patches has been proven to be an effective method for optimizing $|S_{11}|$ for rectangular cavity-backed E-patch antennas. Through the use of simple fitness functions and orthogonal arrays, $|S_{11}|$ at the two main resonance frequencies of the E-patch antenna can be minimized. Computation time and ease of coding make Taguchi's method of optimization an attractive alternative to genetic algorithms and particle swarm optimizations.

Further research using Taguchi's method for E-patch antennas should be done to improve the overall bandwidth between both points of resonance of the antenna. Increasing the bandwidth of the antenna may be accomplished with modification to the fitness functions presented in earlier chapters.

APPENDICES

Appendix A: Setting up Sonnet for E-patch antenna simulations

1. Refer to Figure A.1. Create a new Geometry.

2. Refer to Figure A.2. Left click on circuit tab in the upper left corner of the new geometry layout. Highlight “units" and left click. Once the Unit dialog box has appeared, highlight the length box and scroll until mm is highlighted. Left click when mm is highlighted and then left click on the “Ok" button.

3. Refer to Figure A.3. Locate the circuit tab again and highlight “box" and left click it to bring up the Box dialog box. To initialize the geometries of Figure 2.1 and Figure 2.2 initialization, change the “Cell" size to 1 for the X and Y directions. Change the box size to 150 for the X and Y directions. Change the number of cells to 150 for the X and Y directions. To initialize the geometries Figure 2.3 and Figure 2.4 initialization, change the “Cell" size to 1 for the X and Y directions. Change the box size to 200 for the X and Y directions. Change the number of cells to 200 for the X and Y directions. Change the Top metal to “Free Space" in both cases. Left click on the“apply” button, then left click on the “Ok" button. For L-band E-patch antenna designs, one should specify that the box be at least 1λ x 1λ in the X and Y directions respectively. Here λ is the largest wavelength of L-band. Note that 200 mm is not large enough to cover the 1λ x 1λ; however, 200 mm yields good approximations for $|S_{11}|$.

4. Refer to Figure A.4. Locate the circuit tab again and highlight “dielectric layers" and left click. The dielectric layer box should now appear. Left click the first dielectric layer so that it is highlighted. Then left click the “edit" button and change the material name to “Air". Change the thickness of the dielectric to the designs h height. In this case, the value is 10 mm.

5. Refer to Figure A.5. Left click the second dielectric layer so that it is highlighted. This is the layer above the metallic patches. Then left click the “edit" button and change the material name to “Air". Change the thickness of the dielectric to the height of the antenna. In this case, the value is 10 mm.

6. Refer to Figure A.6. Left click on the “Ok" button on the dielectric layer box. The dielectric box should now appear as below.

7. Refer to Figure A.7. Left click on the Analysis tab in the upper left corner. Highlight and left click setup. A setup box will appear. For the geometry of Figure 2.1, enter 2 and 3 GHz for the start and stop frequencies, respectively. For the geometry of Figure 2.2, enter 1.5 and 3 GHz for the start and stop frequencies, respectively. For the geometry of Figure 2.3 and Figure 2.4 enter 1 and 2 GHz for the start and stop frequencies, respectively. Check the computer density box if you wish to view surface fields and radiation patterns. Leave the Adaptive Sweep drop down box as is. Left click on “Ok" when finished.

8. Refer to Figure A.8. Left click on the tools tab in the upper left corner. Highlight “add metallization". Then highlight “add rectangle".

9. Refer to Figure A.9. The rectangle attributes should appear on your screen. For the width and and length, type in the dimensions of the patch. Left click on “Ok" when finished. For the Sonnet simulations, four patches were used then merged when all were placed in the box.

10. Refer to Figure A.10. Place rectangle.

11. Refer to Figure A.11. Left click on the tools tab once again and highlight “via", then left click on “down to ground". With “via" still highlighted, highlight and left click on “circular".

12. Refer to Figure A.12. Another warning dialog box may appear. Left click on “ok" if it does. The circular attributes dialog box should now appear. Change the diameter to 1.2 mm this is the approximate diameter of a coaxial probe. The circle should be 10 sided. Left click “Ok" when the changes have been made.

13. Refer to Figure A.13. Place the via at the designated X_f and Y_f point by left clicking.

14. Refer to Figure A.14. Left click on the tools tab once again and highlight and left click on “add port". The pointer now has a boxed “n" around it. Left click on the via just made. A warning dialog box will pop up. Left click “ok" to close it. To view the patch geometry in 3-D, left click the 3-D box as shown in the screen shot below.

15. Refer to Figure A.15. When done viewing the 3-D simulation, left click on the “x" in the upper right corner. Now left click on the file tab in the upper left corner and save the geometry file.

16. Refer to Figure A.16. Left click on the project tab in the upper left corner and highlight and click the analyze tab. A view response dialog box should now appear and show your $|S_{11}|$ and other parameters. Click on the view response box to see the $|S_{11}|$ vs Frequency plot.

17. Refer to Figure A.17. Below the final E-patch geometry.

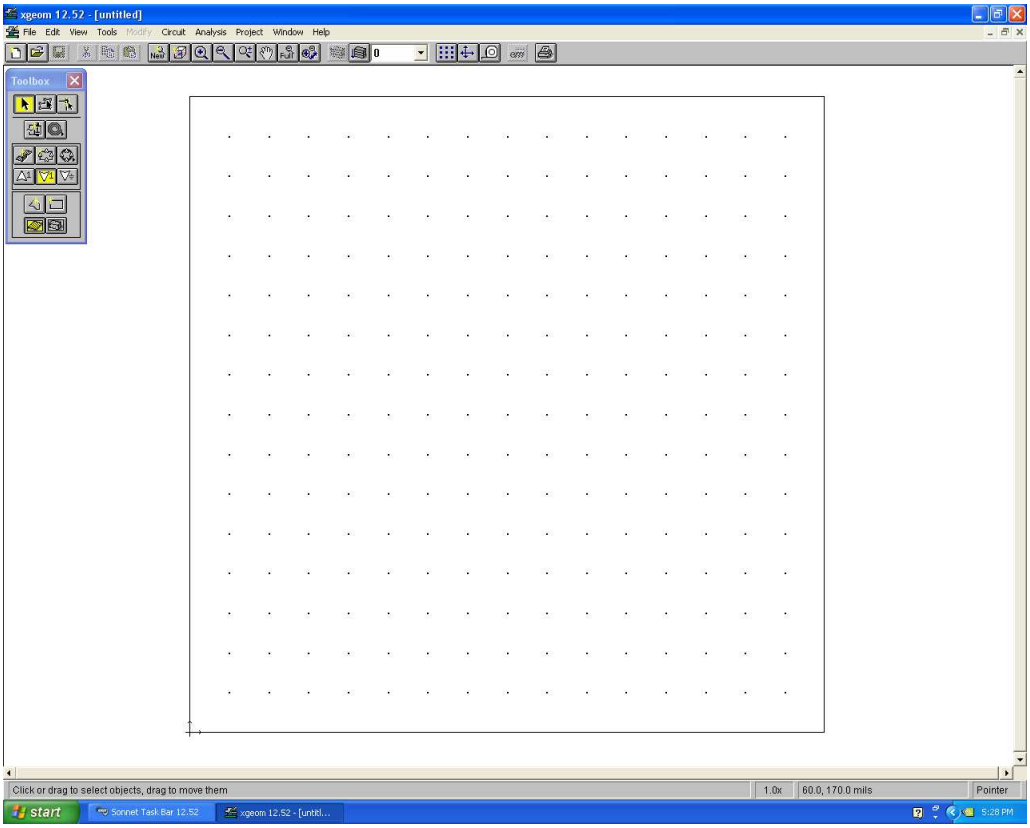

Figure A.1: New Geometry

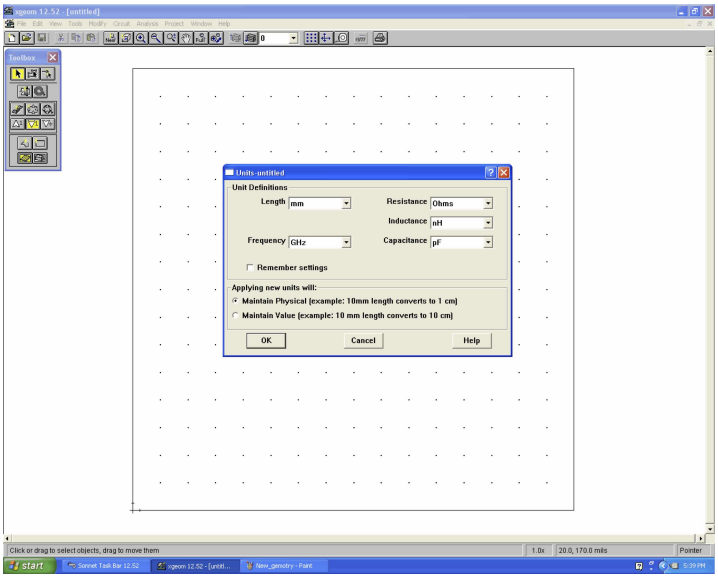

Figure A.2: Circuit Unit box

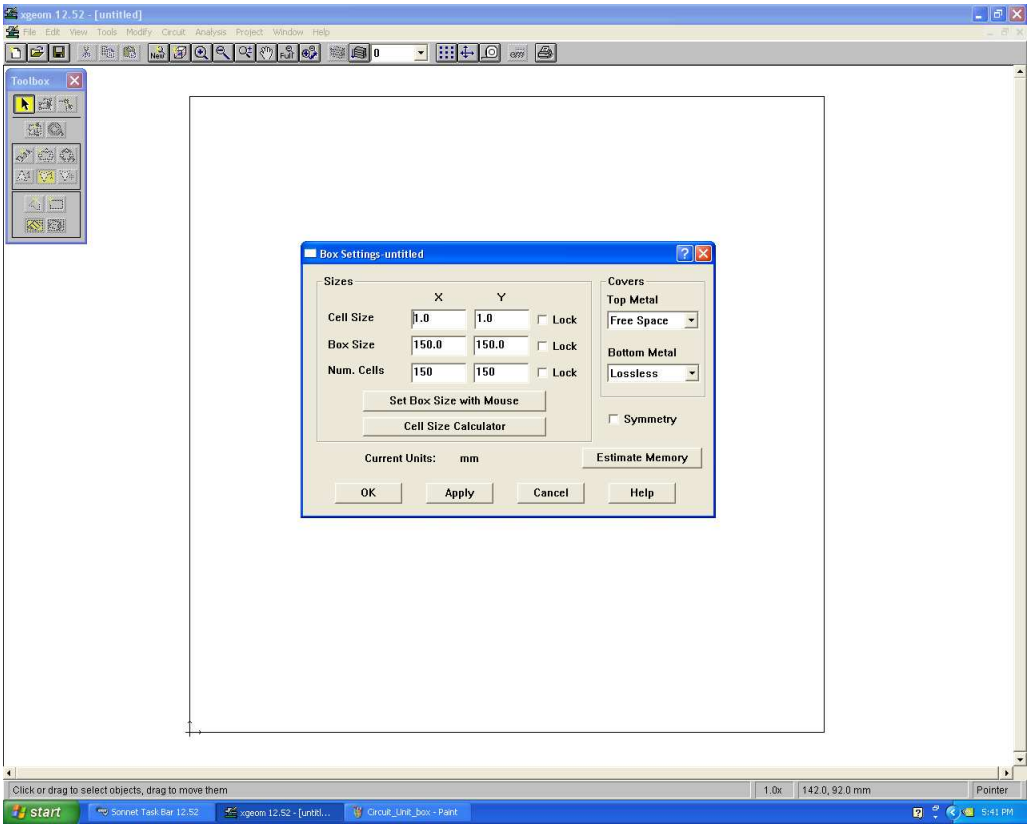

Figure A.3: Circuit box

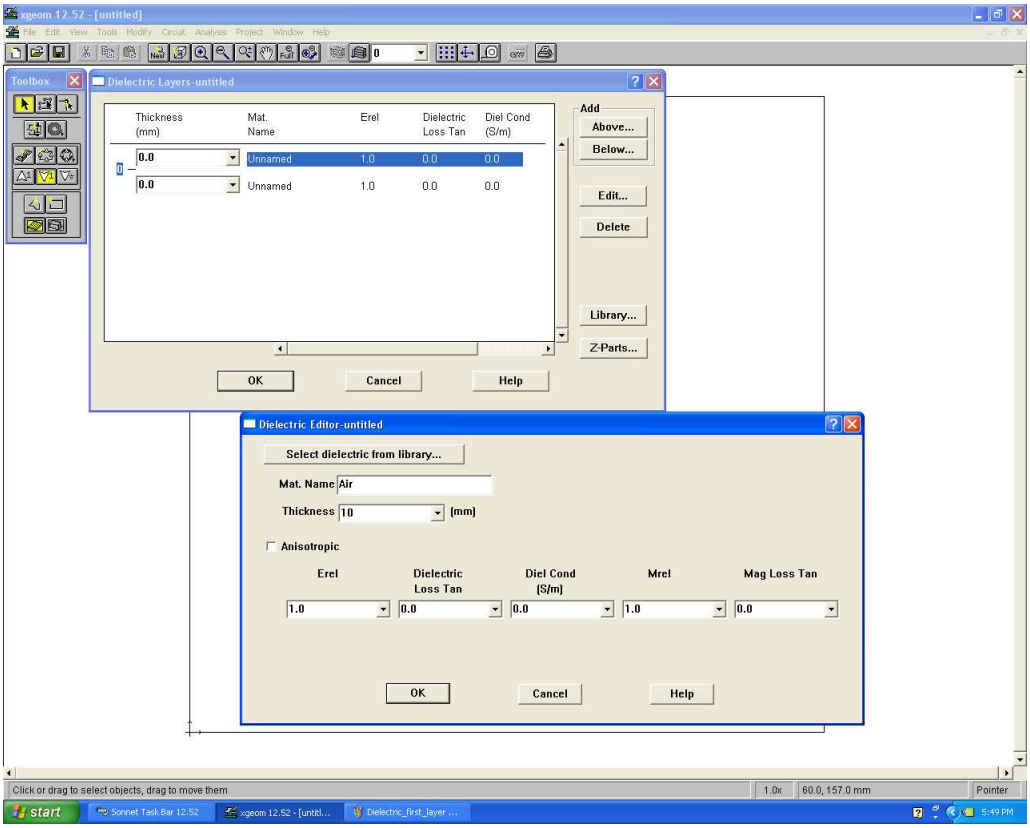

Figure A.4: Dielectric first layer Dielectric selection

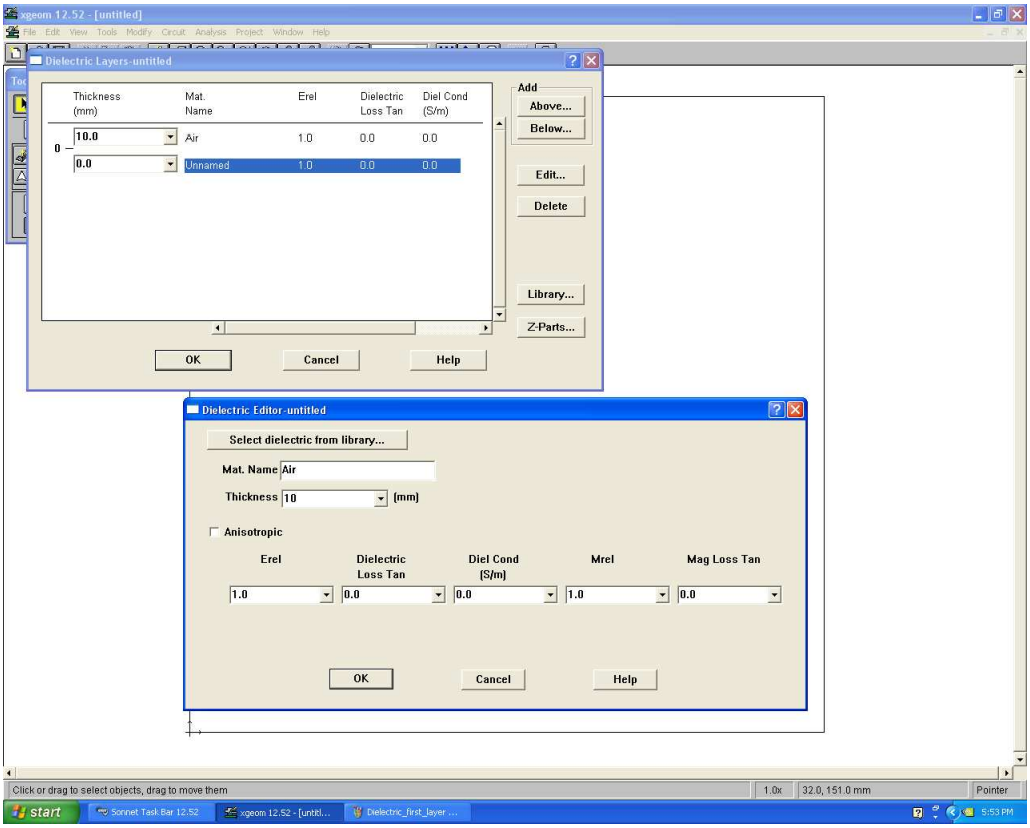

Figure A.5: Dielectric second layer parameter selection

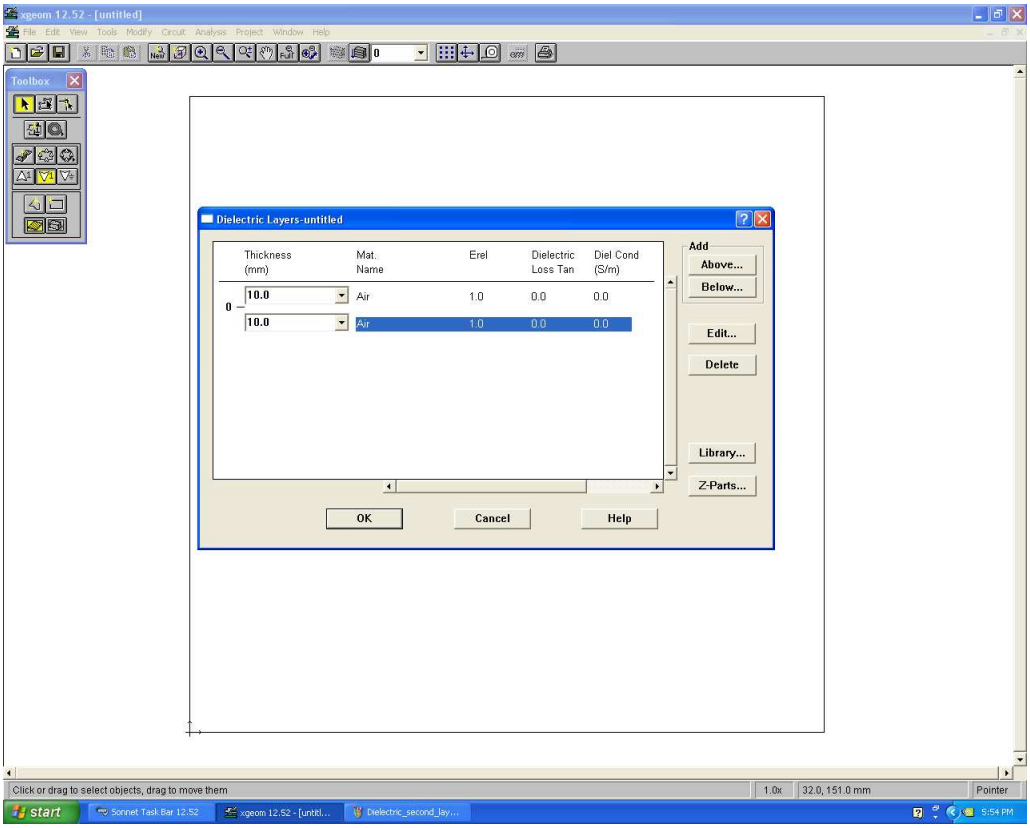

Figure A.6: Completion of Dielectric selection

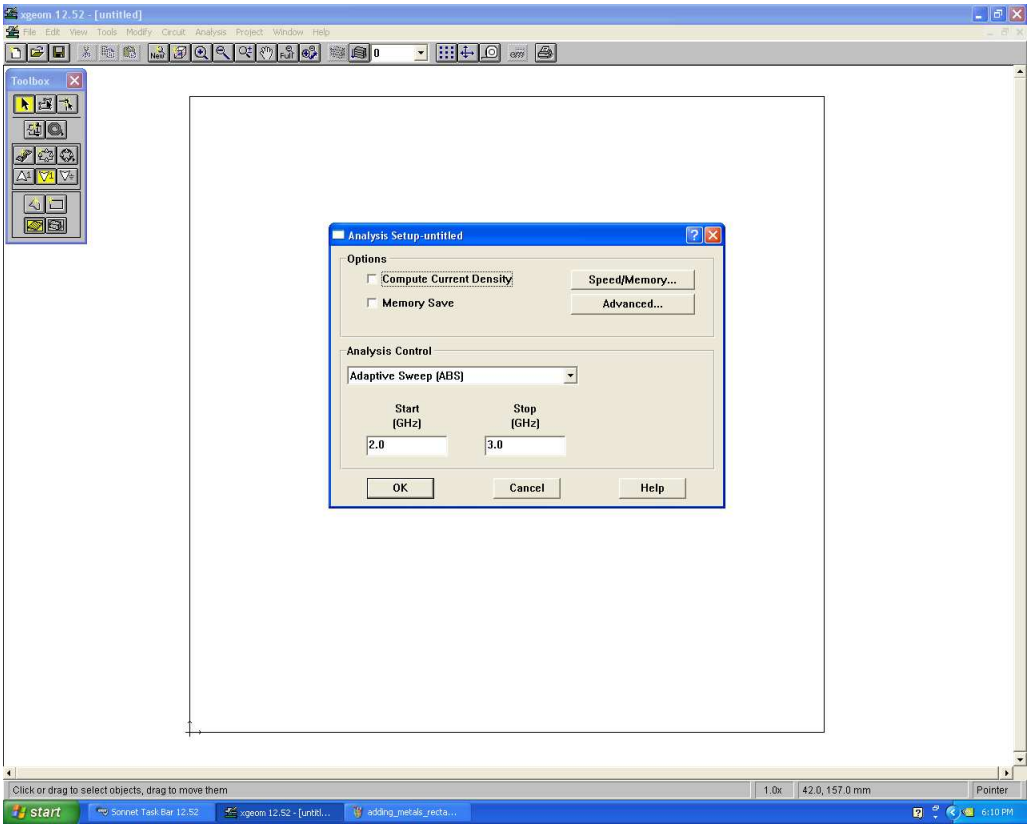

Figure A.7: Analysis Setup

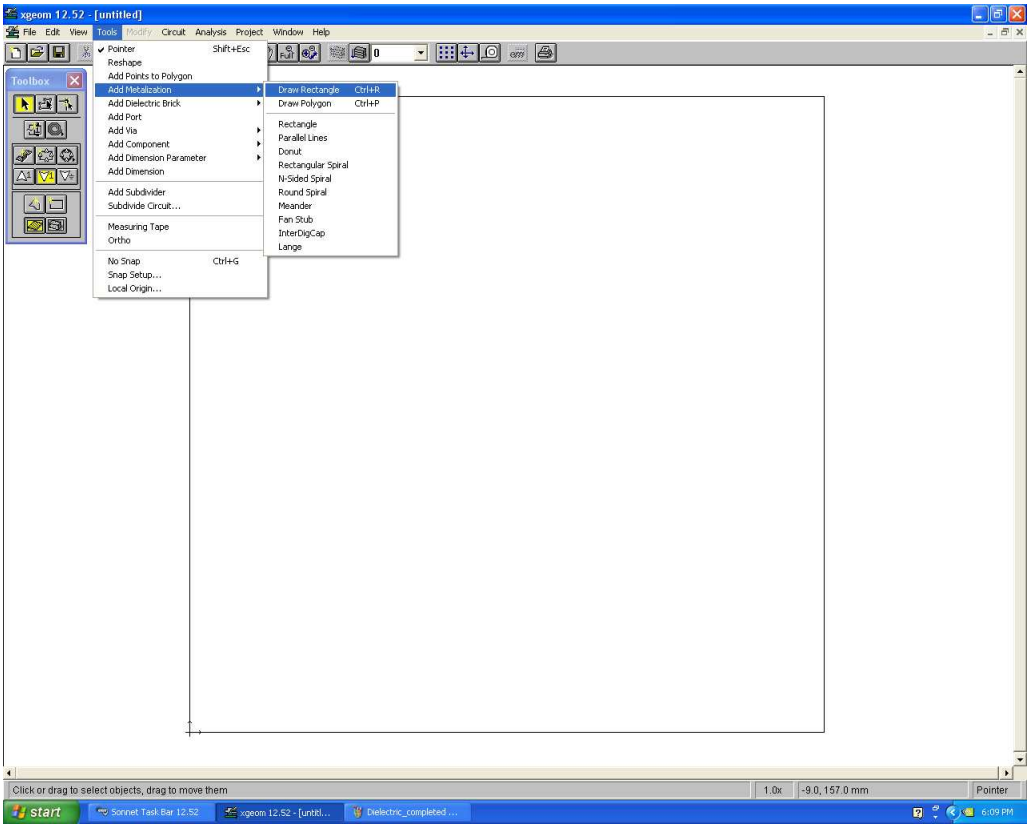

Figure A.8: Adding a metal rectangle

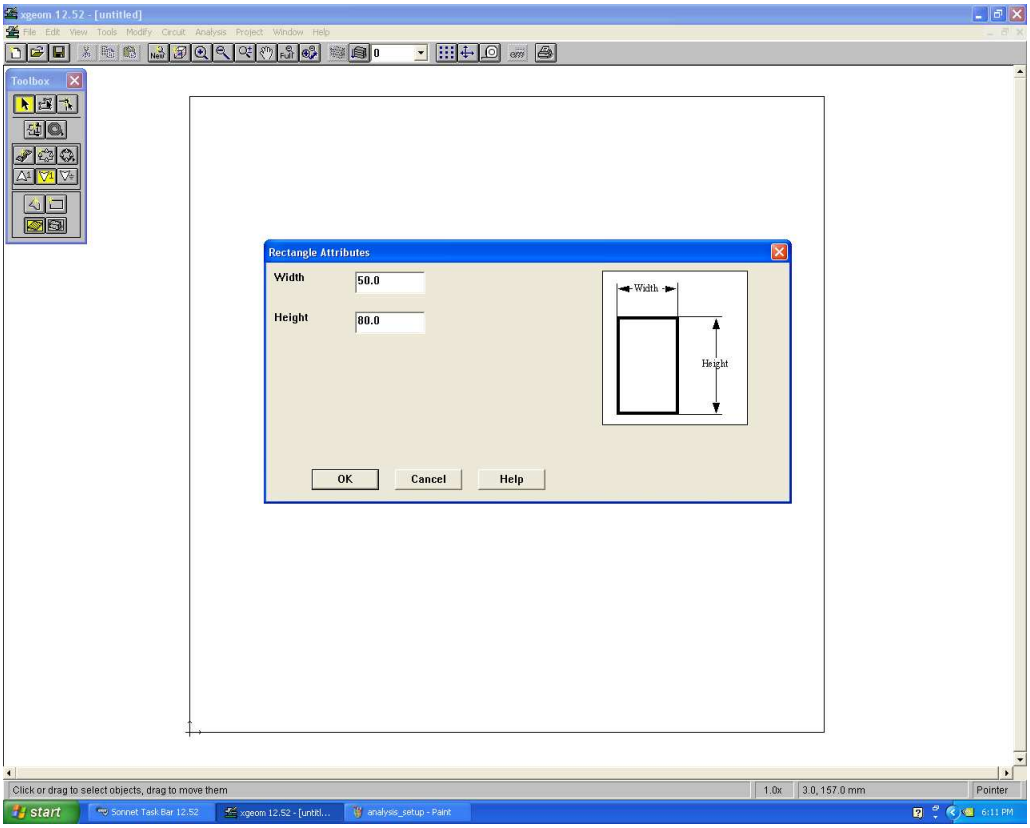

Figure A.9: Rectangle dimension selection

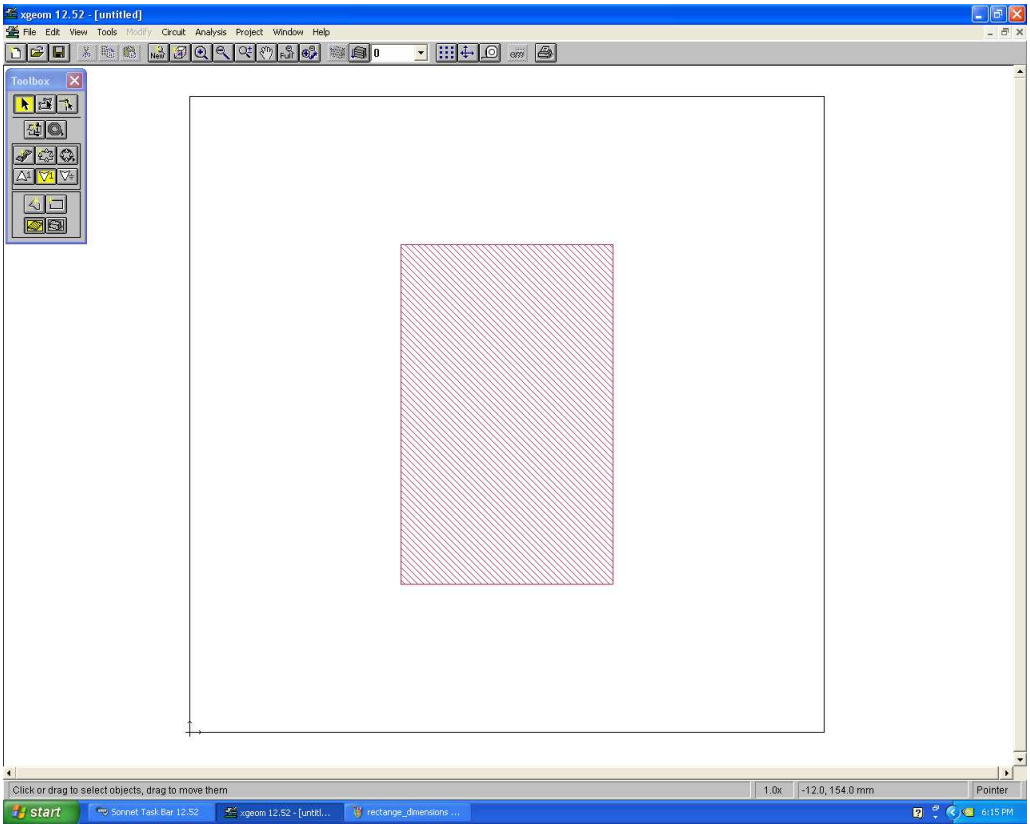

Figure A.10: Rectangle placement

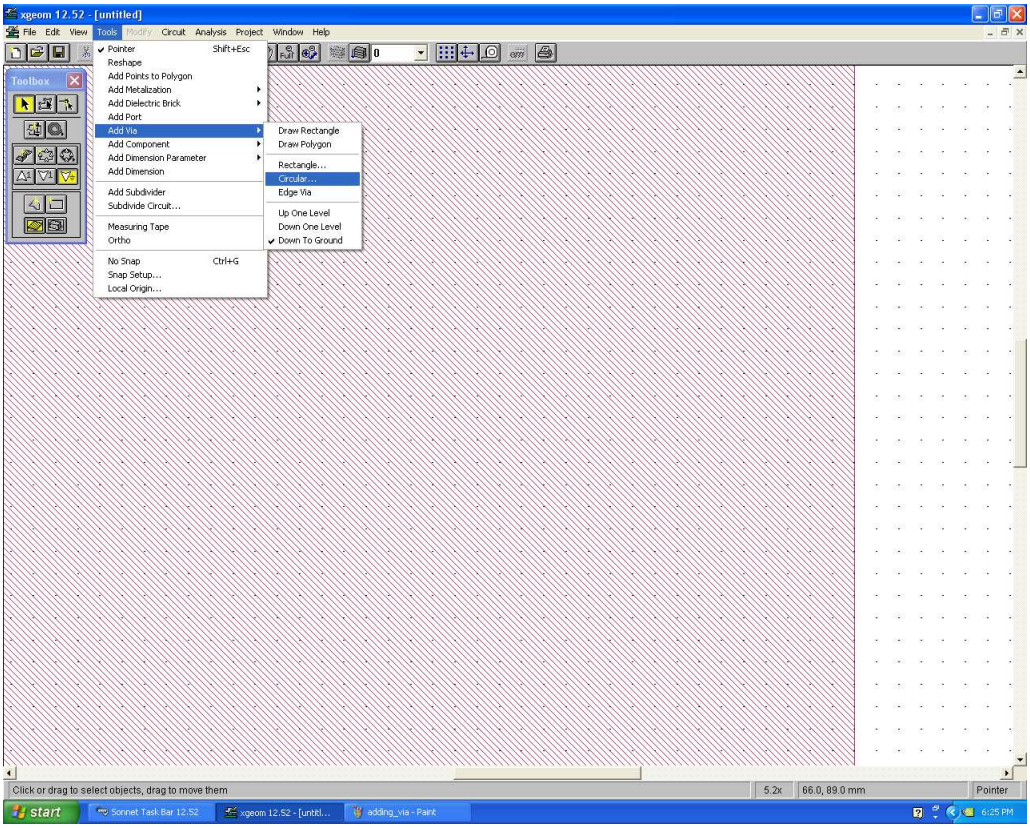

Figure A.11: Adding a Via

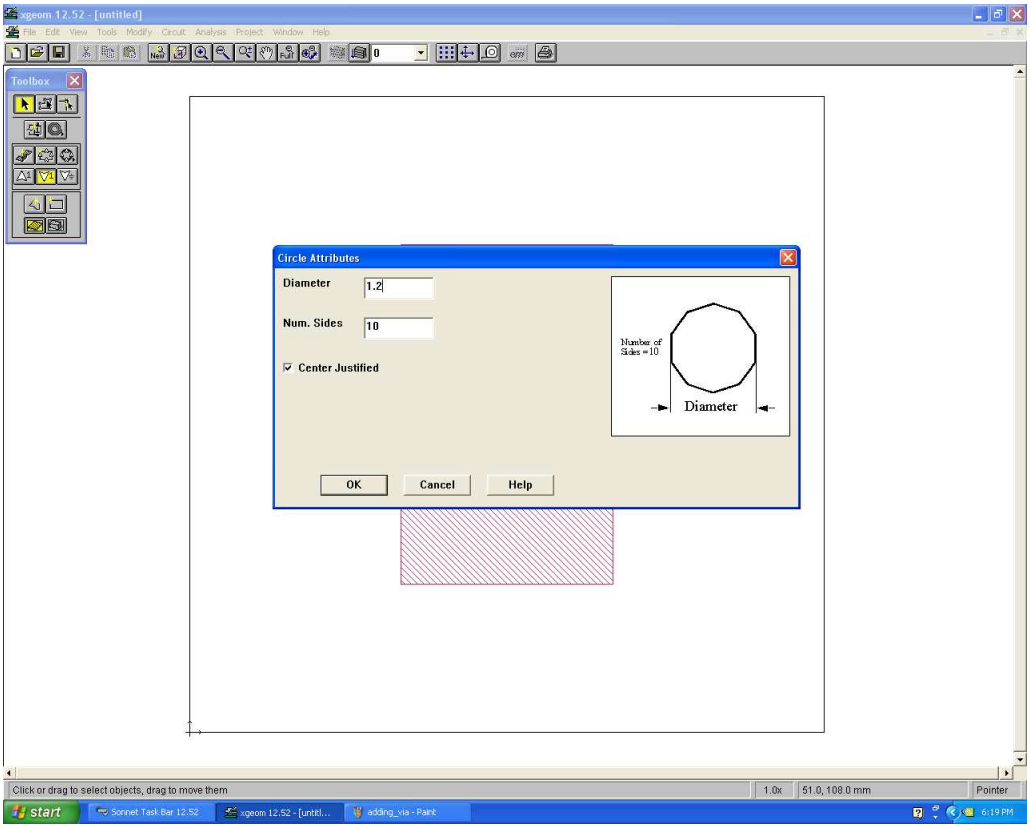

Figure A.12: Via circle parameter selection

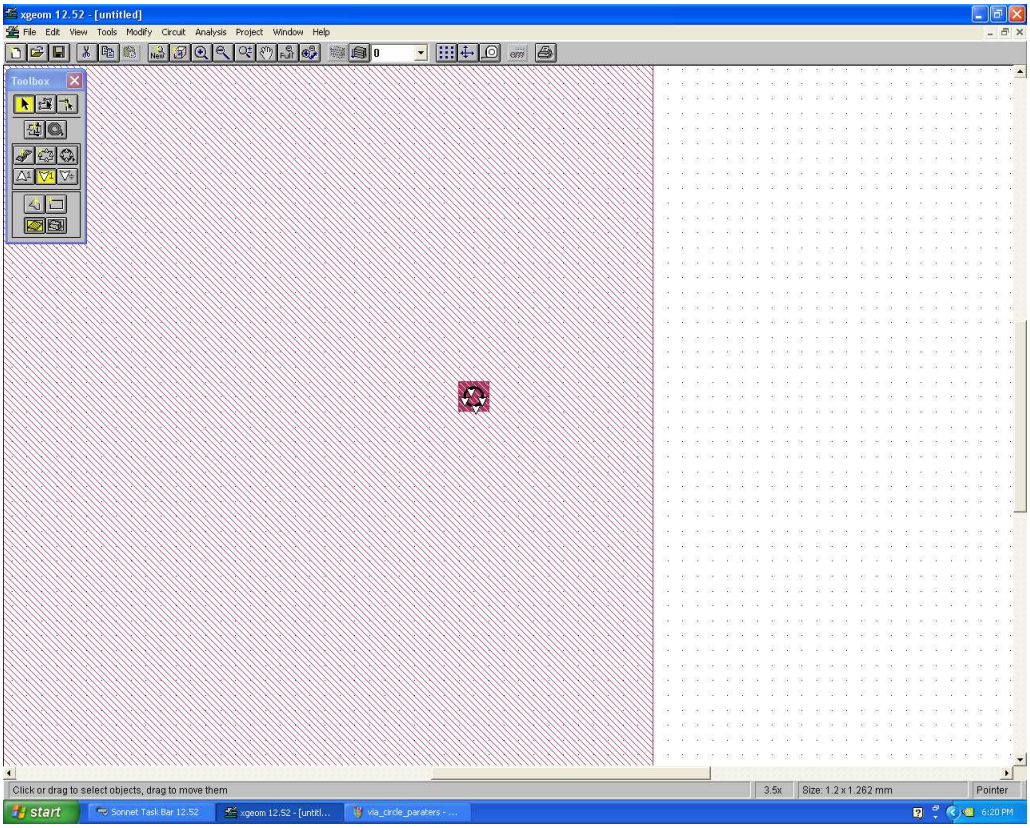

Figure A.13: Via placement on a rectangle

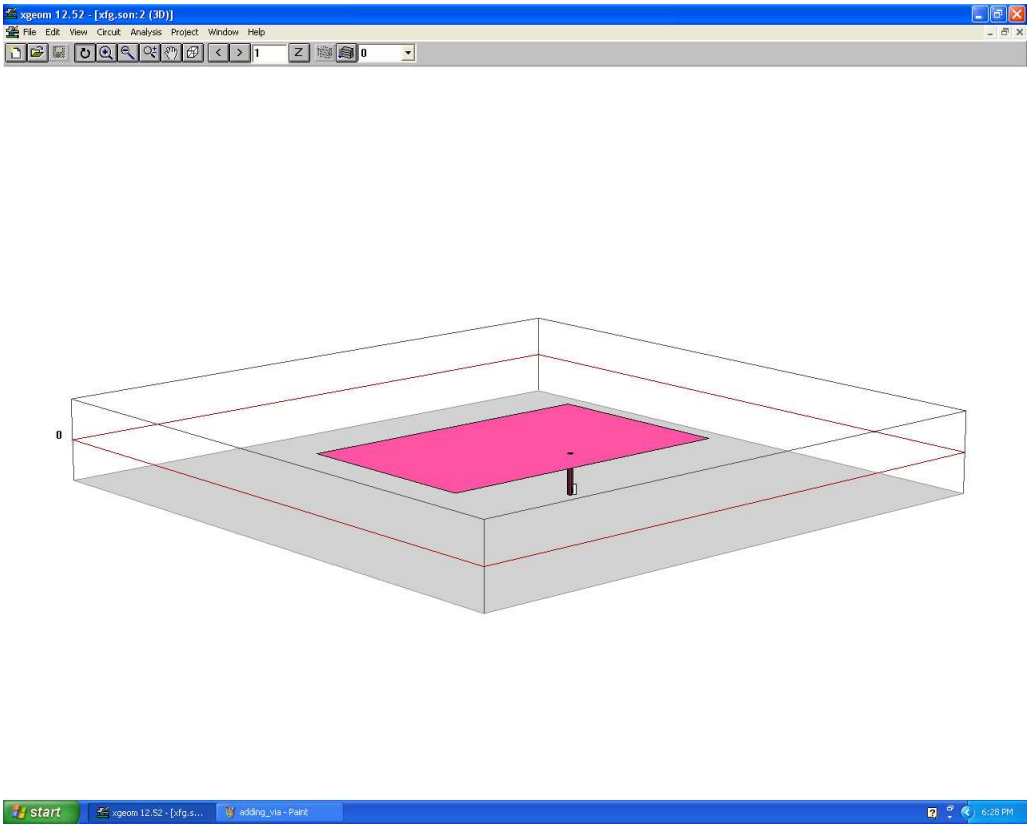

Figure A.14: 3-D view of a patch

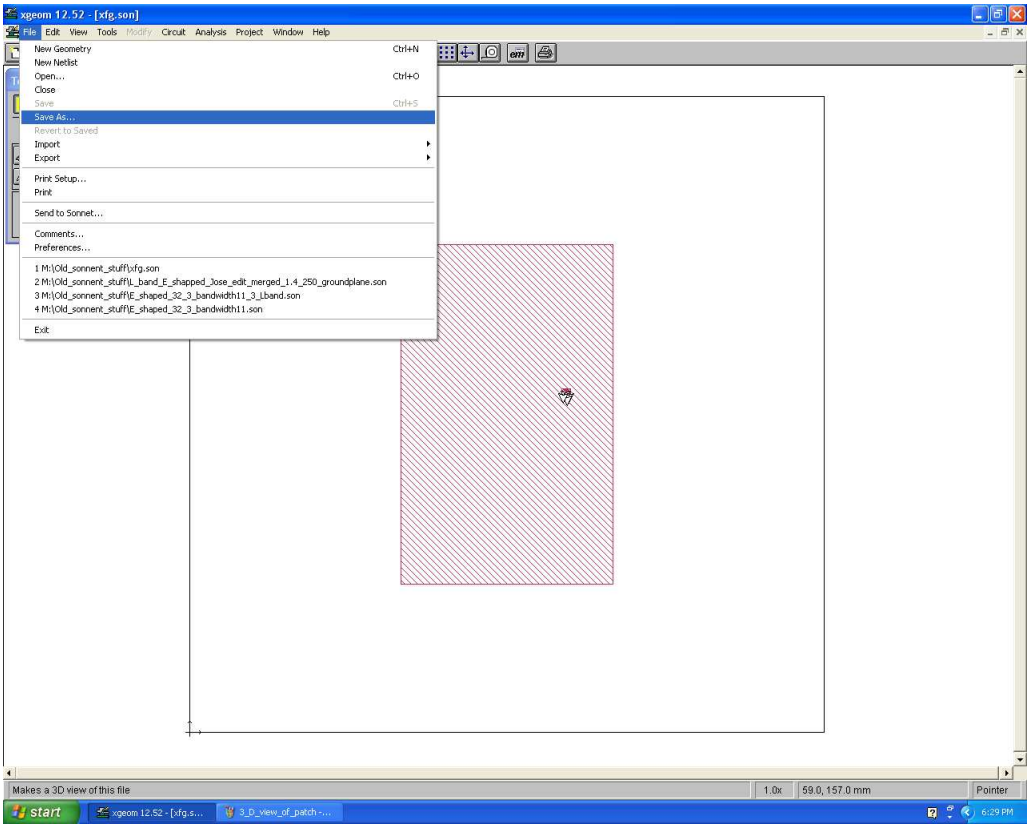

Figure A.15: Save as screen shot

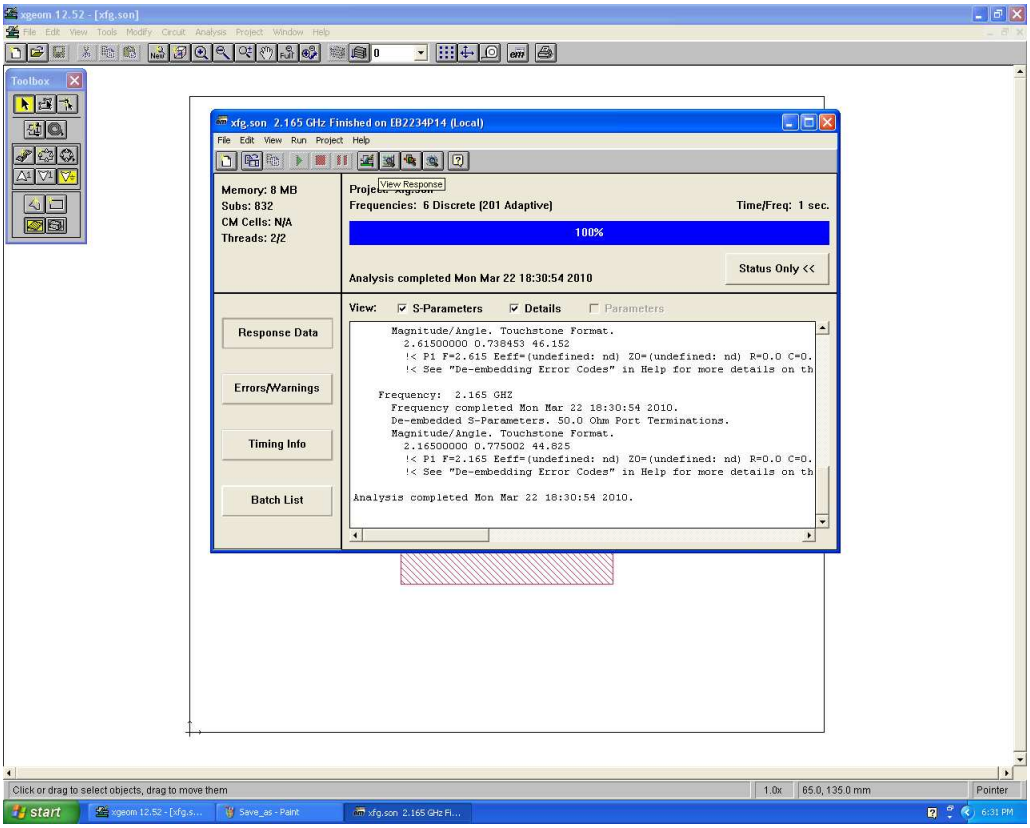

Figure A.16: Viewed Response Box

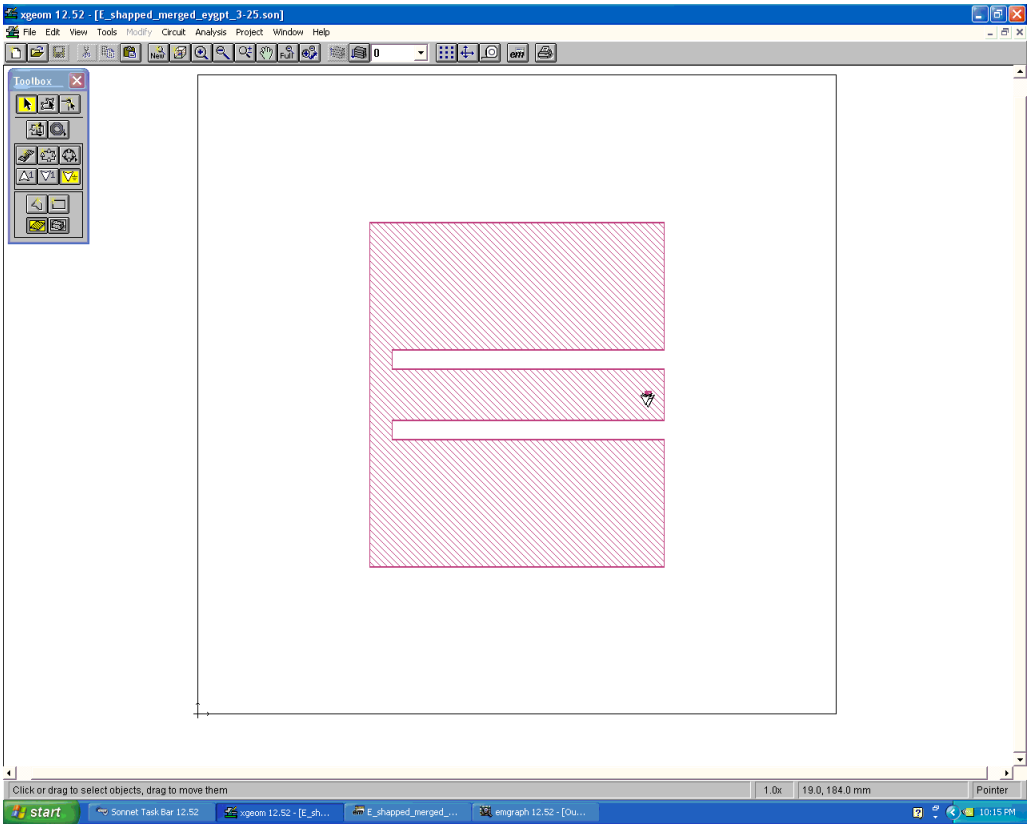

Figure A.17: Finished E-patch topology

Appendix B: Matlab Code for a two variable function

```
close all
clear all

[filename pathname] = uigetfile('*.txt','Choose the OA file from
    online or generated code MUST BE ACII Formated!');

OA=dlmread([pathname filename]);

SizeofOA=size(OA);
Element = 2;% input number of Elements to be tested

if SizeofOA(2) == Element

else
    newColummnsize=SizeofOA(2)-Element;
    for n=1:newColummnsize % reduces the columm size for expriment

        OA(:,SizeofOA(2))=[];

        SizeofOA=size(OA);
    end
end


% Level Design User specifies level
Level=0:2; %Input by user assuming 3 levels
lengthflevel=length(Level);
Max1=10; % Input by User
Min1=0; % Input by User
Max2=10; % Input by User
Min2=0; % Input by User
rr=.9; % Reduction rate used by User
i=0;
diffrentlevels=0:1; % Because not all levels will have the same
    range. Specified here is how many are different

Ld1= (Max1 - Min1)/(lengthflevel+1); % first specifed level differnce
    first interation
```

```
Ldi1=Ld1*rr^i;

Ld2= (Max2 - Min2)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi2=Ld2*rr^i;

while (Ldi1/Ld1)>.0001;

    clear  LevelOA2 LevelOA3 LevelOA1
    Level=0:2;
       clear n b g c


Ld1= (Max1 - Min1)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi1=(1.5*Ld1)*rr^i;

Ld2= (Max2 - Min2)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi2=(1.5*Ld2)*rr^i;



for n=1:SizeofOA(2)



    if i==0
    switch n
        case 1
         Level(2)= (Max1-Min1)/2 + Min1;% Middle of range
         Level(1)=Level(2)-Ldi1;
         Level(3)=Level(2)+Ldi1;

         LevelOA1(n)=Level(1);
         LevelOA2(n)=Level(2);
         LevelOA3(n)=Level(3);

        case 2
         Level(2)=(Max2-Min2)/2 + Min2; %Middle of range
         Level(1)=Level(2)-Ldi2;
         Level(3)=Level(2)+Ldi2;

         LevelOA1(n)=Level(1);
         LevelOA2(n)=Level(2);
         LevelOA3(n)=Level(3);
    end

    else

    switch n
        case 1
         Level(2)= OAr(n);
         Level(1)=Level(2)-Ldi1;
```

```
            Level(3)=Level(2)+Ldi1;

            if Level(1)≤Min1
            Level(1)=Min1;
            else
            end
            if Level(3)≥Max1
            Level(3)=Max1;
            else
            end
            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);

           case 2
            Level(2)= OAr(n);
            Level(1)=Level(2)−Ldi2;
            Level(3)=Level(2)+Ldi2;

            if Level(1)≤Min2
            Level(1)=Min2;
            else
            end
            if Level(3)≥Max2
            Level(3)=Max2;
            else
            end

            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);
    end



    end

end


countoflevel=0;


clear LevelOA


for n=1:SizeofOA(1) % Filling up a Level based OA

    for b=1:SizeofOA(2)


    if OA(n,b)==0
        LevelOA(n,b)=LevelOA1(b);
```

```
    elseif OA(n,b)==1
        LevelOA(n,b)=LevelOA2(b);
    else
        LevelOA(n,b)=LevelOA3(b);
    end

    end


end

clear FitnessF
for n=1:SizeofOA(1)




    %Fitness=Fitness+(LevelOA(n,g))^2-10*cos(2*pi*(LevelOA(n,g)))+10;
   %Fitness=1-abs(sin(pi*(LevelOA(n,1)-3))/(pi*(LevelOA(n,1)-3)))*abs
      (sin(pi*(LevelOA(n,2)-3))/(pi*(LevelOA(n,2)-3)));
   Fitness=exp((LevelOA(n,1)*sin(4*LevelOA(n,1))+1.1*LevelOA(n,2)*sin
      (2*LevelOA(n,2))));

FitnessF(n,1)=Fitness;
end

clear StoN

StoN=-20*log10((FitnessF)); % Signal to Noise Ratio
clear FinalResponse
%Build Response table


for n=1:SizeofOA(2)
    response=0;
    response2=0;
    response3=0;
    countoflevel=0;

    for h=1:SizeofOA(1)

    if OA(h,n)==0
        response=response+StoN(h);
        countoflevel=countoflevel+1;

    elseif OA(h,n)==1
        response2=response2+StoN(h);

    elseif OA(h,n)==2
        response3=response3+StoN(h);
    end

    end
```

```
    for c=1:lengthflevel

    if c==1
    FinalResponse(c,n)=response/countoflevel;

    elseif c==2
    FinalResponse(c,n)=response2/countoflevel;

    elseif c==3
    FinalResponse(c,n)=response3/countoflevel;
    end

    end

end

    clear Optparameters


       clear OptMat
Optparameters=max(FinalResponse);

for n=1:SizeofOA(2) % Matrix use to single out the good s/n's

    for c=1:lengthflevel

        if Optparameters(n)==FinalResponse(c,n)
            OptMat(c,n)=FinalResponse(c,n);
        else
            OptMat(c,n)=pi;
        end

    end
end
 clear OAr
for n=1:SizeofOA(2) % matrix used to specify final levels that are
   optimal

    for c=1:lengthflevel

        if OptMat(c,n)==pi;
        else
            switch c
                case 1
                    OAr(n)=LevelOA1(n);
                case 2
                    OAr(n)=LevelOA2(n);
                case 3
                    OAr(n)=LevelOA3(n);
            end

        end
    end
```

```
end

 % The design for this Fitness Function Raguchi for use only for
    Taguchi



%FitnessG=1-abs(sin(pi*(OAr(1)-3))/(pi*(OAr(1)-3)))*abs(sin(pi*(OAr
   (2)-3))/(pi*(OAr(2)-3)));
FitnessG=(OAr(1)*sin(4*OAr(1))+1.1*OAr(2)*sin(2*OAr(2)));

    meck(i+1)=(FitnessG);

%    if FitnessG >.9
%        break
%    else
i=i+1;
   Store(i+1)=OAr(1);
   Store2(i+1)=OAr(2);

end


x=[0:.01:10];
y=[0:.01:10];

ta=length(x);



for bah=1:ta

    for cah=1:ta
        meckfitness(cah,bah)=x(bah)*sin(4*x(bah))+1.1*y(cah)*sin(2*y(
           cah));

    end

end



 figure;

 mesh(x,y,meckfitness)
 xlabel('x values')
ylabel('y values')
title('global minimun')


meck(i+1)=FitnessG;
```

```
   plotness=1:length(meck);
figure,plot(plotness,meck);
h = legend('Final Fitness');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 27 expriements')
ylabel('Fitness')
title('Fitness vs Runs')


figure,plot(plotness,Store);
h = legend('x values');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('x values')
title('x values')

figure,plot(plotness,Store2);
h = legend('y vaules');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('y values')
title('y values')
```

Appendix C: Start up guide to K-brick

Contained in this appendix are a start up guide to K-brick and a sample input text file. Comments indicated by `^^^` should note appear in an actual input file.

K-brick uses the FE-BI equations to solve for the electric and magnetic fields of a cavity-backed patch antenna. To do this, the cavity is represented using a finite number of cells. The dimensions and number of cells are determined by the user. K-brick starts the cell topology in the lower left (LL) corner of the cavity. In the input file below, the length, width and depth of the cavity are 20 cm x 20 cm x 1.5 cm respectively. The corresponding input line in the text file is 200 200 1. The input line 200 200 1 means that each cell is $\frac{20}{200} = .1\,cm = 1\,mm$ by $\frac{20}{200} = .1\,cm = 1\,mm$ by h (h is the dielectric height, (`(Layer 1 thickness (cm))`of the cavity). Smaller cell sizes increase computation time considerably. For most simulations 1 mm x 1 mm x h cells will yield good results. The layer concept is discussed later in this appendix.

As discussed earlier, K-brick begins constructing the cavity out of the cells starting in LL corner. Due to cells being used there isn't a 0 cell, so the cell numbering starts at 1. This also means that the input into K-brick for cells has to be a whole number. This is important for when metallic patches are added in the input file. Infinitesimally thin rectangular patches are created and placed by cell number within the cavity. In the example text file below, four rectangular patches are added to the top layer of

the cavity. The numbers for placement of these patches are cell positions, and are not in x, y, z coordinates. For example, the cell size in the text file below is 1 mm by 1 mm by 1.5 cm. The patches are each placed by first specifying where they start to enlarge. Patches enlarge from left to right and down to up. So the first patch's LL corner in the text file below is at the 60th cell from the left and 52nd from the bottom of the cavity; at that cell point the patch enlarges by 10 cells to the right and 96 cells to the top of the cavity. This is repeated 4 times until the proper topology of the patch is created. The probe is not placed using cell length or position. The probe is placed relative to the exact middle of the cavity instead of the LL corner of the cavity like metallic patches, material blocks, or metal bricks are. This means that probe placement and h dielectric height can be a real integer. In the example below the probe is placed 3.17 cm to the right, and 0 cm from the middle (center) of the cavity.

K-brick also has the ability to layer the dielectric in the cavity. Typically this is done if metal bricks are added to the simulation in the input file. Metal bricks are perfect electric conductors that are composed of cells (same cell size as initialized in the input file). Metal blocks are constructed from the bottom of the cavity to one layer below where the patch antenna resides. Metal blocks are added so that the probe can rest on top of the metal blocks. The probe is placed on top of the metal blocks to yield results closer to those of a typical coaxial vertical fed probe. However, when metal blocks are added in this way, a considerable capacitance is added and this yields a poor $|S_{11}|$ calculated by the probe. Because of the added capacitance of the metal bricks, all the simulations done in K-brick did not have metal bricks included. At each point of resonance in the E-patch simulation, a sign change was noted in the imaginary portion of the impedance calculated at the probe, indicating a good simulated result. If one were to use metal bricks, they would be added exactly the same way a patch is, i.e by cell position and number.

The rest of the input file is self-explanatory except the convergence tolerance, minimum iterations, and maximum iterations, which deal how close the numerical solutions in the FE-BI equations need to get to before K-brick terminates operation and outputs simulated antenna parameters. The convergence tolerances for the problems considered in this thesis are 0.01 5 60000, which seem to be sufficient.

```
1
^^^ 1 = save geo file, 2=read it, 0=do nothing
patch.geo
^^^ filename for geometry
20.0 20.0
^^^ (x,y) cavity dimensions in cm
200 200 1
^^^ x,y,z dimension cells
0
^^^ 0 = cavity, 1=slot
1.50
^^^ Layer 1 thickness (cm)
4
^^^ Number of patches
60 52 1
^^^ Lower left corner of patch + layer
10 96
^^^ Patch dimension in edges
60 52 1
^^^ Lower left corner of patch + layer
81 33
^^^ Patch dimension in edges
60 93 1
^^^ Lower left corner of patch + layer
81 14
^^^ Patch dimension in edges
60 115 1
^^^ Lower left corner of patch + layer
81 33
^^^ Patch dimension in edges
0
^^^ Number of slots
0
```

```
^^^ Number of shorting pins
0
^^^ Number of metal blocks
0
^^^ Number of material blocks
1
^^^ 1 = uniform material fill
0
^^^ 0 = non-dispersive, 1 = dispersive
1.0 -0.00
^^^ Complex epsilon (real, imaginary)
1.0 -0.00
^^^ Complex mu (real, imaginary)
0.01 5 60000
^^^ convergence tolerance, min. iterations, max. iterations
0
^^^ 0 = BiCG, 1 = QMR
0
^^^ 1 = monitor iterations
0
^^^ 0=initial guess, 1=automatic guess
1.227 1.6 .348
^^^ Start, Stop, INCR frequency [GHz]
0 0 1
^^^ Start, Stop, Incr phi angles [deg]
-90.0 90.0 1.0
^^^ Start, Stop, Incr theta angles [deg]
0
^^^ 0 = do not compute far-zone, 1=compute far-zone fields
patchX_test
^^^ Six (6) character file prefix
0
^^^ 0 = driven, 1=S-paramters, 2=bistatic
E_patch_chad_impedance_Opt.zin
^^^ Input impedance file
50.0 0.0
^^^ Reference impedance
1
^^^ Number of feeds
3
^^^ direction of feed (1=x, 2=y, 3=z)
3.17 0.00
^^^ (x,y) location of feed in cm
1
^^^ Layer of feed
```

```
1.0 0.0
^^^ Magnitude and phase of current
0
^^^ Number of loads
0
^^^ 0 = do NOT save interior fields, 1= save surface fields
0
^^^ Number of field probe
```

Below are the steps taken to get K-brick running on a Unix machine at Michigan State University.

1. Obtain fortran code from Professor Kempel.

2. Create a folder in the Unix station to house input, output, and fortran files.

3. Once the files have been successfully uploaded, type `make` in the command line (you should still be in the same directory where the fortran files are located). This builds and complies the fortran program K-brick. If the program fails to compiles, double check the .mak file. Make sure it uses the proper compiler for the unix station you are using, i.e pgf90, f90, or f77 compilers.

4. Once K-brick has successfully compiled, obtain or create a input text file such as that given earlier in the appendix.

5. K-brick can output many simulated antenna parameters such as radiation pattern, impedance measured at the probe, and magnitude and phase of the electric and magnetic fields. Generally you would want have two different types of input files, one for a frequency sweep of the impedance calculated at the probe, and a second one for the radiation pattern, electric field, and magnetic field at a particular frequency. If you did a frequency sweep that had 100 frequencies, the there would be 100 different radiation patterns and 100 different electric and magnetic magnitude and phase field files. K-brick would

take too long to compute your simulated antenna parameters. Once you have determined what you would like for an output, modify your input file at the appropriate line. For impedance frequency sweeps change the `Start, Stop, INCR frequency [GHz]` to the range you require. For a test case use `1 2 .01`. This will make K-brick simulate the antenna at 100 frequencies between 1 and 2 GHz. Also change `Start, Stop, Incr phi angles [deg]` to `0 0 1` and `Start, Stop, Incr theta angles [deg]` to `-90 90 1`. This will prevent K-brick from computing any E-fields, H-fields, or radiation pattern files because the angle range is zero. To get the electric field (magnitude and phase), magnetic field(magnitude and phase), and radiation pattern files, do the opposite. Fix the `Start, Stop, INCR frequency [GHz]` to sweep one frequency (e.g 2, 2, and 1) and vary the `Start, Stop, Incr phi angles [deg]` and `Start, Stop, Incr theta angles [deg]`.

6. After the input file has been modified, now is the time to run K-brick. The Unix command `nohup` will make K-brick run even if you log out of the Unix machine. It is a useful command when K-brick takes a long time to run. A typical run command for K-brick looks like this: `nohup k_brick<inputfile.txt > out.txt &`. This will make K-brick run in the back round of the server and will allow exit of the server while the program still runs. `out.txt` is an output file containing basic run logs of the inputs from the input file. It also contains impedance sweep information.

7. As mentioned earlier K-brick outputs many antenna parameters. For a frequency sweep that contains impedance information at the probe refer to the .zin file. The 0 above the line, `^^^0 = do not compute far-zone, 1=compute far-zone fields` is what is changed to calculate the radiation pattern. For the magnitude and phase of the electric and magnetic fields the 0 above the line,

`^^^0 = do NOT save interior fields, 1= save surface fields` is what is changed. Then, specify the component of the fields desired.

Appendix D: Matlab Imaging Code for GUI

```
function varargout = gui(varargin)
% GUI M-file for gui.fig
%      GUI, by itself, creates a new GUI or raises the existing
%      singleton*.
%
%      H = GUI returns the handle to a new GUI or the handle to
%      the existing singleton*.
%
%      GUI('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in GUI.M with the given input
   arguments.
%
%      GUI('Property','Value',...) creates a new GUI or raises the
%      existing singleton*.  Starting from the left, property value
   pairs are
%      applied to the GUI before gui_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property
   application
%      stop.  All inputs are passed to gui_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows
   only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui

% Last Modified by GUIDE v2.5 28-Oct-2009 01:13:25

% Begin initialization code - DO NOT EDIT

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_OpeningFcn, ...
```

```
                   'gui_OutputFcn',  @gui_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui is made visible.
function gui_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui (see VARARGIN)

% Choose default command line output for gui
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit1
    as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2
    as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu1.
function popupmenu1_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu1 contents
    as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from
    popupmenu1
if isequal(get(hObject,'Value'),1)
    set(handles.text2,'Visible','off');
    set(handles.edit2,'Visible','off');
else
    set(handles.text2,'Visible','on');
    set(handles.edit2,'Visible','on');
end


% --- Executes during object creation, after setting all properties.
function popupmenu1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: popupmenu controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3
    as a double


% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global results;
sprintf('%10s%10s%10s%10s%10s%10s%10s%10s%10s%10s','1.227 GHz','1.575
    GHz','Width','Length  ','SlotWidth ','SlotLength','Ps','Probe X',
   'Probe Y','Height')
if isequal(get(handles.popupmenu1,'Value'),1)
    lmin=str2num(get(handles.edit1,'String'));
%    lmin=round(str2num(lmin{1}));
    lmax=lmin;
else
    lmin=str2num(get(handles.edit1,'String'));
%    lmin=round(str2num(lmin{1}));
    lmax=str2num(get(handles.edit2,'String'));
%    lmax=round(str2num(lmax{1}));
end
if isequal(get(handles.popupmenu2,'Value'),1)
    wmin=str2num(get(handles.edit3,'String'));
    wmax=wmin;
else
    wmin=str2num(get(handles.edit3,'String'));
    wmax=str2num(get(handles.edit4,'String'));
end
if isequal(get(handles.popupmenu3,'Value'),1)
    slmin=str2num(get(handles.edit5,'String'));
    slmax=slmin;
else
    slmin=str2num(get(handles.edit5,'String'));
    slmax=str2num(get(handles.edit6,'String'));
end
if isequal(get(handles.popupmenu4,'Value'),1)
    swmin=str2num(get(handles.edit7,'String'));
    swmax=swmin;
else
    swmin=str2num(get(handles.edit7,'String'));
    swmax=str2num(get(handles.edit8,'String'));
end
if isequal(get(handles.popupmenu5,'Value'),1)
    psmin=str2num(get(handles.edit9,'String'));
    psmax=psmin;
else
    psmin=str2num(get(handles.edit9,'String'));
    psmax=str2num(get(handles.edit10,'String'));
```

```
end
if isequal(get(handles.popupmenu6,'Value'),1)
    xmin=str2num(get(handles.edit11,'String'));
    xmax=xmin;
    xstep=.05;
else
    xmin=str2num(get(handles.edit11,'String'));
    xmax=str2num(get(handles.edit12,'String'));
    xstep=str2num(get(handles.edit15,'String'));
end
if isequal(get(handles.popupmenu7,'Value'),1)
    hmin=str2num(get(handles.edit13,'String'));
    hmax=hmin;
    hstep=.05;
else
    hmin=str2num(get(handles.edit13,'String'));
    hmax=str2num(get(handles.edit14,'String'));
    hstep=str2num(get(handles.edit16,'String'));
end



num=1;
% E-patch length
for l=lmin:lmax
    % E-patch width (must be even)
for w0=wmin:2:wmax
% slot length
for sl=slmin:slmax
% slot width (must be even)
for sw=swmin:swmax
% Ps (distance from line of symmetry to middle of slot)
for ps=psmin:psmax
% Probe x placement (cm)
for probex=xmin:xstep:xmax
% Dielectric Height
for h=hmin:hstep:hmax

% Define all patches

if not(mod(sw,2)==mod(w0,2))
    w=w0+1;
else
    w=w0;
end


x1=round((200-l)/2);
l1=l-sl;
y1=round((200-w)/2);
w1=w;
x2=round((200-l)/2);
l2=l;
y2=round((200-w)/2);
```

```
w2=round(w/2-sw/2-ps);
x3=round((200-l)/2);
l3=l;
y3=round(100+sw/2-ps);
w3=2*ps-sw;
x4=round((200-l)/2);
l4=l;
y4=round(100+ps+sw/2);
w4=round(w/2-sw/2-ps);
sw1=y3-y2-w2;
sw2=y4-y3-w3;
probey=mod(w3,2)/20


% figure;
% rectangle('position',[0 0 200 200]);
% hold on;
% fill([x1 x1 x1+l1 x1+l1],[y1 y1+w1 y1+w1 y1],'r');
% fill([x2 x2 x2+l2 x2+l2],[y2 y2+w2 y2+w2 y2],'r');
% fill([x3 x3 x3+l3 x3+l3],[y3 y3+w3 y3+w3 y3],'r');
% fill([x4 x4 x4+l4 x4+l4],[y4 y4+w4 y4+w4 y4],'r');
% plot(probex*10+100,probey*10+100,'--rs','MarkerFaceColor','g','
   MarkerSize',5);
%



    % output kbrick input file
    fid=fopen(['text.txt'],'w');
    fprintf(fid,'1\n^^^ 1 = save geo file, 2=read it, 0=do nothing\
       npatch.geo\n^^^ filename for geometry\n20.0 20.0\n^^^ (x,y)
       cavity dimensions in cm\n200 200 1\n^^^ x,y,z dimension cells\
       n0\n^^^ 0 = cavity, 1=slot\n');
    fprintf(fid,'%.2f\n^^^ Layer 1 thickness (cm)\n',h);
    fprintf(fid,'4\n^^^ Number of patches\n');
    fprintf(fid,'%d %d 1\n^^^ Lower left corner of patch + layer\n',[
       x1; y1]);
    fprintf(fid,'%d %d\n^^^ Patch dimension in edges\n',[l1; w1]);
    fprintf(fid,'%d %d 1\n^^^ Lower left corner of patch + layer\n',[
       x2; y2]);
    fprintf(fid,'%d %d\n^^^ Patch dimension in edges\n',[l2; w2]);
    fprintf(fid,'%d %d 1\n^^^ Lower left corner of patch + layer\n',[
       x3; y3]);
    fprintf(fid,'%d %d\n^^^ Patch dimension in edges\n',[l3; w3]);
    fprintf(fid,'%d %d 1\n^^^ Lower left corner of patch + layer\n',[
       x4; y4]);
    fprintf(fid,'%d %d\n^^^ Patch dimension in edges\n',[l4; w4]);
    fprintf(fid,'0\n^^^ Number of slots\n0\n^^^ Number of shorting
       pins\n0\n^^^ Number of metal blocks\n0\n^^^ Number of material
        blocks\n1\n^^^ 1 = uniform material fill\n0\n^^^ 0 = non-
       dispersive, 1 = dispersive\n1.0 -0.00\n^^^ Complex epsilon (
       real, imaginary)\n1.0 -0.00\n^^^ Complex mu (real, imaginary)\
       n');
```

```
    fprintf(fid,'0.01 5 60000\n^^^ convergence tolerance, min.
       iterations, max. iterations\n0\n^^^ 0 = BiCG, 1 = QMR\n0\n^^^
       1 = monitor iterations\n0\n^^^ 0=initial guess, 1=automatic
       guess\n1.227 1.6 .348\n^^^ Start, Stop, INCR frequency [GHz]\
       n0 0 1\n^^^ Start, Stop, Incr phi angles [deg]\n-90.0 90.0 1.0
       \n^^^ Start, Stop, Incr theta angles [deg]\n0\n^^^ 0 = do not
       compute far-zone, 1=compute far-zone fields\npatchX_test\n^^^
       Six (6) character file prefix\n0\n^^^ 0 = driven, 1=S-
       paramters, 2=bistatic\nE_patch_chad_impedance_Opt.zin\n^^^
       Input impedance file\n');
    fprintf(fid,'50.0 0.0\n^^^ Reference impedance\n1\n^^^ Number of
       feeds\n3\n^^^ direction of feed (1=x, 2=y, 3=z)\n');
    fprintf(fid,'%.2f %.2f\n^^^ (x,y) location of feed in cm\n',[
       probex; probey]);
    fprintf(fid,'1\n^^^ Layer of feed\n1.0 0.0\n^^^ Magnitude and
       phase of current\n0\n^^^ Number of loads\n0\n^^^ 0 = do NOT
       save interior fields, 1= save surface fields\n0\n^^^ Number of
        field probe\n');
    fclose(fid);

    %run kbrick
    !kbrick<text.txt >out.txt

    %input db's from kbrick output file
    fid=fopen('out.txt','rt');
while feof(fid) == 0
    clear tline;
    tline = fgetl(fid);
    if length(tline)>6
    if tline(1:7)==' 1.2270'
        db1=str2num(tline(length(tline)-5:length(tline)));
    end
    if tline(1:7)==' 1.5750'
        db2=str2num(tline(length(tline)-5:length(tline)));
    end
    end
end
fclose(fid);


    %save results
    results(num,1)=db1;
    results(num,2)=db2;
    results(num,3)=w;
    results(num,4)=l;
    results(num,5)=sw;
    results(num,6)=sl;
    results(num,7)=ps;
    results(num,8)=probex;
    results(num,9)=probey;
    results(num,10)=h;
    disp(results(num,:))
    num=num+1;
    % !rm out.txt
```

```
end
end
end
end
end
end
end
fid=fopen(['results.txt'],'w');
fprintf(fid,'%10s%10s%10s%10s%10s%10s%10s%10s%10s%10s\n','1.227 GHz',
    '1.575 GHz','Width','Length  ','SlotWidth ','SlotLength','Ps','
    Probe X','Probe Y','Height');
for i=1:num-1
fprintf(fid,'%10g%10g%10g%10g%10g%10g%10g%10g%10g%10g\n',results(i,:)
    );
end
fclose(fid);



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)




function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4
    as a double


% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on selection change in popupmenu2.
function popupmenu2_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu2 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from
   popupmenu2
if isequal(get(hObject,'Value'),1)
    set(handles.text4,'Visible','off');
    set(handles.edit4,'Visible','off');
else
    set(handles.text4,'Visible','on');
    set(handles.edit4,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: popupmenu controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5
   as a double


% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called
```

```
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6
    as a double


% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu3.
function popupmenu3_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu3 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from
    popupmenu3
if isequal(get(hObject,'Value'),1)
    set(handles.text6,'Visible','off');
    set(handles.edit6,'Visible','off');
else
    set(handles.text6,'Visible','on');
    set(handles.edit6,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu3_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to popupmenu3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: popupmenu controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7
   as a double


% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8
   as a double


% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu4.
function popupmenu4_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu4 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from
   popupmenu4
if isequal(get(hObject,'Value'),1)
    set(handles.text8,'Visible','off');
    set(handles.edit8,'Visible','off');
else
    set(handles.text8,'Visible','on');
    set(handles.edit8,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: popupmenu controls usually have a white background on
   Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit9 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit9
    as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10
    as a double


% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu5.
function popupmenu5_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu5 contents
    as cell array
```

```
%        contents{get(hObject,'Value')} returns selected item from
    popupmenu5
if isequal(get(hObject,'Value'),1)
    set(handles.text10,'Visible','off');
    set(handles.edit10,'Visible','off');
else
    set(handles.text10,'Visible','on');
    set(handles.edit10,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: popupmenu controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit11_Callback(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit11 as text
%        str2double(get(hObject,'String')) returns contents of edit11
     as a double


% --- Executes during object creation, after setting all properties.
function edit11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit12_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12
    as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu6.
function popupmenu6_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu6 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from
   popupmenu6
if isequal(get(hObject,'Value'),1)
    set(handles.text12,'Visible','off');
    set(handles.edit12,'Visible','off');
    set(handles.text15,'Visible','off');
    set(handles.edit15,'Visible','off');
else
    set(handles.text12,'Visible','on');
    set(handles.edit12,'Visible','on');
    set(handles.text15,'Visible','on');
    set(handles.edit15,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called
```

```
% Hint: popupmenu controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit13_Callback(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit13 as text
%        str2double(get(hObject,'String')) returns contents of edit13
    as a double


% --- Executes during object creation, after setting all properties.
function edit13_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit14_Callback(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit14 as text
%        str2double(get(hObject,'String')) returns contents of edit14
    as a double


% --- Executes during object creation, after setting all properties.
function edit14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: edit controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on selection change in popupmenu7.
function popupmenu7_Callback(hObject, eventdata, handles)
% hObject    handle to popupmenu7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popupmenu7 contents
    as cell array
%        contents{get(hObject,'Value')} returns selected item from
    popupmenu7
if isequal(get(hObject,'Value'),1)
    set(handles.text14,'Visible','off');
    set(handles.edit14,'Visible','off');
    set(handles.text16,'Visible','off');
    set(handles.edit16,'Visible','off');
else
    set(handles.text14,'Visible','on');
    set(handles.edit14,'Visible','on');
    set(handles.text16,'Visible','on');
    set(handles.edit16,'Visible','on');
end

% --- Executes during object creation, after setting all properties.
function popupmenu7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popupmenu7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
    called

% Hint: popupmenu controls usually have a white background on
    Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
    defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end




function edit15_Callback(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit15 as text
```

```
%        str2double(get(hObject,'String')) returns contents of edit15
    as a double


% --- Executes during object creation, after setting all properties.
function edit15_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit16_Callback(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit16 as text
%        str2double(get(hObject,'String')) returns contents of edit16
    as a double


% --- Executes during object creation, after setting all properties.
function edit16_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns
   called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'
   defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

Appendix E: Matlab Imaging Code for Taguchi-K-brick-Epatch

```
close all
clear all
format short

db1=0;
db2=0;
% 1 Epatch Width
% 2 Epatch Lenght
% 3 Slot Length
% 4 Slot Width
% 5 Ps
% 6 Dielectric Height
% 7 Probe placement x direction

[filename pathname] = uigetfile('*.txt','Choose the OA file from
    online or generated code MUST BE ACII Formated!');

OA=dlmread([pathname filename]);

SizeofOA=size(OA);
Element = 7;% input number of Elements to be tested

if SizeofOA(2) == Element

else
    newColummnsize=SizeofOA(2)-Element;
    for n=1:newColummnsize % reduces the columm size for expriment

        OA(:,SizeofOA(2))=[];

        SizeofOA=size(OA);
    end
end
A=1; % factor for db1
```

```
B=1; % factor for db2

% Level Design User specifies level
Level=0:2; %Input by user assuming 3 levels
lengthflevel=length(Level);
Max1=110; % Input by User patch width in mm
Min1=60; % Input by User patch width in mm

Max2=110; % Input by User patch lenght in mm
Min2=60; % Input by User patch lenght in mm

Max3=95; % Input by User slot lenght in mm
Min3=50; % Input by User slot lenght in mm

Max4=10; % Input by User slot width in mm
Min4=1; % Input by User slot width in mm

Max5=25; % Input by User Ps in mm
Min5=5; % Input by User Ps in mm

Max6=1.5; % Input by User dielectric hight in cm
Min6=.1; % Input by User dielctric height in cm

Max7=7; % Input by User probe placment x in cm
Min7=0; % Input by User probe placement x in cm

rr=.9; % Reduction rate used by User
i=0;
diffrentlevels=0:1; % Because not all levels will have the same
   range. Specified here is how many are different
LDtime=1.5;

Ld1= LDtime*(Max1 - Min1)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld2= LDtime*(Max2 - Min2)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld3= LDtime*(Max3 - Min3)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld4= LDtime*(Max4 - Min4)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld5=LDtime*(Max5 - Min5)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld6= LDtime*(Max6 - Min6)/(lengthflevel+1); % first specifed level
   differnce first interation
```

```
Ld7=LDtime*(Max7 - Min7)/(lengthflevel+1); % first specifed level
   differnce first interation


Ldi1=round(Ld1*rr^i);

Ldi2=round(Ld2*rr^i);

Ldi3=round(Ld3*rr^i);

Ldi4=round(Ld4*rr^i);

Ldi5=round(Ld5*rr^i);

Ldi6=(Ld6*rr^i);

Ldi7=(Ld7*rr^i);




while (Ldi1/Ld1)>.0001;

    clear Level LevelOA2 LevelOA3 LevelOA1 db1 db2
    Level=0:2;
       clear n b g c


Ldi1=round(Ld1*rr^i);

Ldi2=round(Ld2*rr^i);

Ldi3=round(Ld3*rr^i);

Ldi4=round(Ld4*rr^i);

Ldi5=round(Ld5*rr^i);

Ldi6=(Ld6*rr^i);

Ldi7=(Ld7*rr^i);




for n=1:SizeofOA(2)
```

```
if i==0
        switch n

            case 1
     Level(2)= (Max1-Min1)/2 + Min1;% Middle of range
     Level(1)=Level(2)-Ldi1;
     Level(3)=Level(2)+Ldi1;
            case 2
                Level(2)= (Max2-Min2)/2 + Min2;% Middle of range
     Level(1)=Level(2)-Ldi2;
     Level(3)=Level(2)+Ldi2;
            case 3
                Level(2)= (Max3-Min3)/2 + Min3;% Middle of range
     Level(1)=Level(2)-Ldi3;
     Level(3)=Level(2)+Ldi3;
            case 4
                 Level(2)= (Max4-Min4)/2 + Min4;% Middle of range
     Level(1)=Level(2)-Ldi4;
     Level(3)=Level(2)+Ldi4;
            case 5
                Level(2)= (Max5-Min5)/2 + Min5;% Middle of range
     Level(1)=Level(2)-Ldi5;
     Level(3)=Level(2)+Ldi5;
            case 6
                Level(2)= (Max6-Min6)/2 + Min6;% Middle of range
     Level(1)=Level(2)-Ldi6;
     Level(3)=Level(2)+Ldi6;
            case 7
                Level(2)= (Max7-Min7)/2 + Min7;% Middle of range
     Level(1)=Level(2)-Ldi7;
     Level(3)=Level(2)+Ldi7;
        end

     LevelOA1(n)=Level(1);
     LevelOA2(n)=Level(2);
     LevelOA3(n)=Level(3);

else

switch n
    case 1
     Level(2)= OAr(n);
     Level(1)=Level(2)-Ldi1;
     Level(3)=Level(2)+Ldi1;

     if Level(1)≤Min1
     Level(1)=Min1;
     else
     end
     if Level(3)≥Max1
     Level(3)=Max1;
     else
     end
```

```
LevelOA1(n)=Level(1);
LevelOA2(n)=Level(2);
LevelOA3(n)=Level(3);

case 2
Level(2)= OAr(n);
Level(1)=Level(2)−Ldi2;
Level(3)=Level(2)+Ldi2;

if Level(1)≤Min2
Level(1)=Min2;
else
end
if Level(3)≥Max2
Level(3)=Max2;
else
end
LevelOA1(n)=Level(1);
LevelOA2(n)=Level(2);
LevelOA3(n)=Level(3);

      case 3
Level(2)= OAr(n);
Level(1)=Level(2)−Ldi3;
Level(3)=Level(2)+Ldi3;

if Level(1)≤Min3
Level(1)=Min3;
else
end
if Level(3)≥Max3
Level(3)=Max3;
else
end
LevelOA1(n)=Level(1);
LevelOA2(n)=Level(2);
LevelOA3(n)=Level(3);
      case 4
Level(2)= OAr(n);
Level(1)=Level(2)−Ldi4;
Level(3)=Level(2)+Ldi4;

if Level(1)≤Min4
Level(1)=Min4;
else
end
if Level(3)≥Max4
Level(3)=Max4;
else
end
LevelOA1(n)=Level(1);
LevelOA2(n)=Level(2);
LevelOA3(n)=Level(3);
      case 5
```

```
        Level(2)= OAr(n);
        Level(1)=Level(2)-Ldi5;
        Level(3)=Level(2)+Ldi5;

        if Level(1)≤Min5
        Level(1)=Min5;
        else
        end
        if Level(3)≥Max5
        Level(3)=Max5;
        else
        end
        LevelOA1(n)=Level(1);
        LevelOA2(n)=Level(2);
        LevelOA3(n)=Level(3);
              case 6
        Level(2)= OAr(n);
        Level(1)=Level(2)-Ldi6;
        Level(3)=Level(2)+Ldi6;

        if Level(1)≤Min6
        Level(1)=Min6;
        else
        end
        if Level(3)≥Max6
        Level(3)=Max6;
        else
        end
        LevelOA1(n)=Level(1);
        LevelOA2(n)=Level(2);
        LevelOA3(n)=Level(3);
              case 7
        Level(2)= OAr(n);
        Level(1)=Level(2)-Ldi7;
        Level(3)=Level(2)+Ldi7;

        if Level(1)≤Min7
        Level(1)=Min7;
        else
        end
        if Level(3)≥Max7
        Level(3)=Max7;
        else
        end
        LevelOA1(n)=Level(1);
        LevelOA2(n)=Level(2);
        LevelOA3(n)=Level(3);


    end
```

```
    end


end


countoflevel=0;


clear LevelOA


for n=1:SizeofOA(1) % Filling up a Level based OA

    for b=1:SizeofOA(2)


    if OA(n,b)==0
        LevelOA(n,b)=LevelOA1(b);

    elseif OA(n,b)==1
        LevelOA(n,b)=LevelOA2(b);
    else
        LevelOA(n,b)=LevelOA3(b);
    end



    if b==7
      if LevelOA(n,1)≤LevelOA(n,3)
    LevelOA(n,3)=Min1−5;
    else

    end
    if (LevelOA(n,1)/20)≤LevelOA(n,7)
           LevelOA(n,7)=(LevelOA(n,1)/20)−.2;
       else
    end
    else
    end


    end

    LevelOATrack(n,b,(i+1))=LevelOA(n,b);




    end
```

```
clear FitnessF
clear F1 F2 F3
% 1 Epatch Width
% 2 Epatch Lenght
% 3 Slot Length
% 4 Slot Width
% 5 Ps
% 6 Dielectric Height
% 7 Probe placement x direction
FirstT=clock;
for n=1:SizeofOA(1)
tic;

if n==1
else
!rm out.txt
end



if not(mod(LevelOA(n,4),2)==mod(LevelOA(n,2),2))
    w=LevelOA(n,2)+1;
else
    w=LevelOA(n,2);
end




l=LevelOA(n,1);


sl=LevelOA(n,3);
sw=LevelOA(n,4);

ps=LevelOA(n,5);
h=LevelOA(n,6);
probex=LevelOA(n,7);

x1=round((200-l)/2); % probe placement x direction piece 1
l1=(l-sl); % slot length  of the first piece
y1=round((200-w)/2); % probde placement y direction piece 1
w1=(w); % width of E-patch
x2=round((200-l)/2); % probe placement x direction piece 2
l2=(l); % lenght of E-patch
y2=round((200-w)/2); % probe placement y dircetion piece 2
w2=round(w/2-sw/2-ps); %with of piece 2 based on how wide slot needs
   to be
x3=round((200-l)/2); % probe placement x direction piece 3
l3=(l); % lenght of piece 3
```

```
y3=round(100+sw/2-ps); % probe placement y direction of piece 3 based
    on PS
w3=(2*ps-sw); % width of piece 3 based on PS
x4=round((200-l)/2); % probe placement piece 4 x
l4=(l); % lenght of piece 4
y4=round(100+ps+sw/2); % probe placem piece 4 in y
w4=round(w/2-sw/2-ps); % width of piece 4

probey=mod(w3,2)/20; % probe placement in y direction always in the
   symetric center


fid=fopen(['text.txt'],'w');
    fprintf(fid,'1\n^^^ 1 = save geo file, 2=read it, 0=do nothing\
       npatch.geo\n^^^ filename for geometry\n20.0 20.0\n^^^ (x,y)
       cavity dimensions in cm\n200 200 1\n^^^ x,y,z dimension cells\
       n0\n^^^ 0 = cavity, 1=slot\n');
    fprintf(fid,'%.2f\n^^^ Layer 1 thickness (cm)\n',h);
    fprintf(fid,'4\n^^^ Number of patches\n');
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x1; y1]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l1; w1
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x2; y2]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l2 ; w2
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x3; y3]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l3; w3
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x4; y4]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l4; w4
       ]);
    fprintf(fid,'0\n^^^ Number of slots\n0\n^^^ Number of shorting
       pins\n0\n^^^ Number of metal blocks\n0\n^^^ Number of material
        blocks\n1\n^^^ 1 = uniform material fill\n0\n^^^ 0 = non-
       dispersive, 1 = dispersive\n1.0 -0.00\n^^^ Complex epsilon (
       real, imaginary)\n1.0 -0.00\n^^^ Complex mu (real, imaginary)\
       n');
    fprintf(fid,'0.01 5 60000\n^^^ convergence tolerance, min.
       iterations, max. iterations\n0\n^^^ 0 = BiCG, 1 = QMR\n0\n^^^
       1 = monitor iterations\n0\n^^^ 0=initial guess, 1=automatic
       guess\n1.227 1.6 .348\n^^^ Start, Stop, INCR frequency [GHz]\
       n0 0 1\n^^^ Start, Stop, Incr phi angles [deg]\n-90.0 90.0 1.0
       \n^^^ Start, Stop, Incr theta angles [deg]\n0\n^^^ 0 = do not
       compute far-zone, 1=compute far-zone fields\npatchX_test\n^^^
       Six (6) character file prefix\n0\n^^^ 0 = driven, 1=S-
       paramters, 2=bistatic\nE_patch_chad_impedance_Opt.zin\n^^^
       Input impedance file\n');
    fprintf(fid,'50.0 0.0\n^^^ Reference impedance\n1\n^^^ Number of
       feeds\n3\n^^^ direction of feed (1=x, 2=y, 3=z)\n');
```

```
    fprintf(fid,'%.2f %.2f\n^^^ (x,y) location of feed in cm\n',[
       probex; probey]);
    fprintf(fid,'1\n^^^ Layer of feed\n1.0 0.0\n^^^ Magnitude and
       phase of current\n0\n^^^ Number of loads\n0\n^^^ 0 = do NOT
       save interior fields, 1= save surface fields\n0\n^^^ Number of
        field probe\n');
    fclose(fid);

    %run kbrick
    !k_brick <text.txt > out.txt

    %input db's from kbrick output file
    fid=fopen('out.txt','rt');
while feof(fid) == 0
    clear tline;
    tline = fgetl(fid);
    if length(tline)>6
    if tline(1:7)==' 1.2270'
        db1=str2num(tline(length(tline)-5:length(tline)));
    end
    if tline(1:7)==' 1.5750'
        db2=str2num(tline(length(tline)-5:length(tline)));
    end
    end
end
fclose(fid);
 % build fitness function



 %  FitnessF(1,n) =int((db1+15),x,1.177,1.277)+int((db2+15),x,1.525,1
    .625);
   FitnessF(1,n)=abs(A*(db1+40))+abs((db2+40));
    disp('FITNESS for Experiment');disp(n);disp('Iteration');disp(i)
    disp(FitnessF(1,n));



end
Time2=clock;
TimeF=Time2-FirstT;
disp('Time for one whole set of 27 expriments');disp(TimeF)

disp(' LevelOA for Iteration');disp((i+1))
    disp(LevelOATrack(:,:,(i+1)));
clear StoN

StoN=-20*log10((FitnessF)); % Signal to Noise Ratio
clear FinalResponse
%Build Response table


for n=1:SizeofOA(2)
```

```
    response=0;
    response2=0;
    response3=0;
    countoflevel=0;

    for h=1:SizeofOA(1)

    if OA(h,n)==0
        response=response+StoN(h);
        countoflevel=countoflevel+1;

    elseif OA(h,n)==1
        response2=response2+StoN(h);

    elseif OA(h,n)==2
        response3=response3+StoN(h);
    end

    end

    for c=1:lengthflevel

    if c==1
    FinalResponse(c,n)=response/countoflevel;

    elseif c==2
    FinalResponse(c,n)=response2/countoflevel;

    elseif c==3
    FinalResponse(c,n)=response3/countoflevel;
    end

    end

end

    clear Optparameters

      disp('Final Response table for iteration');disp(i)
      disp(FinalResponse)
       clear OptMat
Optparameters=max(FinalResponse);

for n=1:SizeofOA(2) % Matrix use to single out the good s/n's

    for c=1:lengthflevel

        if Optparameters(n)==FinalResponse(c,n)
            OptMat(c,n)=FinalResponse(c,n);
        else
            OptMat(c,n)=pi;
        end

    end
```

```
end
 clear OAr
for n=1:SizeofOA(2) % matrix used to specify final levels that are
   optimal

    for c=1:lengthflevel

        if OptMat(c,n)==pi;
        else
            switch c
                case 1
                    OAr(n)=LevelOA1(n);
                case 2
                    OAr(n)=LevelOA2(n);
                case 3
                    OAr(n)=LevelOA3(n);
            end

        end
    end
end

 % The design for this Fitness Function for use only for Taguchi



 !rm text.txt
 !rm out.txt

 if OAr(1)≤OAr(3) % redundent check of slotlenght and patch width
    OAr(3)=Min1-5;
else
 end

 if (OAr(1)/20)≤OAr(7)%make sure probe is on patch
           OAr(7)=(OAr(1)/20)-.2;
       else
       end

if not(mod(OAr(4),2)==mod(OAr(2),2))
    w=OAr(2)+1;
else
    w=OAr(2);
end




l=OAr(1);

sl=OAr(3);
sw=OAr(4);
```

```
ps=OAr(5);
h=OAr(6);
probex=OAr(7);

x1=round((200-l)/2); % probe placement x direction piece 1
l1=(l-sl); % slot length width of the first piece
y1=round((200-w)/2); % probde placement y direction piece 1
w1=(w); % width of E-patch
x2=round((200-l)/2); % probe placement x direction piece 2
l2=(l); % lenght of E-patch
y2=round((200-w)/2); % probe placement y dircetion piece 2
w2=round(w/2-sw/2-ps); %with of piece 2 based on how wide slot needs
   to be
x3=round((200-l)/2); % probe placement x direction piece 3
l3=(l); % lenght of piece 3
y3=round(100+sw/2-ps); % probe placement y direction of piece 3 based
    on PS
w3=(2*ps-sw); % width of piece 3 based on PS
x4=round((200-l)/2); % probe placement piece 4 x
l4=(l); % lenght of piece 4
y4=round(100+ps+sw/2); % probe placem piece 4 in y
w4=round(w/2-sw/2-ps); % width of piece 4

probey=mod(w3,2)/20; % probe placement in y direction always in the
   symetric center


fid=fopen(['text.txt'],'w');
    fprintf(fid,'1\n^^^ 1 = save geo file, 2=read it, 0=do nothing\
       npatch.geo\n^^^ filename for geometry\n20.0 20.0\n^^^ (x,y)
       cavity dimensions in cm\n200 200 1\n^^^ x,y,z dimension cells\
       n0\n^^^ 0 = cavity, 1=slot\n');
    fprintf(fid,'%.2f\n^^^ Layer 1 thickness (cm)\n',h);
    fprintf(fid,'4\n^^^ Number of patches\n');
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x1; y1]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l1; w1
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x2; y2]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l2 ; w2
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x3; y3]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l3; w3
       ]);
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x4; y4]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l4; w4
       ]);
    fprintf(fid,'0\n^^^ Number of slots\n0\n^^^ Number of shorting
       pins\n0\n^^^ Number of metal blocks\n0\n^^^ Number of material
        blocks\n1\n^^^ 1 = uniform material fill\n0\n^^^ 0 = non-
       dispersive, 1 = dispersive\n1.0 -0.00\n^^^ Complex epsilon (
```

```
        real, imaginary)\n1.0 -0.00\n^^^ Complex mu (real, imaginary)\
        n');
    fprintf(fid,'0.01 5 60000\n^^^ convergence tolerance, min.
        iterations, max. iterations\n0\n^^^ 0 = BiCG, 1 = QMR\n0\n^^^
        1 = monitor iterations\n0\n^^^ 0=initial guess, 1=automatic
        guess\n1.227 1.6 .348\n^^^ Start, Stop, INCR frequency [GHz]\
        n0 0 1\n^^^ Start, Stop, Incr phi angles [deg]\n-90.0 90.0 1.0
        \n^^^ Start, Stop, Incr theta angles [deg]\n0\n^^^ 0 = do not
        compute far-zone, 1=compute far-zone fields\npatchX_test\n^^^
        Six (6) character file prefix\n0\n^^^ 0 = driven, 1=S-
        paramters, 2=bistatic\nE_patch_chad_impedance_Opt.zin\n^^^
        Input impedance file\n');
    fprintf(fid,'50.0 0.0\n^^^ Reference impedance\n1\n^^^ Number of
        feeds\n3\n^^^ direction of feed (1=x, 2=y, 3=z)\n');
    fprintf(fid,'%.2f %.2f\n^^^ (x,y) location of feed in cm\n',[
        probex; probey]);
    fprintf(fid,'1\n^^^ Layer of feed\n1.0 0.0\n^^^ Magnitude and
        phase of current\n0\n^^^ Number of loads\n0\n^^^ 0 = do NOT
        save interior fields, 1= save surface fields\n0\n^^^ Number of
         field probe\n');
    fclose(fid);

    %run kbrick
    !k_brick <text.txt >out.txt

    %input db's from kbrick output file
    fid=fopen('out.txt','rt');
while feof(fid) == 0
    clear tline;
    tline = fgetl(fid);
    if length(tline)>6
    if tline(1:7)==' 1.2270'
        db1=str2num(tline(length(tline)-5:length(tline)));
    end
    if tline(1:7)==' 1.5750'
        db2=str2num(tline(length(tline)-5:length(tline)));
    end
    end
end
fclose(fid);
 % build fitness function

  % FitnessG=int((db1+15),x,1.177,1.277)+int((db2+15),x,1.525,1.625);

FitnessG=abs(A*(db1+40))+abs((db2+40));
disp('This is the Fitness of the "bestfit" function for iteration');
   disp(i);
disp(FitnessG);
meck(i+1)=FitnessG;

for  n=1:Element
FinalDs(i+1,n)=OAr(n);
end
```

```
disp('Values for E-patch for iteration');disp(i);
      disp(OAr);
 !rm out.txt
i=i+1;
   Store_db1(i+1)=(db1)
   Store2_db2(i+1)=(db2)

   %Write and update a output file
   fid=fopen(['Taguchi_Epatch_heightrange_5.txt'],'w+');
   fprintf(fid,'Max1     Min1     Max2     Min2     Max3     Min3
          Max4     Min4     Max5     Min5     Max6     Min6     Max7
          Min7 \n')
   fprintf(fid,'  %.2f     %.2f      %.2f      %.2f      %.2f      %
      .2f      %.2f      %.2f     %.2f      %.2f      %.2f      %.2f
           %.2f      %.2f   ', [ Max1; Min1; Max2; Min2; Max3; Min3;
      Max4; Min4; Max5; Min5; Max6; Min6; Max7; Min7])
fprintf(fid,'\n Iteration   Epatch Width    Epatch Length    Sloth
   Lenght    Slot Width    Ps     Dielectric Height      Probe X  1
   .227 1.575')
for j=1:i


for n=1:Element

    if n==1
    fprintf(fid,'\n    %.2f      %.2f    ', [j; FinalDs(j,n)]);
    else
        fprintf(fid,'    %.2f      ',FinalDs(j,n));
    end
    if n==Element
        fprintf(fid,' %.2f  %.2f ' , [Store_db1(j+1); Store2_db2(j+1)
           ]);
    else
    end

end

end

fclose(fid);
end




meck(i+1)=FitnessG;
   plotness=1:length(meck);
figure,plot(plotness,meck);
h = legend('Final Fitness');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 27 expriements')
ylabel('Fitness')
title('Fitness vs Runs')
```

```
figure,plot(plotness,Store_db1);
h = legend('db1');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 27 expriements')
ylabel('db1 values')
title('db1 values')

figure,plot(plotness,Store2_db2);
h = legend('db2');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 27 expriements')
ylabel('db2 values')
title('db2 values')
```

Appendix F: Matlab Imaging Code for Taguchi-K-brick-Rectangular Patch

This attached code is for a Taguchi optimizer for a rectangular-cavity-backed patch antenna, simulated in K-brick. Figure F.1 shows the iteration of Taguchi-K-brick optimization vs $|S_{11}|$ in dB of the simulated patch antenna. For both frequencies a return loss of at least 35 dB was found. After 27 iterations $|S_{11}|$ was -36.23 dB for 1.5 GHz and -37.27 dB after 24 iteration for 1.2 GHz.

```
close all
clear all
format short

db1=0;
db2=0;
% 1 Epatch Width
% 2 Epatch Lenght
% 3 Slot Length
% 4 Slot Width
% 5 Ps
% 6 Dielectric Height
% 7 Probe placement x direction

[filename pathname] = uigetfile('*.txt','Choose the OA file from
    online or generated code MUST BE ACII Formated!');

OA=dlmread([pathname filename]);

SizeofOA=size(OA);
Element = 4;% input number of Elements to be tested

if SizeofOA(2) == Element

else
    newColummnsize=SizeofOA(2)-Element;
```

```
    for n=1:newColummnsize % reduces the columm size for expriment

        OA(:,SizeofOA(2))=[];

        SizeofOA=size(OA);
    end
end
A=1; % factor for db1
B=1; % factor for db2

% Level Design User specifies level
Level=0:2; %Input by user assuming 3 levels
lengthflevel=length(Level);

Max1=130; % Input by User patch width in mm
Min1=60; % Input by User patch width in mm

Max2=130; % Input by User patch lenght in mm
Min2=60; % Input by User patch lenght in mm

Max3=6.5; % Input by User probe in x in cm
Min3=0; % Input by User probe in x in cm

Max4=1.5; % Input by User dielectric h in cm
Min4=.5; % Input by User tric hdielec in cm


rr=.9; % Reduction rate used by User
i=0;
diffrentlevels=0:1; % Because not all levels will have the same
   range. Specified here is how many are different
LDtime=1.5;

Ld1= LDtime*(Max1 - Min1)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld2= LDtime*(Max2 - Min2)/(lengthflevel+1); % first specifed level
   differnce first interation


Ld3= LDtime*(Max3 - Min3)/(lengthflevel+1); % first specifed level
   differnce first interation



Ld4=LDtime*(Max4 - Min4)/(lengthflevel+1); % first specifed level
   differnce first interation


Ldi1=round(Ld1*rr^i);

Ldi2=round(Ld2*rr^i);
```

```
Ldi3=(Ld3*rr^i);

Ldi4=(Ld4*rr^i);




while (Ldi1/Ld1)>.0001;

    clear Level LevelOA2 LevelOA3 LevelOA1 db1 db2
    Level=0:2;
       clear n b g c


Ldi1=round(Ld1*rr^i);

Ldi2=round(Ld2*rr^i);

Ldi3=(Ld3*rr^i);

Ldi4=(Ld4*rr^i);




for n=1:SizeofOA(2)


    if i==0
            switch n

                case 1
         Level(2)= (Max1-Min1)/2 + Min1;% Middle of range
         Level(1)=Level(2)-Ldi1;
         Level(3)=Level(2)+Ldi1;
                case 2
                    Level(2)= (Max2-Min2)/2 + Min2;% Middle of range
         Level(1)=Level(2)-Ldi2;
         Level(3)=Level(2)+Ldi2;
                case 3
                    Level(2)= (Max3-Min3)/2 + Min3;% Middle of range
         Level(1)=Level(2)-Ldi3;
         Level(3)=Level(2)+Ldi3;
                case 4
                    Level(2)= (Max4-Min4)/2 + Min4;% Middle of range
         Level(1)=Level(2)-Ldi4;
         Level(3)=Level(2)+Ldi4;

                case 5
```

```
        end
      LevelOA1(n)=Level(1);
      LevelOA2(n)=Level(2);
      LevelOA3(n)=Level(3);


else

switch n
    case 1
     Level(2)= OAr(n);
     Level(1)=Level(2)−Ldi1;
     Level(3)=Level(2)+Ldi1;

     if Level(1)≤Min1
     Level(1)=Min1;
     else
     end
     if Level(3)≥Max1
     Level(3)=Max1;
     else
     end
     LevelOA1(n)=Level(1);
     LevelOA2(n)=Level(2);
     LevelOA3(n)=Level(3);

     case 2
     Level(2)= OAr(n);
     Level(1)=Level(2)−Ldi2;
     Level(3)=Level(2)+Ldi2;

     if Level(1)≤Min2
     Level(1)=Min2;
     else
     end
     if Level(3)≥Max2
     Level(3)=Max2;
     else
     end
     LevelOA1(n)=Level(1);
     LevelOA2(n)=Level(2);
     LevelOA3(n)=Level(3);

          case 3
     Level(2)= OAr(n);
     Level(1)=Level(2)−Ldi3;
     Level(3)=Level(2)+Ldi3;

     if Level(1)≤Min3
     Level(1)=Min3;
     else
     end
```

```
            if Level(3)≥Max3
            Level(3)=Max3;
            else
            end
            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);

                  case 4
            Level(2)= OAr(n);
            Level(1)=Level(2)−Ldi4;
            Level(3)=Level(2)+Ldi4;

            if Level(1)≤Min4
            Level(1)=Min4;
            else
            end
            if Level(3)≥Max4
            Level(3)=Max4;
            else
            end
            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);

        end




                end
        end


countoflevel=0;


clear LevelOA


for n=1:SizeofOA(1) % Filling up a Level based OA

        for b=1:SizeofOA(2)


        if OA(n,b)==0
            LevelOA(n,b)=LevelOA1(b);

        elseif OA(n,b)==1
            LevelOA(n,b)=LevelOA2(b);
        else
            LevelOA(n,b)=LevelOA3(b);
        end
```

```
  if b==4 % check for probe is on patch
      if (LevelOA(n,1)/20)≤LevelOA(n,3)
          LevelOA(n,3)=(LevelOA(n,1)/20)−.2;
      else
      end
  end

    LevelOATrack(n,b,(i+1))=LevelOA(n,b);




    end


end



clear FitnessF
clear F1 F2 F3
% 1 Epatch Width
% 2 Epatch Lenght
% 3 Slot Length
% 4 Slot Width
% 5 Ps
% 6 Dielectric Height
% 7 Probe placement x direction
FirstT=clock;
for n=1:SizeofOA(1)
tic;

if n==1
else
!rm out.txt
end


w=LevelOA(n,2);
l=LevelOA(n,1);




h=LevelOA(n,4);
probex=LevelOA(n,3);
BlocksX=floor(probex*10+99);
BlocksY=99;

x1=round((200−l)/2); % probe placement x direction piece 1
```

```
l1=round(l); % patch width of the first piece
y1=round((200-w)/2); % probde placement y direction piece 1
w1=round(w); % lenght  of patch


probey=mod(w,2)/20; % probe placement in y direction always in the
    symetric center

     fid=fopen(['text_normal.txt'],'w');
    fprintf(fid,'1\n^^^ 1 = save geo file, 2=read it, 0=do nothing\
       npatch.geo\n^^^ filename for geometry\n20.0 20.0\n^^^ (x,y)
       cavity dimensions in cm\n200 200 1\n^^^ x,y,z dimension cells\
       n0\n^^^ 0 = cavity, 1=slot\n');
    fprintf(fid,'%.3f\n^^^ Layer 1 thickness (cm)\n',h);
    fprintf(fid,'1\n^^^ Number of patches\n');
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x1; y1]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l1; w1
       ]);
    fprintf(fid,'0\n^^^ Number of slots\n0\n^^^ Number of shorting
       pins\n0\n^^^ Number of metal blocks\n0\n^^^ Number of material
        blocks\n1\n^^^ 1 = uniform material fill\n0\n^^^ 0 = non-
       dispersive, 1 = dispersive\n1.0 -0.00\n^^^ Complex epsilon (
       real, imaginary)\n1.0 -0.00\n^^^ Complex mu (real, imaginary)\
       n');
    fprintf(fid,'0.01 5 60000\n^^^ convergence tolerance, min.
       iterations, max. iterations\n0\n^^^ 0 = BiCG, 1 = QMR\n0\n^^^
       1 = monitor iterations\n0\n^^^ 0=initial guess, 1=automatic
       guess\n1.2 1.6 .448\n^^^ Start, Stop, INCR frequency [GHz]\n0
       0 1\n^^^ Start, Stop, Incr phi angles [deg]\n-90.0 90.0 1.0\n
       ^^^ Start, Stop, Incr theta angles [deg]\n0\n^^^ 0 = do not
       compute far-zone, 1=compute far-zone fields\npatchX_test\n^^^
       Six (6) character file prefix\n0\n^^^ 0 = driven, 1=S-
       paramters, 2=bistatic\nE_patch_chad_impedance_Opt.zin\n^^^
       Input impedance file\n');
    fprintf(fid,'50.0 0.0\n^^^ Reference impedance\n1\n^^^ Number of
       feeds\n3\n^^^ direction of feed (1=x, 2=y, 3=z)\n');
    fprintf(fid,'%.3f %.3f\n^^^ (x,y) location of feed in cm\n',[
       probex; probey]);
    fprintf(fid,'1\n^^^ Layer of feed\n1.0 0.0\n^^^ Magnitude and
       phase of current\n0\n^^^ Number of loads\n0\n^^^ 0 = do NOT
       save interior fields, 1= save surface fields\n0\n^^^ Number of
        field probe\n');
    fclose(fid);
!k_brick <text_normal.txt >out.txt

    fid=fopen('out.txt','rt');
while feof(fid) == 0
    clear tline;
    tline = fgetl(fid);
    if length(tline)>6
    if tline(1:7)==' 1.2000'
        db1=str2num(tline(length(tline)-5:length(tline)));
    end
```

```
    end
end
fclose(fid);
 % build fitness function




 %  FitnessF(1,n) =int((db1+15),x,1.177,1.277)+int((db2+15),x,1.525,1
    .625);
   FitnessF(1,n)=abs(A*(db1+50));
    disp('FITNESS for Experiment');disp(n);disp('Iteration');disp(i)
    disp(FitnessF(1,n));



end
Time2=clock;
TimeF=Time2-FirstT;
disp('Time for one whole set of 27 expriments');disp(TimeF)

disp(' LevelOA for Iteration');disp((i+1))
    disp(LevelOATrack(:,:,(i+1)));
clear StoN

StoN=-20*log10((FitnessF)); % Signal to Noise Ratio
clear FinalResponse
%Build Response table


for n=1:SizeofOA(2)
    response=0;
    response2=0;
    response3=0;
    countoflevel=0;

    for h=1:SizeofOA(1)

    if OA(h,n)==0
        response=response+StoN(h);
        countoflevel=countoflevel+1;

    elseif OA(h,n)==1
        response2=response2+StoN(h);

    elseif OA(h,n)==2
        response3=response3+StoN(h);
    end

    end

    for c=1:lengthflevel
```

```
    if c==1
    FinalResponse(c,n)=response/countoflevel;

    elseif c==2
    FinalResponse(c,n)=response2/countoflevel;

    elseif c==3
    FinalResponse(c,n)=response3/countoflevel;
    end

    end

end

    clear Optparameters

      disp('Final Response table for iteration');disp(i)
      disp(FinalResponse)
       clear OptMat
Optparameters=max(FinalResponse);

for n=1:SizeofOA(2) % Matrix use to single out the good s/n's

    for c=1:lengthflevel

        if Optparameters(n)==FinalResponse(c,n)
            OptMat(c,n)=FinalResponse(c,n);
        else
            OptMat(c,n)=pi;
        end

    end
end
 clear OAr
for n=1:SizeofOA(2) % matrix used to specify final levels that are
   optimal

    for c=1:lengthflevel

        if OptMat(c,n)==pi;
        else
            switch c
                case 1
                    OAr(n)=LevelOA1(n);
                case 2
                    OAr(n)=LevelOA2(n);
                case 3
                    OAr(n)=LevelOA3(n);
            end

        end
    end
end
```

```
 % The design for this Fitness Function for use only for Taguchi



 !rm text.txt
 !rm out.txt




w=OAr(2);
l=OAr(1);


if (OAr(1)/20)≤OAr(3)
          OAr(3)=(OAr(1)/20)-.2;
      else
      end


h=OAr(4);
probex=OAr(3);
BlocksX=floor(probex*10+99);
BlocksY=99;

x1=round((200-l)/2); % probe placement x direction piece 1
l1=round(l); % patch width of the first piece
y1=round((200-w)/2); % probde placement y direction piece 1
w1=round(w); % lenght width of E-patch


probey=mod(w,2)/20; % probe placement in y direction always in the
    symetric center


     fid=fopen(['text_normal.txt'],'w');
    fprintf(fid,'1\n^^^ 1 = save geo file, 2=read it, 0=do nothing\
       npatch.geo\n^^^ filename for geometry\n20.0 20.0\n^^^ (x,y)
       cavity dimensions in cm\n200 200 1\n^^^ x,y,z dimension cells\
       n0\n^^^ 0 = cavity, 1=slot\n');
    fprintf(fid,'%.3f\n^^^ Layer 1 thickness (cm)\n',h);
    fprintf(fid,'1\n^^^ Number of patches\n');
    fprintf(fid,'%.0f %.0f 1\n^^^ Lower left corner of patch + layer\
       n',[x1; y1]);
    fprintf(fid,'%.0f %.0f \n^^^ Patch dimension in edges\n',[l1; w1
       ]);
    fprintf(fid,'0\n^^^ Number of slots\n0\n^^^ Number of shorting
       pins\n0\n^^^ Number of metal blocks\n0\n^^^ Number of material
        blocks\n1\n^^^ 1 = uniform material fill\n0\n^^^ 0 = non-
       dispersive, 1 = dispersive\n1.0 -0.00\n^^^ Complex epsilon (
       real, imaginary)\n1.0 -0.00\n^^^ Complex mu (real, imaginary)\
       n');
```

```
    fprintf(fid,'0.01 5 60000\n^^^ convergence tolerance, min.
       iterations, max. iterations\n0\n^^^ 0 = BiCG, 1 = QMR\n0\n^^^
       1 = monitor iterations\n0\n^^^ 0=initial guess, 1=automatic
       guess\n1.2 1.6 .448\n^^^ Start, Stop, INCR frequency [GHz]\n0
       0 1\n^^^ Start, Stop, Incr phi angles [deg]\n-90.0 90.0 1.0\n
       ^^^ Start, Stop, Incr theta angles [deg]\n0\n^^^ 0 = do not
       compute far-zone, 1=compute far-zone fields\npatchX_test\n^^^
       Six (6) character file prefix\n0\n^^^ 0 = driven, 1=S-
       paramters, 2=bistatic\nE_patch_chad_impedance_Opt.zin\n^^^
       Input impedance file\n');
    fprintf(fid,'50.0 0.0\n^^^ Reference impedance\n1\n^^^ Number of
       feeds\n3\n^^^ direction of feed (1=x, 2=y, 3=z)\n');
    fprintf(fid,'%.3f %.3f\n^^^ (x,y) location of feed in cm\n',[
       probex; probey]);
    fprintf(fid,'1\n^^^ Layer of feed\n1.0 0.0\n^^^ Magnitude and
       phase of current\n0\n^^^ Number of loads\n0\n^^^ 0 = do NOT
       save interior fields, 1= save surface fields\n0\n^^^ Number of
        field probe\n');
    fclose(fid);
!k_brick <text_normal.txt >out.txt



    %run kbrick


    %input db's from kbrick output file
    fid=fopen('out.txt','rt');
while feof(fid) == 0
    clear tline;
    tline = fgetl(fid);
    if length(tline)>6
    if tline(1:7)==' 1.2000'
        db1=str2num(tline(length(tline)-5:length(tline)));
    end
    end
end
fclose(fid);


FitnessG=abs(A*(db1+50));
disp('This is the Fitness of the "bestfit" function for iteration');
   disp(i);
disp(FitnessG);
meck(i+1)=FitnessG;

for  n=1:Element
FinalDs(i+1,n)=OAr(n);
end

disp('Values for E-patch for iteration');disp(i);
      disp(OAr);
 !rm out.txt
i=i+1;
```

```
   Store_db1(i+1)=(db1)



   %Write and update a output file
   fid=fopen(['Rectangular_patch_values_1.2_freq_LD_1_5.txt'],'w+');
   fprintf(fid,'Max1     Min1     Max2     Min2     Max3     Min3
          Max4     Min4      \n')
   fprintf(fid,'  %.2f     %.2f      %.2f      %.2f      %.2f      %
      .3f      %.3f     %.3f     ', [ Max1; Min1; Max2; Min2; Max3;
      Min3; Max4; Min4;])
fprintf(fid,'\n Iteration   patch Width    patch Length     Probe X
     dielectric height   1.2')
for j=1:i


for n=1:Element

    if n==1
    fprintf(fid,'\n    %.3f      %.3f    ', [j; FinalDs(j,n)]);
    else
        fprintf(fid,'    %.3f      ',FinalDs(j,n));
    end
    if n==Element
        fprintf(fid,' %.3f   ' , [Store_db1(j+1)]);
    else
    end

end

end

fclose(fid);

end


meck(i+1)=FitnessG;
   plotness=1:length(meck);
figure,plot(plotness,meck);
h = legend('Final Fitness');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('Fitness')
title('Fitness vs Runs')


figure,plot(plotness,Store_db1);
h = legend('db1');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('db1 values')
```

```
  title('db1 values')
```

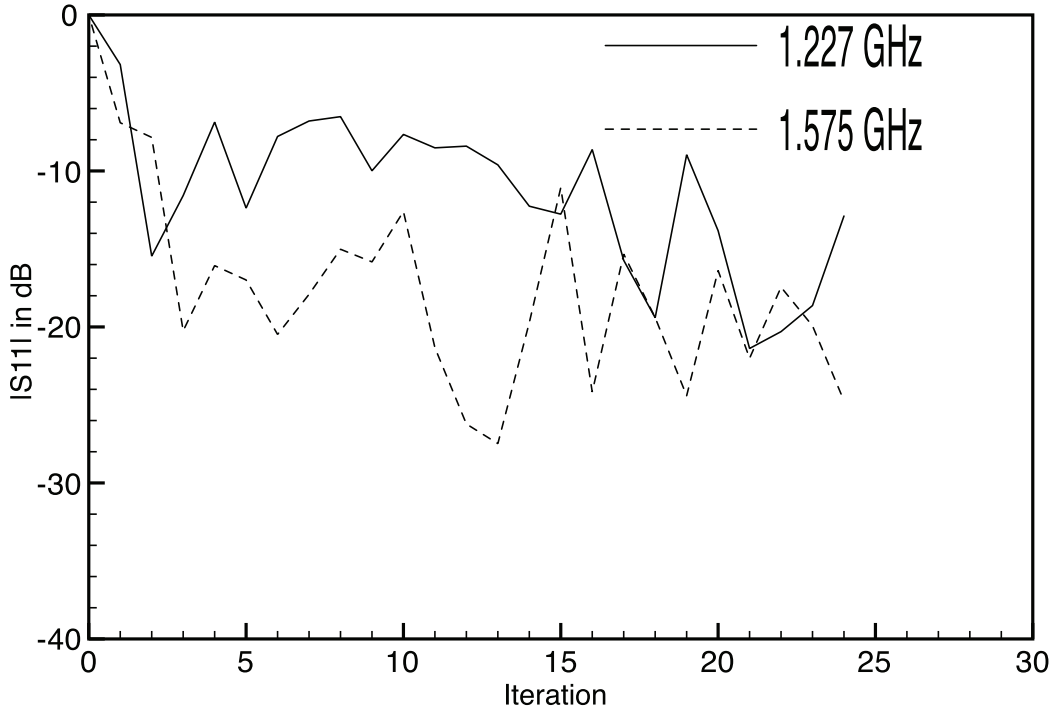

Figure F.1: $|S_{11}|$ as a function of Taguchi iteration number for rectangular patches optimizing at 1.5 and 1.2 GHz.

Appendix G: Matlab code for Rosenbrock Taguchi optimization

```
close all
clear all

[filename pathname] = uigetfile('*.txt','Choose the OA file from
    online or generated code MUST BE ACII Formated!');

OA=dlmread([pathname filename]);

SizeofOA=size(OA);
Element = 2;% input number of Elements to be tested

if SizeofOA(2) == Element

else
    newColummnsize=SizeofOA(2)-Element;
    for n=1:newColummnsize % reduces the columm size for expriment

        OA(:,SizeofOA(2))=[];

        SizeofOA=size(OA);
    end
end


% Level Design User specifies level
Level=0:2; %Input by user assuming 3 levels
lengthflevel=length(Level);
Max1=10; % Input by User
Min1=0; % Input by User
Max2=10; % Input by User
Min2=0; % Input by User
rr=.99; % Reduction rate used by User
i=0;
diffrentlevels=0:1; % Because not all levels will have the same
    range. Specified here is how many are different

A=2; %changes the level times
```

```
Ld1= (Max1 - Min1)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi1=Ld1*rr^i;

Ld2= (Max2 - Min2)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi2=Ld2*rr^i;

while (Ldi1/Ld1)>.01;

    clear  LevelOA2 LevelOA3 LevelOA1
    Level=0:2;
       clear n b g c


Ld1= (Max1 - Min1)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi1=(Ld1*A)*rr^i;

Ld2= (Max2 - Min2)/(lengthflevel+1); % first specifed level differnce
    first interation
Ldi2=(Ld2*A)*rr^i;



for n=1:SizeofOA(2)



    if i==0
    switch n
        case 1
         Level(2)= (Max1-Min1)/2 + Min1;% Middle of range
         Level(1)=Level(2)-Ldi1;
         Level(3)=Level(2)+Ldi1;

         LevelOA1(n)=Level(1);
         LevelOA2(n)=Level(2);
         LevelOA3(n)=Level(3);

        case 2
         Level(2)=(Max2-Min2)/2 + Min2; %Middle of range
         Level(1)=Level(2)-Ldi2;
         Level(3)=Level(2)+Ldi2;

         LevelOA1(n)=Level(1);
         LevelOA2(n)=Level(2);
         LevelOA3(n)=Level(3);
    end

    else

    switch n
        case 1
```

```
            Level(2)= OAr(n);
            Level(1)=Level(2)-Ldi1;
            Level(3)=Level(2)+Ldi1;

            if Level(1)≤Min1
            Level(1)=Min1;
            else
            end
            if Level(3)≥Max1
            Level(3)=Max1;
            else
            end
            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);

           case 2
            Level(2)= OAr(n);
            Level(1)=Level(2)-Ldi2;
            Level(3)=Level(2)+Ldi2;

            if Level(1)≤Min2
            Level(1)=Min2;
            else
            end
            if Level(3)≥Max2
            Level(3)=Max2;
            else
            end

            LevelOA1(n)=Level(1);
            LevelOA2(n)=Level(2);
            LevelOA3(n)=Level(3);
    end



    end

end


countoflevel=0;


clear LevelOA


for n=1:SizeofOA(1) % Filling up a Level based OA

    for b=1:SizeofOA(2)
```

```
    if OA(n,b)==0
        LevelOA(n,b)=LevelOA1(b);
    elseif OA(n,b)==1
        LevelOA(n,b)=LevelOA2(b);
    else
        LevelOA(n,b)=LevelOA3(b);
    end

    end


end

clear FitnessF
for n=1:SizeofOA(1)




    %Fitness=Fitness+(LevelOA(n,g))^2-10*cos(2*pi*(LevelOA(n,g)))+10;
   %Fitness=1-abs(sin(pi*(LevelOA(n,1)-3))/(pi*(LevelOA(n,1)-3)))*abs
      (sin(pi*(LevelOA(n,2)-3))/(pi*(LevelOA(n,2)-3)));
   %Fitness=exp((LevelOA(n,1)*sin(4*LevelOA(n,1))+1.1*LevelOA(n,2)*
      sin(2*LevelOA(n,2))));
  %   Fitness=20+((LevelOA(n,1)*sin(4*LevelOA(n,1))+1.1*LevelOA(n,2)*
     sin(2*LevelOA(n,2))));
%Fitness=exp((1-LevelOA(n,1))^2+100*(LevelOA(n,2)-(LevelOA(n,1))^2)^2
   _;



FitnessG=(1-LevelOA(n,1))^2+100*(LevelOA(n,2)-(LevelOA(n,1))^2)^2;
FitnessF(n,1)=FitnessG;
end

clear StoN

StoN=-20*log10((FitnessF)); % Signal to Noise Ratio
clear FinalResponse
%Build Response table


for n=1:SizeofOA(2)
    response=0;
    response2=0;
    response3=0;
    countoflevel=0;

    for h=1:SizeofOA(1)

    if OA(h,n)==0
        response=response+StoN(h);
        countoflevel=countoflevel+1;
```

```
    elseif OA(h,n)==1
        response2=response2+StoN(h);

    elseif OA(h,n)==2
        response3=response3+StoN(h);
    end

    end

    for c=1:lengthflevel

    if c==1
    FinalResponse(c,n)=response/countoflevel;

    elseif c==2
    FinalResponse(c,n)=response2/countoflevel;

    elseif c==3
    FinalResponse(c,n)=response3/countoflevel;
    end

    end

end

    clear Optparameters


       clear OptMat
Optparameters=max(FinalResponse);

for n=1:SizeofOA(2) % Matrix use to single out the good s/n's

    for c=1:lengthflevel

        if Optparameters(n)==FinalResponse(c,n)
            OptMat(c,n)=FinalResponse(c,n);
        else
            OptMat(c,n)=pi;
        end

    end
end
 clear OAr
for n=1:SizeofOA(2) % matrix used to specify final levels that are
   optimal

    for c=1:lengthflevel

        if OptMat(c,n)==pi;
        else
            switch c
                case 1
```

```
                    OAr(n)=LevelOA1(n);
                case 2
                    OAr(n)=LevelOA2(n);
                case 3
                    OAr(n)=LevelOA3(n);
            end

        end
    end
end

 % The design for this Fitness Function Raguchi for use only for
    Taguchi



%FitnessG=1-abs(sin(pi*(OAr(1)-3))/(pi*(OAr(1)-3)))*abs(sin(pi*(OAr
   (2)-3))/(pi*(OAr(2)-3)));
FitnessG=(1-OAr(1))^2+100*(OAr(2)-(OAr(1))^2)^2;

    meck(i+1)=(FitnessG);

    if FitnessG == 0
        Store(i+1)=OAr(1);
   Store2(i+1)=OAr(2);
        break
    else
i=i+1;
   Store(i+1)=OAr(1);
   Store2(i+1)=OAr(2);

end
end

x=[0:.01:10];
y=[0:.01:10];

ta=length(x);


%
%
% for bah=1:ta
%
%     for cah=1:ta
%         meckfitness(cah,bah)=x(bah)*sin(4*x(bah))+1.1*y(cah)*sin(2*
   y(cah));
%
%     end
%
% end

fid=fopen(['Taguchi_Opt_varibles_LD_2_Rosenbrock.txt'],'w+');
```

```
for j=1:i




    fprintf(fid,' %.9f            %.9f                %.9f        \n
          ', [meck(j); Store(j); Store2(j)]);



end

fclose(fid);

 %figure;
%
%  mesh(x,y,meckfitness)
%  xlabel('x values')
% ylabel('y values')
% title('global minimun')


meck(i+1)=FitnessG;
   plotness=1:length(meck);
figure,plot(plotness,meck);
h = legend('Final Fitness');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('Fitness')
title('Fitness vs Runs')


figure,plot(plotness,Store);
h = legend('x values');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('x values')
title('x values')

figure,plot(plotness,Store2);
h = legend('y vaules');
set(h,'Interpreter','none')
axis auto
xlabel('Number of Runs of 9 expriements')
ylabel('y values')
title('y values')
```

Appendix H: Matlab code for gain measurements at MSU engineering building

```
clear all
close all
%converter for radtion plots
R=1.90500; %seperation distance in meters from antennas
c=3e8; %constant speed of light
f=1300E6; %frequency of operation
lambda=c/f; %wavelenght
%Pr= ; %recieved power
%Pt= ; %transmitted power
S21_antenna= -22.367187500 ; %magnatidue of s21 in dB  of antenna
S21_cable= .0340366; %magnatidude of s21 in dB of cable

Gt_1300=1/2*(S21_antenna-S21_cable)+ 10*log10((4*pi*R)/lambda);
Gt_1300_lol=1/2*(S21_antenna-S21_cable+ 20*log10((4*pi*R)/lambda));

R=1.90500; %seperation distance in meters from antennas
c=3e8; %constant speed of light
f=1227.5E6; %frequency of operation
lambda=c/f; %wavelenght
%Pr= ; %recieved power
%Pt= ; %transmitted power
S21_antenna= -22.298828125; %magnatidue of s21 in dB  of antenna
S21_cable= .029809952; %magnatidude of s21 in dB of cable

Gt_1227=1/2*(S21_antenna-S21_cable)+ 10*log10((4*pi*R)/lambda);

R=1.90500; %seperation distance in meters from antennas
c=3e8; %constant speed of light
f=1575E6; %frequency of operation
lambda=c/f; %wavelenght
%Pr= ; %recieved power
%Pt= ; %transmitted power
S21_antenna= -25.266601562; %magnatidue of s21 in dB  of antenna
S21_cable= .040971756; %magnatidude of s21 in dB of cable

Gt_1575=1/2*(S21_antenna-S21_cable)+ 10*log10((4*pi*R)/lambda);
```

```
R=1.90500; %seperation distance in meters from antennas
c=3e8; %constant speed of light
f=1640E6; %frequency of operation
lambda=c/f; %wavelenght
%Pr= ; %recieved power
%Pt= ; %transmitted power
S21_antenna= -25.250976562; %magnatidue of s21 in dB  of antenna
S21_cable= .046300888; %magnatidude of s21 in dB of cable

Gt_1640=1/2*(S21_antenna-S21_cable)+ 10*log10((4*pi*R)/lambda);

Horn_gain_upstairs=[.11 .01 .022 .023; .037 .01 .009 .007; .04 .004
    .009 .005; .015 .008 .0032 .002];

% Horn_gain_upstairs_1300=
% Horn_gain_upstairs_1575=
% Horn_gain_upstairs_1640=
%
% Z_X_cross_plane_1227_cylinder.txt
% Z_Y_cross_plane_1227_cylinder.txt
%
% Z_X_cross_plane_1575_cylinder.txt
% Z_Y_cross_plane_1575_cylinder.txt
%
% Z_X_plane_1227_cylinder.txt
% Z_Y_co_plane_1227_cylinder.txt
%
% Z_X_plane_1575_cylinder.txt
% Z_Y_co_plane_1575_cylinder.txt

%converter for radtion plots

[degree,gain]= textread(['Z_X_co_polar_1300.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_co_ploar_1300_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f     %f  ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100
```

```
    fprintf(fid,' \n  %f     %f  ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_X_co_ploar_1300_cylinder_FINAL.txt'], '%f
   %f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_co_ploar_1300_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f     %f  ', [degree(n); 10*log10((10^(Gt_1300
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(2,1)))]);


    j=j+1;

end
fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
[degree,gain]= textread(['Z_Y_co_polar_1300.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_co_ploar_1300_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f     %f  ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100
```

```
    fprintf(fid,' \n  %f     %f  ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_Y_co_ploar_1300_cylinder_FINAL.txt'], '%f
   %f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_co_ploar_1300_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f     %f  ', [degree(n); 10*log10((10^(Gt_1300
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(2,1)))]);


    j=j+1;

end
fclose(fid);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[degree,gain]= textread(['Z_Y_co_polar_1640.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_co_ploar_1640_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f     %f  ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100
```

```
    fprintf(fid,' \n  %f     %f  ', [degree(n); (10*log10(gain(n)))])
        ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_Y_co_ploar_1640_cylinder_FINAL.txt'], '%f
    %f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_co_ploar_1640_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f     %f  ', [degree(n); 10*log10((10^(Gt_1640
        /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(4,1)))]);


    j=j+1;

end
fclose(fid);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



[degree,gain]= textread(['Z_X_co_polar_1640.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_co_ploar_1640_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f     %f  ', [-degree(y); (10*log10(gain(n+90)
        ))]);


    j=j+1;
```

```
end

for n=1:100


    fprintf(fid,' \n  %f      %f   ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_X_co_ploar_1640_cylinder_FINAL.txt'], '%f
   %f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_co_ploar_1640_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f      %f   ', [degree(n); 10*log10((10^(Gt_1640
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(4,1)))]);


    j=j+1;

end
fclose(fid);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




[degree,gain]= textread(['Z_X_cross_polar_1300.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_cross_ploar_1300_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;
```

```
    fprintf(fid,' \n  %f      %f   ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100


    fprintf(fid,' \n  %f      %f   ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_X_cross_ploar_1300_cylinder_FINAL.txt'],
   '%f%f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_cross_ploar_1300_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f      %f   ', [degree(n); 10*log10((10^(Gt_1300
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(2,1)))]);


    j=j+1;

end
fclose(fid);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




[degree,gain]= textread(['Z_Y_cross_polar_1300.txt'], '%f%f');
```

```
l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_cross_ploar_1300_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f     %f  ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100


    fprintf(fid,' \n  %f     %f  ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_Y_cross_ploar_1300_cylinder_FINAL.txt'],
   '%f%f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_cross_ploar_1300_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f     %f  ', [degree(n); 10*log10((10^(Gt_1300
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(2,1)))]);


    j=j+1;

end
fclose(fid);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%



[degree,gain]= textread(['Z_X_cross_polar_1640.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_cross_ploar_1640_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f      %f   ', [-degree(y); (10*log10(gain(n+90)
        ))]);


    j=j+1;

end

for n=1:100


    fprintf(fid,' \n  %f      %f   ', [degree(n); (10*log10(gain(n)))])
        ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_X_cross_ploar_1640_cylinder_FINAL.txt'],
    '%f%f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_X_cross_ploar_1640_cylinder_FINAL_102610.txt'],'w+');

for n=1:200
y=92-n;

    fprintf(fid,' \n  %f      %f   ', [degree(n); 10*log10((10^(Gt_1640
        /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(4,1)))]);
```

```
    j=j+1;

end
fclose(fid);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%




[degree,gain]= textread(['Z_Y_cross_polar_1640.txt'], '%f%f');

l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_cross_ploar_1640_cylinder_FINAL.txt'],'w+');

for n=1:100
y=101-n;

    fprintf(fid,' \n  %f      %f  ', [-degree(y); (10*log10(gain(n+90)
       ))]);


    j=j+1;

end

for n=1:100


    fprintf(fid,' \n  %f      %f  ', [degree(n); (10*log10(gain(n)))])
       ;


    j=j+1;

end


fclose(fid);


[degree,gain]= textread(['Z_Y_cross_ploar_1640_cylinder_FINAL.txt'],
    '%f%f');


l=length(degree);
j=1;
n=0;
fid=fopen(['Z_Y_cross_ploar_1640_cylinder_FINAL_102610.txt'],'w+');
```

```
for n=1:200
y=92-n;

    fprintf(fid,' \n  %f     %f  ', [degree(n); 10*log10((10^(Gt_1640
       /10))*(10^(gain(n)/10))/(Horn_gain_upstairs(4,1)))]);


    j=j+1;

end
fclose(fid);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

BIBLIOGRAPHY

Bibliography

[1] Xiaoning Ye Yahya Rahmat-Samii Fan Yang, Xue-Xia Zhang. Wide-band e-shaped patch antennas for wireless communications. *IEEE Transactions on Antennas and Propagation*, 49(7), July 2001.

[2] http://www.emtalk.com/mpacalc.php.

[3] Fan Yang Demir Veysel Elsherbeni Atef Wei-Chung Weng. Optimization using taguchi method for electromagnetic applications. pages 1 –6, Nov. 2006.

[4] Chi Ho Chan, Sai Ho Yeung, Wing Shing Chan, and Kim Fung Man. Uwb sickle-shape patch dipolar antenna with stable radiation pattern. pages 1993 –1996, june 2007.

[5] A.A. Alshehri and A.R. Sebak. A novel uwb planar patch antenna for wireless communications. pages 1 –4, july 2008.

[6] Nasimuddin, Zhi Ning Chen, T.S.P. See, and Xianming Qing. Multi-dielectric layer multi-patches microstrip antenna for uwb applications. pages 1019 –1021, oct. 2007.

[7] E. Chang, SA Long, and WF Richards. An experimental investigation of electrically thick rectangular microstrip antennas. *IEEE transactions on antennas and propagation*, 34(6):767–772, 1986.

[8] Mook-Seng Leong Ban-Leong Ooi, Shen Qin. Novel design of broad-band stacked patch antenna. *Antennas and Propagation, IEEE Transactions on*, 50(10):1391 – 1395, Oct 2002.

[9] Bulla G. Serafin P.-Fernandez C.R. Monser-G. de Salles A.A.A. Pedra, A.C.O. Bandwidth and size optimisation of a wide-band e-shaped patch antenna. pages 422 –426, 29 2007-Nov. 1 2007.

[10] Khalil K. Excell-P.S. See C.H. Abd-Alhameed, R.A. Simulation and measurement of broadband microstrip patch antenna for 3g wireless communications. volume 2, pages 477 – 480 vol.2, March-3 April 2003.

[11] Chahine S.A. Kabalan K.Y.-El-Hajj A.-Chehab A. Bzeih, A. Empirical formulation and design of a broadband enhanced e-patch antenna. pages 1 –9, March 2007.

[12] Misran N. Take T.C.-Moniruzzaman-M. Islam, M.T. Optimization of microstrip patch antenna using particle swarm optimization with curve fitting. volume 02, pages 711 –714, Aug. 2009.

[13] Cwik T. Rahmat-Samii-Y.-Manteghi M. Villegas, F.J. Parallel genetic-algorithm optimization of a dual-band patch antenna for wireless communications. volume 1, pages 334 – 337 vol.1, 2002.

[14] A. Hedayat, N.J.A. Sloane, and J. Stufken. *Orthogonal arrays: theory and applications*. Springer Verlag, 1999.

[15] Ranjit K. Roy. *Design of Experiments Using the Taguchi Approach*. John Wiley and Sons inc., 2001.

[16] CR Rao. Hypercubes of strength d leading to confounded designs in factorial experiments. *Bull. Calcutta Math. Soc*, 38:67–78, 1946.

[17] C.R. Rao. Factorial experiments derivable from combinatorial arrangements of arrays. *Supplement to the Journal of the Royal Statistical Society*, 9(1):128–139, 1947.

[18] C.R. Rao. On a class of arrangements. *Proceedings of the Edinburgh Mathematical Society*, 8(03):119–125, 1949.

[19] Elsherbeni-A.Z. Wei-Chung Weng, Fan Yang. Linear antenna array synthesis using taguchi's method: A novel optimization technique in electromagnetics. *Antennas and Propagation, IEEE Transactions on*, 55(3):723 –730, March 2007.

[20] Wei-Chung Weng, Fan Yang, and Atef Elsherbeni. Electromagnetics and antenna optimization using taguchi's method. *Synthesis Lectures on Computational Electromagnetics*, 2(1):1–94, 2007.

[21] Wei-Chung Weng, Fan Yang, V. Demir, and A. Elsherbeni. Electromagnetic optimization using taguchi method: a case study of linear antenna array design. pages 2063 –2066, july 2006.

[22] C.A. Balanis. *Antenna theory*. Wiley New York, 1997.

[23] L.C. Kempel, J.L. Volakis, and R.J. Sliva. Radiation by cavity-backed antennas on a circular cylinder. *Microwaves, Antennas and Propagation, IEE Proceedings* -, 142(3):233 –239, jun 1995.

[24] J.S. Dahele, R.J. Mitchell, K.M. Luk, and K.F. Lee. Effect of curvature on characteristics of rectangular patch antenna. *Electronics Letters*, 23(14):748 – 749, jul. 1987.

[25] N. J. A. Sloane. http://www2.research.att.com/ njas/oadir/.

[26] R.L. Haupt and SE Haupt. *Practical genetic algorithms*. Wiley-Interscience, 2004.