ANALYSIS OF EMISSION SOURCE MICROSCOPY THROUGH SIMULATION

By

JACOB DIXON

Bachelor of Science in Electrical Engineering

Oklahoma State University

Stillwater, Oklahoma

2015

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
May, 2018

ANALYSIS OF EMISSION SOURCE MICROSCOPY
THROUGH SIMULATION

Thesis Approved:

Dr. Chuck Bunting

Thesis Adviser

Dr. Jim West

Dr. Sabit Ekin

ACKNOWLEDGEMENTS

I thank my Mother and Father, Teresa and David Dixon and my sisters Maddi and Audrey for their love and support through my formative years, and unconditional willingness to lend an ear to talk through life's challenges.

I would also like to thank my thesis chair and advisors Drs. Chuck Bunting, Jim West, and Sabit Ekin for their guidance and experience that enabled the development of my thesis.

I would like to thank all members of the REFTAS lab past and present who provided me personal growth through applied and theoretical expertise. Specifically, I would like to thank Dr. Vignesh Rajamani who was a mentor during undergraduate capstone courses and then taught me the fundamentals of performing electromagnetic measurements. I also want to thank Neda Nourshamsi, who was always eager to give advice when needed.

Finally, I want to thank Sam Connor of International Business Machines (IBM), who provided unwavering mentorship while constructing the emission source microscope, which yielded physical results and the inspiration that lead to the development of this thesis.

Name: Jacob Dixon

Date of Degree: May, 2018

Title of Study: Analysis of Emission Source Microscopy Through Simulation

Major Field: Electrical Engineering

Abstract:
In 2016, a method of resolving an electric field radiation source by sampling in the far field was published. The result of this publication was a device, deemed the Emission Source Microscope (ESM). Following these developments, this paper presents a comparison and validation study between results created with an ESM, and a simulated environment which is designed to mimic the operation of a physical ESM. From the developmental process of this study, this paper addresses the issues that arise in Emission Source Microscopy and present best practices associated with application. This paper concludes with a study of this assertion by creating a simulation environment that compares ESM performance using two antennas of differing gain and beamwidth characteristics.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

The Emission Source Microscope (ESM) is a tool for researchers and engineers that is designed to provide a further understanding of the electromagnetic environment around us. It does so by providing graphical insight into RF emission sources on a macro scale. This is in contrast to a near field scanner, which is limited to close proximity measurements and can interfere with the scanned target. The ESM employs an optical algorithm to 'focus' the image. This is referred to as the back-propagation technique by its developers at Missouri S&T, and first detailed in a publication by Maheshwari [1].

An automated version of the ESM was desired by International Business Machines (IBM) because of its ability to provide an EMC engineer insight into the source of radiation from a device during compliance testing. This practical application of the ESM reduces the debug time spent by an EMC engineer to locate emissions sources, because it has finer resolution and is more repeatable. An emission source microscope was developed at IBM in the spring of 2017, and expands on MST's work by:

- Automating the data collection process to increase the EMC engineer's time efficiency, and notifies the user via a messaging application *Slack* the resultant scan image.

- Automating the data processing – the raw data collected from the microscope is processed after the collection without any user input.
- Utilizing a larger plotter to scan large equipment, such as full sized server racks.

This thesis provides a detailed analysis of the construction and operation of a physical ESM by providing a detailed discussion of ESM operation that is complemented with an array of graphics and figures. The spirit of this thesis is not to simply apply research already conducted, but also explore beyond prior art relating to Emission Source Microscopy by:

- Performing a validation survey of ESM results by observing the two-dimensional cross section of un-perturbed electric field data to provide an indication of how much the measurement antennas affect the results of an ESM scan.
- Building a simulation environment that is designed to mimic the operation of a physical ESM for direct comparison between measurement in simulation results.

Chapter 2 introduces the fundamentals of emission source microscopy. The chapter begins by discussing history and development of synthetic aperture radar (SAR), which provided the inspiration for the back propagation method, and is the computation engine behind the emission source microscope. The chapter goes on to derive the back propagation equation using plane wave theory. The chapter concludes with a discussion covering the theoretical limitations of an emission source microscope. Chapter 3 covers implementation of ESM with an in-depth discussion of the hardware required as well as the mathematics and data processing behind the software that runs an ESM. This chapter includes implementation details for two different hardware sets from both IBM and Oklahoma State University. Chapter 4 introduces a simulation

environment designed to measure the electric field in space without the influence of any receiving antennas. The results of this chapter serve as the inspiration for the work documented in chapter 5. Chapter 5 presents a comparison analysis between two different receiver antennas and their effects on the results of the emission source microscope after post processing. Chapter 6 concludes this work with an overview of results and closing remarks.

CHAPTER 2

FUNDAMENTALS OF EMISSION SOURCE MICROSCOPY

Synthetic aperture radar was developed by Carl Wiley in the early 1950's at Goodyear Aircraft Corporation. Wiley found a relationship between the instantaneous Doppler shift reflected back to the aircraft by an object, and its location with respect to a moving radar system [4]. Synthetic aperture radars are primarily used in air and spacecraft in a 'side-looking' radar configuration, which exploits the motion of the craft to increase the equivalent aperture size by coherently integrating over discrete time samples. This coherent integration allows synthetic aperture radars to achieve a finer spatial resolution than would be otherwise possible within the constraints of physical aperture size onboard an air or spacecraft [5].

Brown [4] illustrates the Fourier property applied to a coherently illuminated optical system:

"This interesting relation follows directly from the Kirchhoff diffraction integral and simply says that the amplitude spectrum of a two-dimensional function displayed at plan P_1 can be observed at plane P_2."

The electromagnetic analog to Brown's [4] work is applied in Maheshwari's work [1]. His back propagation algorithm is derived from a Fourier transform property first applied to synthetic

aperture radars (SARs) to enable measurement of radar antenna patterns, as well as radar antenna diagnostics and faulty antenna detection [1, 2].

The following two subsections cover supporting plane wave spectrum theory, and build up to the back propagation algorithm [1].

2-1 PLANE WAVE SPECTRUM THEORY

The relationship between the near-field and far-field electric field measurements in a planar form is represented by [3]:

$$E(x,y,z) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\mathbf{f}(k_x,k_y)e^{-j\mathbf{k}\cdot\mathbf{r}}dk_x dk_y \tag{2.1}$$

Where:

$$\mathbf{f}(k_x,k_y) = \hat{\mathbf{a}}_x f_x(k_x,k_y) + \hat{\mathbf{a}}_y f_y(k_x,k_y) + \hat{\mathbf{a}}_z f_z(k_x k_y) \tag{2.2}$$

$$\mathbf{k} = \hat{\mathbf{a}}_x k_x + \hat{\mathbf{a}}_y k_y + \hat{\mathbf{a}}_z k_z \tag{2.3}$$

$$\mathbf{r} = \hat{\mathbf{a}}_x x + \hat{\mathbf{a}}_y y + \hat{\mathbf{a}}_z \tag{2.4}$$

Equation (2.1) represents a modal expansion for a two dimensional planar system, the resultant wave spectrum equations will be directly applied to back propagation. The discussion to follow describes how discrete sampling of a two dimensional plane connects to that used by synthetic aperture radar.

In (2.2), $\mathbf{f}(k_x, k_y)$ represents the plane wave spectrum in the field of interest. For a planar, two dimensional wave spectrum, the z component of the electric field is set equal to zero. This results in the following planar wave spectrum equations:

$$E_x(x, y, 0) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{f_x}(k_x, k_y) e^{-j(\mathrm{k_x}x + k_y y)} dk_x dk_y \tag{2.5a}$$

$$E_y(x, y, 0) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathrm{f_y}(k_x, k_y) e^{-j(\mathrm{k_x}x + k_y y)} dk_x dk_y \tag{2.5b}$$

Where $f_x(k_x, k_y)$ and $f_y(k_x, k_y)$ are the following two dimensional Fourier transforms of their respective electric fields $E_x(x, y, 0)$, and $E_y(x, y, 0)$:

$$f_x(k_x, k_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_x(x, y, 0) e^{j(k_x x + k_y y)} dx dy \tag{2.6a}$$

$$f_y(k_x, k_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} E_y(x, y, 0) e^{j(k_x x + k_y y)} dx dy \tag{2.6b}$$

Where the x and y direction spectrum wavenumbers are represented by k_x and k_y.

The spectrum in a plane (x, y, z_0) can be found by [4]:

$$f_x(k_z, k_y, z_0) = f_x(k_x, k_y) * e^{-jk_z z_0} \tag{2.7a}$$

$$f_y(k_z, k_y, z_0) = f_y(k_x, k_y) * e^{-jk_z z_0} \tag{2.7b}$$

Where z_0 is an arbitrary distance from a source point $z = 0$, and satisfies the condition (x, y, z_0) and is perpendicular to $(x, y, 0)$.

k_z is the z axis propagation vector, which is defined as:

$$k_z = \begin{cases} \sqrt{k^2 - k_x^2 - k_y^2} & , \ k_x^2 + k_y^2 \leq k^2 \\ -j\sqrt{k_x^2 + k_y^2 - k^2} & , \ elsewhere \end{cases} \tag{2.8}$$

Where $k = \frac{2\pi}{\lambda}$, the free space wavenumber, and λ is the wavelength of interest, radiating from the source/device under test. The first conditional term defining k_z describes propagating waves. The second term describes evanescent waves. Because the ESM is designed to be used in the far field, the evanescent term is ignored [2]. Applied to the back propagation algorithm, the imaginary (evanescent) k_z terms are set to zero, yielding a purely real k_z [1].

2-2 BACK PROPAGATION EQUATION

The crux of the emission source microscope is the back propagation equation:

$$E(x, y, 0) = \mathrm{F}^{-1}[\mathrm{F}\{E(x, y, z_0)\} \cdot \ e^{jk_z z_0}] \tag{2.9}$$

This equation is a synthesis of plane-wave spectrum theory, where the field distribution can be represented as a superposition of plane waves, and the Fourier transform property in its synthetic

aperture radar derivation. This equation is applied to the electric field at the two dimensional plane z_0, which is perpendicular to the source point $z = 0$. This Fourier transform product is then multiplied by the exponential function, which represents the phase offset associated with the axis propagation vector k_z and distance from scan plane to the source plane $z = 0$. Finally, the inverse Fourier transform is applied to the previous product to yield the 2-D electric field plane at its source [1].

A graphical representation of the scan and source planes is shown in Figure 2.1.

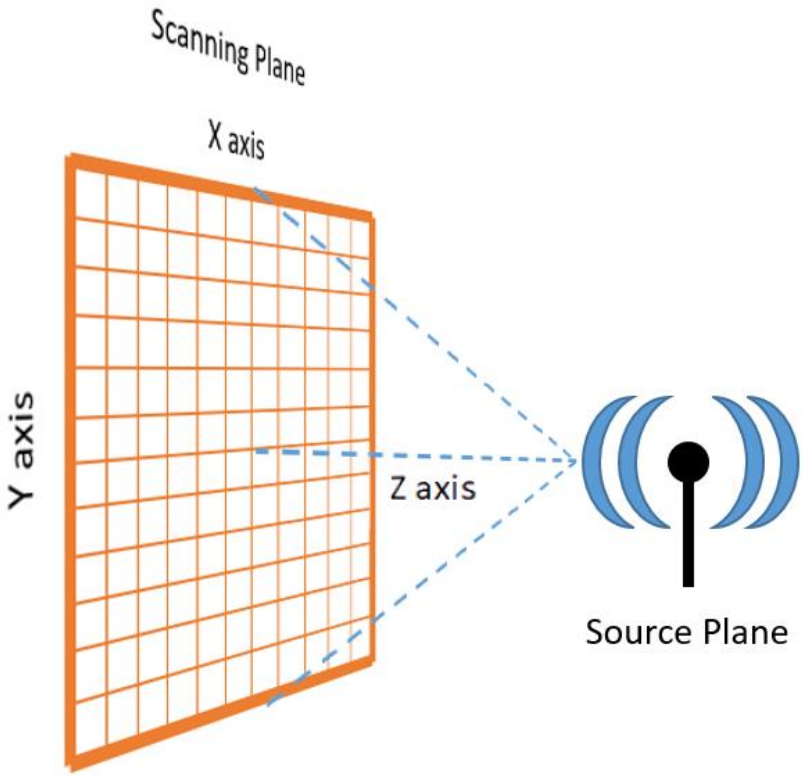

Figure 2.1: Illustration of ESM source to scan plane.

As shown in the illustration above, the plotter scans in the two dimensional space of X and Y. Z is the separation space between the source and scan planes. This distance must be at least multiple λ away from the source to guarantee only far field energy is sampled by the microscope.

2-3 LIMITATIONS

Using a plotter system allows the collection of high resolution images using only a single detector antenna and sweeping that detector in space. However, ESM relies on the emission source to be time invariant, or have a period well known and synchronize the ESM to the field variance. In practice, this limitation means the best source to study is a continuous wave signal. To overcome this limitation, an arrayed detector system could be implemented to study emissions in near real time, however such a device would be costly to implement, because it would require a complex signal detector/analyzer for each discrete array element.

Like optical microscopy, the ESM's maximum resolution is limited by frequency, numerical aperture, and minimum number of samples per wavelength i.e. Nyquist's theorem. More specifically, the ideal upper bound of scan resolution is limited to $\frac{\lambda}{2}$, under the idealistic conditions that the aperture angle is 90 degrees, vacuum medium, and numerical aperture (defined in equation 1.11) is equal to 1. The resolution of a reconstructed image is given by the equation [1]:

$$R = \frac{\lambda}{2NA} \tag{2.10}$$

where R is the resolution, and NA (numerical aperture) is defined as:

$$NA = n\sin(\theta) \tag{2.11}$$

where n is the refractive index of media, and θ is ½ the aperture angle (in radians). While the realistic resolution is greater than the theoretical lower bound, the physical scan resolution should be targeted at $\frac{\lambda}{2}$ to satisfy Nyquist's spatial sampling theorem [1].

A formula for finding the spectral resolution is as follows:

$$\Delta s_x = \frac{2\pi}{N\Delta x} \tag{2.12}$$

$$\Delta s_y = \frac{2\pi}{M\Delta y} \tag{2.13}$$

Where (N, M) is the total number of scan points in the (x, y) axis. Δs_x , Δs_y represent the spectral resolution, and Δx, Δy is the physical distance between scan points.

Finally, the ESM requires a direct line of sight to take accurate measurements. Reflective or noisy environments at the frequency of interest can produce results that will not be accurate at the source plane.

The fundamental theory behind emission source microscopy presented in this chapter will be applied in the following chapters to explain how a physical emission source microscope was

developed (chapter 3), and to dig deeper into the operation of the device via a simulation analysis conducted in chapters 4 and 5.

CHAPTER 3

EXPERIMENTAL EMISSION SOURCE MICROSCOPE

This chapter explores the construction of an emission source microscope by outlining the equipment required to build an ESM, and a discussion of the software control required to perform data collection and processing. This software analysis is given in the form of both high-level software flow charts and an in depth discussion of the software.

A proof of concept emission source microscope was presented in 2017 by Maheshwari [1]. While this microscope provided results, their microscope had shortcomings that this work aims to overcome. Specifically, (1) their scanning receiver was controlled by hand, and due to this human control, the repeatability of measurements cannot be guaranteed. (2) The VNA was set to sample as the receiving antenna was in motion, leading to positional uncertainty and uneven separation of samples. (3) Their approach lacked a data processing scheme that could be repeated with ease. In short, the goal is an improved ESM that overcomes the issues of locational uncertainty, repeatability, and automation. This chapter explores the construction and practical application of such an emission source microscope.

3-1 CONSTRUCTION

An ESM was constructed at IBM in the spring of 2017 using an existing three-axis plotter system. Originally used as a near field scanner, the plotter was modified to accept dual ridge horn antennas designed to be used as a near field scanner.

Figure 3.1 below illustrates the wiring diagram of the RF input to the VNA. This RF network is general for both IBM and OSU emission source microscopes.

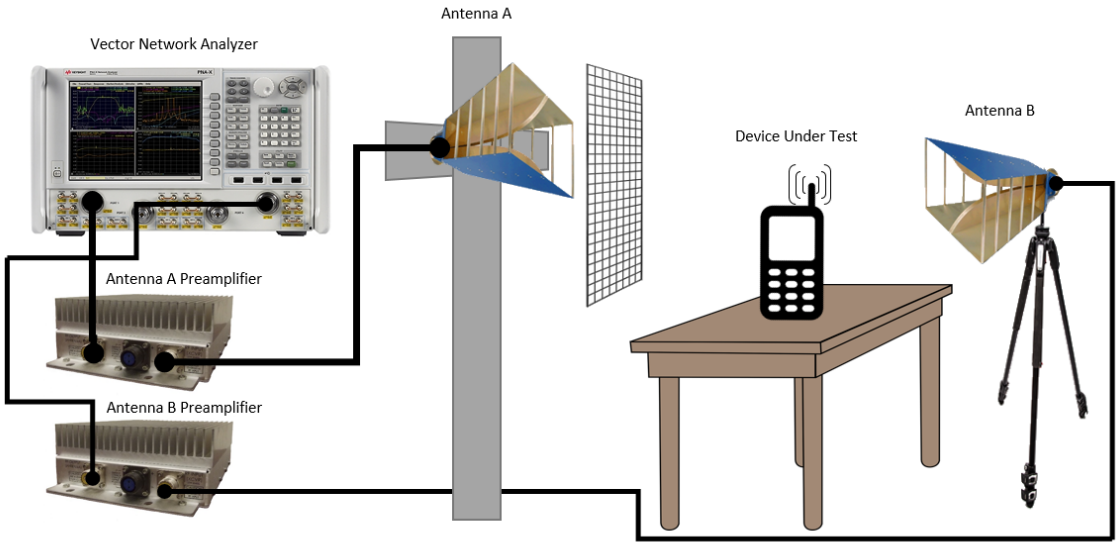

Figure 3.1: ESM RF connection diagram.

Figure 3.2 illustrates the IBM ESM in a typical operating environment. A complete list of components and hardware used to construct the IBM ESM is located in Appendix I.

Figure 3.2: Typical ESM test configuration. In this photo, debugging data processing system of the ESM.

3-2 ESM AT OSU

Oklahoma State's plotter controller has improved on IBM's by including support for ramping velocity while moving between positions. This modification reduces vibrations of the antenna, providing a more stable measurement. Additionally, the stepper motor controller supports the industry standard G-code CNC language, making the data collection program compatible with many more industrial motor controller offerings.

Figure 3.3 depicts the OSU ESM in operation. In this photo, the tablet/laptop is controlling both the motor controller. An effort was made to reduce reflection off of the reflective metallic surfaces by lining the surrounding surfaces with electromagnetic absorber material to both deflect

and absorb energy that would otherwise reflect from a metallic surface and alter the measurement environment. Two single axis linear actuators were used to create a two-dimensional axis system my mounting the end stop of one actuator to the movement table of the other. The axes were controlled with a repurposed 3D printer stepper motor driver/controller system using the universal CNC G-code language. Standard gain horns were used for both receivers A and B. More detailed information about the parts used to construct this emission source microscope can be found in Appendix I.

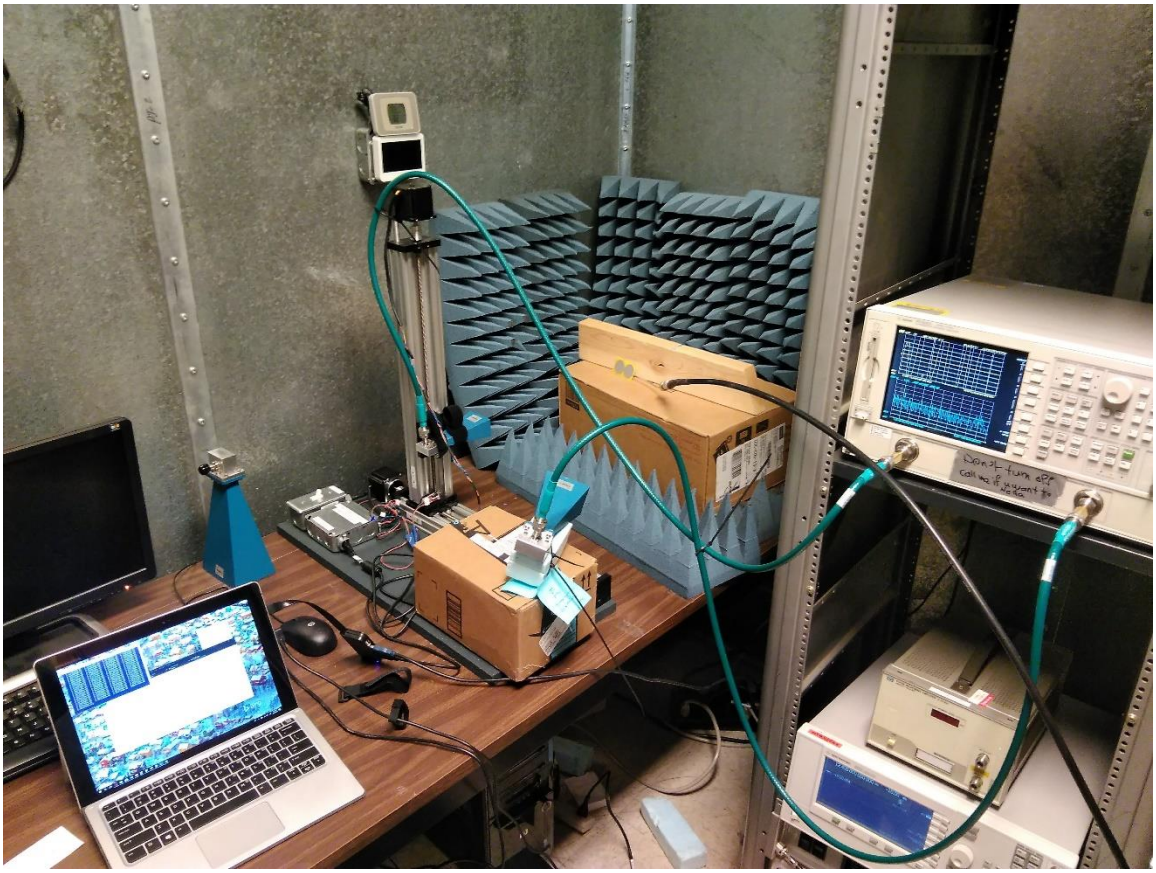

Figure 3.3: Oklahoma State ESM.

3-3 DATA COLLECTION

An Emission Source Microscope is the application of the back propagation algorithm. In practice, a fixed "anchor" point is required to establish a baseline magnitude and phase reference for the data collected. Situating a statically placed antenna within the line of sight of the scanned object accomplishes this baseline positioning. For notation, we will call this reference $Antenna\ B$. In contrast, another antenna is mounted to a plotter, and moves in a discrete (x, y) grid, and is referred to as $Antenna\ A$. Then, a $\frac{Antenna\ A}{Antenna\ B}$ field strength measurement is taken utilizing the $\frac{A}{B}$ detector function of a vector network analyzer (VNA). The plotter movement is controlled via a stepper motor driver that allows the master controlling PC to move the motors to the desired positions. Figure 3.4 illustrates a system level diagram detailing the RF connection and motor control systems of an emission source microscope.

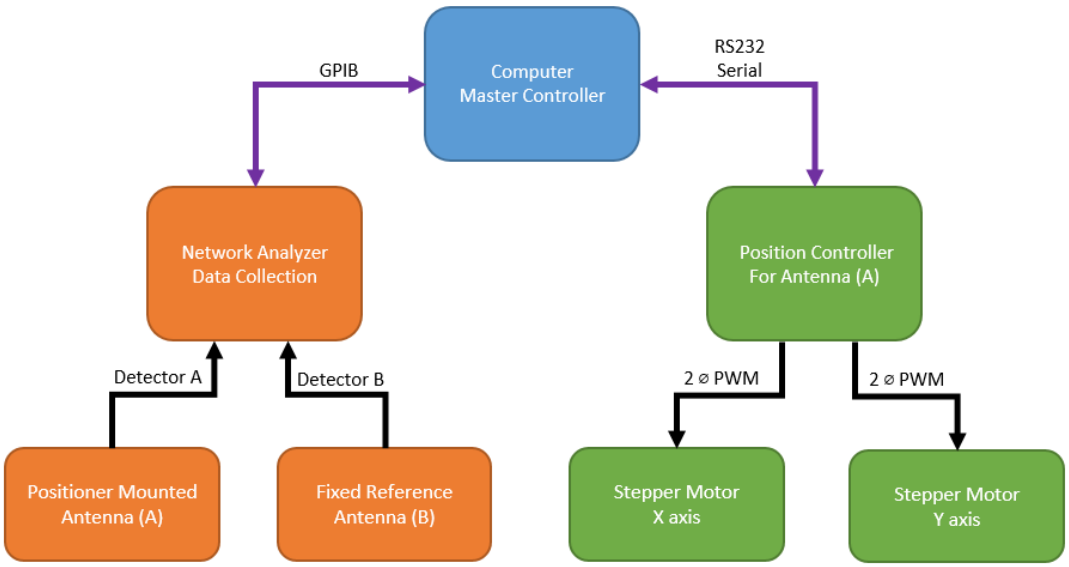

Figure 3.4: General ESM System Diagram.

A *vector* network analyzer was chosen to perform the measurements because the phase component is critical to resolve the focused image back to the source plane [1]. Scalar analysis equipment could resolve the amplitude of the scan plane, but the lack of complex information would make the back propagation to the source plane impossible to compute. The requirement of phase information is evident in equation 2.9 as the phase term of the inverse Fourier transform. The network analyzer is set to collect data in $\frac{detector\ A}{detector\ B}$ mode, or the ratio between receiver port A; the left most port on VNA in Figure 3.2, and B; the rightmost port in the figure. Note that an ESM measurement only uses the detector mode of a VNA, and that the stimulus features are not used. In the case of the IBM ESM, the detector A and B ports were connected through pre-amplifiers. These preamplifiers were used in the IBM ESM to raise the dynamic range seen at the VNA, which would allow the ESM to resolve radiation sources with lower radiated power. A sufficiently powerful radiation source above the instrument's noise floor would allow ESM operation without preamplifiers. This configuration is used at OSU, as preamplifiers are unavailable.

Antenna A is connected to the XYZ positioner, which allows it to move in space. During a scan, the antenna will move in a 2D grid pattern depicted above. Antenna B is kept in a fixed location and acts as a reference in the A/B measurement. Both antennas are pointed at the center of the DUT.

3-4 ESM CONTROL PROGRAM

The control system of the emission source microscope is broken into two separate programs: the data collector, and the radiation source plotter. State diagrams were illustrated to explain the steps involved in data collection and interpretation in each program. The first part of this section will present the corresponding state diagrams for each program, then follow with an in depth discussion of each program. Source code for the data collector and radiation source plotter programs has been made available in Appendix II A and II D, respectively.

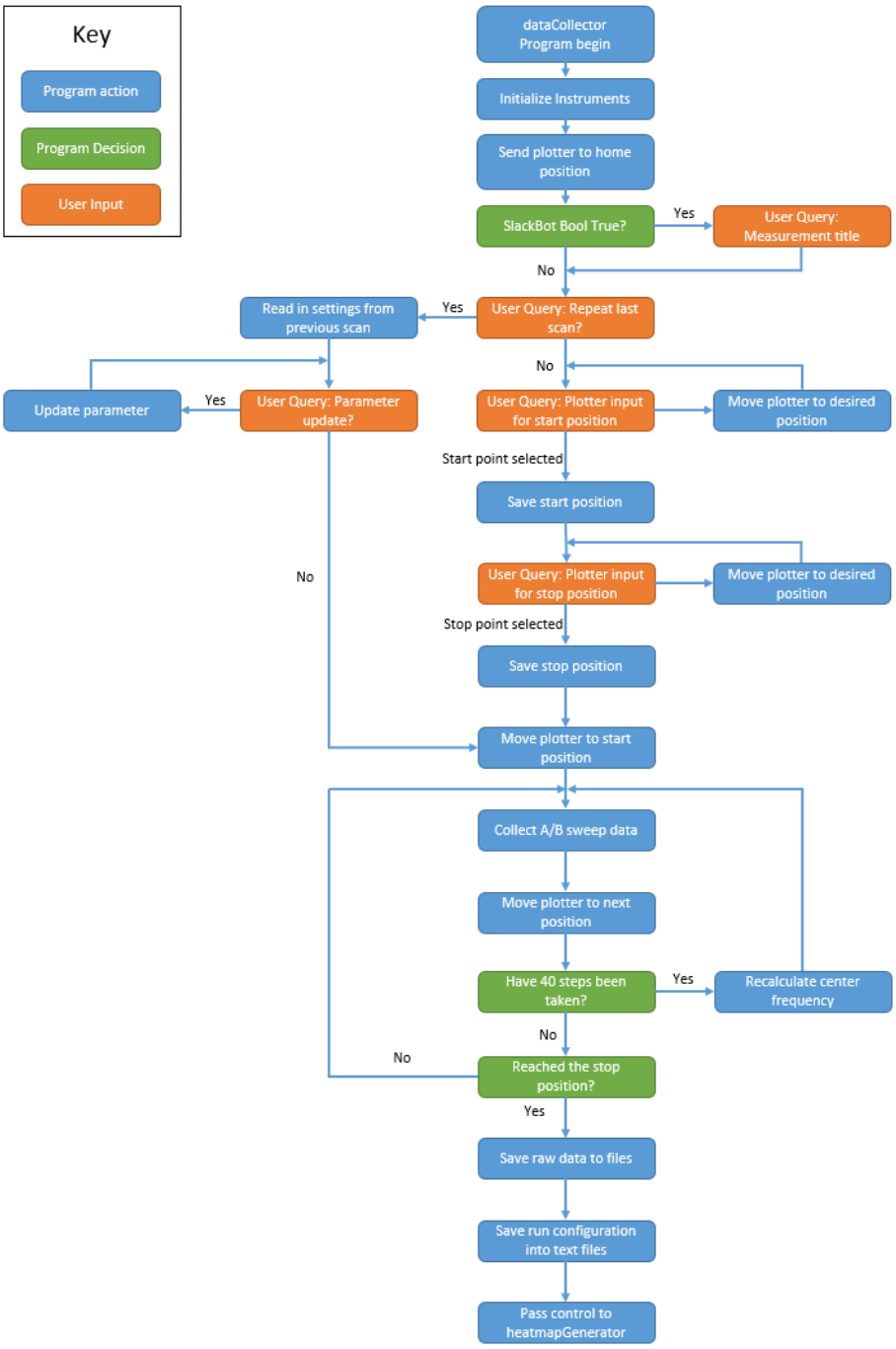

Figure 3.5: State diagram of the data collector program.

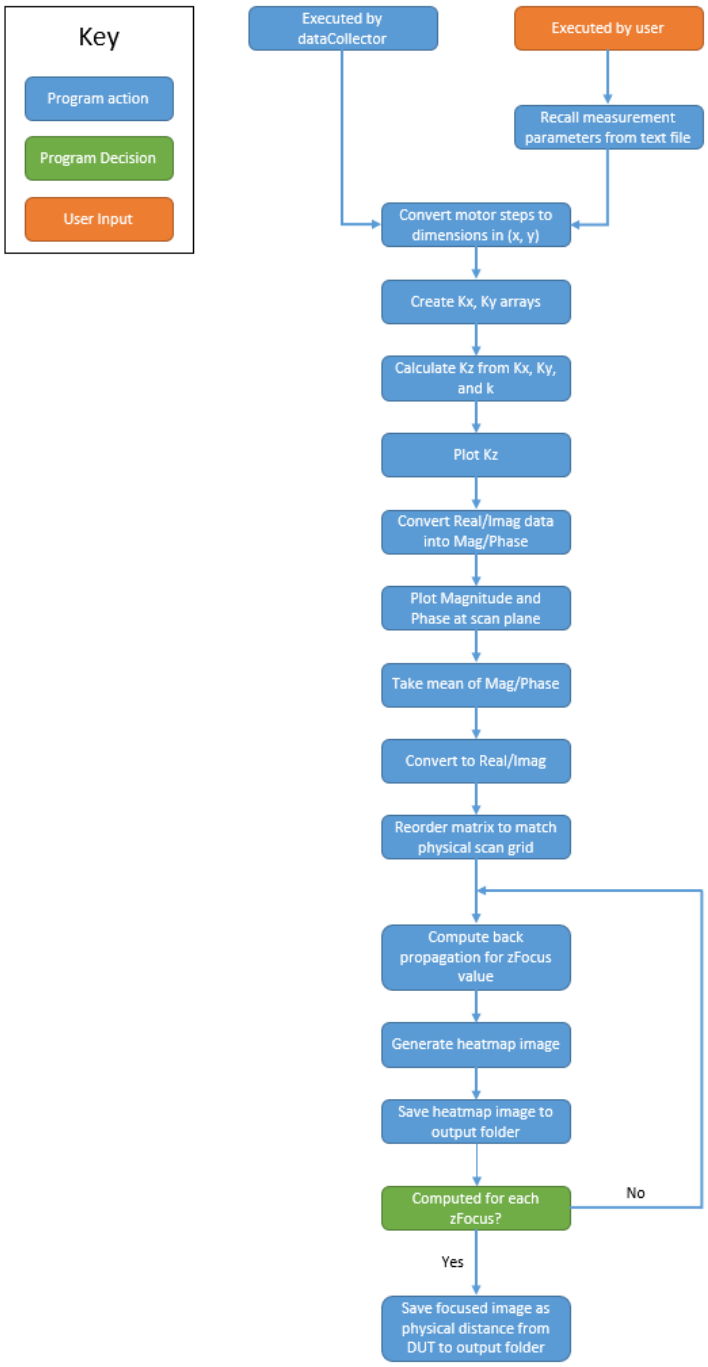

Figure 3.6: State diagram of the electric field plotter program.

These programs are named after their respective functions. The data collector controls the plotter and VNA and organizes the data into .csv files for the electric field plotter to interpret. The electric field plotter reads in this data and processes/performs the back propagation algorithm on the data and generates the visual output.

As compared to the original work produced at IBM, Oklahoma State's data collector program has been modified to work with different collection hardware readily available, which has been outlined in the construction section of this chapter and parts detailed in Appendix I. The electric field plotter program has also been modified to (1) add an autofocusing algorithm, and (2) to accept simulated source data, which is used extensively in chapters 4 and 5.

To honor a confidentiality agreement with IBM, the lines of code directly pertaining to the back propagation algorithm have been redacted from the Appendix. However, the following program explanation is fully describes the arithmetic processed by the back propagation algorithm.

3-5 PROGRAM DISCRIPTION

Two-dimensional matrices are created to represent the spectrum wavenumbers in the x and y-axis. Let these matrices be KX and KY, respectively. The dimensions of KX and KY are dependent on the number of scan points in each axis.

Next, let there be a scaling factor S, determined by:

$$S = \frac{\frac{2\pi}{|dx, dy|}}{shape(x, y)} \tag{3.1}$$

One-dimensional arrays kx_1 and ky_1 are created from the element wise multiplication of the list of K values in the X and Y axis, multiplied by the scaling factor S. Giving:

$$kx_1, ky_1 = Klist[X, Y] \cdot scale[X, Y] \tag{3.2}$$

1. First, the program converts motor steps into traveled millimeters in the X and Y axis. The program uses a conversion rate which is proportional to the mechanical gear to stepper position ratio. This conversion rate is reliable because the stepper motors position are both precise and repeatable.

2. Next, two mesh grids (evenly spaced 2D array of values) defined as KX and KY are created. These mesh grids represent the spectrum wavenumbers in the X and Y axis based on the axis dimension and numerical spacing between each dimension element. A detailed explanation of calculation is: 1D arrays ($kx1$, $ky1$) are created from the element wise multiplication of the list of K values in the X and Y axis, and the scaling factor. The scaling factor is determined by (2.1). Giving (2.2).

3. K_z is the z component of the wave vector. It is calculated through the element wise operation $K_z = \sqrt{K - K_x^2 - K_y^2}$ where $K = \frac{2\pi}{\lambda}$, the free space wavenumber

4. K_z is plotted on a real/imaginary axis and saved to the measurement run output folder.

5. The measurement data from the VNA was collected in real and imaginary format. At each sampling location, 201 time domain (0 span frequency) measurements were made.

To preserve phase coherency, the mean of these 201 real/imaginary data points is taken prior to the program converting to magnitude & phase for the resultant graphs.

6. This time-averaged real and imaginary data is used in the following back-propagation calculations.

7. The resultant 1D real and imaginary arrays are of length number of scanned points. These arrays are combined element wise by $abmatrix = (realDataAB + j \cdot imaginaryDataAB)$. This array is then re-ordered to match the dimensions of the physical scan plane of size.

8. The data is now in the final format of the emission source at the scanning plane. The back-propagation method is now applied to the data to focus the image to the emission source. The following is the data processing equations used:

$$Essx = \left| abMatrix \cdot meanDataB \cdot e^{jangle(abMatrix)} \right| \tag{3.3}$$

Where $abMatrix$ is the 2D data at the scan plane, and $meanDataB$ is the phase reference data from receiver B.

$$Sx = FFT_{2D}(Essx) \tag{3.4}$$

Sx is the Fourier transform of $Essx$.

$$Sx_h = Sx \cdot e^{-jK_z - zFocus} \tag{3.5}$$

Sx_h incorporates the Kz array and focusing distance $zFocus$ (in meters).

$$Final\ Image\ Matrix = IFFT_{2D}(Sx_h) \tag{3.6}$$

The final image is a result of the inverse Fourier Transform on the Sx_h matrix.

9. The program calculates the correct value of $zFocus$ through the direct correlation between the magnitude of the receive energy and the precise focal point. As the magnitude increases, so does the convergence to the focal point. The program finds this focal point by generating datasets from 1 centimeter through 1 meter in 1 centimeter increments and sorting by highest received magnitude. The program names the generated electric field plot at distance $zFocus$ FocusedImage.png for the operator's convenience.

A list of technical notes associated with operating the emission source microscope is located in Appendix III. This appendix addresses tradeoffs and operational limitations including dynamic range of the system, scan resolution, environmental effects on scan quality, and limitations on the source signal to produce reliable results.

3-6 IBM RESULTS

This section highlights the utility of the ESM by presenting results taken a proof of concept experiment conducted at IBM. It consists of two bowtie antennas being driven by a signal generator with a continuous waveform at 12.4 GHz. The first scan, labeled “IBM Bowtie Antennas” had a scanning distance (defined as distance from antennas under test to scanning receiver antenna) of approximately 15 cm. The second scan had a distance of 20 cm.

Table 3.1: IBM Bowtie test 1 configuration.

IBM Bowtie Antennas	
Frequency:	12.4 GHz
RX Antennas Polarization:	Horizontal
Focal Length:	14 cm
Horizontal Scan Distance:	54 cm
Vertical Scan Distance:	27 cm
Horizontal Resolution:	76 points
Vertical Resolution:	46 points
Total Scan Resolution:	3496 points

The graphical results of this scenario are shown in Figures 3.7 – 3.10. Figure 3.7 represents the physical antennas scanned in this scenario. The real and imaginary data compiled by the data collection program is converted into phase and linear magnitude to generate figures 3.8 and 3.9 respectively. The back propagation algorithm is applied using the scan data to synthesize the source plane, shown in figure 8. Figure 3.8 provides a graphical representation of the phase change across the scan plane relative to the fixed position reference antenna B. The general alignment in registration of both antennas are evident when comparing figure 3.7 to figures 3.8-10. In figure 3.8, the phase change increases as the distance away from the radiation sources increases. Figure 3.9 represents the linear magnitude of electric field data observed at the scan plane. This graph can be used to compare the 'before' and 'after' results of applying the back propagation algorithm, whose results are shown in figure 3.10.

Figure 3.7: Bowtie antenna configuration.

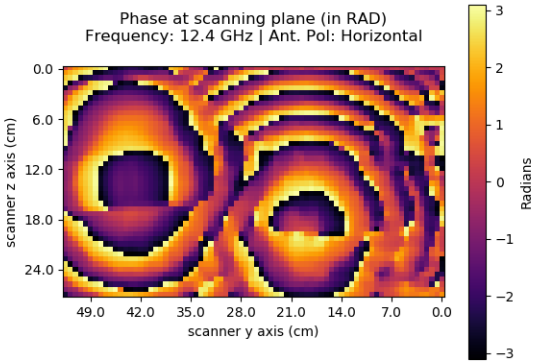

Figure 3.8: Bowtie phase (wrapped) at scan plane.

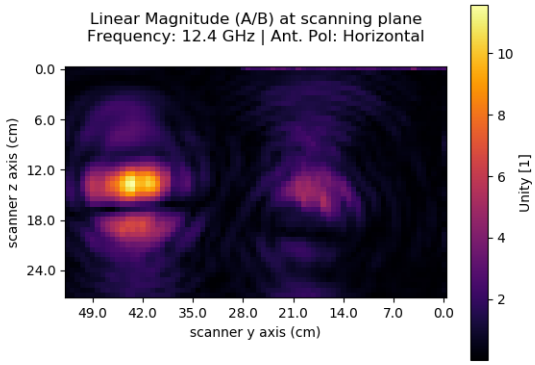

Figure 3.9: Scan plane magnitude result.

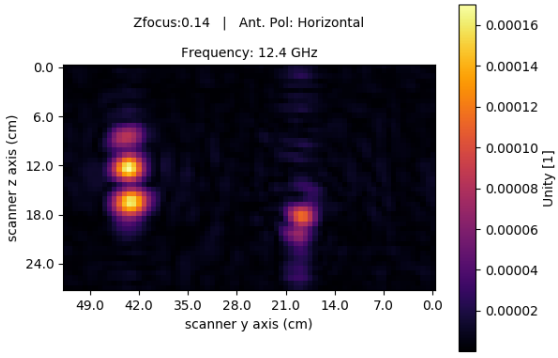

Figure 3.10: Source plane synthesized via back propagation algorithm.

Figure 3.11 overlays the image of the bowtie antennas on top of the synthesized source plane electric field plot. The registration is good for the left antenna. However, exact placement of the radiating antennas was not logged at the time of measurement. As a result, the registration of the second antenna is off by about 5 cm. In subsequent experiments, measuring the dimensions of the area under test was better controlled.

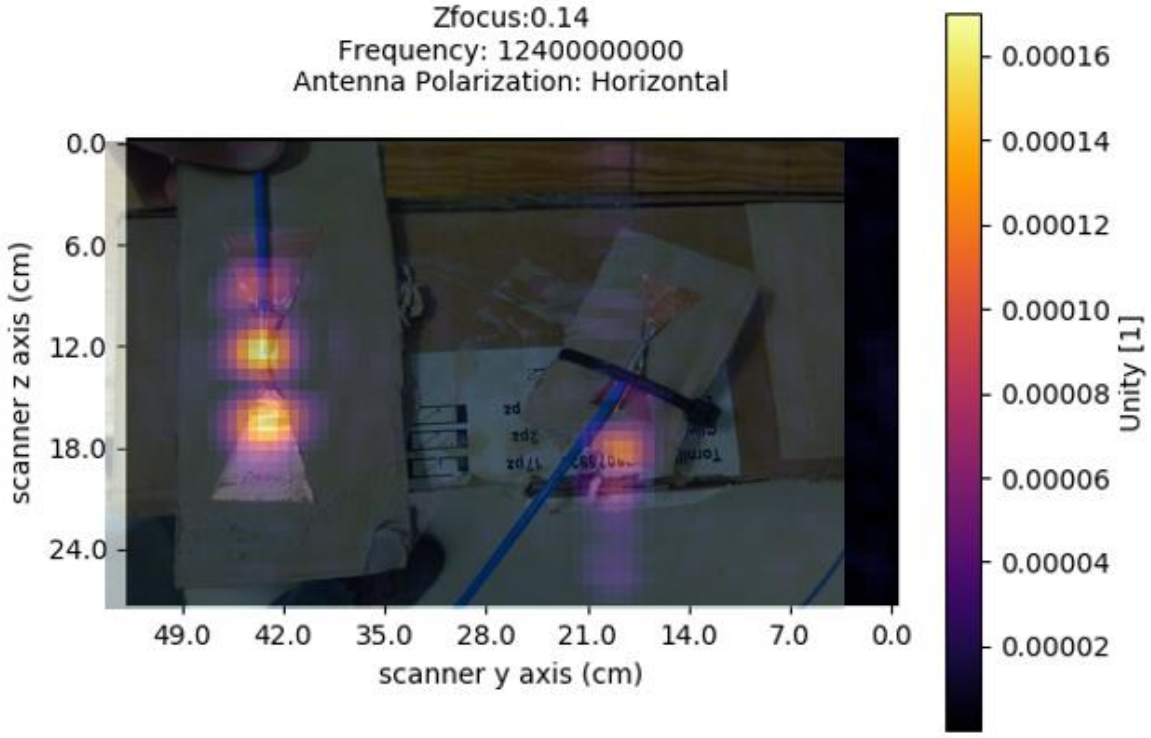

Figure 3.11: Antenna graphic overlaid on relative electric field plot.

Test 2 with the bowtie antenna kept all variables but the antenna location the same. The antennas under test were translated 5 cm backwards, bringing the distance from the antennas under test to the scanning antenna to 20 cm. As an unintended result of this movement, the antennas were also shifted upwards 6 cm as well.

Table 3.2: IBM Bowtie test 2 configuration.

IBM Bowtie Antennas: Test 2	
Frequency:	12.4 GHz
RX Antennas Polarization:	Horizontal
Focal Length:	23 cm
Horizontal Scan Distance:	54 cm
Vertical Scan Distance:	27 cm
Horizontal Resolution:	76 points
Vertical Resolution:	46 points
Total Scan Resolution:	3496 points

The result was a movement in the focal length from 14cm to 23 cm, shown in figure 3.13. The back propagation algorithm successfully resolved the tight radiation pattern. The phase and linear magnitude of this can also be found in figures 3.11 and 3.12, respectively.

Figure 3.12: Bowtie antenna configuration.

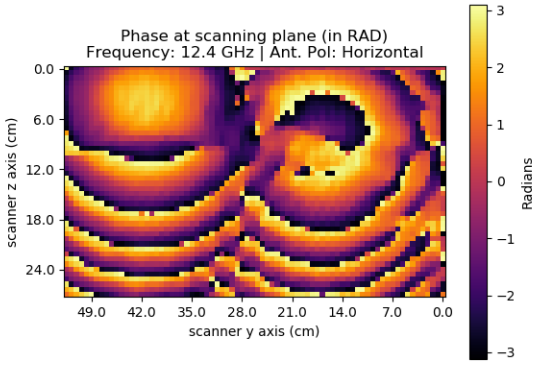

Figure 3.13: Bowtie phase (wrapped) at scan plane.

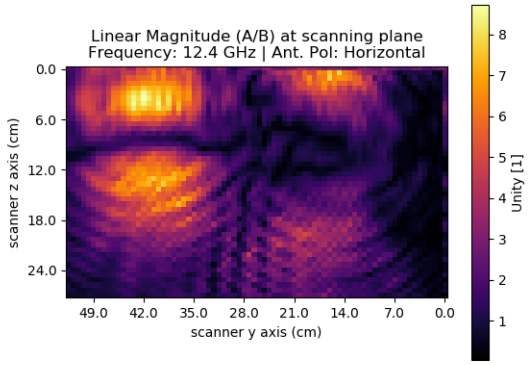

Figure 3.14: Scan plane magnitude result.

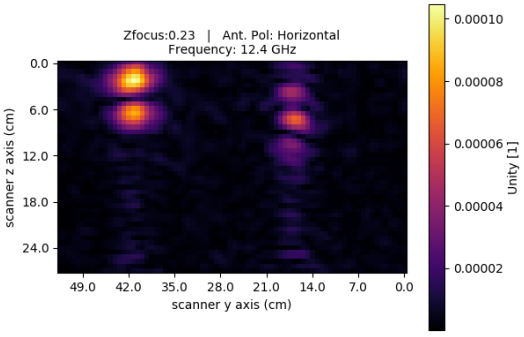

Figure 3.15: Source plane synthesized via back propagation algorithm.

3-7 OSU RESULTS

The intent to create an ESM at Oklahoma State was to reproduce results gathered using IBM's equipment. In order to accomplish this goal, versions of the data collection and radiation plotting programs were re-written and/or modified to operate on the hardware readily available at Oklahoma State to use. The OSU ESM system was de-bugged and refined to a point where it was collecting scan data and automatically generating the resultant electric field plots after the back propagation algorithm was applied. However, the results received from this ESM were not consistent with those seen when using IBM hardware. Further analysis of results and debugging has uncovered a problem with the results given by the VNA while in detection (tuned receiver) mode. Figure 3.16 depicts the frequency domain measurement of a 7.5GHz CW input signal at +10 dBm.

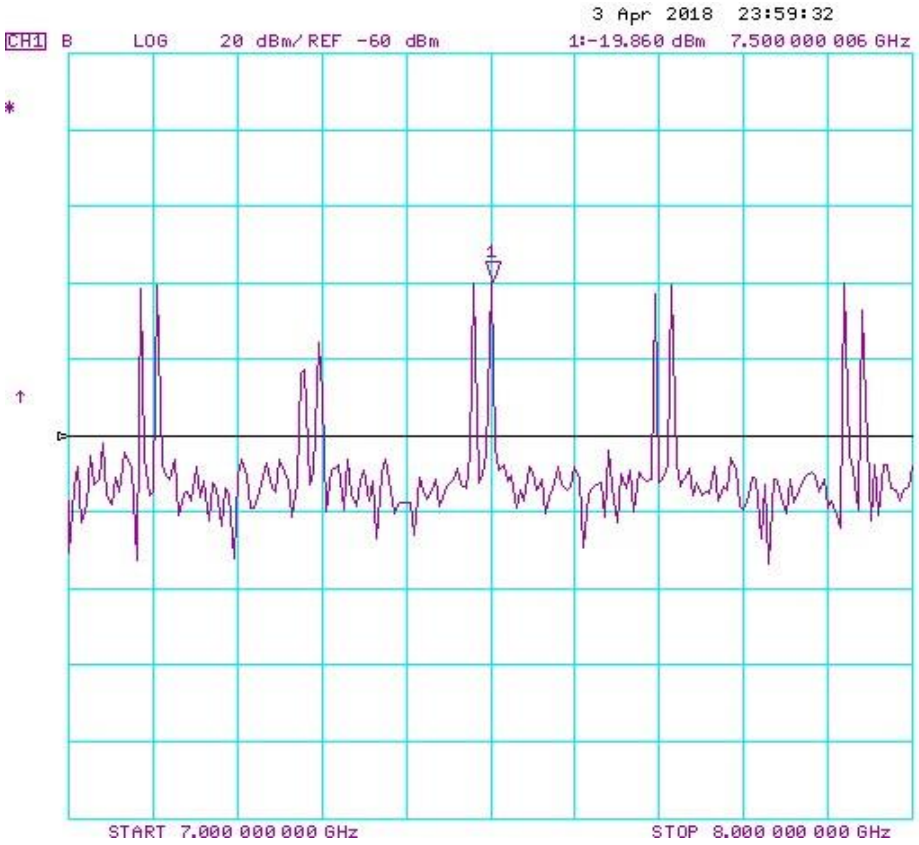

Figure 3.16: VNA measurement of channel detector B.

The receiver's frequency domain graph contained periodic impulses indicative of harmonics that were not rejected by the receiver. To verify that this issue was related to the VNA and not the source generation, a second VNA (model Agilent E5071C) was employed. The results with this second VNA is shown in figure 3.17.

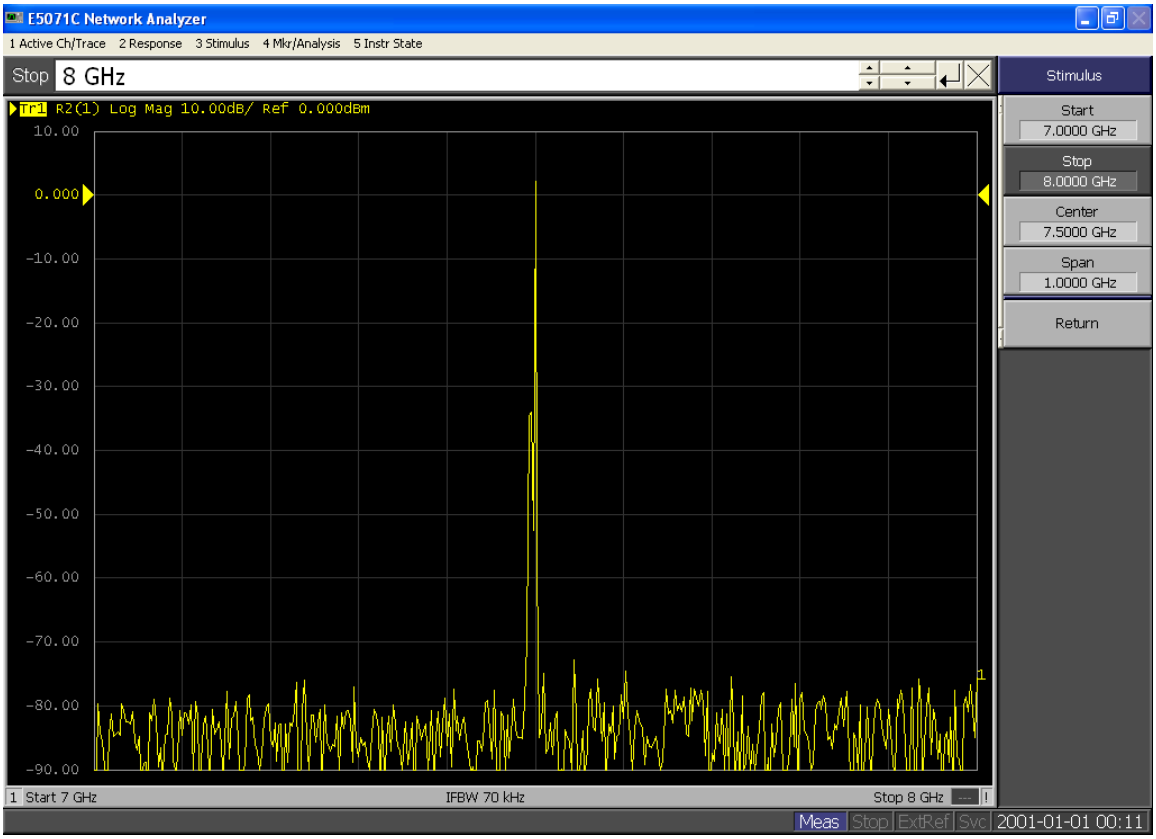

Figure 3.17: Second VNA detector B measurement for comparison.

With all input conditions were held constant, the detector mode on this second VNA worked as expected and produced a single did not produce the periodic signal seen on the HP 8722ES. It's possible that this phenomenon is related to the time domain gate filter option installed on the HP 8722ES, but steps to verify of this claim has not been explored [6].

3-8 CONCLUSION

The results from this chapter provide insight in the radiation characteristics of different antennas, environments, and frequencies. Additionally, a spatial analysis was conducted to build confidence in the registration of the electric field plot data to the actual positioning of the device(s) under test.

The next chapter documents the pursuit of creating an emission source microscope in simulation. This pursuit is to verify our understanding of the theory of operation behind the ESM and allow us to conduct research on emission source microscopy without the aforementioned problems encountered with the physical equipment.

CHAPTER 4

COMPARISON OF PHYSICAL VS. SIMULATED ESM

The first half of this chapter explores the creation of a scenario built in Dassault Systèmes / Computer Simulation Technologies Microwave Studio to closely resemble the physical measurement environment introduced in chapter 3. The second half of the chapter performs the comparison analysis of the physical ESM vs a simulation environment by forcing a large simulation boundary region (44 cm^3) and discretely sampling the magnitude of the electric field in a two dimensional plane normal to the direction of propagation and 20 cm away from the radiating bowtie antennas. The set of discretely sampled data was then exported from CST Microwave Studio to be processed by the same back propagation program that created the physical ESM results.

This measurement scenario is made with two bowtie antennas of differing physical size. This inequality in dimension was chosen because it provides an indicator of antenna efficiency mismatch as shown in the resultant electric field magnitude. This antenna efficiency indication can then be compared to the reflection mismatch shown in the SWR (Standing Wave Ratio) reflection parameters of the two bowtie antennas.

4-1 THE BOWTIE ANTENNA

A bowtie antenna is a type of broadband dipole. It is practical in applications that require a planar surface to mount an antenna. In the case of the following experiment, a bowtie antenna was chosen it is inexpensive and simple to fabricate, and its broadband nature would allow for some fabrication error in its geometry and still be an efficient radiator. This first scenario was a proof of concept experiment conducted at IBM. It consists of two bow tie antennas driven by a signal generator with a continuous waveform at 12.4 GHz. The scan plane was approximately 15 cm above the source antennas. Figure 4.1 depicts the layout physically of the two antennas under test during this first scenario, with the scan plane being parallel to the antennas as they are pictured.

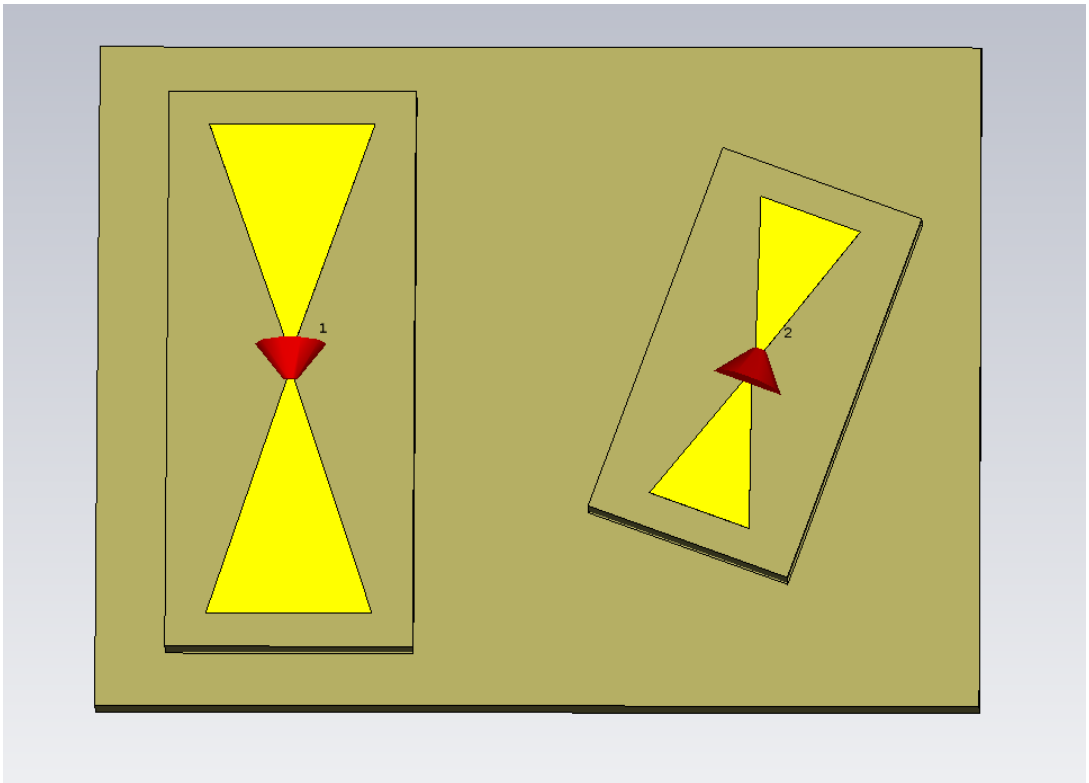

Figure 4.1: Bowtie antennas: 8 GHz tuned (left), 12.4 GHz tuned (right).

The larger of the two bow tie antennas shown in figure 1 was designed to resonate at 8 GHz. It has a total length of 28.2 mm, the maximum width at the ends of the bowtie of 9.4mm, and has a center gap spacing of 0.8 mm. The smaller antenna was designed to resonate at 12.4 GHz, the frequency of interest in the scan, and has a total length of 18.2 mm, maximum width of 6 mm, and center element spacing of 0.5 mm [7]. Both antennas use the built-in material property for lossy copper, with an electrical conductivity value of $5.96 \cdot 10^6 \frac{S}{m}$. The brown material behind both antennas is cardboard, with a relative permittivity of $\varepsilon_r = 1.8$ [8].

Figure 4.2 presents the SWR (standing wave ratio) of both bowtie antennas across frequency from 8 GHz to 15 GHz.

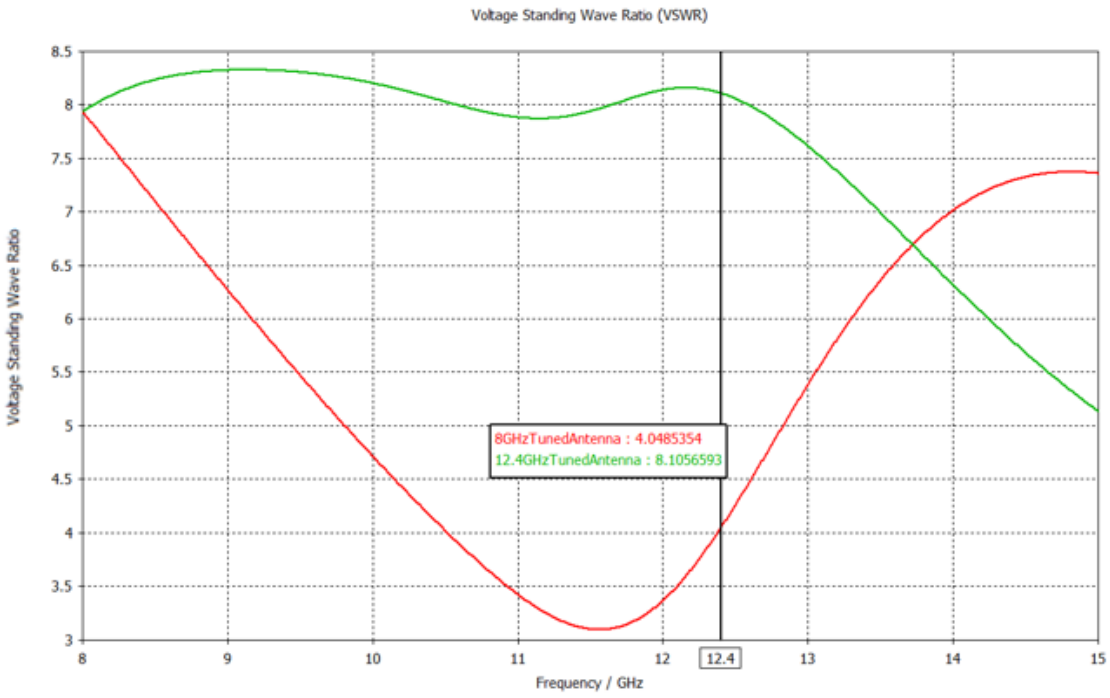

Figure 4.2: SWR of both bowtie antennas.

SWR, defined in equation (4.1), is the ratio of forward to reflected energy across an antenna. It is measured by exciting the antenna with a well-known energy source and then detecting the energy reflected back to the source [9].

$$SWR = \frac{1 + \sqrt{\frac{P_{reflected}}{P_{forward}}}}{1 - \sqrt{\frac{P_{reflected}}{P_{forward}}}} \tag{4.1}$$

This metric gives a complete figure of antenna performance because this data is based on a simulation designed to neglect energy loss due to antenna structure. In a physical antenna performance measurement, P_{foward} and $P_{reflected}$ must be driven and detected, respectively, with sensitive, calibrated network analysis equipment. The results presented above were generated in simulation, where the network analysis is performed in an ideal environment that accounts for non-ideal material performance as outlined above, and assumes all other environmental variables as ideal.

In these results, neither antenna had an excellent SWR value. At 12.4 GHz, the frequency of interest, the larger (more efficient) antenna with a SWR value of 4.04:1 and a reflection loss of 36%. The smaller antenna with a SWR of 8.1:1 lost 60.9% of its energy. The SWR and graphical data had good agreement. In figure 4.3, the left most antenna had a larger, more pronounced radiation signature than the smaller antenna. While the numerical solution will give you the exact amount of energy lost due to reflection, the graphical solution fulfils its goal of giving a user an ‘at a glance’ answer to the question ‘which antenna is more efficient at this frequency?’ [6]. This

relative, side-by-side comparison analysis is where the emission source microscope is most useful.

4-2 BOWTIE SIMULATION RESULTS & COMPARISON

This section of the chapter compares the results of the physical ESM to the simulation results. The following method was devised to extract the two dimensional electric field data from the simulation results to feed into the ESM radiation plotting program. (1) The stimulus sources are combined if multiple radiators are present in the simulation. Their excitations need to be combined so that the electric field can be observed as a sum of all contributing radiators. (2) Export a two dimensional cross section of the three dimensional electric field solution set, taken at the same cross sectional scan plane as the physical ESM data. (3) From the exported data, combine the electric field data E_x, E_y, E_z into the magnitude vector product of the electric field E, characterized by equation 4.2 [3].

$$\mathrm{E} = \sqrt{|E_x|^2 + \left|E_y\right|^2 + |E_z|^2} \tag{4.2}$$

(4) Importing electric field data into the ESM radiation plotting software to perform the back propagation functions on the data. The result is a side-by-side comparison to samples collected by physical hardware and samples generated in simulation.

The following results use scan plane data was taken 20 cm above the antennas, as illustrated in figure 4.3. The scan plane had dimensions of 44 cm^2. The volume above and below the scan plane was set to be 44 cm across as well, making the total bounding region 44 cm^3. The boundaries were set as the *open with space* condition on all sides.

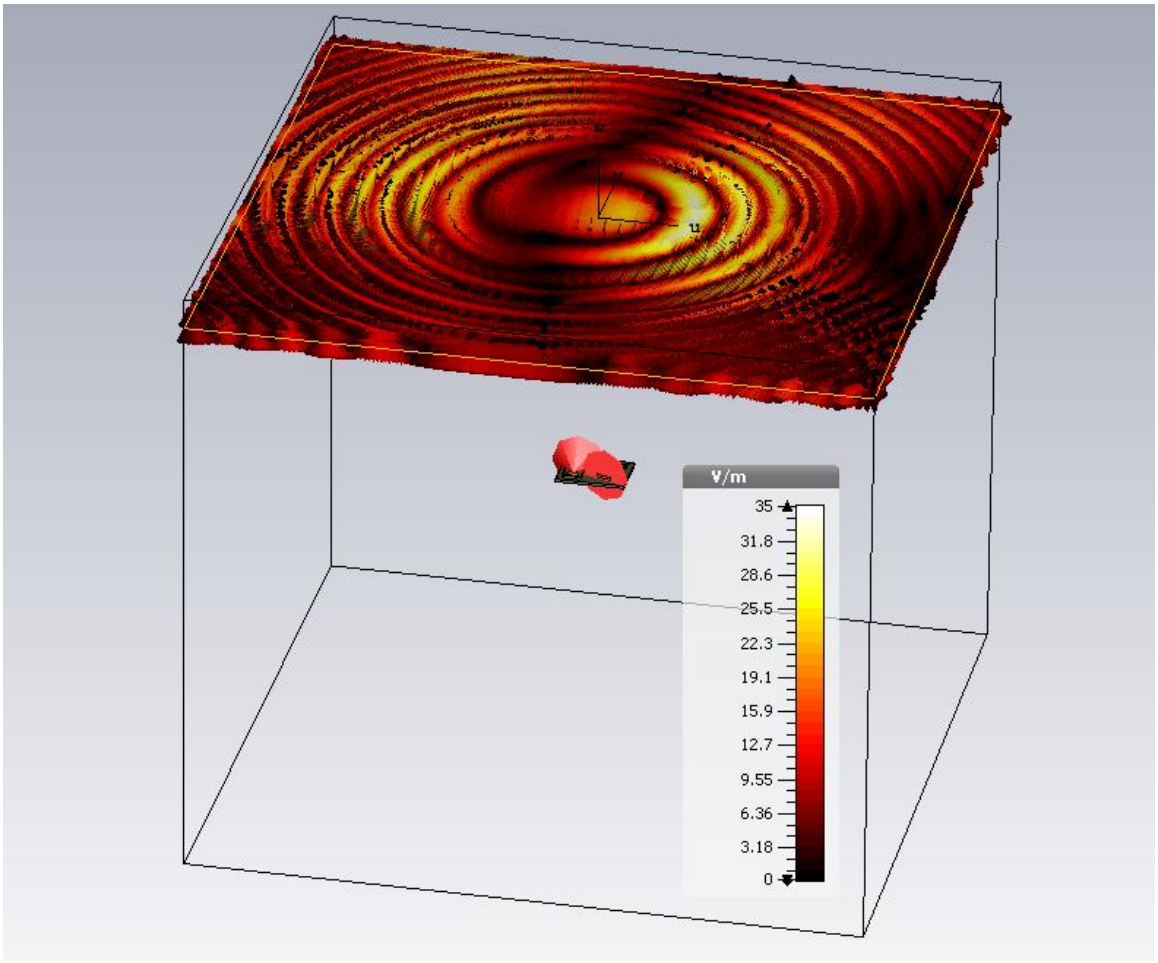

Figure 4.3: 2D cross section of electric field, 20 cm above radiation sources.

Figures 4.4, 4.5, and 4.6 contain the results of processing the two dimensional electric field cross section scenario as shown in figure 4.3. Figures 4.4 and 4.5 provide the phase and linear magnitude, respectively, of the two dimensional scan plane. Figure 4.4 indicates there is a lack of complexity in the phase at the center of the plot, where the radiation source is coming from. This would indicate that the energy is in the far field by the time the energy reaches scan plane. Figure 4.6 represents the focused magnitude of the electric field as a result of the back propagation program. Note that the most focused image is defined as the focal length that produces the maximum $\Delta\, Unity$ value.

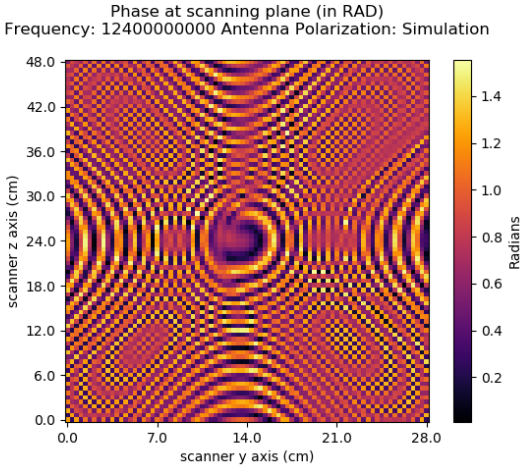

Figure 4.4: Relative phase across the scan plane.

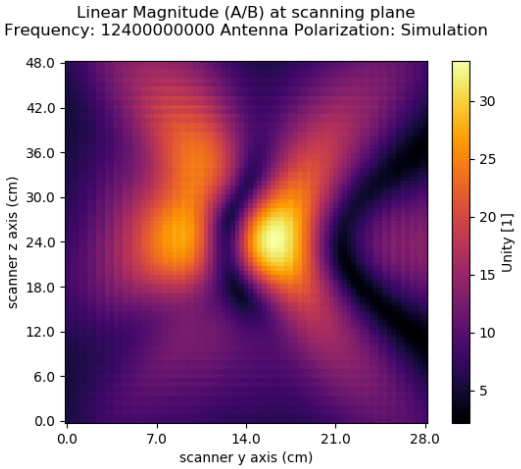

Figure 4.5: Scan plane magnitude result in simulation.

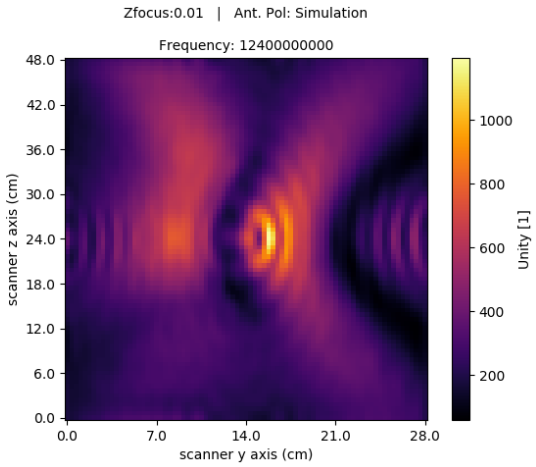

Figure 4.6: Source plane synthesized via back propagation algorithm from simulated results.

Physical ESM Result of bowtie antennas
Scan Distance: 20cm
Focal distance: 14cm

E-Field above simulated bowtie antennas
E-Field sample distance: 20cm
Focal distance: 10cm

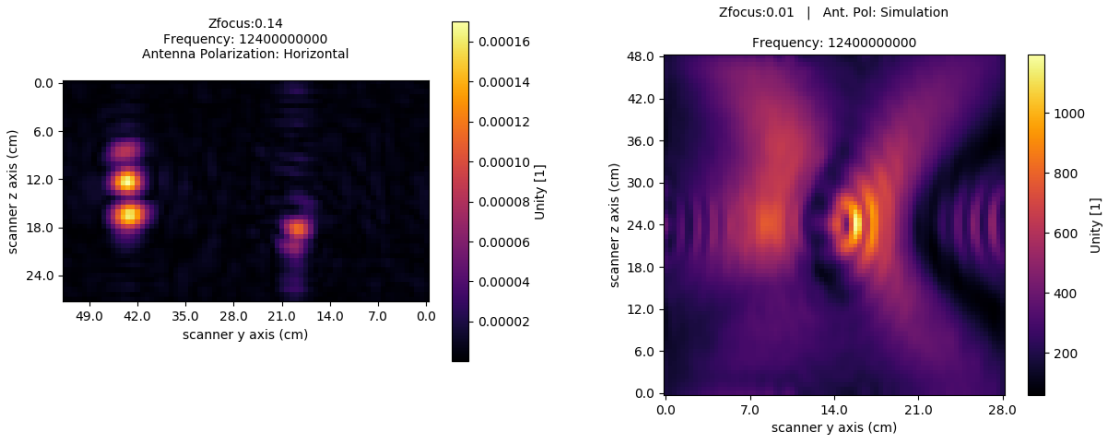

Figure 4.7: Comparison between physical scan vs. simulation back propagation results.

The comparison of these two resultant back propagation graphs in figure 4.7 suggests that a large difference between the data collected in a real measurement vs simulated electric field data exists. While the data collected in simulation did ‘focus’, the electric field at a distance of 20 cm, the rapid phase variance at that distance from the radiator inhibits the back propagation from scan to source plane.

To illustrate the contrast in scan plane positioning simulation results, figure 4.8 contains the comparison between contrast to taking a two-dimensional cross section of electric field at 10mm separation. Figure 4.9 compares the real data taken from the ESM and simulation results.

Simulation result, before (left) and after post processing (right).

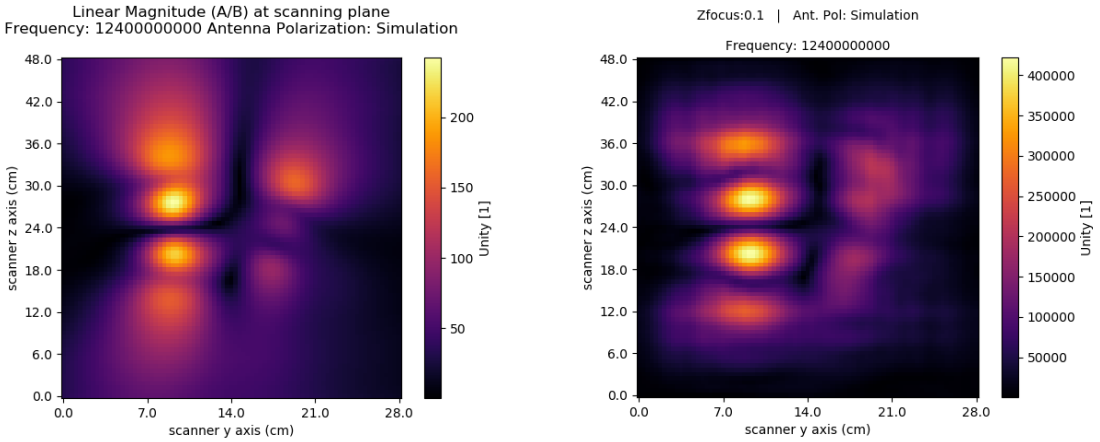

Figure 4.8: Simulation results at 10 mm spacing from source, both pre- and post-processing.

Comparison of post processing results. Left is measured, right is simulated.

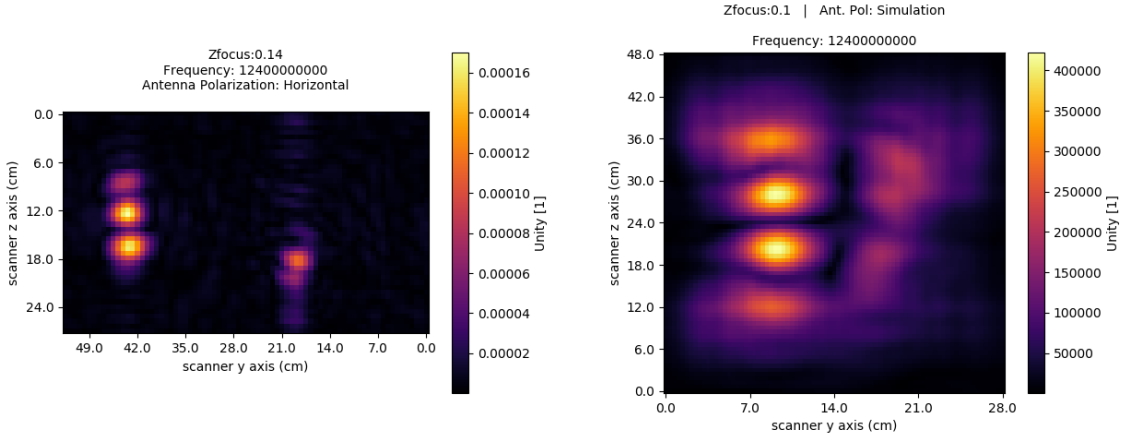

Figure 4.9: Comparison of measurement vs simulation results after post-processing.

The simulation result after post processing shown in figure 4.9 indicates a much better agreement in geometry compared to the simulation results shown in figure 4.6. This result reinforces the theory that the energy is approaching a uniform phase front as the distance between the radiator and sampled electric field plane increases. In the physical scan, antennas (dual ridge horn waveguides) were used as the measurement antennas. In simulation, there are no antenna effects applied to the electric field at all.

4-3 CONCLUSION

This discrepancy in results will be further examined in Chapter 5. The role of directivity of the physical ESM measurement antennas will be examined with a new set of simulation environments. Specifically, physical results will be compared to simulated open face and horn waveguides as they are parametrically swept across the scan plane in a similar manner to the physical ESM device.

CHAPTER 5

ANTENNA EFFECTS ANALYSIS THROUGH SIMULATION

This chapter presents an antenna effects theory that explores the idea that the type of antenna used in an emission source microscopy has an effect on scan performance. To test this theory, this chapter documents the development and presents results of a simulation environment that is operationally equivalent to a physical ESM system.

In Chapter 4, a comparison analysis was performed between measurement and simulation scenarios that studied a two dimensional cross-section of the electric field at a fixed distance away from radiating sources. A discrepancy in results between these scenarios was observed, and indicates that simply observing the normal component of the electric field is not a realistic approximation to the operation of an emission source microscope.

This discrepancy leads to the theory that the observation antennas used in ESM systems play a role recovering information of the radiation source by integrating power received over the beamwidth of the antenna. This property yields more dynamic range (i.e. range of minimum to maximum values) across the scanned region, which under marginal signal-to-noise ratio conditions would yield more focused back propagation results.

The first part of this chapter will cover the construction and characterization of horn waveguide antennas used as the ESM scanning antennas. Next, a simulation environment designed to match

the electromagnetic conditions of a physical ESM measurement is detailed. Lastly, results of the simulated ESM environment are discussed and presented.

5-1 SIMULATION ANTENNAS

The following tables outline the horn waveguide design properties using WR62 waveguide geometry. WR62 was chosen because its cutoff frequency of 9.488 GHz was comfortably above the target frequency of 12.4 GHz [10]. This waveguide is used in both the 60 mm long horn antenna, as well as the open face waveguide, as shown respectively in figures 5.3 and 5.4.

Table 5.1: 60 mm Horn waveguide properties.

60 mm Horn Waveguide Antenna	
Property	Parameter
Horn Length	60 mm
Taper angle	15 degrees
Wall thickness	2 mm
Feed waveguide dimensions	WR62 (15.79 x 7.89) mm
Waveguide aperture dimensions	47.95 x 40.05 mm (Width x Height)

Table 5.2: Open Face waveguide properties.

Open Face Waveguide Antenna	
Property	Parameter
Waveguide length	5.5 mm
Taper angle	N/A
Wall thickness	2 mm
Waveguide dimensions	WR62 (15.79 x 7.89) mm

As shown in figure 5.1, the reflection characteristics of the 60 mm horn waveguide antenna confirms excellent operation at 12.4 GHz with a VSWR of 1.074:1, reflecting only 0.036% of the incident energy back to the source. Unlike the 60mm waveguide antenna, the antenna efficiency of the open face waveguide (figure 5.2) was 1.626:1, reflecting 23.7% of the incident energy.

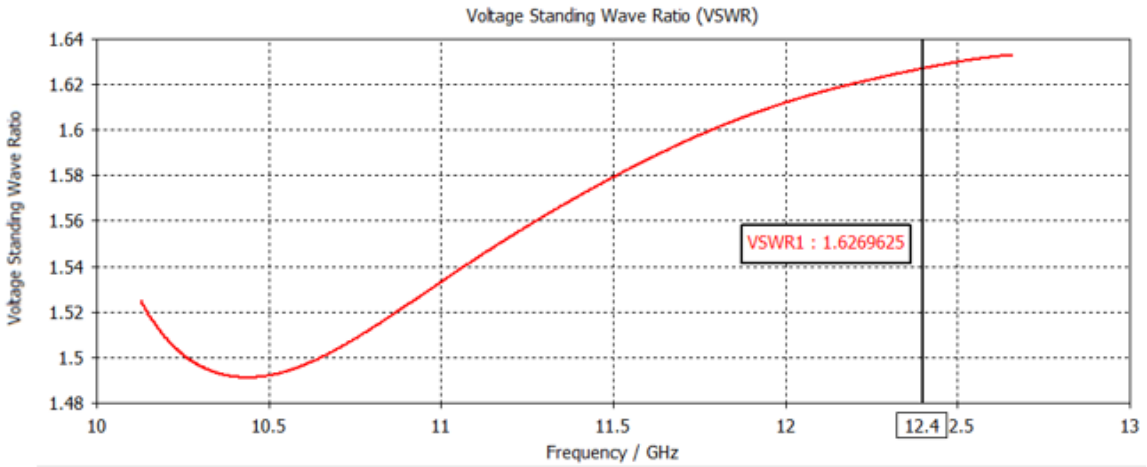

Figure 5.1: VSWR of the open face waveguide antenna.

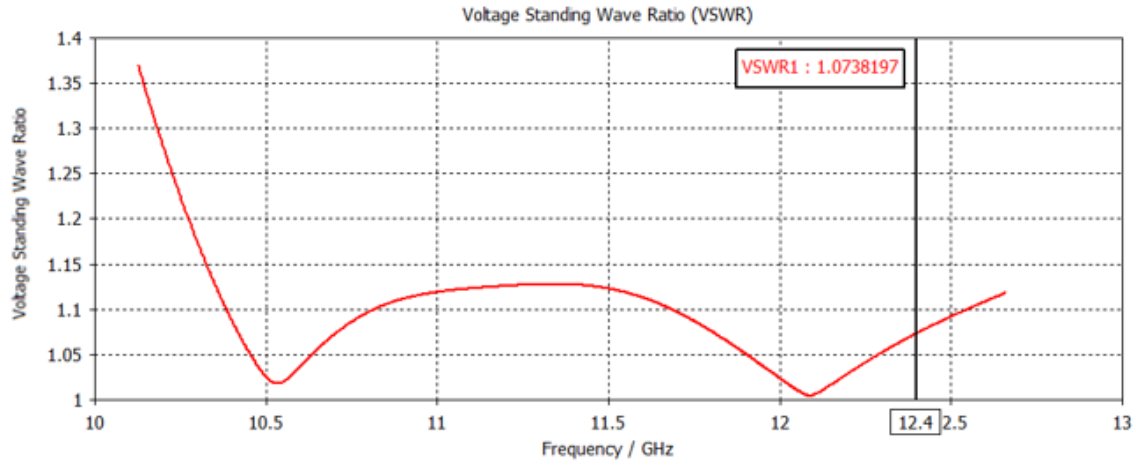

Figure 5.2: VSWR of 60 mm horn waveguide antenna.

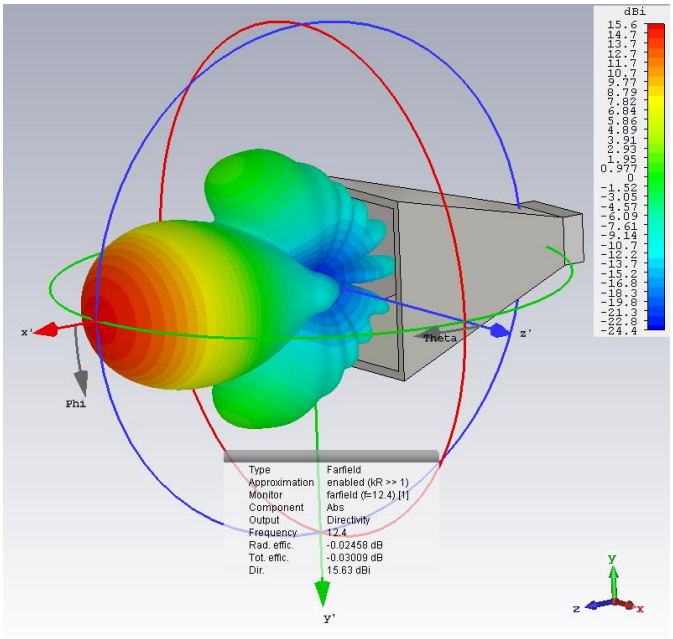

Figure 5.3: 60 mm Horn Waveguide antenna 3D lobe pattern at 12.4 GHz.

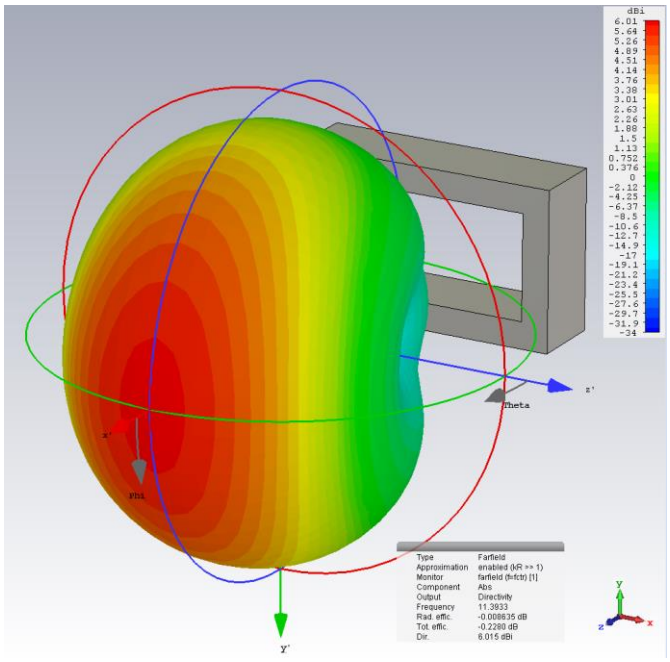

Figure 5.4: 3D gain plot of open face waveguide antenna.

Figures 5.3 and 5.4 illustrate the gain and beamwidth lobing patterns of each waveguide antenna. From figure 5.3, it can be observed that the gain of the 60 mm horn waveguide antenna has a maximum value of 15.6 dB over isotropic. In total, three side lobe pairs and one back lobe exist for this antenna. First (and most significant) side lobes are seen at ± 55° in the ϕ (phi) direction, with a gain of 1.2 dB over isotropic, which is 14.4 dB down from the maximum gain observed at boresight. The corresponding first nulls on either side of the main lobe exist at ± 35° in ϕ with a minima of -6 dB over isotropic. In contrast, the open face waveguide antenna has a lower maximum gain of 6.01 dB over isotropic. The only secondary lobe present is a back lobe, with a gain value of -2.59 dB over isotropic, 8.6 dB down from maximum gain. The corresponding nulls exist on either side of the main lobe at ± 140° in ϕ with a minima of -10 dB over isotropic.

A clear representation of maximum gain can be seen in figures 5 and 6. These graphs represent is a two dimensional cross section of gain across all ϕ with a fixed value θ (theta). In both figures, the θ axis is set to the value in the direction of maximal gain, which is parallel to the direction of wave propagation for both antennas presented ($\theta = 90°$).

Also evident when comparing figures 5.5 and 5.6 is the difference in directivity in the form of half power bandwidth (HPBW). HPBW is defined as the angle, in direction ϕ, in which the magnitude of the radiation pattern decreases by 50%, or -3 dB from its peak on boresight [11]. Comparing the directivity of the 60 mm horn and the open face waveguide antennas, there is a large difference in directivity. The 60 mm horn antenna has a HPBW (half power bandwidth) of 27.6 degrees, and the open face waveguide has a HPBW of 143.4 degrees. This large difference in beamwidth values will result in an evident difference between ESM results. The larger beamwidth of the open face waveguide will result in less variation across the scan plane. Less

variation across the scan plane means that more accurate measurements are necessary to resolve changes between samples.

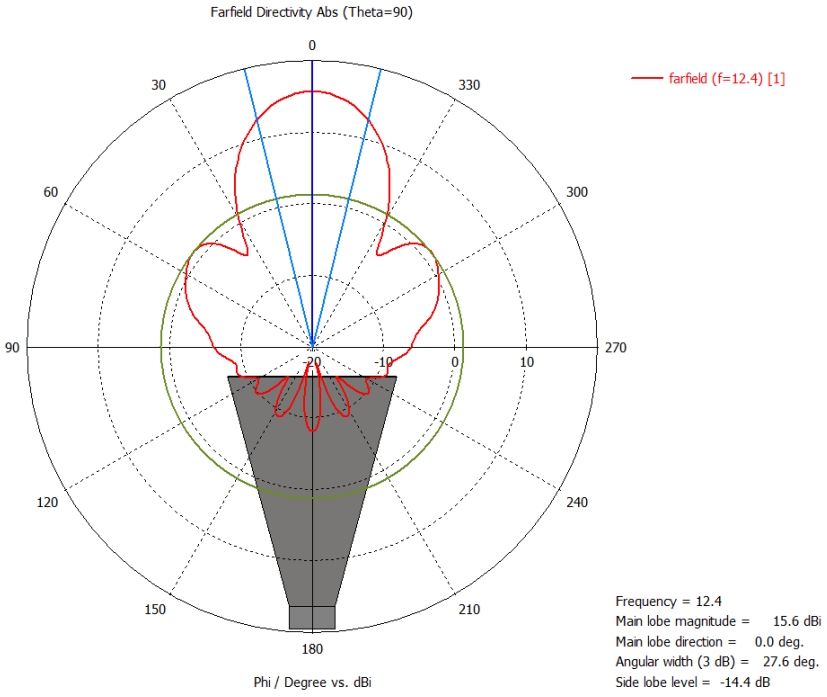

Figure 5.5: 2D directivity plot of 60 mm Horn Waveguide antenna.

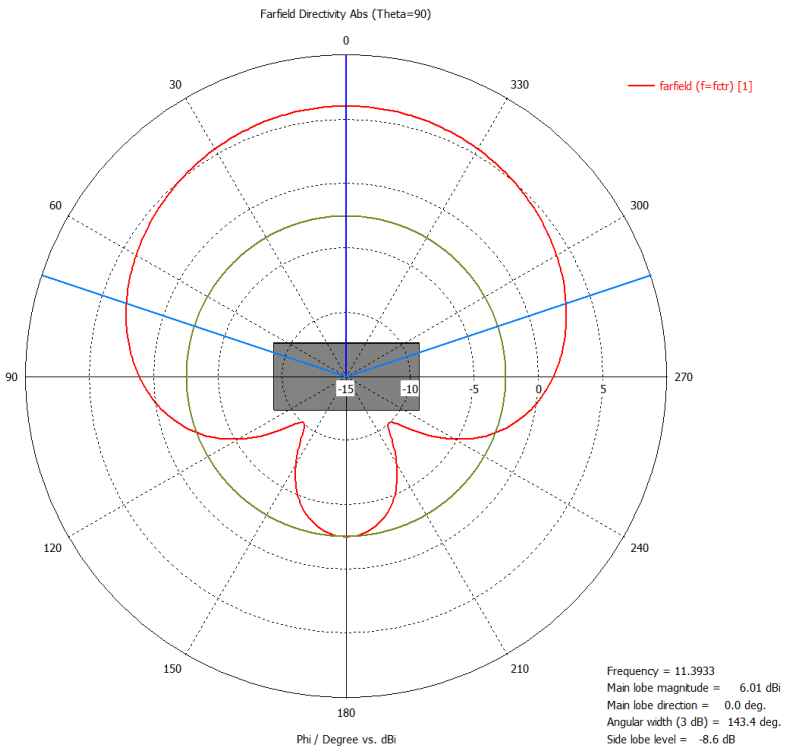

Figure 5.6: 2D far field directivity plot of open face waveguide antenna.

It is important that both antennas will be receiving far field energy from the source. All simulations used in this paper had a distance from source radiator to field probe detector of 25 cm. However, the phase center of each antenna differed. The Fraunhofer far field definition is used to verify both antennas were in the far field [15], is given by:

$$R > \frac{2D^2}{\lambda} \tag{5.1}$$

Where R is the distance from the radiator, D is the largest dimension of the antenna, and λ is the wavelength of interest. The open waveguide antenna had a maximum aperture dimension of 15.8

mm in the x direction, and the distance from the radiating source to the open aperture was R = 24.45 cm. Computing (5.1) for the open waveguide antenna yields an $R > 2.063$ cm, therefore, the open face waveguide antenna is comfortably measuring the far field of the radiated sources used in this paper.

The aperture of the 60 mm horn waveguide antenna is located 18.45 cm from the radiating body. Computing (5.1) for this antenna yields $R > 19.01$ cm. This antenna is within 1 cm of the far field by the Fraunhofer definition.

5-2 SIMULATION ENVIRONMENT

To accomplish the goal mimicking the operation of a physical emission source microscope, a method of discretely sampling the electric field with an antenna structure within a simulation environment was needed. This was accomplished by using parametric sweep solver function within CST. A two-stage parametric sweep was devised to sweep through all parameters using the following resolution equation:

$$X\ Axis\ Samples = \frac{Ant\_posX_{min} \leq Ant\_posX \leq Ant\_posX_{max}}{sample\ resolution} \tag{5.1}$$

$$Y\ Axis\ Samples = \frac{Ant\_posY_{min} \leq Ant\_posY \leq Ant\_posY_{max}}{sample\ resolution} \tag{5.2}$$

Where Ant_posX and Ant_posY, units millimeters, represent the two dimensional position of the center of the receiving antenna, which was either the 60mm horn waveguide or open face waveguide, depending on the simulation scenario. The $sample\ resolution$, in millimeters, is the delta between each sample. The result: $< X, Y >\ Axis\ Samples$ represents the number of samples in each axis. Resolution terms defined in figure 5.7 graphically describe the resolution terms used in this section of the chapter.

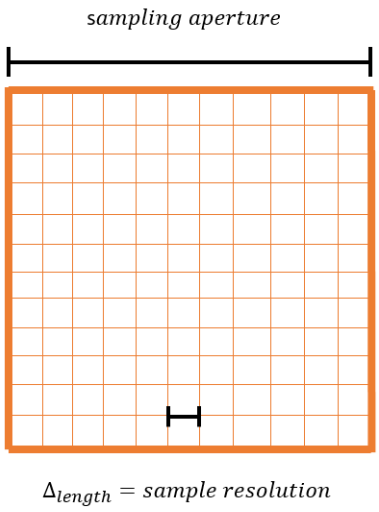

Figure 5.7: definition of sampling aperture and sample resolution.

A graphical representation of both the 60mm horn waveguide and open face waveguide scenarios are depicted in figures 5.8 and 5.9, respectively. A top-down view of the 2D measurement window is illustrated in figure 5.10, where the orange box defines the bounding region of the sampling space for the ESM measurements.

60 mm Horn Waveguide Scenario

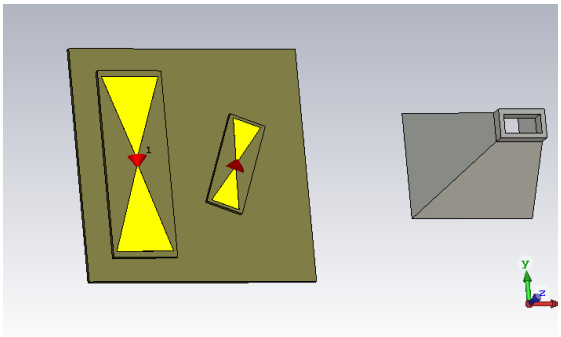

Figure 5.8: Typical simulation environment with 60 mm horn waveguide antenna.

Open Face Waveguide Scenario

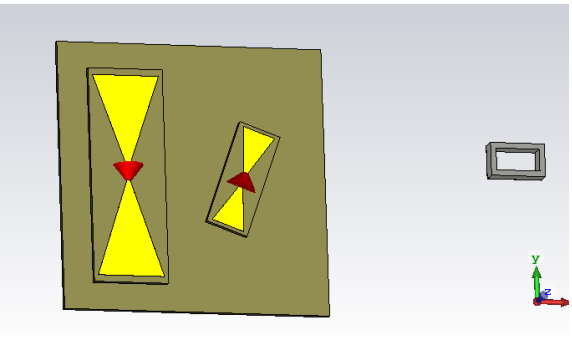

Figure 5.9: Typical simulation environment with the open face waveguide antenna.

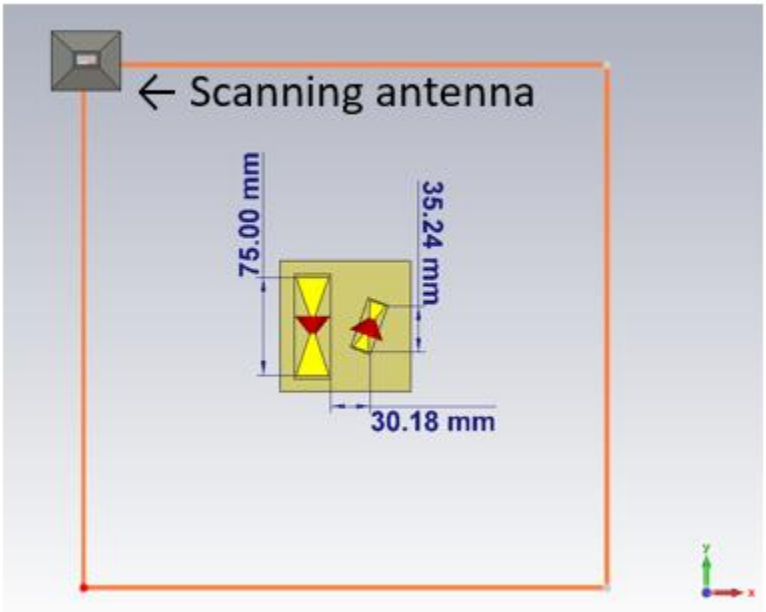

Figure 5.10: Scan region (orange) and radiating element dimensions & spacing.

The electric field was sampled using a field probe situated in the center of the antenna aperture as illustrated in figure 5.11.

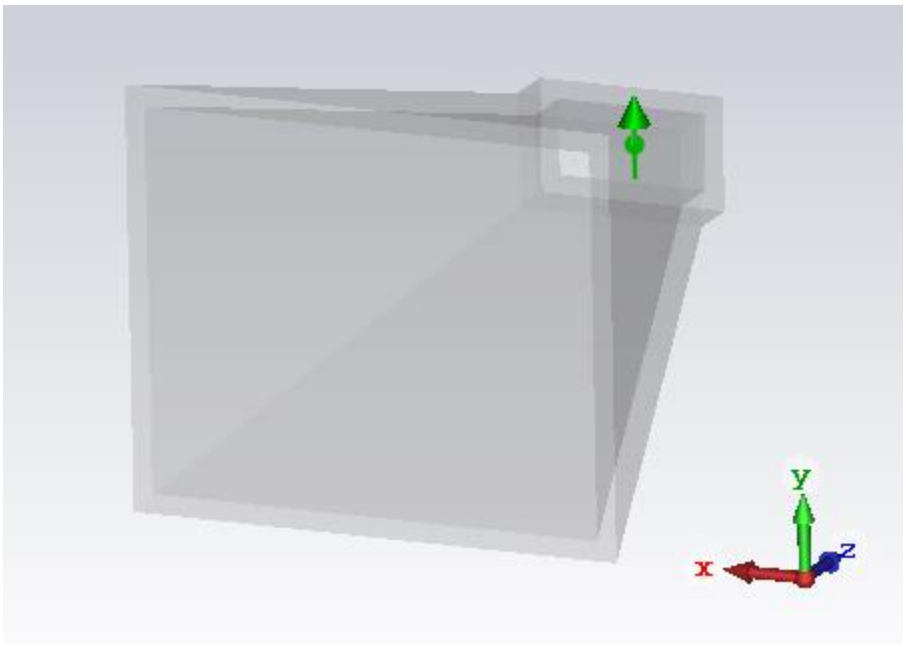

Figure 5.11: Location of field probe, located in center-back of waveguide.

The field probe location was constant at 250 mm away from the radiating bowtie antennas in both simulation environments, and in all simulation scenarios, the field probe moves along with the antenna structure in each parametric sweep. At each of these discrete samples, the receiver antenna moves its position as defined by the $sample\ resolution$ (figure 5.7). Since the geometry has changed, the mesh needs to be recalculated, meaning each sample is an independent simulation.

A large number of simulations need to be run to generate a single emission source microscopy result. For example, an 11 x 11 measurement would be 121 simulations alone, in the final results

of this chapter, a *sample resolution* of 41 x 41 simulations resulted in a total sample count of 1681 simulations. Because of the vast number of simulations to run, and the limited parallelization resources available, great care was taken to optimize the simulation time and disk space required for each simulation.

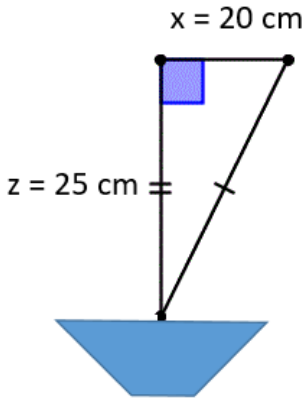

Figure 5.12: geometry from receiver antenna. z is the distance from source to scan planes, x is the width of the scan plane.

The illumination area of both antennas has been calculated using (1) the HPBW results shown in figures 5.5 and 5.6, (2) the distance from the source to scan plane and the width of the scanned plane is defined in Figure 5.12. The illumination area was calculated using Angle-Side-Angle triangle geometry and the law of sines to compute the length of the illumination area along the scan plane. The open face waveguide, with a HPBW of 143.4 degrees, has a very large illumination area of 147.86 cm. This means that the total scan area of 40 cm is within the HPBW of the open face waveguide at every point in its scan range. This beamwidth stability across the

scan range leads to greater coherency in phase across the scan region, leading to better results as seen in section 5-6.

In contrast, the horn waveguide antenna has a HPBW of just 27.6 degrees. The distance from the source to the scan planes remained the same in the computations, but as a result the total illuminated area of the antenna is only 9.36 cm. At best, the horn waveguide antenna will cover ¼ of the total scan area at any given sample position, with the edge samples of the scan plane covering less than that. This results in lesser coherence in phase across the scan region, leading to a lower measurement dynamic range and non-optimal scan results.

5-3 IMAGE RESOLUTION

Next, two approaches were used to ascertain (1) the sampling resolution required by the emission source microscope system, and (2) the expected feature resolution after source reconstruction. The approach to determining the required sampling resolution uses synthetic aperture radar theory [16] to construct an image spatial resolution required to resolve the source. The second approach uses the equations given in prior art [1] to find the maximum feature resolution expected after back propagation has been applied.

To validate the sampling resolution, we define the two way phase shift relative to scan range, as:

$$\phi = kR = k\sqrt{x^2 + z^2} \tag{5.3}$$

$$\phi \approx k\left(z + \frac{x^2}{2z}\right) \tag{5.4}$$

where k is the free space wavenumber $\frac{2\pi}{\lambda}$, and R is range to target, and x and z are defined in figure 5.12. Next, Ω is defined as:

$$\Omega = \frac{\partial \phi}{\partial x} = \frac{kx}{z} = \frac{2\pi x}{\lambda z} \tag{5.5}$$

which is applied to the definition of spatial frequency:

$$F = \frac{1}{2\pi}\Omega = \frac{x}{\lambda z} \; ; \; F_{max} = \frac{20}{25\lambda} = \frac{4}{5\lambda} \tag{5.6}$$

and the spatial bandwidth is defined as:

$$B_s = 2F_{max} \tag{5.7}$$

finally, the image spatial resolution can be found as the inverse of the spatial bandwidth:

$$\rho_x = \frac{1}{B_s} = \frac{1}{2F_{max}} \tag{5.8}$$

computing (5.3) – (5.8) yields an image spatial resolution of 15.125 mm. This result confirms that a sample resolution of 10 mm is sufficiently oversampling the sources radiation.

In addition to the analytical analysis of the sampling resolution, a simulation study was performed to validate that a sample resolution of 10 mm was sufficient to resolve features in the received energy by comparing 5 mm and 10 mm sample resolution across a one-dimensional spatial cross section of the sampling plane. The results from this study are shown in figure 5.13:

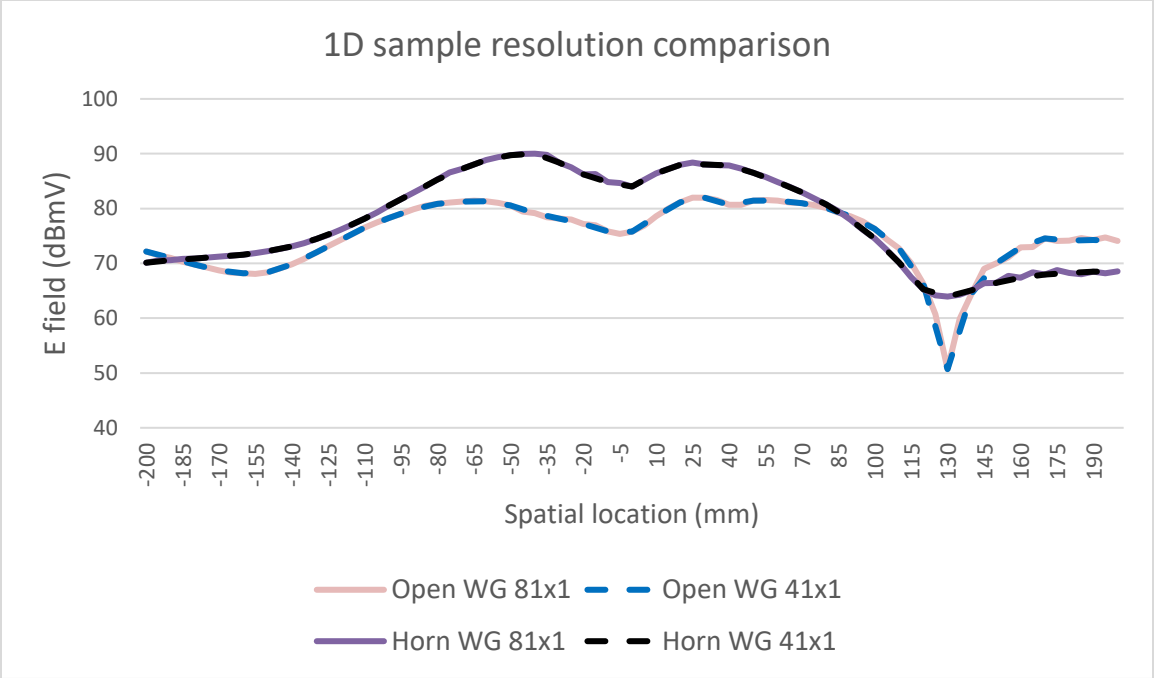

Figure 5.13: 1D cross section comparing sample resolution for each antenna.

Figure 5.13 confirms that a sample resolution of 10 mm is capable of resolving the features in the received energy across this 1D scan plane. In addition, this graph gives insight into antenna performance across the scan plane. The horn waveguide antenna has increased received energy within its illumination region in front of the radiating body, but has similar or reduced performance when located out of its HPBW illumination area, compared to the open face waveguide.

Prior art [1] suggests that the maximum resolution of the reconstructed image can be found using equations 2.10 and 2.11. From equation 2.11, the numerical aperture of a sampling plane is calculated by finding the angle θ based on the range of the sampling plane (x axis in figure 5.12)

and the distance from the source to the observation planes (z axis in figure 5.12). The angle is found to be $\theta = 38.66°$. Vacuum is the propagation media used in these simulations, so the refractive index is 1. With this data, the numerical aperture can be computed and the reconstruction image resolution can be found using the wavelength of the frequency of interest λ = 24.2 mm (f = 12.4 GHz). The resultant feature resolution of the measurement system is 19.23 mm. However, based on the results of a 41 x 41 simulation shown in section 5-6, this is a conservative estimate in a simulation environment, as the results obtained had a feature resolution of approximately 5 mm.

5-4 SIMULATION OPTIMIZATION

The finite element time domain solver in CST Microwave Studio was chosen because the simulation model required measurements within the open air volume across the relatively large area compared to minima geometry, which would be the thickness of the copper elements of the bowtie antennas. General optimization was achieved by reducing the "far-from-model" (CST terminology) to three mesh cells per wavelength. This feature saved a significant amount of computation time as the model contains considerable open space in the bounding region. To maintain accuracy, the "near-to-model" cell count was set to 10 cells per wavelength. In addition, a cell within the thickness of the bowtie antenna elements was forced to help ensure an accurate representation of the antennas radiation properties.

Finally, an accuracy analysis was performed in appendix IV. The conclusion of this accuracy analysis is that a decay threshold of -20 dB provided sufficient dynamic range to meet the

convergence criteria for these simulations. However, if a lower amplitude was chosen or external noise injected into the simulation scenarios, a more accurate (thus greater) decay threshold may be required, which would increase the computation time by an exponential factor.

Both bowtie antennas were driven with cross-fed discrete ports, these ports are represented in Figure 5.9 as the red arrows at the center of the bowtie elements. To further save simulation time, the excitations were merged so that the simulation scenario solves the radiation from both antennas as a unit instead of the default method of solving excitations individually and generating results as a network. The ports were driven with identical amplitudes and with zero time shift, ensuring that any resultant phase offset seen in results were a result of the geometry of the antennas and not the excitation source.

A frequency span of 11 to 15 GHz was used in the simulation environment. This frequency span enables visual representation of dynamic range across frequency, and is a component in the accuracy analysis performed and detailed in Appendix IV.

As a result of these optimizations, each simulation took between 2.5 and 3.5 minutes to solve. The solution time variance is a result of the changing boundary region as the receiving antenna is swept through space.

5-5 RESULT POST-PROCESSING

The result of each scan produced a one-dimensional array of electric field data across frequency. A Visual Basic script, documented in Appendix IIB, was developed to iterate through the results of each parametric simulation and save the one-dimensional data into individual text files. Then, a

Python script, documented in Appendix IIC, was written to perform the following functions; import all text files, sort by simulation number, extract zero-dimensional (i.e. single value) electric field data at the frequency of interest. Finally, the resultant data is sorted into a two-dimensional array of size $(xDim, yDim) = (X\ Axis\ samples, Y\ Axis\ Samples)$. Where $X\ Axis\ Samples$ and $Y\ Axis\ Samples$ are the result of equations 5.1 and 5.2. This entire process was repeated for both real and imaginary datasets. Next, an arbitrary simulation is chosen to represent the normalization factor for the back propagation equation, and the real and imaginary data is saved into corresponding .csv files.

The ESM post processing program is then executed. It reads the .csv files and applies the back propagation algorithm (Section 2-2) on the dataset and generates electric field radiation plots for an array of focal points.

5-6 ANTENNA RESULTS

The benchmark for an optimal scan would resolve the 30 degree offset of the right hand (smaller) bowtie antenna, this benchmark was chosen because it closely resembles results seen in in physical ESM measurements.

The first measurement scenario used 121 (11 x 11) simulations, with a sampling aperture of 100 x 100 mm, and a sample resolution of 10 mm. From these results (not included), it was apparent that a wider spatial resolution was required to capture enough energy to resolve the image. Subsequent measurements were made with an increased sampling aperture of 200 x 200 mm (4x larger area) and maintained the sampling resolution of 10 mm. As a result, the number of simulations increased to 1,681 (41 x 41). These results, shown in figures 5.11 and 5.12,

demonstrated that the source plane could be resolved by both open and horn waveguide receiver antennas.

60 mm Horn Waveguide Scenario

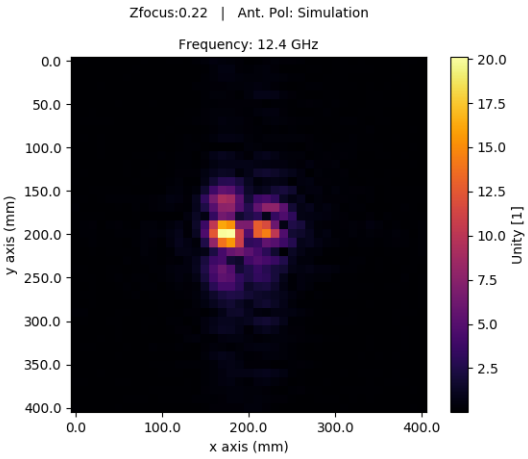

Figure 5.14: Focused electric field across scan plane using horn waveguide as receiving antenna.

Open Face Waveguide Scenario

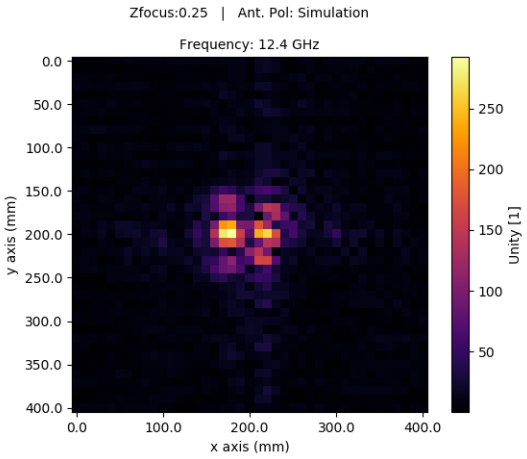

Figure 5.15: Focused electric field across scan plane using open face waveguide as receiving antenna.

The spatial spectral results in figures 5.14 and 5.15 show similar magnitudes across the scanning plane. However, the open face waveguide case (figure 5.15) shows increased contrast of the radiating elements above and below the center feed point of the bowtie antennas, as seen as the points of highest magnitude in the images. The colors of each simulation result are defined using the max-to-min ratio, defined as $\Delta = \left(Max_{E_z} - Min_{E_z}\right)$. While the color distribution of both images look similar, observe the unit scale of the color bars to the right of the images. The scale of the open face waveguide antenna is 10 dB greater in magnitude than that of the horn

waveguide across the scan region. Even with the reduced performance of the horn waveguide case, the graphical spectral results look similar to one another because they were taken under ideal, simulated conditions. In a real ESM measurement, noise, reflections, and frequency drift would cause measurement error that would require a robust signal-to-noise (SNR) ratio to still resolve the geometry of the radiating bodies.

Figure 5.16 is created from the open face waveguide results shown in figure 5.15 to create a graphical overlay of the source radiation from its matching geometry.

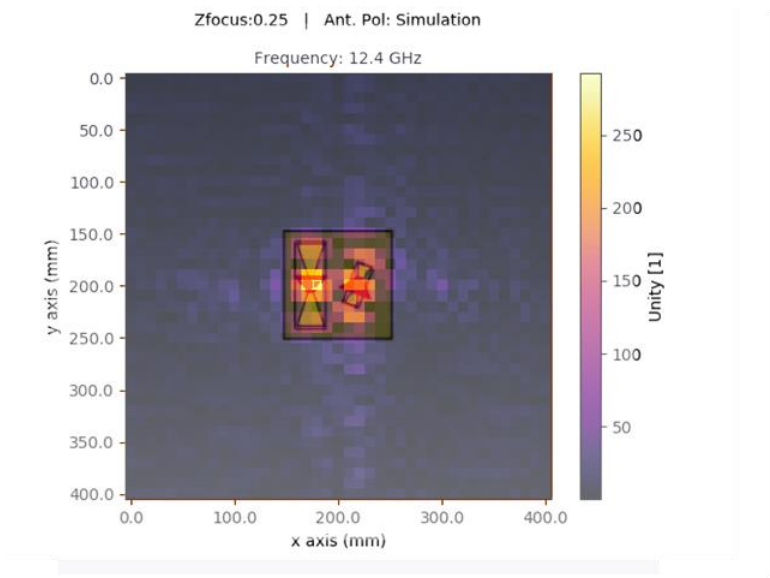

Figure 5.16: Radiating geometry overlaid on electric field radiation plot

Figure 5.16 confirms excellent registration of the radiating elements of the bowtie antennas. The outline of the simulation bounding region is aligned to the edge of the radiation plot graph to ensure 1:1 spatial alignment for both graphs.

Videos provided in [13] (60 mm horn waveguide), and [14] (open face waveguide) give visual representations of the change in electric field across the scan plane as the focal length increases with time. These videos portray the 'focusing effect' that is analogous to optical focusing.

5-7 CONCLUSION

Results from this series of experiments provide the methodology of building a simulation environment to emulate a physical emission source microscope. The result of these simulations shown in section 5-6 reinforce the theory that the type of receiver antenna used in the ESM has an effect on the dynamic range of the resultant electric field radiation graph. Specifically, the narrow beamwidth on the horn waveguide antenna rejected energy critical to provide coherent results into the Fourier transforms as required in the back propagation algorithm.

To maintain simplicity and reduce computation time, these simulations do not include any noise sources. It is suggested that future work add uniform noise to the equation to experiment with the tolerance of dynamic range of the emission source microscope system, particularly when using the a directive horn as a receiving antenna.

CHAPTER 6

CONCLUSION

This thesis has presented the development of emission source microscopy systems in both physical and simulation environments. The efforts applied to this work began as a means to enhance measurement capabilities at IBM, but grew into an academic pursuit by exploring the validation of ESM results though simulation. Additionally, this thesis aims to extend prior art by exploring the use of more directive antennas for ESM reception. The result of comparing an open face waveguide antenna to a more directive horn antenna concluded that the horn reduces the signal-to-noise ratio by a factor of ten. This decreased measurement sensitivity seen with the directive (horn waveguide) antenna means that marginal radiators relative to an open face waveguide would be within the noise of the system using a horn waveguide. This reduction in signal-to-noise ratio with the horn antenna is a result of the phase incoherency in samples across the scan region due to its narrow beamwidth, as detailed by the lack of illumination shown in section 5-1.

6.1 FUTURE WORK: VALIDATION OF RESULTS WITH PHYSICAL ESM

The results of the ESM in simulation concluded that the type of receive antenna used has an effect on measurement sensitivity. Therefore, it is the author's recommendation that a direct comparison with an open face waveguide and a horn antenna on an ESM system in a real-world measurement environment be performed in order to validate conclusions. It was the author's intent to perform and present these measurements as part of this work, and construction of an ESM at Oklahoma State University has been completed. However, irregular results obtained by the VNA's tuned receiver mode led to the focus of efforts on the simulation environment instead. In order to enable future researchers, complete source code of the OSU ESM data collection system has been documented in Appendix IIA. Additional information about the components of OSU's ESM and challenges encountered can be found in chapter 3.

6.2 FUTURE WORK: EXPANSION OF ESM IN SIMULATION

The simulated emission source microscope assumes a free space environment, free of any unintentional radiators or reflecting objects. A study exploring the lower bound of signal-to-noise necessary for the ESM to resolve geometric structures could be performed by expanding the complexity of the simulation environment through the introduction of random noise sources and reflective structures. The results of such as study would provide valuable insight into the operational conditions necessary for a successful scan. Additionally, different radiation structures could be introduced to the physical and simulation ESM environments to study the resultant representation of those structures.

In chapter 3, a method of compensating for frequency drift of a continuous wave (CW) radiation source was presented. However, recent work [12] has been conducted to use a stochastic model to

resolve sources with time variant emissions. The ability of an ESM to resolve time variant sources would be greatly beneficial in practice, since the majority of radiation sources are not ideal CW emissions, but are instead random and periodic.

REFERENCES

[1] P. Maheshwari, H. Kajbaf, V. V. Khilkevich and D. Pommerenke, "Emission Source Microscopy Technique for EMI Source Localization," in IEEE Transactions on Electromagnetic Compatibility, vol. 58, no. 3, pp. 729-737, June 2016.

[2] J. Hanfling, G. Borgiotti and L. Kaplan, "The backward transform of the near field for reconstruction of aperture fields," 1979 Antennas and Propagation Society International Symposium, Seattle, WA, USA, 1979, pp. 764-767.

[3] C. A. Balanis, Antenna Theory - Analysis and Design. New Jersey: John Wiley & Sons, 2016. pp. 996-1000.

[4] W. M. Brown and L. J. Porcello, "An introduction to synthetic-aperture radar," in IEEE Spectrum, vol. 6, no. 9, pp. 52-62, Sept. 1969.

[5] C Wolff, "Radar Basics - Synthetic Aperture Radar", Radartutorial.eu, 2018. [Online]. Available: http://www.radartutorial.eu/20.airborne/ab07.en.html. [Accessed: 01- Feb- 2018].

[6] “User’s Guide HP 8753E Network Analyzer”, Hewlett-Packard Company, 1999.[Online].Available: http://literature.cdn.keysight.com/litweb/pdf/08753-90367.pdf. [Accessed: 3-Mar-2018] pp.2.83-2.91

[7] T. Williams, The AARL Antenna Book For Radio Communications. Newington CT: AARL, 2016. pp. (23-10) – (23-14)

[8] Saghlatoon, Hossein & Sydänheimo, Lauri & Ukkonen, Leena & Tentzeris, Manos. (2014). Optimization of Inkjet Printing of Patch Antennas on Low-Cost Fibrous Substrates. Antennas and Wireless Propagation Letters, IEEE. 13. 915-918. 10.1109/LAWP.2014.2322572.

[9] P. Bevelacqua, “Bow Tie Antennas”, antenna-theory.com, 2013. [Online]. Available: http://www.antenna-theory.com/antennas/wideband/bowtie.php. [Accessed: 18-Feb-2018].

[10] WR62 | WG18 | R140 – Rectangular Waveguide Size, everythingrf.com, 2015. [Online]. Available: https://www.everythingrf.com/tech-resources/waveguides-sizes/wr62. [Accessed: 25-Mar-2018].

[11] P. Bevelacqua, “Beamwidths and Sidelobe Levels”, antenna-theory.com, 2013. [Online]. Available: http://www.antenna-theory.com/basics/radPatDefs.php. [Accessed: 25-Mar-2018].

[12] G. Gradoni et al., "Propagation methods for stochastic field emissions and source reconstruction," 2017 International Symposium on Electromagnetic Compatibility - EMC EUROPE, Angers, 2017, pp. 1-6.doi: 10.1109/EMCEurope.2017.8094693

[13] J. Dixon, “Simulated ESM - 60mm Horn Waveguide Receiver”. (2018). [video]. Internet: https://youtu.be/NMbpbT8fnCg.

[14] J. Dixon, “Simulated ESM – Open Face Waveguide Receiver”. (2018). [video]. Internet: https://youtu.be/iejsY8Tp8Hw.

[15] P. Bevelacqua, “Field Regions”, antenna-theory.com, 2013. [Online]. Available: http://www.antenna-theory.com/basics/fieldRegions.php. [Accessed: 04-Apr-2018].

[16] N. Levanon, Radar Principles. Tel-Aviv University: John Wiley & Sons, 1988. pp. 273 - 282.

APPENDICES

APPENDIX I

I-A: IBM ESM PARTS LIST

Table A-3: IBM ESM parts list

Device and Materials List		
Quantity	Device	Description
1	Keysight N5224 PNA	Vector network analyzer, used to collect data
2	ETS Lindgren 3115	Receiver antennas, 1-18 GHz dual ridge horns
1	CNC-Kontroller C1421	Stepper motor driver and computer interface
1	Plotter cart	Wooden cart that held all measurement equipment
1	MetroLogic 3 axis plotter	Three axis plotter assembly, mounted to cart
1	Hewlett Packard 8449B	Pre-amplifier for antenna A
1	AML113P2001	Pre-amplifier for antenna B
4	Male – Male N type cables	RF cables to connect antennas to pre-amplifiers to the VNA
1	Agilent E8257D	Signal generator – up to 20 GHz

I-B: OSU ESM PARTS LIST

Table A-4: OSU ESM parts list

Device and Materials List		
Quantity	Device	Description
1	Agilent 8722ES VNA	Vector network analyzer, used to collect data
2	Pasternak PE9855/PE9858 Standard gain horns or open face WR-90 waveguide – N type adapters	Receiver antennas
1	Arduino Uno – gShield v5b	Stepper motor driver and computer interface
2	Open Builds linear actuator	Two linear actuators mounted to each other to make the X and Y axis
2	Male – Male N type cables	RF cables to connect antennas to pre-amplifiers to the VNA
1	Agilent E8257D	Signal generator – up to 20 GHz

APPENDIX II

II-A: OSU ESM DATA COLLECTOR PROGRAM

Language: Python

```
#Data Collection program for Emission Source Microscopy with both VNA
and XYZ positioner
#Written February 6, 2018, Jacob Dixon

import serial
import visa
import time
import numpy
import os
import time
import datetime
from slackclient import SlackClient
import heatmapGenerator_OSU

#important user variables
stepSize = 1 #defines resolution of the plotter steps for sampling
movementResolution = 10 #defines the step resolution of the scanner
when setting coordinates
recalcTriggerCount = 1 #number of datasets to collect when
recalculating max frequency
#targetFrequency = 12400000000 #Hz
targetFrequency = 5000000000
vnaAddress = 'GPIB0::16::INSTR'
slackBotEnable = False #enables slackbot notification services

#init resources
resourceList = []
vnaResource = ''
measurementTitle = ''
rm = visa.ResourceManager()
frequencyList = [] #frequency list array
startPosition = [] #array for x, y, z position to start a scan
endPosition = [] #array for x, y, z stop position


###### VNA related functions
##############################################
def listAndSelectVNA():#list resources and select VNA
    resourceList = rm.list_resources()
    print('\n' + str(resourceList))
    print('\n\n\nAvailable Resources: ("example inst 0, example inst 1,
example inst 2") = (0, 1, 2 ...)')
    vnaResourceSelect = resourceList.index(vnaAddress)
    vnaResource = resourceList[vnaResourceSelect] #select resource
    print('Selecting Resource #' + str(vnaResource))
    vna = rm.open_resource(vnaResource)
```

```
    vna.timeout = 120000 #2 minute timeout
    return vna

def calculateFrequencies(): #generates discrete list of frequencies the
VNA samples
    i = 1
    startFreq = float(vna.query('SENS:FREQ:STAR?')) #ask VNA what is
the sweep start frequency
    stopFreq = float(vna.query('SENS:FREQ:STOP?')) #ask VNA what is the
sweep stop frequency
    numPoints = float(vna.query('SENS:SWE:POIN?')) #ask VNA # points
    stepSize = float(vna.query('SENS:SWE:STEP?')) #ask VNA frequency
step size
    while i < int(numPoints):
        if i == 1:
            frequencyList.append(startFreq)
            incremFreq = startFreq
        incremFreq = incremFreq + stepSize
        frequencyList.append(incremFreq)
        i = i + 1
    return frequencyList

def configureVNA(): #Recall Reg 8 and set output format
    vna.write('OPC?;RECAREG05;')
    vna.write('OPC?;CHAN1;')
    vna.write('OPC?;AB;')
    vna.write('OPC?;CHAN2;')
    vna.write('OPC?;MEASB;')
    vna.write('OPC?;DUACON;')
    vna.write('OPC?;FORM4;')
    vna.write('OPC?;SING;')


def triggerVNA(): #trigger VNA
    vna.write('OPC?;SING;')
    waitForClear()

def collectRawData(): #collects raw data
    dataArr = numpy.zeros((numPoints,2), dtype = numpy.float)
    data = vna.query("OUTPFORM;") #collect VNA data (formatted)
    data = data.splitlines()
    data = numpy.asarray(data)
    for y in range(0,data.shape[0]): #for each column of mag data
        dataArr[y,:] = numpy.fromstring(data[y], dtype=numpy.float,
sep=',')
    return dataArr[:,0]

def collectReal(channelSelect):
    if channelSelect == 'AB':
        vna.write('OPC?;CHAN1;')
        vna.write('OPC?;REAL;')
        vna.write('OPC?;WAIT;')
        realData = collectRawData()
        waitForClear()
        return realData
    elif channelSelect == 'B': #select channel 2
        vna.write('OPC?;CHAN2;')
```

```
        vna.write('OPC?;REAL;')
        vna.write('OPC?;WAIT;')
        realData = collectRawData()
        waitForClear()
        return realData
    else:
        print('Error: improper measurement selection passed in
collectReal function.')
        exit()

def collectImaginary(channelSelect):
    if channelSelect == 'AB':
        vna.write('OPC?;CHAN1')
        vna.write('OPC?;IMAG')
        vna.write('OPC?;WAIT')
        imagData = collectRawData()
        waitForClear()
        return imagData
    elif channelSelect == 'B': #select channel 2
        vna.write('OPC?;CHAN2')
        vna.write('OPC?;IMAG')
        vna.write('OPC?;WAIT')
        imagData = collectRawData()
        waitForClear()
        return imagData
    else:
        print('Error: improper measurement selection passed in
collectImaginary function.')
        exit()

def getNumPoints():
    numPoints = float(vna.query('POIN?')) #ask VNA # points
    return numPoints

def waitForClear(): #query's the VNA status register until busy flag
clear.
    opComplete = False
    vna.write('WAIT;')
    while True: #VNA returns 1 when not busy
        opStatus = vna.query('OPC?') #operation complete query
        if '1' in opStatus:
            break

def recalculateMaxFrequency(start):

    freqRecalcData = numpy.zeros((numPoints,recalcTriggerCount), dtype
= numpy.float) #data for point recalculation
    vna.write('OPC?;CHAN2')
    waitForClear()
    vna.write('OPC?;LINM')  # sets data format to linear magnitude
    waitForClear()
    if start == True:
        vna.write('OPC?;SPAN20000000') #set frequency span to 20 MHz
for initial scan
    else:
        vna.write('OPC?;SPAN5000') #subsequent searches have a 5 kHz
span
```

```
    vna.write('OPC?;MARK1') #enable the marker readout on the display

    print('Calculating frequency peak...')
    for x in range(0,(recalcTriggerCount)): #perform max hold
measurement
        vna.write('OPC?;SING') #single sweep
        vna.write('OPC?;WAIT')
        freqRecalcData[:,x] = collectRawData()
        print(freqRecalcData[:,x])
        vna.write('OPC?;AUTO')  # auto-scales the display
    numRows, numColumns = freqRecalcData.shape
    freqRecalcData = numpy.amax(freqRecalcData, axis=1)
    maxFreqPoint = numpy.argmax(freqRecalcData)

    vna.write('OPC?;MARKCOUP') #couple markers between channel 1 & 2
    vna.write('OPC?;CHAN1') #switch to channel 1
    vna.write('OPC?;MARKBUCK ' + str(maxFreqPoint)) #set marker to
freq. with max RX
    vna.write('OPC?;MARKCENT') #set center of stimulus to maximum value
    vna.write('OPC?;SPAN0') #set stimulus range to 0 Hz
    vna.write('OPC?;CHAN2') #switch to channel 2
    vna.write('OPC?;MARKBUCK ' + str(maxFreqPoint)) #set marker to
freq. with max RX
    vna.write('OPC?;MARKCENT') #set center of stimulus to maximum value
    vna.write('OPC?;MARKUNCO') #decouple marker between channels
    vna.write('OPC?;MARKOFF') # turns off all markers


##########################################################################
####


###### XYZ plotter related functions
#####################################
def serialReturnHandler():#checks ACK of CNC controller, displays error
message (if any)
    while True:
        SerialPort.flush() #clear input buffer
        SerialPort.write("?".encode('ascii'))
        time.sleep(.1) #wait for controller to reply
        bytesIn = SerialPort.inWaiting()
        reply = ((SerialPort.read(bytesIn)).decode(encoding='ascii'))
        if 'Idle' in reply:
            break

def positionCalculator(val, bits): #converts plotters hex coordinate
position into moter steps with respect to origin
    if (val & (1 << (bits - 1))) != 0:  #two's complement manipulation
        val = val - (1 << bits)
    return val

def getCurrentPosition(): #this function prints the plotters
coordinates
    SerialPort.flush() #clear input buffer before checking position
    SerialPort.write("?".encode('ascii'))
    time.sleep(.5) #wait for controller to reply
    bytesIn = SerialPort.inWaiting()
```

```
    position = ((SerialPort.read(bytesIn)).decode(encoding='ascii'))
    print(position)
    #parse position data
    position = position.split('|')
    position = position[1]
    position = position.split(':')
    position = position[1]
    print(str(position))
    xPosition,yPosition,zPosition = position.split(',')
    position = (float(xPosition), float(yPosition), float(zPosition))
#position data packaged in tuple form

    return position

def initSerialPort(): #setup serial port
    SerialPort = serial.Serial('COM4',115200, timeout=60000)
#Baud=115200,8 bits, No parity, 1 stop
    print('Opening serial port...')
    time.sleep(5)
    return SerialPort
def initPlotter():
    print('Homing plotter...')
    SerialPort.write("$H\n".encode('ascii')) #Home the plotter
    time.sleep(.5)
    #wait for homing cycle to finish
    while True:
        bytesIn = SerialPort.inWaiting()
        serRead = ((SerialPort.read(bytesIn)).decode(encoding='ascii'))
        if 'ok' in serRead:
            break
    SerialPort.write("G92 X0 Y0\n".encode('ascii')) #Verify home
position is set to X0, Y0
    serialReturnHandler()

def setPlotterPosition():
    global movementResolution
    distanceInput = 10 #default motor steps per movement command
    position = getCurrentPosition() #get starting position
    currentXpos = position[0] + 1 # +1 accounts for homing offset
    currentYpos = position[1] + 1
    print('\n\n Use keys (WASD) to control the X and Y axis on the
plotter')
    print('[c] to change the step size \n[p] to print current position
\n[q] to quit\n[m] to move to position \n[x] to set current position
\n~~Press [Enter] after each command.~~')
    userInput = 0 #User input init
    userInputArr = [0,1]
    while(True):
        userInput = input()
        if userInput == 'c':
            print('Enter desired movement distance (0 - 321)\nNote:
default movement is 10 units.')
            movementResolution = int(input())
            print('Distance input now set to: ' +
str(movementResolution))
        elif userInput == 'w': # character for up => -Y on plotter
            currentYpos = currentYpos - movementResolution
```

```
            yDown = 'G0 Y' + str(currentYpos) + '\n'
            SerialPort.write(yDown.encode('ascii'))
            serialReturnHandler()
        elif userInput == 's': # character for down => +Y on plotter
            currentYpos = currentYpos + movementResolution
            yUp = 'G0 Y' + str(currentYpos) + '\n'
            SerialPort.write(yUp.encode('ascii'))
            serialReturnHandler()
        elif userInput == 'd': # character for right => X- on plotter
            currentXpos = currentXpos - movementResolution
            xRight = 'G0 X' + str(currentXpos) + '\n'
            SerialPort.write(xRight.encode('ascii'))
            serialReturnHandler()
        elif userInput == 'a': # character for left => X+ on plotter
            currentXpos = currentXpos + movementResolution
            xLeft = 'G0 X' + str(currentXpos) + '\n'
            SerialPort.write(xLeft.encode('ascii'))
            serialReturnHandler()
        elif userInput == 'm': #Move to user specified location
            print('Input x')
            userInputArr[0] = input()
            print('Input y')
            userInputArr[1] = input()
            SerialPort.write(('G0 X' + str(userInputArr[0]) + ' Y' +
str(userInputArr[1])).encode('ascii'))
            currentXpos,currentYpos = userInputArr #update current
position variables
            serialReturnHandler()
        elif userInput == 'p': #print out current position to user
            position = getCurrentPosition()
            print('The current X position is ' + str(position[0]))
            print('The current Y position is ' + str(position[1]))
        elif userInput == 'x': #set position
            position = getCurrentPosition()
            return position #send current position back from function
        elif userInput == 'q':
            break #break out of while loop
        else:
            print('Invalid character entered. (WASD) keys control Z and
Y axis\n[c] changes the step size \n[p] prints the current position
\n[q] quits the manual plotter control \n[x] sets the current
position')
#########################################################################
####



###### Program functions
################################################
def initSlackBot(slackBotEnable):
    if slackBotEnable == True:
        try:
            slackToken = 'KEY_REDACTED' #slack API token
            slackBot = SlackClient(slackToken)
            return slackBot
        except IOError: #can't communicate with slack, not a mission
critical problem
```

```
            print('Warning: cannot connect to slack webhook services.')
            slackBotEnable = False #disable for duration of test
            pass

def sendSlackMessage(slackBot,message):
    if slackBotEnable == True:
        try:

slackBot.api_call('chat.postMessage',parse='full',channel='#emc-
robot',text=str(message),username='EMC
Robot',icon_url='https://avatars.slack-edge.com/2017-03-
01/148937281671_b4cbaa0a5ff77ec67626_48.png') #note, bug workaround,
as_user=True did not work when written.
        except IOError:
            print('Warning: cannot post message to channel.')
            pass
def uploadSlackPhoto(slackBot,fileName):
    if slackBotEnable == True:
        try:
            slackBot.api_call('files.upload',channels='#emc-
robot',filename=str(fileName),file=open(str(fileName),'rb'))
        except IOError:
            print('Warning: cannot upload image.')
            pass
def calculateSamplingPoints():
    numRows = int((abs(endPosition[0] - startPosition[0])) / stepSize)
    numColumns = int((abs((endPosition[1] - startPosition[1]))) /
stepSize)
    samplingPoints = (numRows * numColumns)
    print('samplingPointNumbers')
    print(numRows)
    print(numColumns)
    print(samplingPoints)
    return (numRows,numColumns,samplingPoints)

def checkFileStatus():
    try:
        fqFile = open("frequencies.csv", "w+")
        magFile = open("magnitudeData.csv", "w+")
        phaseFile = open("phaseData.csv", "w+")
        fqFile.close()
        magFile.close()
        phaseFile.close()

    except IOError:
        print('Error: One or more output files are open. The program
will finish once they are closed.')
        time.sleep(2)
        checkFileStatus()

def createDirectory():
    folderName = time.strftime("%Y%m%d-%H%M%S")
    directory = os.getcwd()
    path = (directory + '\\' + folderName)
    if not os.path.exists(path):
        os.makedirs(path)
    return path
```

```
def writeConfigFile(): #config file resides in output folder path,
contains info requred to re-run heatmapGenerator.py
    configFile = open(path + '\\configFile.txt', 'w')
    configFile.write(str(startPosition[0]) + ',' +
str(startPosition[1]) + ',' + str(startPosition[2]) + '\n')
    configFile.write(str(endPosition[0]) + ',' + str(endPosition[1]) +
',' + str(endPosition[2]) + '\n')
    configFile.write(str(positionData[0]) + ',' + str(positionData[1])
+ ',' + str(positionData[2]) + '\n')
    configFile.write(str(stepSize) + '\n')
    configFile.write(str(targetFrequency) + '\n')
    configFile.write(str(distanceToDUT) + '\n')
    configFile.write(str(antennaPolarization) + '\n')
    configFile.close()

def writeSettingsFile(): #settings file resides in program directory
and holds information from last scan
    settingsFile = open('scanSettings.txt', 'w')
    settingsFile.write(str(startPosition[0]) + ',' +
str(startPosition[1]) + ',' + str(startPosition[2]) + '\n')
    settingsFile.write(str(endPosition[0]) + ',' + str(endPosition[1])
+ ',' + str(endPosition[2]) + '\n')
    settingsFile.write(str(stepSize) + '\n')
    settingsFile.write(str(targetFrequency) + '\n')
    settingsFile.write(str(distanceToDUT) + '\n')
    settingsFile.write(str(antennaPolarization) + '\n')
    settingsFile.close()

def getPlotterPositions():
    #ask if user wants to repeat last scan
    #global stepSize, targetFrequency, startPosition, endPosition
    print('Do you want to repeat the last measurement (Y/N)?\nStart and
stop points will be carried over.\nFrequency and step size can be
optionally changed.\n\n')
    userInput = input()
    if userInput == 'Y' or userInput == 'y':
        global stepSize, targetFrequency, distanceToDUT,
antennaPolarization #allow modification of global variable within
function
        try:
            print('Reading from scan settings file...')
            settingsFile = open('scanSettings.txt', 'r')
            startPosition = settingsFile.readline()
            startPosition = list(map(int,startPosition.split(',')))
#separate .csv values, convert into tuple
            endPosition = settingsFile.readline()
            endPosition = list(map(int,endPosition.split(',')))
            stepSize = int(settingsFile.readline())
            targetFrequency = int(settingsFile.readline())
            print('Do you want to change the target frequency (Y/N)?')
            qInput = input()
            if qInput == 'Y' or qInput == 'y':
                print('Type new frequency (in Hz) to be scanned:')
                targetFrequency = int(input())
            elif qInput == 'N' or qInput == 'n':
                pass
```

```
            else:
                print('Invalid entry made, value of previous scan will
be kept.')
            print('Do you want to change the step size (Y/N)?')
            qInput = input()
            if qInput == 'Y' or qInput == 'y':
                print('Input new step size:')
                stepSize = int(input())
                print('New step size is: ' + str(stepSize))
            elif qInput == 'N' or qInput == 'n':
                pass
            else:
                print('Invalid entry made, value of preivous scan will
be kept.')

            distanceToDUT = int(settingsFile.readline()) #read zFocus
distance from antenna A
            antennaPolarization = settingsFile.readline() #read in
antenna polarization


        except IOError:
            print('Error reading from scanSettings.txt, does it exist
in the program directory?')
            print('Continuing with new scan setup.')
            setupQuestions()
            #print('\n\nMove the plotter to the starting point in the
upper right hand corner of the DUT')
            print('\n\nMove the plotter to the starting point (lower
left hand corner of DUT')
            startPosition = setPlotterPosition()
            print(startPosition)
            #print('\n\nMove the plotter to the end point in the lower
left hand corner of the DUT')
            print('\n\nMove the plotter to the end point in teh upper
right hand corner of the DUT')
            endPosition = setPlotterPosition()
            print(endPosition)
            pass
    elif userInput == 'N' or userInput == 'n':
        setupQuestions()
        print('\n\nMove the plotter to the starting point in the upper
right hand corner of the DUT')
        startPosition = setPlotterPosition()
        print(startPosition)
        print('\n\nMove the plotter to the end point in the lower left
hand corner of the DUT')
        endPosition = setPlotterPosition()
        print(endPosition)
    else:
        getPlotterPositions()
    return startPosition, endPosition

def setupQuestions():
    global distanceToDUT, antennaPolarization
    print('Enter the distance (in cm) from the measurement antenna A to
the DUT')
```

```
    distanceToDUT = input()
    print('Enter the reference antennas polarization (Vertical or
Horizontal)')
    antennaPolarization = input()

###### Program begin
########################################################

startTime = time.time()
SerialPort = initSerialPort()
#serialReturnHandler()
initPlotter()
slackBot = initSlackBot(slackBotEnable)
getCurrentPosition()
vna = listAndSelectVNA()
print(vna.query('*IDN?')) #query command handles GPIB write and read
functionality
vna.write('*CLS;*RST') #reset VNA to default state

startPosition, endPosition = getPlotterPositions() #get start and stop
positions from either file (last run) or directly from plotter (new
scan)
print('StartPos: ' + str(startPosition) + ' EndPos: ' +
str(endPosition))
positionData = calculateSamplingPoints()
samplingPoints = positionData[2]
print('SamplingPoints: ' + str(samplingPoints))

if slackBotEnable == True:
    print('Type a title for the scan, when finished, press [ENTER]')
    measurementTitle = input()




time.sleep(1)
vna.write('FORM:DATA ASC,0') #verify the VNA will return ASCII
formatted data



configureVNA() #setup VNA for test
##frequencyList = calculateFrequencies() #generate list of frequencies
numPoints = int(getNumPoints()) #get number of sampled frequencies from
VNA
time.sleep(1)
realDataAB = numpy.zeros((numPoints,samplingPoints), dtype=numpy.float)
#array of magnitude components
imagDataAB = numpy.zeros((numPoints,samplingPoints), dtype=numpy.float)
#array of phase components
realDataB = numpy.zeros((numPoints,samplingPoints), dtype=numpy.float)
#arrays for reference B measurement
imagDataB = numpy.zeros((numPoints,samplingPoints), dtype=numpy.float)

phaseABData = numpy.zeros((numPoints,samplingPoints),
dtype=numpy.float)
```

```
i = 0
recalcCounter = 0
while i < samplingPoints: #main program loop to sample the scanning
range
    #time.sleep(1)
    if i == 0: #do init tasks: Calculate max mag point, find reference
B data
        time.sleep(.5) #allow antenna to settle, mechanically
        recalculateMaxFrequency(True) #calculate peak emissions
frequency prior to starting the scan
        currentPosition = startPosition
        currentPosition = list(currentPosition) #convert tuple to a
list so modifications are supported
        time.sleep(1) #allow vna to catch up with command requests
before beginning data collection
        #collect B data for reference
        triggerVNA()
        realDataB = collectReal('B')
        imagDataB = collectImaginary('B')
        print(realDataB)
        print(imagDataB)
        #format received data
        numpy.asarray(realDataB,order='F') #column major order
representation
        numpy.asarray(imagDataB,order='F')

    #if currentPosition[1] >= endPosition[1]: #we have hit a boundary
in the X axis
        #currentPosition[1] = startPosition[1] #send back for the next
scan line
        #currentPosition[2] = currentPosition[2] + stepSize #send the Y
axis down one line by stepSize

    if (abs(currentPosition[0]) >= abs(endPosition[0])) and i != 0: #we
have hit a boundary in the X axis and not the starting point
        currentPosition[0] = startPosition[0] #send back for the next
scan line
        currentPosition[1] = currentPosition[1] - stepSize #send the Y
axis down one line by stepSize
    nextPos = 'G0 X' + str(currentPosition[0]) + ' Y' +
str(currentPosition[1]) + '\n'
    SerialPort.write(nextPos.encode('ascii')) #send plotter to position
    time.sleep(.25)
    serialReturnHandler()
    print(currentPosition)



    if recalcCounter >= 40:
        recalculateMaxFrequency(False) #subsequent center frequency
recalcs
        recalcCounter = 0
    #trigger & collect VNA data
    triggerVNA()
    realSweepData = collectReal('AB')
    time.sleep(.25)
```

```
    imagSweepData = collectImaginary('AB')

    #format received data
    numpy.asarray(realSweepData,order='F') #column major order
representation
    numpy.asarray(imagSweepData,order='F')
    realDataAB[:,i] = realSweepData
    imagDataAB[:,i] = imagSweepData

    i += 1
    recalcCounter += 1
    #print progress
    print('Sample ' + str(i) + ' of ' + str(samplingPoints) + '. ' +
str(round((i/samplingPoints)*100,2)) + '%')

    currentPosition[0] = currentPosition[0] - stepSize #send the X axis
to the next position by stepSize



time.sleep(.25)
#vna.write('*CLS;*RST') #reset VNA to default state
time.sleep(.25)
vna.close() #close VISA session

path = createDirectory()
writeConfigFile()
writeSettingsFile()

#save the arrays into files
print(realDataAB.shape)
numpy.savetxt(path + "\\realDataAB.csv",realDataAB,delimiter = ',', fmt
= "%s")
numpy.savetxt(path + "\\imagDataAB.csv",imagDataAB,delimiter = ',', fmt
= "%s")

print(realDataB)
print(imagDataB)
numpy.savetxt(path + "\\realDataB.csv",realDataB,delimiter = ',', fmt =
"%s")
numpy.savetxt(path + "\\imagDataB.csv",imagDataB,delimiter = ',', fmt =
"%s")


#Passes xyz positions for start and end points, total steps in x and y
directions (positionData), target frequency, and path to folder
heatmapGenerator_OSU.calculateBackPropagation(startPosition,endPosition
,positionData,stepSize,targetFrequency,distanceToDUT,antennaPolarizatio
n,path)
endTime = time.time()
initSlackBot(slackBotEnable) #initiate slackbot again bacause the
connection may time out
sendSlackMessage(slackBot,'Back Propagation Scanner: Measurement
completed!\nMeasurement Title: ' + measurementTitle + '\nOutput folder:
' + path + '\nTest duration (H:MM:SS): ' +
str(datetime.timedelta(seconds=int(endTime-startTime))))
uploadSlackPhoto(slackBot,path + '\\focusedImage.png')
```

II-B: CST DATA EXPORT PROGRAM

Programming Language: Visual Basic for Applications

```
' Parametric data export for CST tree items
' Jacob Dixon

Sub Main ()
    Dim numSamples As Integer
    numSamples = 6561
    For i = 1 To numSamples
        SelectTreeItem "1D Results\result\Efield [merged]_(" & i & ")"
'change tree item to the data you want to export
        ExportPlotData "C:\Users\YOUR\PATH\HERE\paramExport" & i &
".txt"
        fileNumber = fileNumber + 1
    Next
    MsgBox "Data export complete."
End Sub
```

II-C: DATA POST-PROCESSING FORMATTER

Programming Language: Python

```
#searches all .txt files in directory and selects the particular column
desired.
#Jacob Dixon
#2/8/2018

import glob
import csv
import numpy
import re
import os

#startRow = 49
#endRow = 678
#numDataPoints = endRow - startRow
column = 1
dataRow = 353 # 12.4 GHz
#dataRow = 1003 #15 GHz
numColumns = 81 #number of columns to wrap resultant 1D data into 2D
array
directory = 'output'
numbers = re.compile(r'(\d+)')

def numericalSort(value):
    parts = numbers.split(value)
    parts[1::2] = map(int, parts[1::2])
    return parts

files = sorted(glob.glob('*.txt'), key=numericalSort)
```

```
data = numpy.zeros(len(files), dtype='f')
print(data.shape)
#print(str(files))
fileCounter = 0
for x in range(len(files)):
    currentFile = open(files[x], 'r')
    fileContents = csv.reader(currentFile, delimiter = ' ',
skipinitialspace=True)
    fileContents = list(fileContents)
    #print(str(len(fileContents[0])) + str(len(fileContents[1])))

    data[fileCounter] = fileContents[dataRow - 1][column]
    print(data[fileCounter])
    print(fileCounter)
    fileCounter += 1

numpy.array(data)
data = numpy.reshape(data, (-1,numColumns))

if not os.path.exists(directory):
    os.makedirs(directory)

numpy.savetxt(directory + '\dataFile.csv',data,delimiter = ',')
```

II-D: OSU ESM RADIATION SOURCE PLOTTER PROGRAM

Language: Python

```
#Back Propagation Microscopy Algorithm
#Written February 6, 2018, Jacob Dixon


import matplotlib.pyplot as plot
from multiprocessing import Process
import numpy
from shutil import copyfile




def
calculateBackPropagation(startPosition,endPosition,positionData,stepSiz
e,targetFrequency,distanceToDUT,antennaPolarization,path):
    #arrays to store AB and B mean data
    dataAB = numpy.zeros((positionData[0] * positionData[1], 1),
dtype=float)#check to verify using right tuple values
    dataB = numpy.zeros((positionData[0], positionData[1]),
dtype=float)
    abMatrix = numpy.zeros((positionData[0], positionData[1]),
dtype=float)
    phaseMatrix = numpy.zeros((positionData[0],positionData[1]),
dtype=float)
    magABMatrix = numpy.zeros((positionData[0],positionData[1]),
dtype=float)
```

```
    imageMatrix = numpy.zeros((positionData[0], positionData[1]),
dtype=float)
    magData = numpy.zeros((positionData[0], positionData[1]),
dtype=float)

    #recall stored capture data
    if path == '': #recall files locally
        realDataAB = numpy.genfromtxt('realDataAB.csv', delimiter=',')
        imagDataAB = numpy.genfromtxt('imagDataAB.csv', delimiter=',')
        realDataB = numpy.genfromtxt('realDataB.csv', delimiter=',')
        imagDataB = numpy.genfromtxt('imagDataB.csv', delimiter=',')

    else: #recall files from folder
        realDataAB = numpy.genfromtxt(path + '//realDataAB.csv',
delimiter=',')
        imagDataAB = numpy.genfromtxt(path + '//imagDataAB.csv',
delimiter=',')
        realDataB = numpy.genfromtxt(path + '//realDataB.csv',
delimiter=',')
        imagDataB = numpy.genfromtxt(path + '//imagDataB.csv',
delimiter=',')

    #Get start and stop positions in terms of mm
    startPositionX_mm = startPosition[0] # units of mm. Note:
corresponds to plotter X axis
    endPositionX_mm = endPosition[0]
    print(startPositionX_mm)
    startPositionY_mm = startPosition[1] # units of mm. Note:
corresponds to plotter Y axis
    endPositionY_mm = endPosition[1]
    XX, YY = numpy.meshgrid(numpy.linspace(startPositionX_mm,
endPositionX_mm, positionData[0]),
                            numpy.linspace(startPositionY_mm,
endPositionY_mm, positionData[1]))  # generate meshgrid by given
dimensions
    Exf = numpy.zeros(XX.size)
    x = numpy.unique(XX)
    y = numpy.unique(YY)
    print('Sizes: x: ' + str(x.size) + ' y: ' + str(y.size))
    dx = (1 * 10 ** -3) * (x[1] - x[0])  # Note, must have at least 4
samples. Finds the delta between two points * e^-3 to convert from mm
to meters.
    dy = (1 * 10 ** -3) * (y[1] - y[0])
    print('dx: ' + str(dx) + '\ndy: ' + str(dy))


    Scale = ((2 * numpy.pi / numpy.absolute(dx)) / (x.shape[0]))
    CutX = numpy.floor(((x.shape[0] - 1) / 2) + 1)  # find midpoint of
array
    KlistX = list(range(0, x.shape[0]))
    i = 0
    while i < x.shape[0]:
        if KlistX[i] >= CutX:
            KlistX[i] = KlistX[i] - x.shape[0]
        i += 1
    kx1 = [KlistX[i] * Scale for i in KlistX] # kx1 = K * Scale,
multiplies each element by scaling factor
```

```
    i = 0
    Scale = ((2 * numpy.pi / numpy.absolute(dy)) / (y.shape[0]))
    CutY = numpy.floor(((y.shape[0] - 1) / 2) + 1)  # half of the array
    KlistY = list(range(0,y.shape[0]))
    while i < y.shape[0]:
        if KlistY[i] >= CutY:
            KlistY[i] = KlistY[i] - y.shape[0]
        i += 1
    ky1 = [KlistY[i] * Scale for i in KlistY] # ky1 = K * Scale

    KX, KY = numpy.meshgrid(kx1, ky1, indexing='ij')  # generate mesh
using the scaled K x and y values

    # back propagation definitions
    freq = targetFrequency  # Hz
    c = 3 * (10**8)  # m/s
    #c = 3 * (10**11)  # mm/s
    lam = c / freq  # lambda

    Essx = numpy.zeros((len(kx1),len(ky1)))
    k = (2 * numpy.pi) / lam
    k2 = (k**2)+0j #calculate square of k

    KZinstr = ((k2)-(KX**2)-(KY**2))
    KZ = numpy.sqrt(KZinstr)


    # plot Kz
    plot.figure()
    CS = plot.contour(KX, KY, KZ)
    plot.clabel(CS, inline=1, fontsize=10)
    plot.title('Contour Plot of Kz')
    plot.xlabel('KX')
    plot.ylabel('KY')
    if path == '': #print to current working directory
        plot.savefig('KzContourPlot.png')
    else: #print to sub directory
        plot.savefig(path + '\\KzContourPlot.png')

    #take mean of real, imag
    realDataMean = numpy.mean(realDataAB, axis=0)
    imagDataMean = numpy.mean(imagDataAB, axis=0)
    #convert to mag + phase
    magData = numpy.sqrt(numpy.add(realDataMean**2,imagDataMean**2))
#collected data in magnitude form
    phaseData = numpy.arctan2(imagDataMean,realDataMean) #phase of
collected data


    dataAB = numpy.add(realDataAB,1j*imagDataAB)
    print(dataAB.shape)

    #re-organize AB matrix and phase matrix into scanned grid format
    abMatrix = numpy.reshape(dataAB, (-1,positionData[1]),order='F')
    phaseMatrix = numpy.reshape(phaseData, (-
1,positionData[1]),order='F')
```

```
    magABMatrix = numpy.reshape(magData, (-
1,positionData[1]),order='F')
    print(abMatrix.shape)
    print(phaseMatrix.shape)
    print(magABMatrix.shape)


    ######## plot scan plane phase########
    plot.clf() #clear last plot
    plot.title('Phase at scanning plane (in RAD)\nFrequency: ' +
str(numpy.round(targetFrequency * 1e-9,1)) + ' GHz | Antenna
Polarization: ' + str(antennaPolarization))
    plot.ylabel('y axis (cm)')
    plot.xlabel('x axis (cm)')
    plot.imshow(phaseMatrix.T, origin='lower',
cmap='inferno',interpolation='none') #plot heatmap of phase
    plot.gca().set_aspect('equal')
    #plot.gca().invert_xaxis()
    plot.gca().invert_yaxis()
    scaledTicks = plot.gca().get_xticks()*stepSize*2 #converts motor
steps to cm
    plot.gca().set_xticklabels(numpy.round(scaledTicks,1))
    scaledTicks = plot.gca().get_yticks()*stepSize # #converts motor
steps to cm
    plot.gca().set_yticklabels(numpy.round(scaledTicks,1))
    plot.colorbar(label='Radians')
    if path == '': #print to current working directory
        plot.savefig('ABphasePlotFromRealImag.png')
    else: #print to sub directory
        plot.savefig(path + '\\ABphasePlotFromRealImag.png')

    ###Plot Mag A
    plot.clf() #clear last plot
    plot.title('Linear Magnitude (A/B) at scanning plane\nFrequency: '
+ str(numpy.round(targetFrequency * 1e-9,1)) + ' GHz | Antenna
Polarization: ' + str(antennaPolarization))
    plot.ylabel('y axis (cm)')
    plot.xlabel('x axis (cm)')
    plot.imshow(abs(abMatrix.T), origin='lower',
cmap='inferno',interpolation='none') #plot heatmap of phase
    plot.gca().set_aspect('equal')
    #plot.gca().invert_xaxis()
    plot.gca().invert_yaxis()
    scaledTicks = plot.gca().get_xticks()*stepSize*2 #converts motor
steps to cm
    plot.gca().set_xticklabels(numpy.round(scaledTicks,1))
    scaledTicks = plot.gca().get_yticks()*stepSize #converts motor
steps to cm
    plot.gca().set_yticklabels(numpy.round(scaledTicks,1))
    plot.colorbar(label='Unity [1]')
    if path == '': #print to current working directory
        plot.savefig('aMagPlot.png')
    else: #print to sub directory
        plot.savefig(path + '\\aMagPlot.png')

    #no mean data needed for plot from CST
    dataB = numpy.multiply(realDataB,1j*imagDataB)
```

```
    meanDataB = numpy.absolute(dataB) #calculate the mean values of the
B data set

    Essx =
numpy.absolute(abMatrix)*meanDataB*numpy.exp(1j*numpy.angle(abMatrix))
    Sx = numpy.fft.fft2(Essx)

    #auto focusing algorithm

    #### REDACTED TO ADHERE TO CONFIDENTIALITY AGREEMENT ####

heatmapPlotter(imageMatrix,zFocusArray[i],path,stepSize,targetFrequency
,antennaPolarization)
        i += 1

    #i = int(distanceToDUT) #user selects zFocus scan plane on startup
    i = numpy.argmax(contrastArray) + 1 #set i to be the index of
maximum average "contrast"
    #i = numpy.argmin(contrastArray)  # set i to be the index of min
average "contrast"
    zFocus = zFocusArray[i] #set zFocus value to value of best focus
point at index i-1

    if path == '': # print to current working directory
        copyfile('heatmapWzFocus' + str(zFocus) +
'.png','focusedImage.png') #copy and rename file too send via slack
    else: # print to sub directory
        copyfile(path + '\\heatmapWzFocus' + str(zFocus) + '.png',path
+ '\\focusedImage.png') #copy and rename file too send via slack

def
heatmapPlotter(imageMatrix,zFocus,path,stepSize,targetFrequency,antenna
Polarization):
    #build plot
    plot.clf()
    plot.title('Zfocus:' + str(numpy.round(zFocus,2)) + '   |   Ant.
Pol: ' + str(antennaPolarization) + '\nFrequency: ' +
str(numpy.round(targetFrequency * 1e-9,1)) + ' GHz',fontsize=10)
    plot.ylabel('y axis (cm)')
    plot.xlabel('x axis (cm)')
    #plot.imshow(20*numpy.log10(numpy.absolute(imageMatrix.T)),
origin='upper', cmap='inferno',interpolation='none') #plot heatmap in
log format
    plot.imshow(numpy.absolute(imageMatrix.T), origin='lower',
cmap='inferno',interpolation='none') #plot heatmap in linear format
    plot.gca().set_aspect('equal')
    #plot.gca().invert_xaxis()
    scaledTicks = plot.gca().get_xticks()*stepSize*2
    plot.gca().set_xticklabels(numpy.round(scaledTicks,1))
    scaledTicks = plot.gca().get_yticks()*stepSize
    plot.gca().set_yticklabels(numpy.round(scaledTicks,1))
    plot.gca().invert_yaxis()
    plot.colorbar(label='Unity [1]')
    if path == '': #print to current working directory
        plot.savefig('heatmapWzFocus' +
str("{0:.2f}".format(numpy.round(zFocus,2)) + '.png'))
    else: #print to sub directory
```

```
        plot.savefig(path + '\\heatmapWzFocus' +
str("{0:.2f}".format(numpy.round(zFocus,2)) + '.png'))

#if called directly:
if __name__ == '__main__':
    configFile = open('configFile.txt','r') #open config file in read
mode
    startPosition = configFile.readline() #plotter start position
    startPosition = list(map(int,startPosition.split(',')))
    endPosition = configFile.readline() #plotter stop position
    endPosition = list(map(int,endPosition.split(',')))
    positionData = configFile.readline() #delta of start & stop
    positionData = list(map(int,positionData.split(',')))
    stepSize = int(configFile.readline()) #step size of plotter
    targetFrequency = int(configFile.readline()) #in Hz
    distanceToDUT = int(configFile.readline()) #read in distance (in
cm) to DUT
    antennaPolarization = configFile.readline() #read in antenna
polarization
    path = '' #will save outputs to current working directory if called
directly


calculateBackPropagation(startPosition,endPosition,positionData,stepSiz
e,targetFrequency,distanceToDUT,antennaPolarization,path)
```

APPENDIX III

Technical Notes

Theoretical challenges associated with the ESM were outlined at the end of chapter 2. The following outlines the technical notes associated with a physical ESM device including limitations and solutions to challenges that were observed.

Dynamic range of the scan is dependent on the field variance across the scan plane – The ESM's measurements are relative to a fixed reference point. If the deviation across the scan plane is small, the results are more susceptible to noise affecting the results. This limitation can be combatted by scanning a larger area to collect more "features".

Best scan resolution is $\frac{\lambda}{2}$ – By the Nyquist sampling theorem, this is the best resolution the ESM can achieve. The current version of the data collector relies on the user to calculate the lowest achievable resolution prior to scanning to verify they are not needlessly oversampling. A proposed extension of this work is to implement an auto resolution adjustment by calculating lambda and comparing that result to the plotters stepping resolution.

Requires direct line of sight measurements of DUT – Both scan antennas need to be in direct line of sight of the device under test to help ensure that the direct path is the dominant contributor of energy to the antennas.

Reflective or EM noisy environments can produce unreliable results – A reflective environment will introduce multipath effects which could result in peaks and fades that would otherwise not exist if the scan was conducted in an environmentally 'clean' environment. Satisfactory results have been collected in a typical office environment where the transmitted signal dominated the

noise by at least 40 dB. More care should be given if the signal to noise margins are narrower, such as performing the test in a quiet environment such as an open air test site or anechoic chamber.

Time varying fields will produce inaccurate results - One of the methods implemented to overcome a slowly changing time variant field was to rescan the electromagnetic spectrum and lock onto the maximum emission frequency for sampling. This rescan method was implemented to counteract the ‘dithering’ seen in frequency on the signal generator equipment used in measurements. It should be noted that this only accounts for very minor changes in frequency in what would otherwise be a static frequency continuous wave signal.

APPENDIX IV

ESM In simulation: Accuracy analysis

The following simulation results are all measured at 12.4 GHz. In previous simulation iterations, a smaller scan range of 200 x 200 mm was used. However, the results produced by these simulations did not provide sufficient data to accurately perform back propagation on the resultant data.

In this simulation campaign, the scan range was doubled to 400 x 400mm. This yielded relative electric field results that were more consistent with physical geometry of the antennas.

Figures A-IV.1 and A-IV.2 represent the total system energy over time (in nanoseconds) within the simulation environment. The maximum energy injected to the system is represented by 0 dB. The solver will keep track of system energy losses until it reaches the desired accuracy threshold. The comparison analysis performed in this appendix observed the effects on the simulation result by allowing more computation time to process the excitation and allow the total system energy to decay to 40dB, instead of the original 20dB threshold used in earlier simulations. This increased accuracy came at the cost of a 3 fold increase in simulation computation time.

The result of this analysis concludes that a decay threshold of 20 dB is sufficient to resolve the geometry of the ESM scan under these ideal measurement conditions. In conditions where there is a more marginal dynamic range, a greater simulation accuracy (via lower cutoff threshold for system energy) would yield a greater dynamic range of measurement.

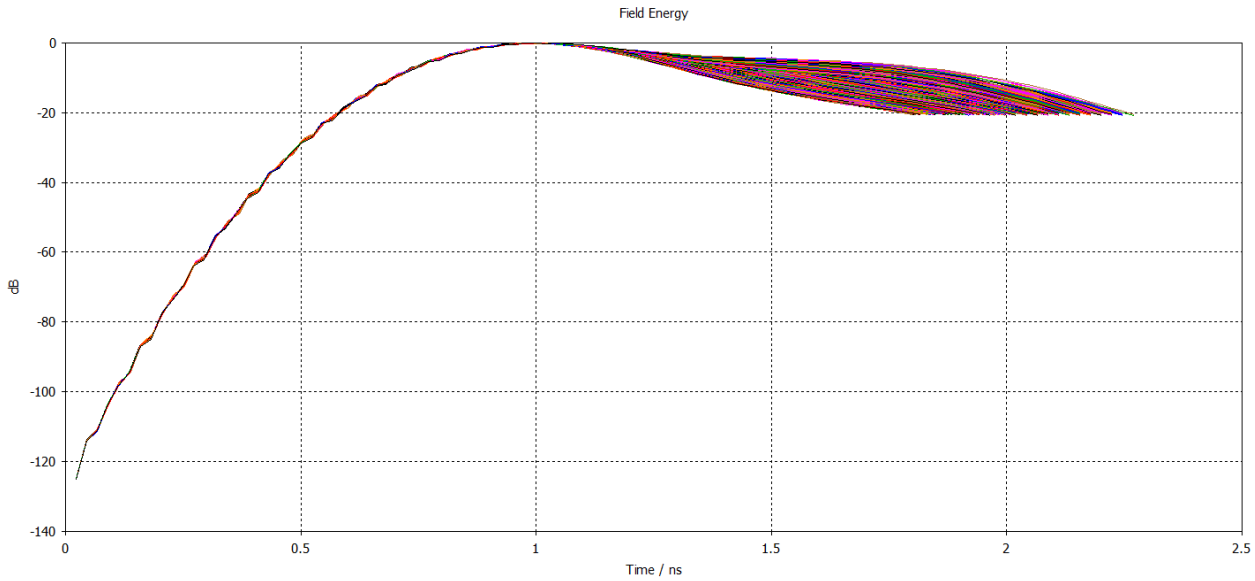

Figure A-6.1: Field energy at the original computation time setting of 20 dB decay threshold.

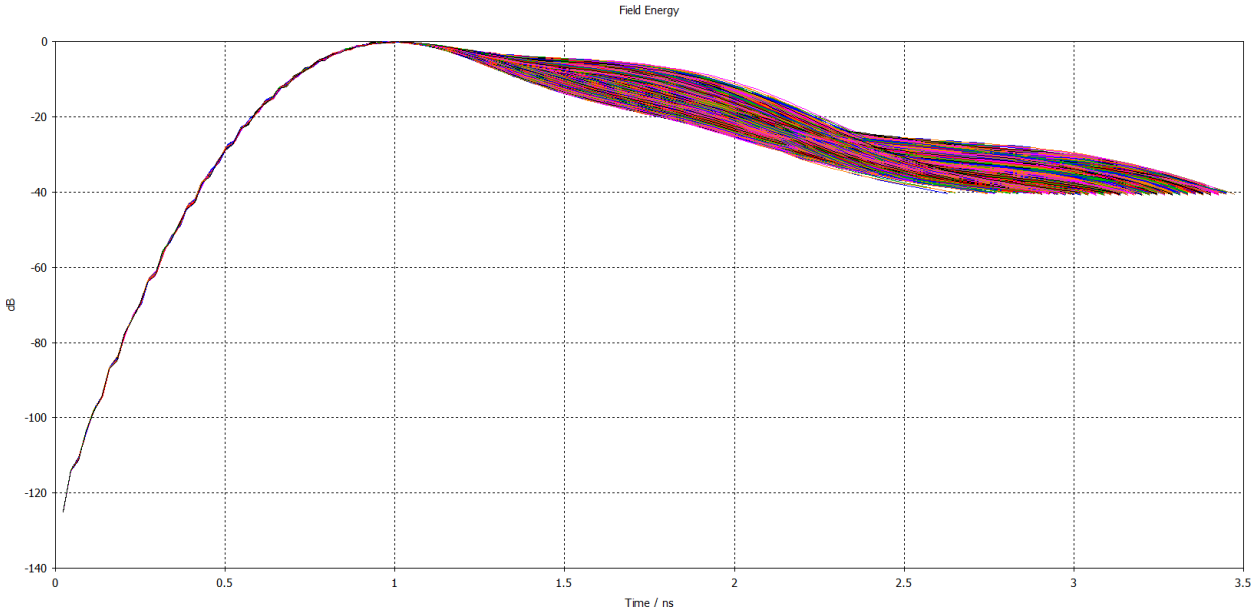

Figure A-6.2: Field energy of an increased computation time setting of 40 dB decay threshold.

The result of this longer simulation time yielded results that were more variant. Figures A-IV.3 and A-IV.4 below represent the real part of the electric field, note that the variance with the longer decay time experienced a 5x increase in variance at 12.4 GHz.

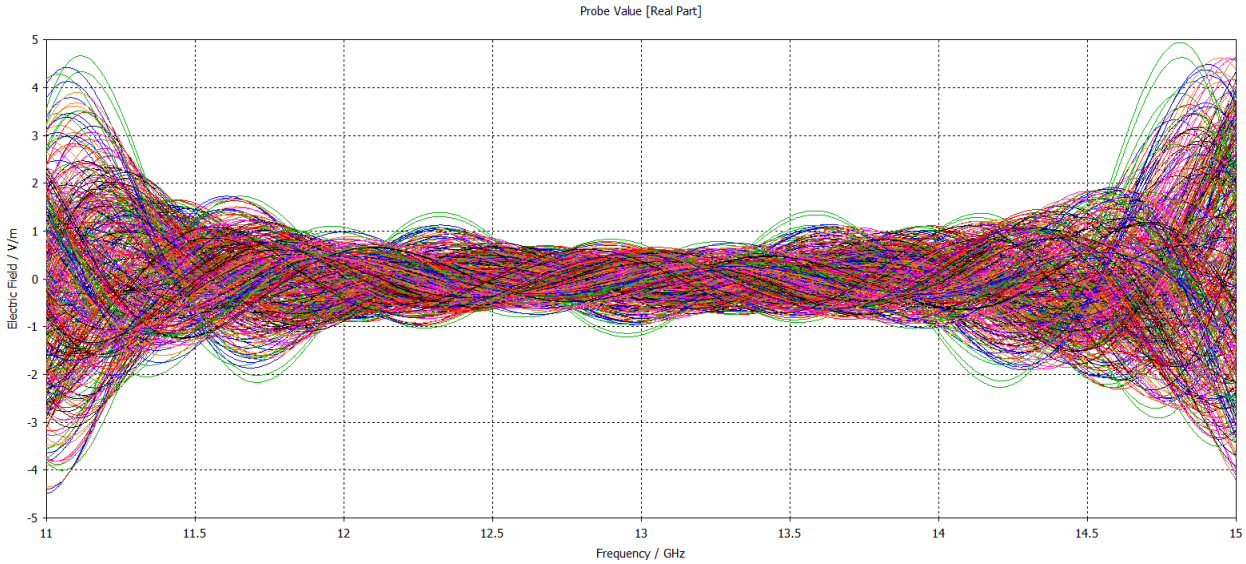

Figure A-6.3: Real part of electric field (V/m) with 20dB decay setting.

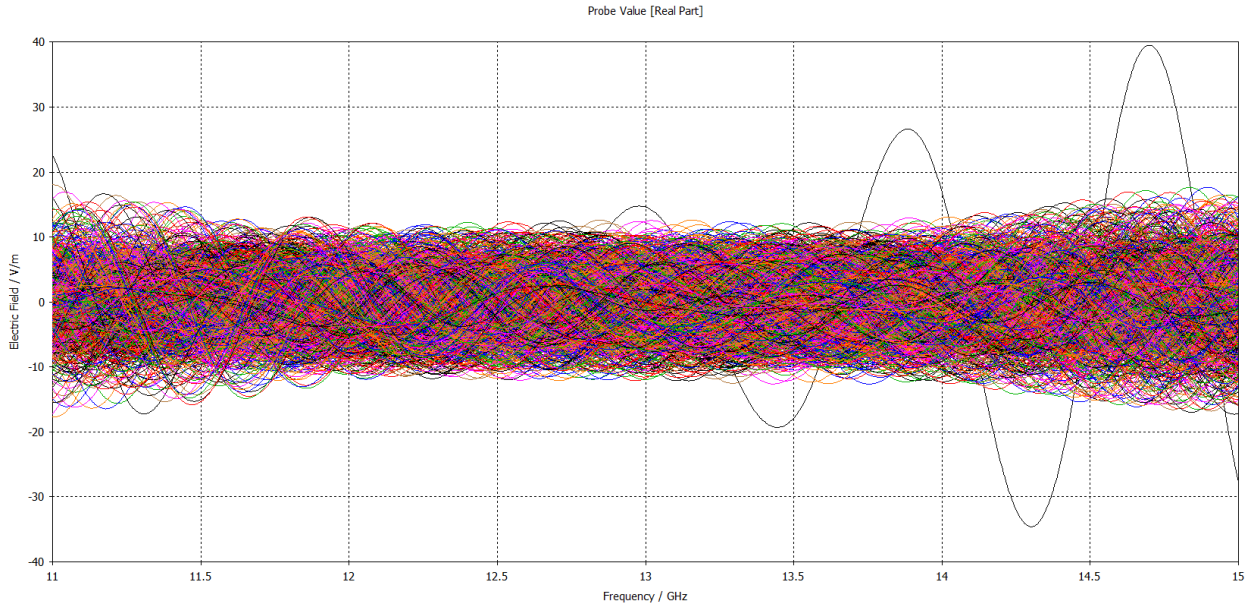

Figure A-6.4: Real part of electric field (V/m) with 40 dB decay setting.

Simulation Scenario 1: 20dB decay

In this scenario, the original decay setting of 20dB is applied to an emission source microscope scan scenario. This scenario, depicted in Figure A-IV.5, collects a 0D data point of the Z

component of the electric field in the center of the horn waveguide antenna. The location of the horn waveguide antenna is parametrically swept in a -200 to 200 mm space in both X and Y axis, yielding a 400x400mm cross sectional scanned area, discretely sampled every 10mm in both X and Y.

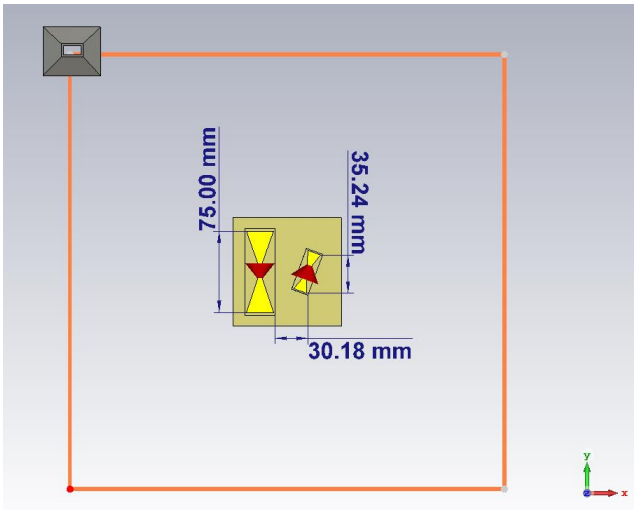

Figure A-6.5: Scan region (orange) and radiating element dimensions & spacing.

The distance from the antennas to the field probe inside the horn waveguide as 25 cm. The most focused image, as judged by $\Delta = \left(Max_{E_z} - Min_{E_z}\right)$ occurs at 25 cm. This represents good agreement between physical distance to target and focal length.

In scenario 2, the only change applied was setting the accuracy settings to a 40 dB threshold. All other variables were held constant from simulation 1. The following figures A-IV.6 through A-IV.9 present a side-by-side comparison of results gained by comparing the 20 dB and 40 dB threshold cases. The delta between max and min is higher in the 40 dB threshold case, but this fact does not translate into a more resolved output image under these simulation conditions. In a more marginal measurement scenario (lower dynamic range of measurement, or in a physical measurement, a low SNR), the greater delta between max and min would be useful.

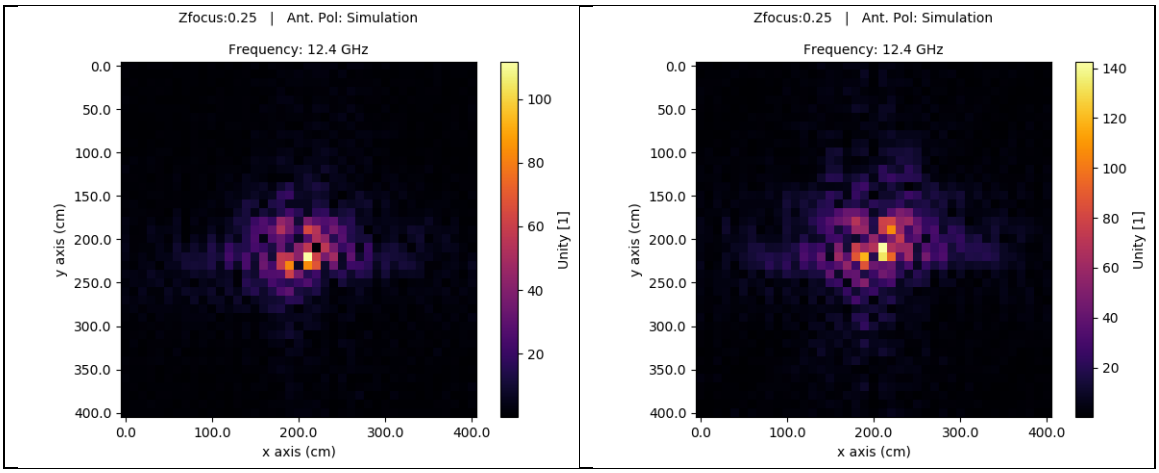

Figure A-6.6: Focal image comparison. Left: 20 dB threshold. Right: 40 dB threshold.

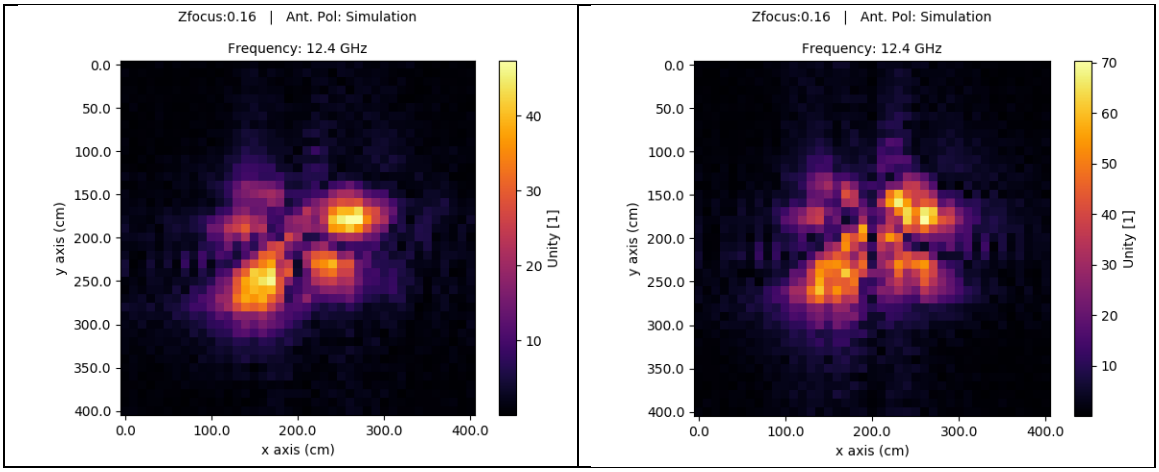

Figure A-6.7: Registration representation (1 of 2) Left: 20 dB threshold. Right: 40 dB threshold.

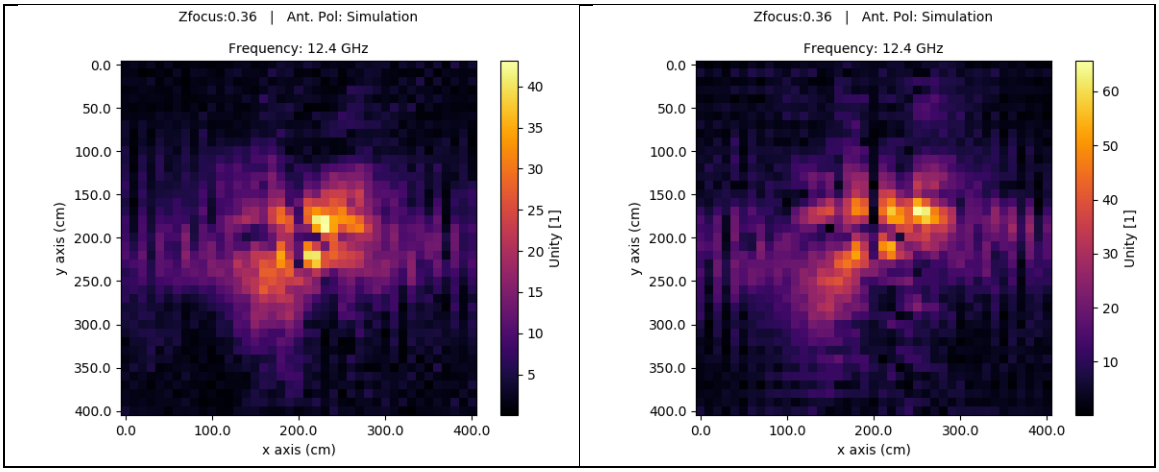

Figure A-6.8: Registration Representation (2 of 2) Left: 20 dB threshold. Right: 40 dB threshold.

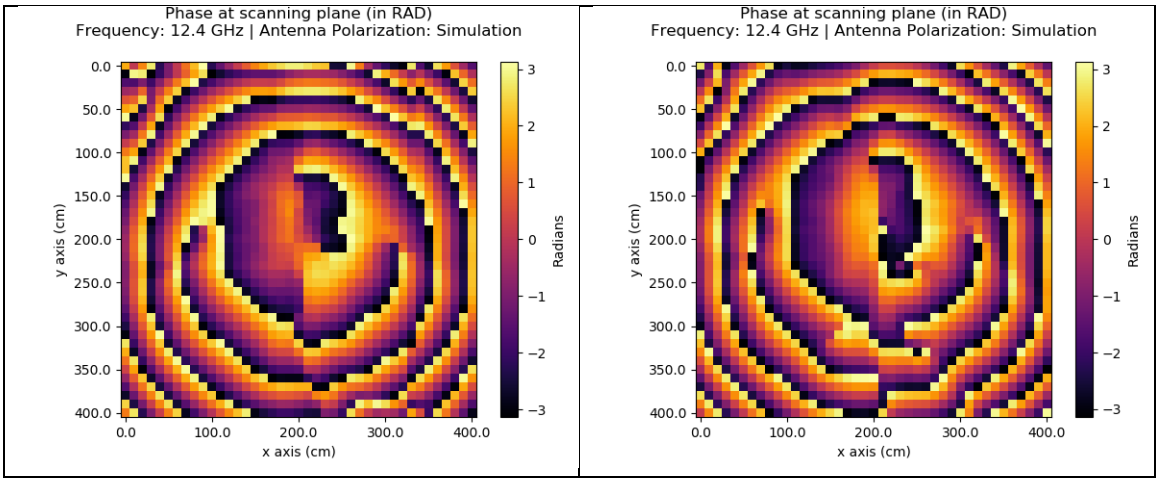

Figure A-6.9: Phase across scan plane. Left: 20 dB threshold. Right: 40 dB threshold.

VITA

Jacob Dixon

Candidate for the Degree of

Master of Science

Thesis: Analysis of Emission Source Microscopy Through Simulation

Major Field: Electrical Engineering

Biographical:

Education:

Completed the requirements for the Master of Science in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in May, 2018.

Completed the requirements for the Bachelor of Science in Electrical and Computer Engineering at Oklahoma State University, Stillwater, Oklahoma in 2015.

Experience:
Research Assistant, Oklahoma State University, Stillwater, OK (2016-2018)
Teaching Assistant, Oklahoma State University, Stillwater, OK (2016-2017)
EMC Design Aid and Automation Engineer, International Business Machines, Research Triangle Park, NC (2017)
Software Engineer, Frontier Electronics Incorporated, Stillwater, OK (2016)

Professional Memberships:
IEEE Student Member
IEEE EMC Society Member