Differential Near Field Holography for Small Antenna Arrays

by

Brian A. Janice

A Thesis
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Master of Science
in
Electrical and Computer Engineering
by

____

August 2011

APPROVED:

____

Dr. Jeffrey Herd (MIT Lincoln Laboratory)

____

Professor Reinhold Ludwig

____

Professor Sergey Makarov, Major Advisor

Abstract

Near-field diagnosis of antenna arrays is often done using microwave holography; however, the technique of near-field to near-field back-propagation quickly loses its accuracy with measurements taken farther than one wavelength from the aperture. The loss of accuracy is partially due to windowing, but may also be attributed to the decay of evanescent modes responsible for the fine distribution of the fields close to the array. In an effort to achieve better resolution, the difference between these two phase-synchronized near-field measurements is used and propagated back. The performance of such a method is established for different conditions; the extension of this technique to the calibration of small antenna arrays is also discussed.

The method is based on the idea of *differential* backpropagation using the measured/simulated/analytical data in the near field. After completing the corresponding literature search authors have found that the same idea was first proposed by P. L. Ransom and R. Mittra in 1971, at that point with the Univ. of Illinois [1],[2]. This method is basically the same, but it includes a few distinct features:

1. The near field of a (faulty) array under test is measured at $1.5\lambda - 2.5\lambda$ via a near field antenna range.
2. The template (non-faulty) near field of an array is simulated numerically (full-wave FDTD solver or FEM Ansoft/ANSYS HFSS solver) at the same distance – an alternative is to use measurements for a non-faulty array.
3. Both fields are assumed (or made) to be coherent (synchronized in phase).
4. A difference between two fields is formed and is then propagated back to array surface using the angular spectrum method (inverse Fourier propagator). The corresponding result is the surface (aperture) error field, F_Z. This approach is more precise than the inverse Rayleigh formula used in [1] since the evanescent spectrum may be included into consideration.
5. The error field magnitude, $|F_Z|$, peaks at faulty elements (both amplitude and phase excitation fault).
6. The method inherently includes all mutual coupling effects since both the template field and the measured field are full-wave results.

Acknowledgements

I would like to thank the Electrical & Computer Engineering and Mathematical Sciences Departments of Worcester Polytechnic Institute for the knowledge they have endowed me during my graduate and undergraduate studies; without their contributions, I would have not been able to complete this work. In particular, Professor Sergey Makarov has provided me with more knowledge, motivation, and confidence in my abilities than any student could ask for.

I would also like to thank Dr. Francesca Sciré-Scappuzzo for her belief in my abilities, support, and insistence on my completion of a graduate degree.

To the staff of MIT Lincoln Laboratory, in particular Drs. Jeffrey Herd and Sean Duffy: I have never been so pleased to learn so much in what felt like such little time. Your patience and manner is grand. Also, if not for Stephen Targonski and Hans Kvinlaug allowing me to use their lab and near-field range, this work would be incomplete.

To my committee: Professor Reinhold Ludwig, Dr. Jeffrey Herd, and Professor Sergey Makarov. Thank you for your time and patience – your belief in my work is an honor.

This work is a tribute to the sacrifices my mother has made in recognition of my potential. She has put me in the position I am in today.

Abstract ii
Acknowledgements iii
Table of Figures vii
Table of Tables x
Chapter 1: Introduction 1
Fundamental Antenna Quantities 2
Antenna Pattern 2
Radiation Lobes and Beamwidth 4
Polarization 6
Antenna Arrays 9
Chapter 2: Algorithm Specification and Numerical Modeling 13
The FDTD Algorithm 13
Source Modeling 18
Boundary Conditions 19
Array Geometry Under Study and Near Field Structure 22
Chapter 3: Near to Near/Far Field Transformations 26
Near to Far Field Transformations 28
Fraunhofer Diffraction 28
Equivalent Magnetic Currents and MoM 29
Rayleigh Diffraction Integral 35
Near to Nearer Field Transformations 37
Chapter 4: Idea of Differential Backpropagation 43
Algorithm 46
Simulated Backpropagation Results 49

Extensions 51
Chapter 5: Measured Results 53
Antenna Measurements 53
Hologram Error Sources 55
Array Under Study 59
Numerical vs. experimental backpropagation 61
Hybrid (Numerical – Experimental) Backpropagation 71
Chapter 6: Potential Extension to Array Calibration & Conclusions 74
Conclusions 76
References 78
Appendix A: Backpropagation for all array elements. 81
Appendix B: Backpropagation for partially attenuated element. 82
Appendix C: Backpropagation for element partially out of phase elements 83
Appendix D: Backpropagation for all array elements with 0.32λ spacing. 85
Appendix E: FDTD MATLAB Codes 86
main.m 86
array_4x4_cw.m 89
constructor.m 93
viewer.m 101
fdtd.m 103
abc_murfirst.m 129
abc_super.m 130
fc.m 131
plot_nearfield.m 131

Appendix F: Backpropagation MATLAB Code 138

Table of Figures

Figure 1 Antenna pattern representations of the same antenna array: Top left: the field pattern in a linear scale represents the magnitude of the electric or magnetic field as a function of angular space. Top right: the power pattern in a linear scale represents a plot of the square of the magnitude of the electric or magnetic field as a function of the angular space. Bottom: the power pattern in the dB scale represents the magnitude of the electric or magnetic field (in decibels) as a function of angular space. 3

Figure 2 Half-Power Beamwidth (HPBW) (top) First Null Beamwidth (FNBW) (bottom) Example for the same antenna array as shown in Fig. 1. The HPBW is approximately 26° while the FNBW is approximately 61° 6

Figure 3 Examples of linear (left) and circular (right) polarization. The polarization is the curve traced by the end point of the vector representing the instantaneous electric field. 7

Figure 4 Frequency dependent transmission coefficient (dB) of two dipoles (transmit/receive network). The red curve pertains to the two dipoles with the same polarization plane (top left). The light blue curve pertains to the network when one dipole is rotated 30° (top right). The violet curve pertains to the network when one dipole is rotated 60° (bottom left). The dark blue curve pertains to the network when one dipole is rotated 90° (bottom right). 9

Figure 5 uv-space plot of the array factor for a 10x10 array of $\lambda/2$ spaced elements. Top left shows a uniformly excited array; Top right shows an array with a progressive 60° phase shift on one axis; Bottom shows an array with a progressive 60° phase shift on both axes. 11

Figure 6 Array factors of a 10x10 array of $\lambda/2$ spaced elements with no progressive scan angle, but an amplitude taper. The graph on the left shows a uniformly excited array with the -13dB side lobes. The graph on the right shows an array with a Dolph-Tschebyscheff taper applied for -26dB side lobes. 12

Figure 7 1D visual example of the FDTD “leap frog” algorithm 15

Figure 8 3D visual example of the FDTD “leap frog” algorithm 16

Figure 9 4x4 array geometry under study. 22

Figure 10 Geometry of Fraunhofer diffraction problem. The blue section is the aperture. 28

Figure 11 Example of non-uniform amplitude pattern of individual elements in a 4x4 array of patch antennas. The top left is a corner element, the top right is an inner element, and the bottom is an edge element. 38

Figure 12 Three numerically computed spatial Fourier spectra (spectrum magnitudes) in *k*-space for a 4×4 array of λ/2 spaced patch antennas over a larger ground plane/reflector. All magnitude spectra are obtained for the co-polar electric field at the distance of λ/8 from the aperture plane. The observation plane is approximately twice as large as the array itself . The first plot (top left) is the spectrum for the non faulty array with all radiators driven by identical generators; the second plot (top right) is the spectrum for a faulty array where the generator for radiator 22 is shorted out; the third plot (bottom) is the difference spectrum between first two, which shall be used for the identification of a faulty element. 45

Figure 13 Lattice representation of surface with points stored within a matrix. Each square represents the differential area. 47

Figure 14 The differential area of matrix that should not be included in the surface area calculation is shown in red. Overlapping red regions must be subtracted twice. 48

Figure 15 Example of how the placement of the transmitting antenna along the grid changes the orientation relative to the test antenna. This directive property, along with the polarization, must be taken into account when observing measurements. The red circles portray the different part of the horn pattern seen by the array when the horn is at different locations. 54

Figure 16 Example of smoothed hologram due to windowing in *k*-space, an effect of a finite measurement plane [8] 56

Figure 17 Hologram of 4x4 patch array modeled in MATLAB with different windows in k-space. Top left pertains to $k_x^2 + k_y^2 < 0.55k$ with a 42.5% error; Top right pertains to $k_x^2 + k_y^2 < 0.65k$ with a 42.0% error; Middle left pertains to $k_x^2 + k_y^2 < 0.75k$ with a 39.9% error; Middle right pertains to $k_x^2 + k_y^2 < 0.85k$ with a 39.1% error (fake); Bottom pertains to $k_x^2 + k_y^2 < 0.95k$ with a 405% error. 58

Figure 18 Top – A 4x4 array of patches with a corporate feed and posterior-fastened aluminum ground plane; bottom – the same array in the near-field range. 60

Figure 19 Simulated (Ansoft/ANSYS HFSS) current distribution on the ground plane when one of the array elements is detuned.. 61
Figure 20 Squared differential hologram of non-faulty measured and simulated (HFSS) data with the simulated data multiplied by a variable phase. From top left, the phases are -40°, -30°, -25°, -20°, -14° (optimal). It can be seen that the measured array may not have been completely parallel with the plane of elevation. 72
Figure 21 Top - Array factor of uniformly excited 4x4 array with $\lambda / 2$ spacing (black) and array with corner element attenuated by 3dB (red). Bottom - Array factor of uniformly excited 4x4 array with $\lambda / 2$ spacing (black) and array with inner element attenuated by 3dB (red). 75

Table of Tables

Table 1 Electric field distributions in different observation planes ($\lambda/2$ spacing)........................ 23

Table 2 Backpropagation parameters corresponding to a minimum restoration error of the co-polar *E*-field at the distance of $\lambda/8$ from the top of the antenna array in the near field................ 49

Table 3 Backpropagated fields in three cases ($\lambda/2$ spacing). Element 22 is shorted out for a faulty array. 50

Table 4 Backpropagation results (num. to num. and experiment to experiment) – last row....... 63

Table 5 Hybrid backpropagation results (numerical to experiment) – last row. 73

Chapter 1: Introduction

The goal of this paper is to describe a simple yet powerful and promising method of locating partially or fully malfunctioning elements in an antenna array. The method is based on the idea of *differential* backpropagation using the measured/simulated/analytical data in the near field. After completing the corresponding literature search authors have found that the same idea was first proposed by P. L. Ransom and R. Mittra in 1971, at that point with the Univ. of Illinois [1],[2]. This method is basically the same, but it includes a few distinct features:

7. The near field of a (faulty) array under test is measured at $1.5\lambda - 2.5\lambda$ via a near field antenna range.
8. The template (non-faulty) near field of an array is simulated numerically (full-wave FDTD solver or FEM Ansoft/ANSYS HFSS solver) at the same distance – an alternative is to use measurements for a non-faulty array.
9. Both fields are assumed (or made) to be coherent (synchronized in phase).
10. A difference between two fields is formed and is then propagated back to array surface using the angular spectrum method (inverse Fourier propagator). The corresponding result is the surface (aperture) error field, F_z. This approach is more precise than the inverse Rayleigh formula used in [1] since the evanescent spectrum may be included into consideration.
11. The error field magnitude, $|F_z|$, peaks at faulty elements (both amplitude and phase excitation fault).
12. The method inherently includes all mutual coupling effects since both the template field and the measured field are full-wave results.

Under ideal conditions, the method is weakly limited by diffraction since it is based on the exact solution to Maxwell's equations. For example, if the measurement plane is chosen as the corresponding aperture and the probe height is chosen as the focal length, the well-known Rayleigh resolution criterion yields typical resolution values on the order of $\lambda/4$ or better.

Microwave holography is a well known method for diagnosis and alignment of phased array antennas [4]-[10]. The hologram, a backpropagation of the complex near field from a probe measurement, is often used as a first-look at the structural quality of an aperture. In arrays, the hologram may provide maps of aperture illumination, element weights, and geometric faults. Element weights are a primary concern when attempting to align an array, but several uncertainties are intrinsic to the hologram [4]-[8]. Some uncertainties, such as probe relative pattern and cable phase stability, are impossible to compensate for as they are specified by the manufacturer [5], [6]; however, other uncertainties such as probe positioning and aliasing [7], [8] may be minimized using certain measurement precautions. One of these precautions is to place the probe farther away from the array in order to avoid mutual coupling between elements of the array and the probe. As the probe is placed farther from the array, we will find that the hologram's accuracy degrades and may sometimes become confusing to read. This degradation is normally attributed to measurement plane truncation [8], noise, and probe inaccuracies [5], [7]; however, we will show that the evanescent modes of the array also play a role in the fine distribution of the field close to the aperture plane.

Although the hologram of an aperture gives insight to potential flaws, requirements concerning the alignment of newer arrays have become more rigid. This has called the use of holography into question as a sufficiently accurate tool for diagnosis of such systems, which are today tested with other longer and more costly procedures [11]-[13]. Although these testing procedures are very accurate, they become impractical when projecting costs to production units, as the time taken to diagnose alignment problems becomes the most costly portion of the project.

Fundamental Antenna Quantities

Antenna Pattern

The antenna pattern is defined as "a mathematical function or a graphical representation of the radiation properties of the antenna as a function of space coordinates. In most cases, the radiation pattern is determined in the far-field region and is represented as a function of the directional coordinates. Radiation properties include power flux density, radiation intensity, field strength, directivity, phase, or polarization." [3] The trace of a received electric or magnetic field at a constant radius is called the amplitude field pattern, while a graph of the spatial variation of the power density along a constant radius is called an amplitude power pattern.

These field and power patterns are often normalized with respect to their maximum value, yielding "normalized" field/power patterns. These plots are usually presented in a dB (decibels) scale, that is, $10 \times \log_{10}$ of the quantity in question. This is done to accentuate any portions of the antenna pattern that may have low values. Thus, for all antennas, we have three typical pattern plots of the same antenna,

a. The field pattern in a linear scale represents the magnitude of the electric or magnetic field as a function of angular space;
b. The power pattern in a linear scale represents a plot of the square of the magnitude of the electric or magnetic field as a function of the angular space
c. The power pattern in the dB scale represents the magnitude of the electric or magnetic field (in decibels) as a function of angular space.

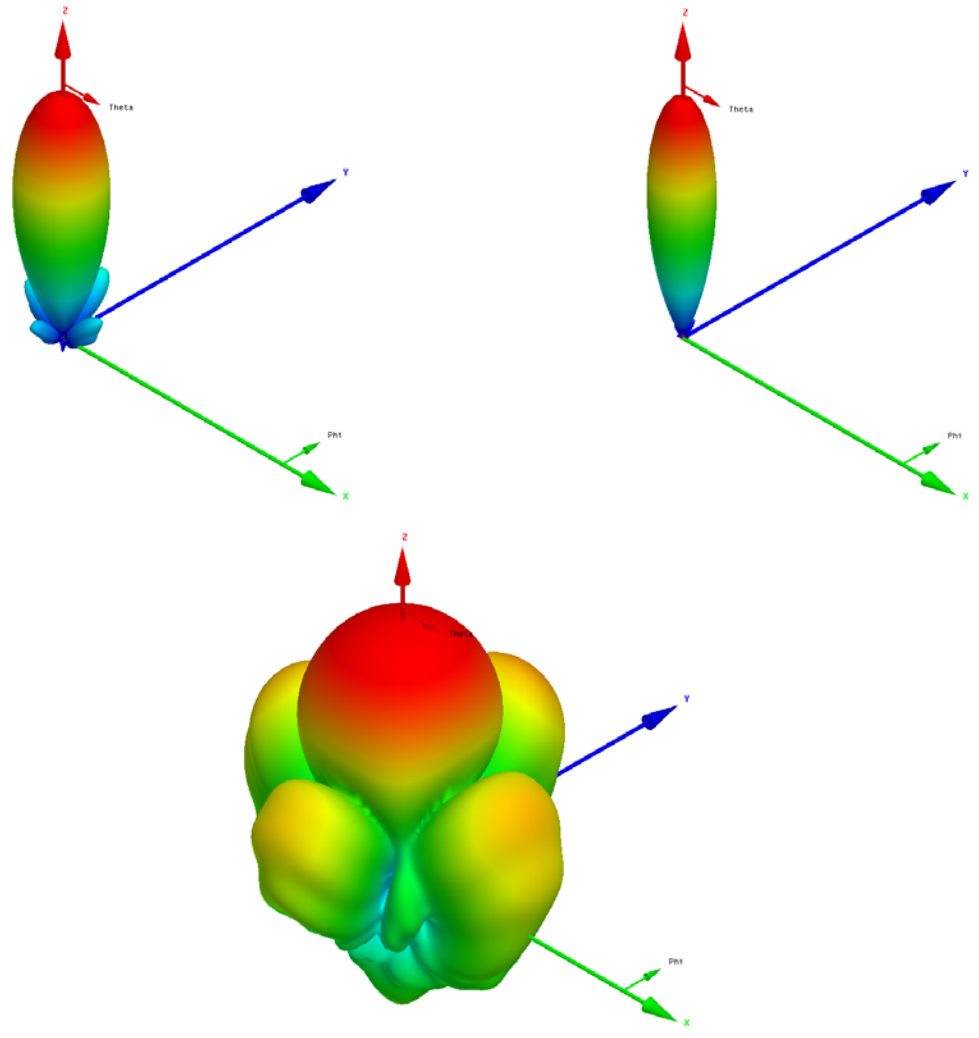


Figure 1 Antenna pattern representations of the same antenna array: Top left: the field pattern in a linear scale represents the magnitude of the electric or magnetic field as a function of angular space. Top right: the power pattern in a linear scale represents a plot of the square of the magnitude of the electric or magnetic field as a function of the angular space. Bottom: the power pattern in the dB scale represents the magnitude of the electric or magnetic field (in decibels) as a function of angular space.

The performance of an antenna is often given in terms of its principal E - and H - plane patterns. The E-plane is defined as "The plane containing the electric field vector and the direction of maximum radiation;" while the H-plane is defined as "the plane containing the magnetic field vector and the direction of maximum radiation." [3] It is possible for an antenna to have just one principle plane, or an infinite number. In a linearly polarized antenna, the E-plane may also be thought of as the plane in which current flows in the antenna, while the H-plane is the plane perpendicular to that.

Radiation Lobes and Beamwidth

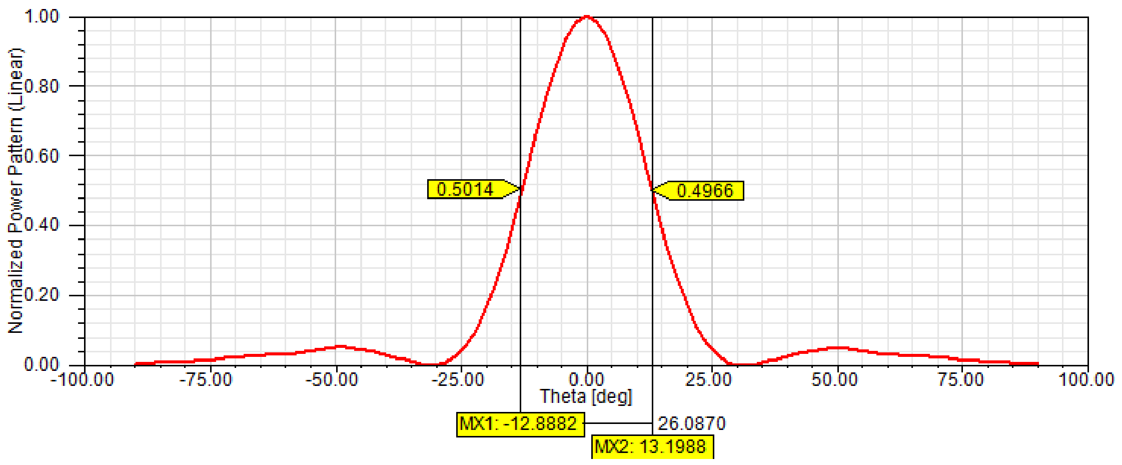
When observing a radiation pattern, there may be one portion of the pattern which has a very high value compared to those portions surrounding it. This portion of the pattern is known as a "radiation lobe." Antenna patterns may have multiple lobes; for example, Fig. 1 has five lobes.

A "major lobe," also known as "main beam," is defined as "the radiation lobe containing the direction of maximum radiation." [3] Fig.1 demonstrates an antenna whose main beam is directed in the $\theta = 0$ direction. Although this is the case for Fig. 1, antennas may have a main beam directed in any direction and may even have multiple beams pointed in several directions. Any lobe except the major lobe is called a minor lobe; for example, Fig. 1 has four minor lobes surrounding its main beam.

A side lobe is defined as "a radiation lobe in any other direction than the intended lobe." [3] This definition may be further specified as any lobe adjacent to the main beam located in the same hemisphere in the direction of the main beam. Thus, it may be said that Fig. 1 has four side lobes. A back lobe is a "radiation lobe whose axis makes an angle of approximately 180 degrees with respect to the main beam of the antenna." [3] This term usually applies to any minor lobe located in the hemisphere pointed in the opposite of the main beam.

Minor lobes typically represent radiation of power in undesired locations; thus, many designers seek to minimize them as part of their design. Minor lobes are normally characterized by taking the ratio of the power density in the minor lobe to that of the main beam; usually this is desired to be below -20dB. Fulfilling this requirement is very important in radar systems, as side lobes may increase the number of false target detections.

Associated with the main beam is the beamwidth. This parameter is known to be the angular separation between two identical points on the main beam. [3] There are several ways of choosing these "identical points;" however, one of the most popular choices is that point which the radiation intensity is half of its maximum value. This is known as Half-Power Beamwidth (HPBW) and is defined by the IEEE as "In a plane containing the direction of the maximum of a beam, the angle between the two directions in which the radiation intensity is one half value of the beam." Another popular choice for beamwidth is the angular separation at which the first null on each side of the main beam appears, or First Null Beamwidth (FNBW). One common approximation made by engineers is $HPBW \approx FNBW / 2$. This is an important quantity for antennas, as it is also describes the resolution capabilities of the antenna to distinguish between two adjacent radiating sources or radar targets.



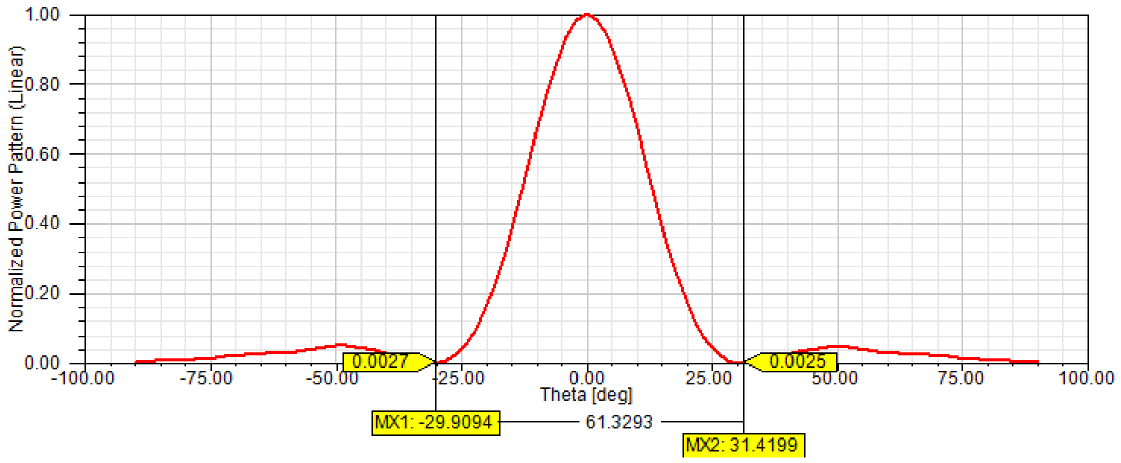


Figure 2 Half-Power Beamwidth (HPBW) (top) First Null Beamwidth (FNBW) (bottom) Example for the same antenna array as shown in Fig. 1. The HPBW is approximately 26° while the FNBW is approximately 61°

Although a narrow beamwidth would be desirable in many cases, there is a tradeoff with respect to this parameter and that of the side lobe level, i.e. the more narrow the beamwidth, the higher the side lobe level and vice versa.

Polarization

"The polarization of the wave transmitted (radiated) by the antenna. *Note:* When the direction is not stated, the polarization is taken to be the polarization in the direction of maximum gain." In reality, polarization varies with the direction from the center of the antenna such that parts of the pattern may have different polarizations.

The polarization of a wave is defined as "that property of an electromagnetic wave describing the time-varying direction and relative magnitude of the electric-field vector; specifically, the figured traced as a function of time by the extremity of the vector at a fixed location in space, and the sense in which is it traced, *as observed along the direction of propagation*;" [3] thus, the polarization is the curve traced by the end point of the vector representing the instantaneous electric field. This may be defined in terms of waves either radiated or received by an antenna. The polarization of a wave radiated by an antenna in the far field is defined as "the polarization of the (locally) plane wave which is used to represent the radiated wave at that point. At any point in the far field of an antenna, the radiated wave can be represented by a plane wave whose electric-field strength is the same as that of the wave and whose direction of propagation is in the radial direction from the antenna. As the radial distance approaches infinity, the radius of curvature of the radiated wave's phase front also approaches infinity and thus in any specified direction the wave appears locally as a plane wave." [3]

Polarization falls into three categories: linear, circular, and elliptical. A wave is linearly polarized if the vector that describes the electric field always falls in the same plane; a wave is circularly polarized if the endpoint of the vector that describes the instantaneous electric field traces a perfect circle; a wave is elliptically polarized if the endpoint of the vector that describes the instantaneous electric field traces an ellipse.

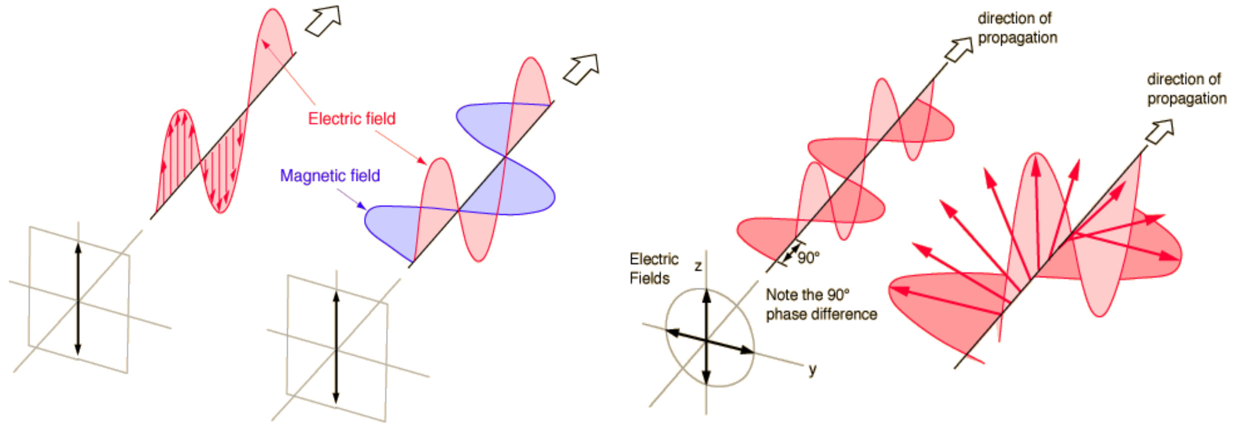


Figure 3 Examples of linear (left) and circular (right) polarization. The polarization is the curve traced by the end point of the vector representing the instantaneous electric field.

In order for an electric field vector to be linearly polarized, it must possess

a. Only one component, or
b. Two orthogonal linear components that are in time phase or 180° (or multiples of 180°) out of phase.

When an antenna is linearly polarized, the (principle) E-plane pattern is directly related to the polarization axis.

In order for an electric field vector to be circularly polarized, it must possess all of the following

a. The field must have two orthogonal components
b. The two components must have the same magnitude
c. The two components must have a time phase difference of odd multiples of 90°.

A field is elliptically polarized if it is neither linearly of circularly polarize; however, the necessary and sufficient conditions to create an elliptically polarized electric field vector are

a. The field must have two orthogonal linear components,
b. The two components can be of the same or different magnitude.
c. If the two components are not of the same magnitude, the time phase difference between the two components must not be 0° or multiples of 180°. If the two components are of the same magnitude, the time phase difference between the two components must not be odd multiples of 90°

The "sense of rotation" for a circular or elliptical polarized wave is always determined by rotating the phase-leading component toward the phase-lagging component and observing the field rotation as the wave is viewed as it travels away from the observer. If rotation is clockwise, the wave is right-hand (CW) polarized, if rotation is counter clockwise, the wave is left-hand (CCW) polarized [3].

Polarization is important because if an antenna were trying to transmit a linearly polarized signal to an identical antenna rotated 90° in the plane orthogonal to the direction of propagation (orthogonal polarization), it would receive no signal at all; thus, it is important to be aware of the polarization of an antenna. Fig. 4 shows a network of two dipoles – one transmit, one receive. One dipole is rotated to show how the polarization affects power transmission. The top left image pertains to both dipoles with the same polarization; the top right image pertains to one dipole being rotated by 30°; the bottom left image pertains to one dipole being rotated by 60°; the bottom right image pertains to both dipoles having polarizations orthogonal to each other. The S21 parameter, a network parameter directly related to power transmission between two ports, is shown in a plot below for each configuration versus frequency. One can clearly see the degradation of transmitted power as a function of rotation as both dipoles begin to have orthogonal polarizations.

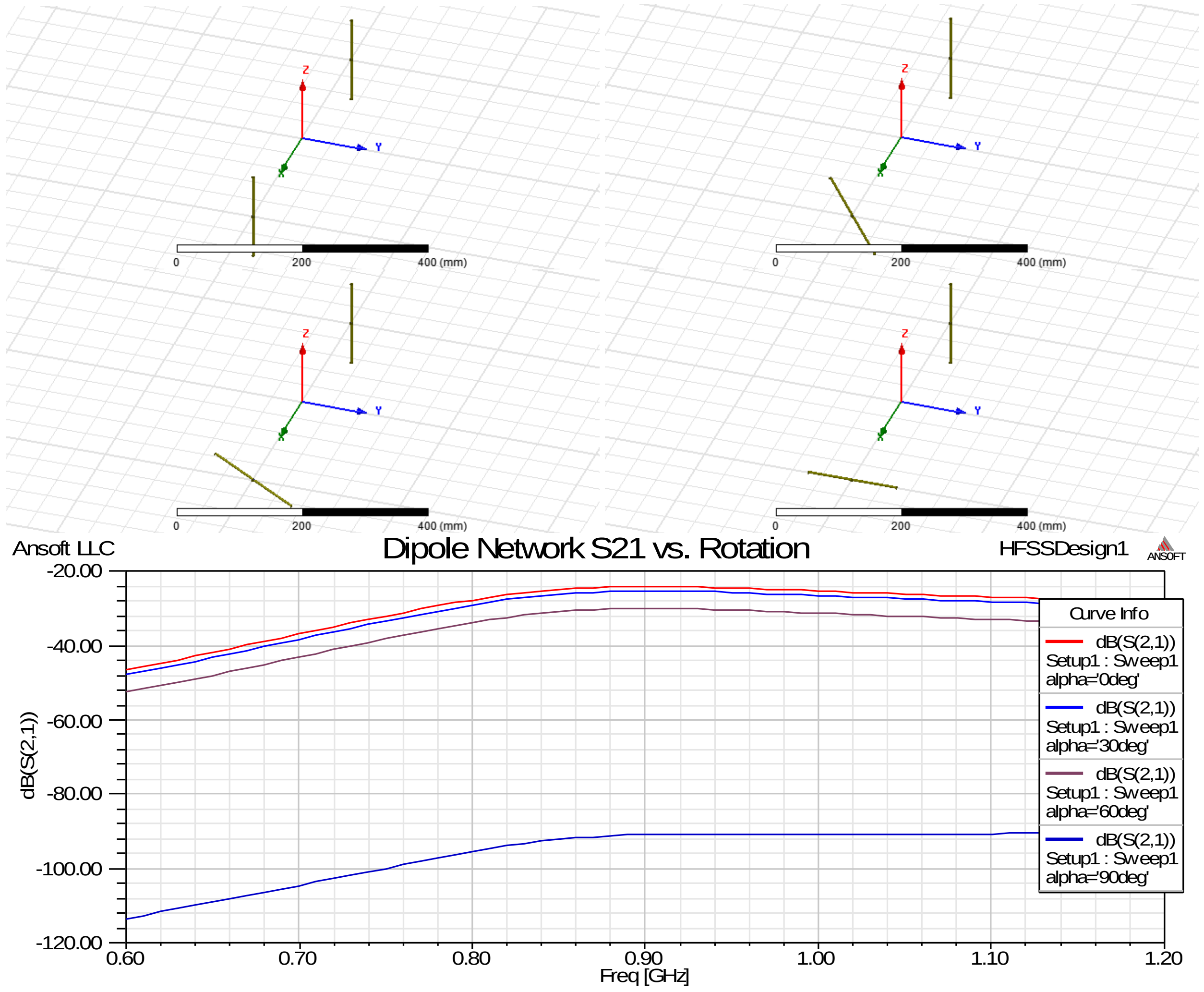


Figure 4 Frequency dependent transmission coefficient (dB) of two dipoles (transmit/receive network). The red curve pertains to the two dipoles with the same polarization plane (top left). The light blue curve pertains to the network when one dipole is rotated 30° (top right). The violet curve pertains to the network when one dipole is rotated 60° (bottom left). The dark blue curve pertains to the network when one dipole is rotated 90° (bottom right).

Antenna Arrays

As mentioned in the previous section, narrow beamwidth of an antenna system is helpful for improving resolution in radar systems. Unfortunately, this is not very easy to accomplish with a single antenna element; however, if we construct an antenna made of several antennas, called an antenna array, this turns out to be an easier task. Another way of narrowing the beam of an antenna is by using a parabolic reflector. This is a very directional type of antenna; however, it must be scanned mechanically, and thus is not as quick as scanning electrically with a phased array.

The total field of an antenna array is usually analytically determined as the vector sum of the fields radiated by each individual element. This sum is not entirely accurate, as the elements in an array will likely couple to one another, creating element-specific fields. [3] This approximation may be avoided by modeling the fields produced each element in the array separately while all other elements are match-terminated, called the "active element" pattern. Adding the vector sum of all active element patterns in an array will yield the proper array pattern. It is possible to shape the pattern of an antenna array using several design parameters [3], some of which are:

a. Element spacing;
b. Element excitation amplitude;
c. Element excitation phase;
d. Element geometry (i.e. individual element pattern);
e. Array geometry (elements aligned in a linear fashion, rectangular grid, triangular grid, spherical grid, etc.).

As mentioned, the radiation pattern of an antenna array may be broken into the sum of the patterns for each individual element. Ignoring mutual coupling, let's consider an array of elements located on a grid. The phase of each element as represented in the far field is expressed as the complex exponential $e^{j\mathbf{k}\cdot\mathbf{r}_i}$ where $\mathbf{k} = \frac{2\pi}{\lambda}(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)$ is the wave vector, and $\mathbf{r}_i$ is the special location of the element in question. If we are to sum up all patterns multiplied by the proper phase, we will end up with the following expression

$$\mathbf{E}_{ARRAY} = \mathbf{E}_{ELEMENT} I_1 e^{j\mathbf{k}\cdot\mathbf{r}_1} + \mathbf{E}_{ELEMENT} I_2 e^{j\mathbf{k}\cdot\mathbf{r}_2} + I_3 \mathbf{E}_{ELEMENT} e^{j\mathbf{k}\cdot\mathbf{r}_3} + \cdots = \mathbf{E}_{ELEMENT} \times \sum_{i=1}^{N} I_i e^{j\mathbf{k}\cdot\mathbf{r}_i}$$

The summation in eq (1) is known as the array factor for an array with no individual phase shifts. If we are to introduce two *progressive* phase shifts in the x and y directions on a rectangular grid, the total array factor for a rectangular grid of antennas is

$$AF = \sum_{m=1}^{M} I_m \exp\left[j(m-1)\left(kd_x \sin\theta\cos\varphi + \beta_x\right)\right]\sum_{n=1}^{N} I_n \exp\left[j(n-1)\left(kd_y \sin\theta\sin\varphi + \beta_y\right)\right]$$

This expression is used to approximate the far field radiation pattern of a rectangular *phased* antenna array, if the array consists of dipole elements, the total pattern would be $\mathbf{E}_{DIPOLE} \times AF$, etc. One tool that some engineers use to view the array factor is called "uv-space." This is a coordinate transform which shows the angular shift of an array beam as a translation on the Cartesian plane. The transform is

$$u = \cos\varphi\sin\theta$$
$$v = \sin\varphi\sin\theta$$

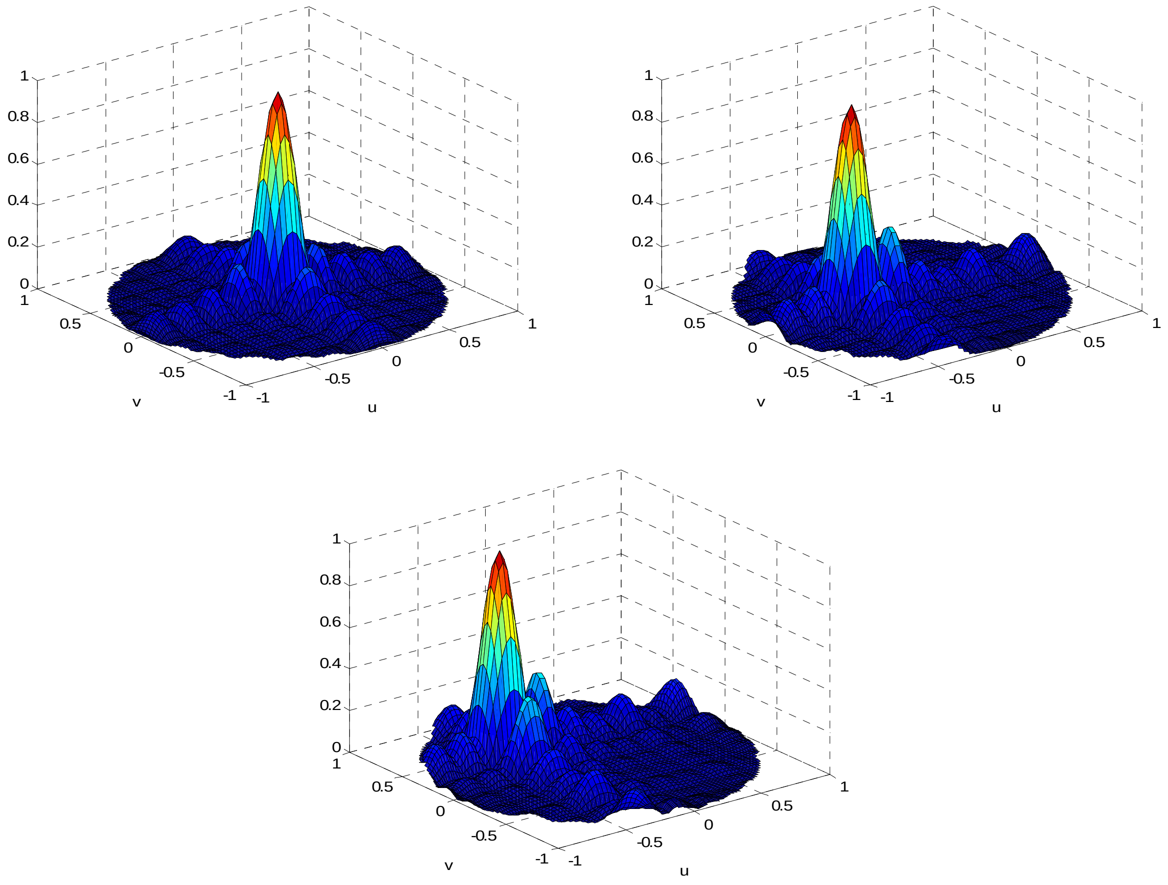


Figure 5 uv-space plot of the array factor for a 10x10 array of $\lambda/2$ spaced elements. Top left shows a uniformly excited array; Top right shows an array with a progressive 60° phase shift on one axis; Bottom shows an array with a progressive 60° phase shift on both axes.

One way of minimizing side lobes with arrays with the tradeoff of transmitted power is by applying a taper to the gain of elements; the Dolph-Tschebyscheff taper is a popular method of producing very low side lobes with acceptable field strength loss. This is done by applying the weights of the n^{th} degree Tschebyscheff polynomial to each element in the array. If low side lobes are only needed on one axis, the weights may be applied to only one axis; however, if the low side lobes are desired for all axes, the weights must be applied to each other on both axes and will result in less transmitted power.

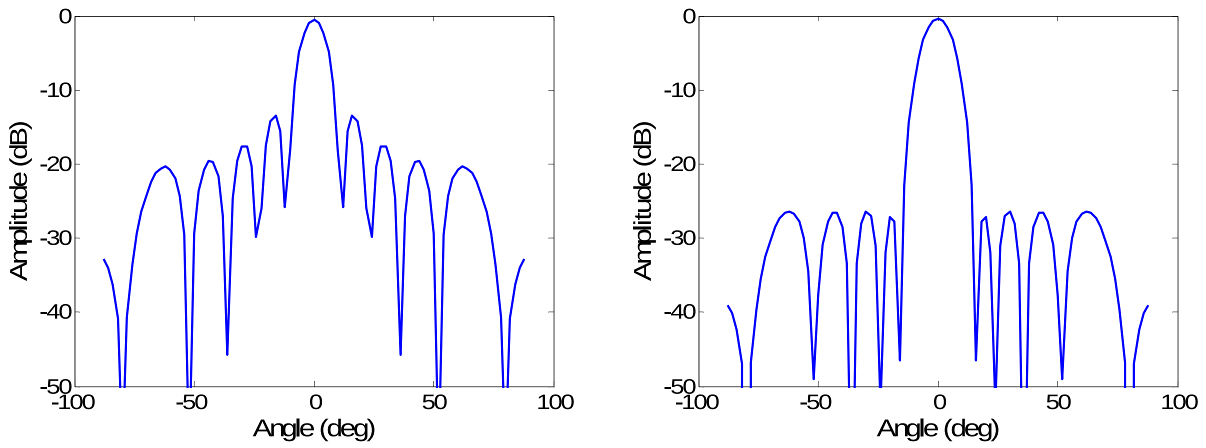


Figure 6 Array factors of a 10x10 array of $\lambda / 2$ spaced elements with no progressive scan angle, but an amplitude taper. The graph on the left shows a uniformly excited array with the -13dB side lobes. The graph on the right shows an array with a Dolph-Tschebyscheff taper applied for -26dB side lobes.

Chapter 2: Algorithm Specification and Numerical Modeling

The FDTD Algorithm

In order to solve Maxwell's equations, we will consider their differential form. The most natural way to solve these equations is by finite differences.

Finite difference methods for solving differential equations utilize the Taylor expansion of a function, *f*, in the following form:

$$f(x+\delta) = f(x) + \delta\frac{d}{dx}f(x) + \frac{\delta^2}{2}\frac{d^2}{dx^2}f(x) + \frac{\delta^3}{6}\frac{d^3}{dx^3}f(x) + \cdots$$

Rearranging (X) in the form of a central difference, we have

$$\frac{f(x+h) - f(x)}{h} = f'(x+\frac{h}{2}) + O(h^2)$$

These equations are the foundation of the finite-difference time-domain (FDTD) method of solving differential equations. When solving electromagnetic problems, finite-difference time-domain method (FDTD) is used frequently due to its efficiency and adaptability to different problems. FDTD is especially useful when dealing with problems defined on a Cartesian grid. This condition is useful when dealing with rectangular patch antennas, as they rarely have oblique boundaries. When considering the discretization of the problem, the spatial meshing may be chosen at the programmer's discretion; however, the time step must maintain the following inequality

$$\Delta t < \frac{h}{c\sqrt{3}} \tag{1}$$

In order for us to start solving Maxwell's equations for the quantities $\mathbf{E}$ and $\mathbf{H}$, we will express Faraday's Law and Ampere's Law in their form as two coupled first order differential equations

$$\frac{\partial \mathbf{D}}{\partial t} = \nabla \times \mathbf{H}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

which translates to the following system of six scalar equations (for a homogeneous domain)

$$\begin{aligned}
\varepsilon \frac{\partial E_x}{\partial t} &= \frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z} \\
\varepsilon \frac{\partial E_y}{\partial t} &= \frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x} \\
\varepsilon \frac{\partial E_z}{\partial t} &= \frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}
\end{aligned} \tag{2}$$

$$\begin{aligned}
\mu \frac{\partial H_x}{\partial t} &= \frac{\partial E_y}{\partial z} - \frac{\partial E_z}{\partial y} \\
\mu \frac{\partial H_y}{\partial t} &= \frac{\partial E_z}{\partial x} - \frac{\partial E_x}{\partial z} \\
\mu \frac{\partial H_z}{\partial t} &= \frac{\partial E_x}{\partial y} - \frac{\partial E_y}{\partial x}
\end{aligned} \tag{3}$$

This leaves us with six equations to solve for six unknowns.

The finite difference approximation of a two dimensional problem is best expressed on a "staggered grid" where $\mathbf{E}$ is expressed at integer multiples of the spatial step, while $\mathbf{H}$ is expressed on half-integer multiples of the spatial step. In order to denote such a scheme, let the subscript r denote an index that refers to the z-coordinate and the superscript n refer to the time coordinate such that $f_r^n \equiv f(r\Delta z, n\Delta t)$.

Expressing the x component of eq. 2 and eq. 3 in finite differences yields

$$\frac{E_{x\,r}^{\,n+1} - E_{x\,r}^{\,n}}{\Delta t} = -\frac{1}{\varepsilon} \frac{H_{y\,r+\frac{1}{2}}^{\,n+\frac{1}{2}} - H_{y\,r-\frac{1}{2}}^{\,n+\frac{1}{2}}}{\Delta z}$$

$$\frac{H_{y\,r+\frac{1}{2}}^{\,n+\frac{1}{2}} - H_{y\,r+\frac{1}{2}}^{\,n-\frac{1}{2}}}{\Delta t} = -\frac{1}{\mu}\frac{E_{x\,r+1}^{\,n} - E_{x\,r}^{\,n}}{\Delta z}$$

The analog for the other two dimensions (four more equations) may be found in the same way.

In order to better visualize this process in the one dimensional problem, consider Fig. 7 portraying the so-called "leap-frog algorithm."

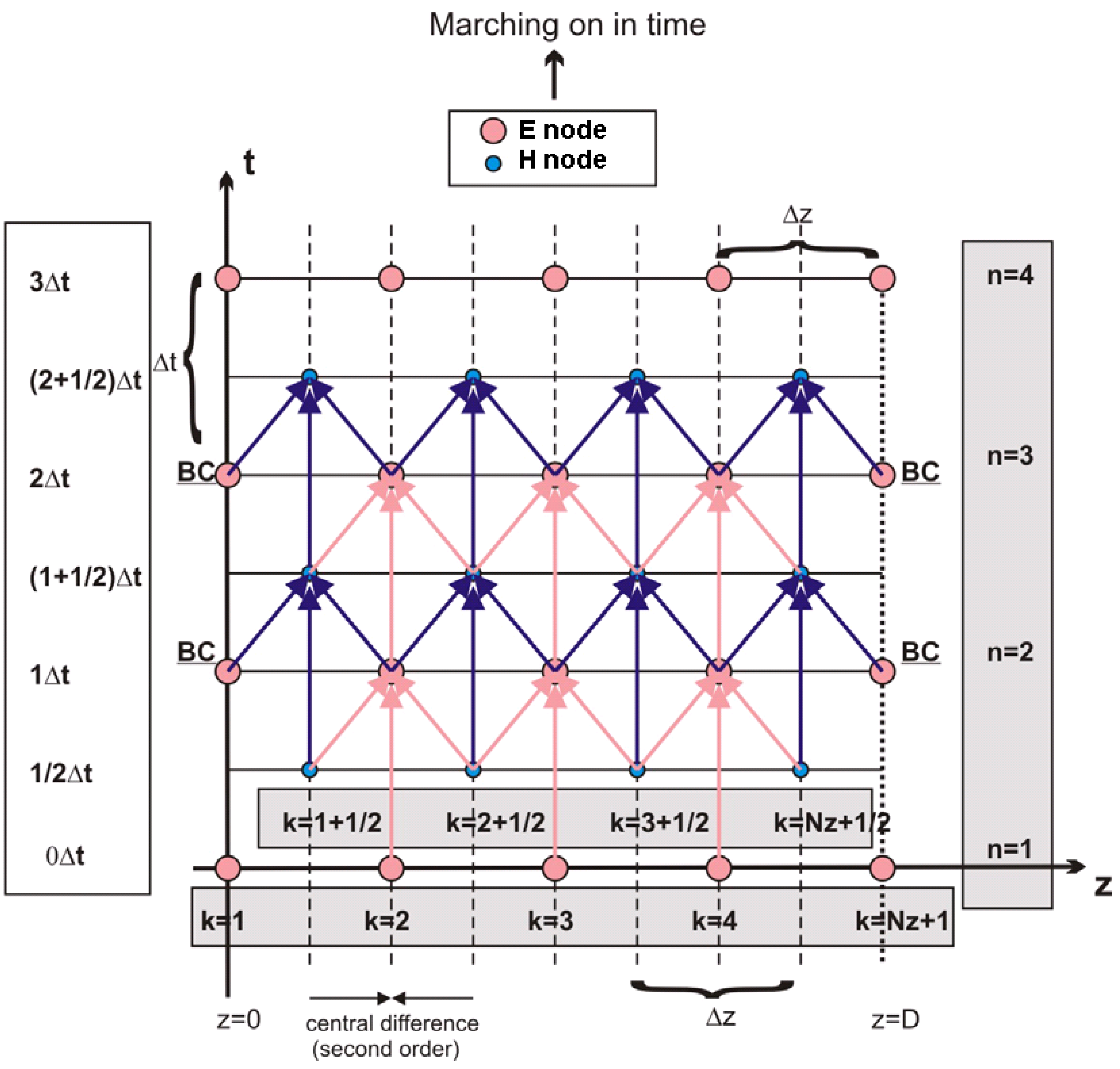


Figure 7 1D visual example of the FDTD "leap frog" algorithm

Extrapolating this to three dimensions yields a hexagonal lattice with half-integer step sizes shown in Fig. 8

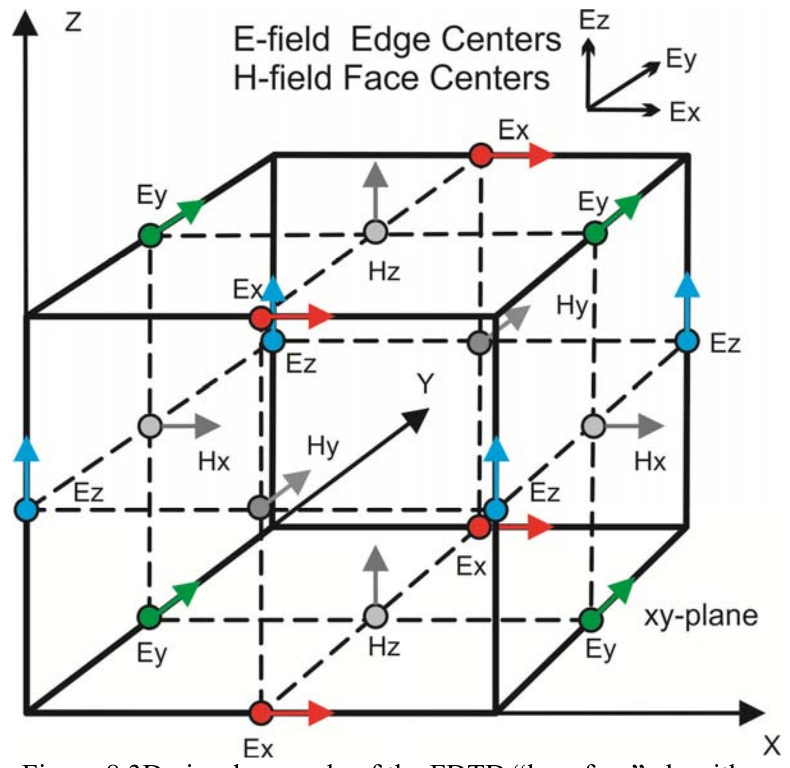


Figure 8 3D visual example of the FDTD "leap frog" algorithm

The equations which govern the 3D FDTD algorithm are as follows with the same notation as above with $f^{n}_{p,q,r} \equiv f(p\Delta x, q\Delta y, r\Delta z, n\Delta t)$. This is known as the Yee FDTD scheme.

$$\varepsilon \frac{E_x{}^{n+1}_{p+\frac{1}{2},q,r} - E_x{}^{n}_{p+\frac{1}{2},q,r}}{\Delta t} = \frac{H_z{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q+\frac{1}{2},r} - H_z{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q-\frac{1}{2},r}}{\Delta y} - \frac{H_y{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q,r+\frac{1}{2}} - H_y{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q,r-\frac{1}{2}}}{\Delta z}$$

$$\varepsilon \frac{E_y{}^{n+1}_{p,q+\frac{1}{2},r} - E_y{}^{n}_{p,q+\frac{1}{2},r}}{\Delta t} = \frac{H_x{}^{n+\frac{1}{2}}_{p,q+\frac{1}{2},r+\frac{1}{2}} - H_x{}^{n+\frac{1}{2}}_{p,q+\frac{1}{2},r-\frac{1}{2}}}{\Delta z} - \frac{H_z{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q+\frac{1}{2},r} - H_z{}^{n+\frac{1}{2}}_{p-\frac{1}{2},q+\frac{1}{2},r}}{\Delta x}$$

$$\varepsilon \frac{E_z{}^{n+1}_{p,q,r+\frac{1}{2}} - E_z{}^{n}_{p,q,r+\frac{1}{2}}}{\Delta t} = \frac{H_y{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q,r+\frac{1}{2}} - H_y{}^{n+\frac{1}{2}}_{p-\frac{1}{2},q,r+\frac{1}{2}}}{\Delta x} - \frac{H_z{}^{n+\frac{1}{2}}_{p,q+\frac{1}{2},r+\frac{1}{2}} - H_z{}^{n+\frac{1}{2}}_{p,q-\frac{1}{2},r+\frac{1}{2}}}{\Delta y} \tag{4}$$

$$\mu \frac{H_x{}^{n+\frac{1}{2}}_{p,q+\frac{1}{2},r+\frac{1}{2}} - H_x{}^{n}_{p,q+\frac{1}{2},r+\frac{1}{2}}}{\Delta t} = \frac{E_y{}^{n}_{p+\frac{1}{2},q,r+1} - E_y{}^{n}_{p-\frac{1}{2},q,r}}{\Delta z} - \frac{E_z{}^{n}_{p,q+1,r+\frac{1}{2}} - E_z{}^{n}_{p,q,r+\frac{1}{2}}}{\Delta y}$$

$$\mu \frac{H_y{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q,r+\frac{1}{2}} - H_y{}^{n}_{p+\frac{1}{2},q,r+\frac{1}{2}}}{\Delta t} = \frac{E_z{}^{n}_{p+1,q,r+\frac{1}{2}} - E_z{}^{n}_{p,q,r+\frac{1}{2}}}{\Delta x} - \frac{E_x{}^{n}_{p+\frac{1}{2},q+1,r+1} - E_x{}^{n}_{p+\frac{1}{2},q,r}}{\Delta z}$$

$$\mu \frac{H_z{}^{n+\frac{1}{2}}_{p+\frac{1}{2},q+\frac{1}{2},r} - H_z{}^{n}_{p+\frac{1}{2},q+\frac{1}{2},r}}{\Delta t} = \frac{E_x{}^{n}_{p+\frac{1}{2},q+1,r} - E_x{}^{n}_{p,q,r+\frac{1}{2}}}{\Delta y} - \frac{E_y{}^{n}_{p+1,q+\frac{1}{2},r+1} - E_y{}^{n}_{p,q+\frac{1}{2},r}}{\Delta x}$$

The following code shows how Yee's FDTD scheme is realized in MATLAB.

```
%----------------------------------------------------------------
Dt       % Time step
Cx       % 1/dx
Cy       % 1/dy
Cz       % 1/dz
mu0      % permeability
eps0     % permittivity

% Allocate field matrices
Ex = zeros(Nx  , Ny+1, Nz+1);
Ey = zeros(Nx+1, Ny  , Nz+1);
Ez = zeros(Nx+1, Ny+1, Nz  );
Hx = zeros(Nx+1, Ny  , Nz  );
Hy = zeros(Nx  , Ny+1, Nz  );
Hz = zeros(Nx  , Ny  , Nz+1);

for n = 1:Nt;
    Hx = Hx + (Dt/mu0)*(diff(Ey,1,3)*Cz - diff(Ez,1,2)*Cy);
    Hy = Hy + (Dt/mu0)*(diff(Ey,1,1)*Cx - diff(Ez,1,3)*Cz);
    Hz = Hz + (Dt/mu0)*(diff(Ey,1,2)*Cy - diff(Ez,1,1)*Cx);
    Ex(:,2:Ny,2:Nz) = Ex(:,2:Ny,2:Nz) + (Dt /eps0) * ...
        (diff(Hz(:,:,2:Nz),1,2)*Cy - diff(Hy(:,2:Ny,:),1,3)*Cz);
    Ey(2:Nx,:,2:Nz) = Ey(2:Nx,:,2:Nz) + (Dt /eps0) * ...
        (diff(Hx(2:Nx,:,:),1,3)*Cy - diff(Hz(:,:,2:Nz),1,1)*Cx);
    Ez(2:Nx,2:Ny,:) = Ez(2:Nx,2:Ny,:) + (Dt /eps0) * ...
        (diff(Hy(:,2:Ny,:),1,1)*Cx - diff(Hx(2:Nx,:,:),1,2)*Cy);
end
%---------------------------------------------------------------
```

Source Modeling

Although we are able to simulate the time progression of the fields using Yee's FDTD scheme, we still must provide "initial values" to the boundary value problem, otherwise known as the source. Sources are classified into two basic categories: hard sources and soft sources. The hard source, or "replaced" source 0 is equivalent to defining one of the components of the electric field at a certain point, x_e, y_e, in the form

$$E_{x,y,z}(t, x_e, y_e) = \sin \omega t\,, \qquad 0 < t < T_{OFF}$$

This type of source was used more in the past when exciting coaxial and waveguide structures; however, in some systems, the hard source may cause reflections of waves propagating back to the source location. It can be shown that the hard source is analogous to placing a voltage across a lossless transmission line.

The soft current source, on the other hand, is expressed as a lumped current source added to Maxwell's equations. Let's assume the case of an electric field source pointed in the z direction. The electric field, as calculated by Ampere's law, can be written in the form

$$\frac{\partial E_z}{\partial t} = \frac{1}{\varepsilon}\frac{\partial H_y}{\partial x} - \frac{1}{\varepsilon}\frac{\partial H_x}{\partial y} - \frac{\sigma}{\varepsilon}E_z \rightarrow \varepsilon\frac{\partial E_z}{\partial t} = \nabla \times \mathbf{H} - J_z^c\,, \qquad J_z^c = \sigma E_z$$

Where J_z^c is the conduction current density directed in the z direction. Similarly, we can introduce a lumped current density into Ampere's law as the source of EM radiation,

$$\varepsilon \frac{\partial E_z}{\partial t} = \nabla \times \mathbf{H} - J_z^c, \quad J_z^c = \frac{I_z^L}{\Delta x \Delta y} \tag{5}$$

where I_z^L is the total lumped current. As mentioned, it is possible to write eq. (5) in terms of finite differences with a known lumped current as an added source. The defined impressed current density may exist in free space or it may be supported by a physical conductor - in either case, the initial source is a voltage.

Boundary Conditions

When solving Maxwell's equations, the boundary conditions must be satisfied numerically as well. If the boundary condition of $\mathbf{E} = \mathbf{0}$ is chosen for the electric field, this indicates that the edges of the domain are composed of perfect electric conductors (PEC) since the tangential component of the electric field on a PEC is 0. In the case of an open boundary, a boundary which absorbs all incident electromagnetic waves is desired; this condition is known as absorbing boundary conditions (ABCs).

Let us observe the wave equation for an unknown quantity $\mathbf{W}$.

$$\frac{\partial^2 \mathbf{W}}{\partial t^2} - c_0^2 \mathbf{\Delta W} = \mathbf{0}$$

$$\frac{\partial^2 \mathbf{W}}{\partial t^2} - c_0^2 \left(\frac{\partial^2 \mathbf{W}}{\partial x^2} + \frac{\partial^2 \mathbf{W}}{\partial y^2} + \frac{\partial^2 \mathbf{W}}{\partial z^2} \right) = \mathbf{0}$$

Rearranging this equation to emphasize dominant propagation along the x -axis, we have

$$\left(\frac{\partial}{\partial t} - c_0 \frac{\partial}{\partial x} \right) \left(\frac{\partial}{\partial t} + c_0 \frac{\partial}{\partial x} \right) \mathbf{W} - c_0^2 \left(\frac{\partial^2 \mathbf{W}}{\partial y^2} + \frac{\partial^2 \mathbf{W}}{\partial z^2} \right) = \mathbf{0}$$

If we wished to nullify the propagation of $\mathbf{W}$ in the $-x$ direction, we could impose the condition

$$\left(\frac{\partial}{\partial t} - c_0 \frac{\partial}{\partial x} \right) \mathbf{W} = \mathbf{0} \tag{6}$$

Likewise, if we wished to nullify the propagation of $\mathbf{W}$ in the $+x$ direction, we would use the opposite signs as in eq. (6). This relates to our problem in the sense that, on the boundary, if we want the plane-wave portion of $\mathbf{E}$ to be nullified so there is no reflection, we could impose the following boundary conditions known as Mur's first order ABCs.

$$\frac{\partial E_z}{\partial t} - c_0 \frac{\partial E_z}{\partial x} = 0 \tag{7}$$

$$E_{z\,1,m}^{\,n+1} = E_{z\,2,m}^{\,n} + \frac{c_0\Delta t - \Delta x}{c_0\Delta t + \Delta x}\left(E_{z\,2,m}^{\,n+1} - E_{z\,1,m}^{\,n}\right)$$

$$\frac{\partial E_z}{\partial t} + c_0 \frac{\partial E_z}{\partial x} = 0 \tag{8}$$

$$E_{z\,N_x+1,m}^{\,n+1} = E_{z\,N_x,m}^{\,n} + \frac{c_0\Delta t - \Delta x}{c_0\Delta t + \Delta x}\left(E_{z\,N_x,m}^{\,n+1} - E_{z\,N_x+1,m}^{\,n}\right)$$

This result may also be extrapolated to the other two spatial dimensions similar to that of the FDTD equations' permutation of *x*, *y*, and *z*. The 3D implementation in MATLAB is as follows

```
%-------------------------------------------------------------------------------
m1      = (c0*dt - d)/(c0*dt + d);
%   Left
EyN(1, :,:)   =  EyP(2,:,:)  + m1*(EyN(2,:,:) - EyP(1,:,:));        %   left - Ey;
EzN(1, :,:)   =  EzP(2,:,:)  + m1*(EzN(2,:,:) - EzP(1,:,:));        %   left - Ez;
%   Right
EyN(Nx+1, :,:)=  EyP(Nx,:,:) + m1*(EyN(Nx, :,:) - EyP(Nx+1,:,:));   %   right - Ey;
EzN(Nx+1, :,:)=  EzP(Nx,:,:) + m1*(EzN(Nx, :,:) - EzP(Nx+1,:,:));   %   right - Ez;
%   Front
ExN(:, 1,:)   =  ExP(:,2,:)  + m1*(ExN(:,2,:) - ExP(:,1,:));        %   front - Ex;
EzN(:, 1,:)   =  EzP(:,2,:)  + m1*(EzN(:,2,:) - EzP(:,1,:));        %   front - Ez;
%   Rear
ExN(:, Ny+1,:)=  ExP(:,Ny,:) + m1*(ExN(:,Ny,:) - ExP(:,Ny+1,:));    %   rear - Ex;
EzN(:, Ny+1,:)=  EzP(:,Ny,:) + m1*(EzN(:,Ny,:) - EzP(:,Ny+1,:));    %   rear - Ey;
%   Bottom
ExN(:, :,1)   =  ExP(:, :,2)  + m1*(ExN(:,:,2) - ExP(:,:,1));       %   bottom - Ex;
EyN(:, :,1)   =  EyP(:, :,2)  + m1*(EyN(:,:,2) - EyP(:,:,1));       %   bottom - Ey;
%   Top
```

```
ExN(:, :, Nz+1)=  ExP(:,:,Nz) + m1*(ExN(:,:,Nz) - ExP(:,:,Nz+1));   %   top - Ex;
EyN(:, :, Nz+1)=  EyP(:,:,Nz) + m1*(EyN(:,:,Nz) - EyP(:,:,Nz+1));   %   top - Ex;
%-----------------------------------------------------------------------------------
```

Although these conditions on $\mathbf{E}$ are sufficient to solve our problem, we may also impose conditions on $\mathbf{H}$ and near the boundary which, depending on the polarization observed, will cancel some of the errors imposed on Mur's ABCs. Consider the 2D TM case where we have polarization in the z direction. The rightmost boundary conditions of the domain may be improved by imposing the same conditions on H_y as in eq. (7) and eq. (8) *one half step from* the boundary

$$H_{y\,N_x,m}^{n+\frac{1}{2}(2)} = H_{y\,N_x,m}^{n-\frac{1}{2}} + \frac{c_0\Delta t - \Delta x}{c_0\Delta t + \Delta x}\left(H_{y\,N_x-1,m}^{n+\frac{1}{2}} - E_{z\,N_x,m}^{n-\frac{1}{2}}\right)$$

Next, the H_y value is calculated near the boundary as it ordinarily would be, using the FDTD scheme, to obtain $H_{y\,N_x,m}^{n+\frac{1}{2}(1)}$

Then, the final updated H_y value on the boundary may be calculated as

$$H_{y\,N_x,m}^{n+\frac{1}{2}} = \frac{H_{y\,N_x,m}^{n+\frac{1}{2}(1)} + \rho H_{y\,N_x,m}^{n+\frac{1}{2}(2)}}{1+\rho}$$

$$\rho = \frac{c_0\Delta t}{\Delta}$$

Mei's super ABCs may be realized in 3D in the following way with MATLAB

```
%-------------------------------------------------------------------
HyP(1, :) = HyP(2, :) + m1*(HyN(2, :) - HyP(1, :)); % x = 0;
HyP(Nx, :) = HyP(Nx-1,:) + m1*(HyN(Nx-1,:) - HyP(Nx, :)); % x = Lx;
HxP(:, 1) = HxP(:, 2) + m1*(HxN(:, 2) - HxP(:, 1)); % y = 0;
HxP(:, Ny) = HxP(:,Ny-1) + m1*(HxN(:, Ny-1)- HxP(:, Ny)); % y = Ly;
% H(1) and H
HyN(1, :) = (rho*HyP(1, :) + HyN(1, :))/(1+rho); % x = 0;
HyN(Nx, :) = (rho*HyP(Nx,:) + HyN(Nx,:))/(1+rho); % x = Lx;
```

```
HxN(:, 1) = (rho*HxP(:, 1) + HxN(:, 1))/(1+rho); % y = 0;
HxN(:, Ny) = (rho*HxP(:,Ny) + HxN(:,Ny))/(1+rho); % y = Ly;
%-----------------------------------------------------------------
```

Simpler boundary conditions met when modeling microwave problems are interfaces between regions of different permittivity and permeability. The staggered grid may simply have the permittivity/permeability defined on the boundary, and the average value between the two values at the point which both media occupy the same space. This type of boundary, along with PEC may be simply defined as a lattice of constants since the FDTD equations are scalar multiples of these values.

Array Geometry Under Study and Near Field Structure

A 4x4 planar array of linearly-polarized square patch antennas spaced at λ/2 or less is chosen as shown in Fig. 9. The ground plane (or the reflecting plane) extends to approximately twice the array size. The large reflector size is important for accurate restoration results.

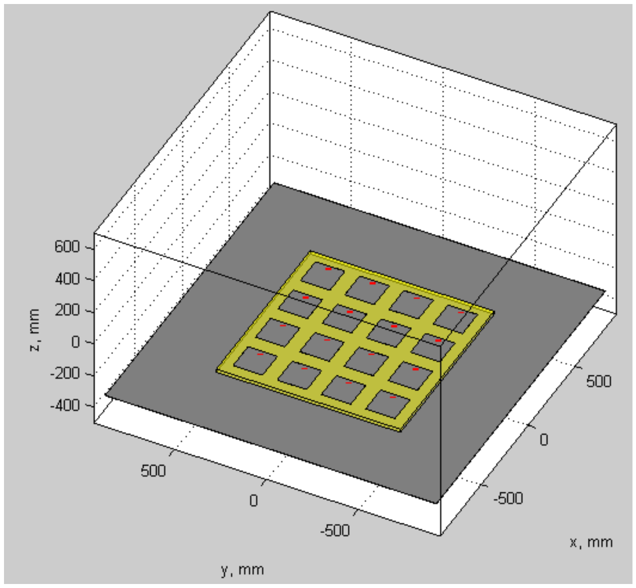


Figure 9 4x4 array geometry under study.

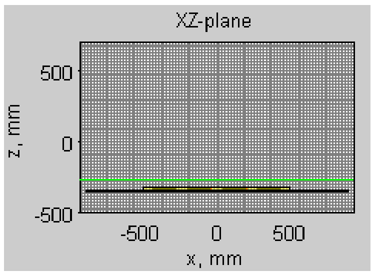
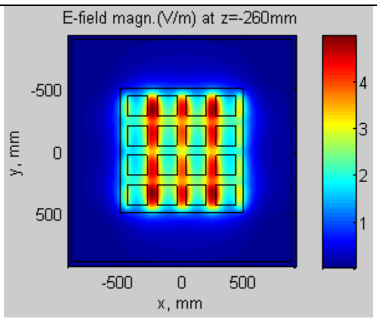
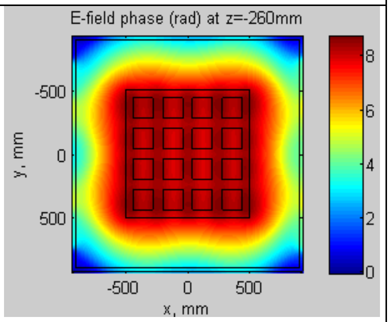
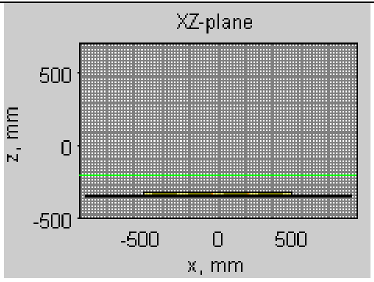
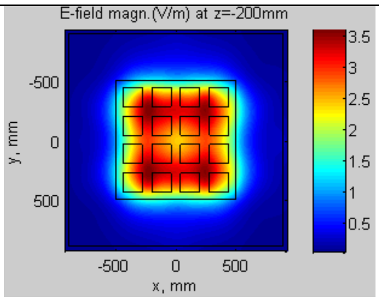
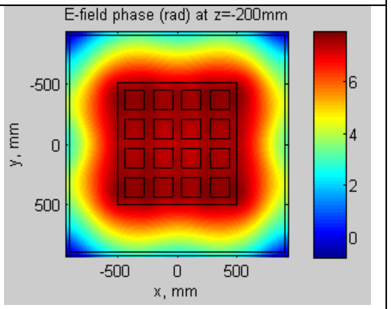
Six observation planes used for sampling transversal electric (and magnetic) fields are shown in Table 1. They are spaced at distances

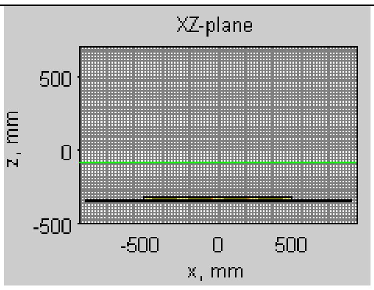
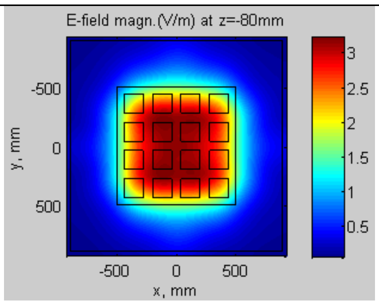
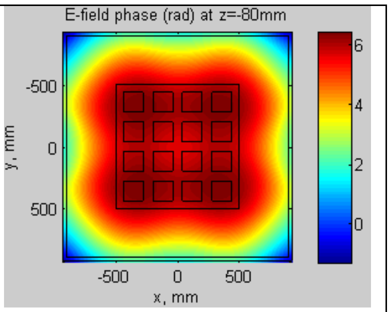
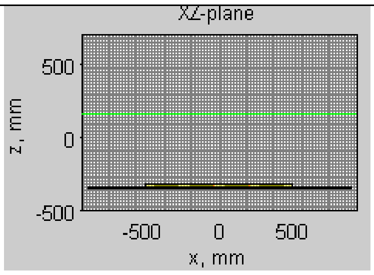
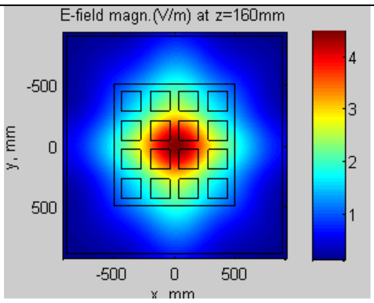
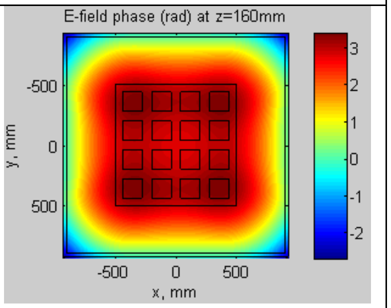
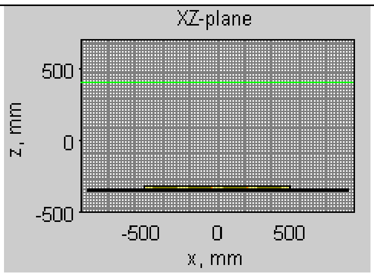
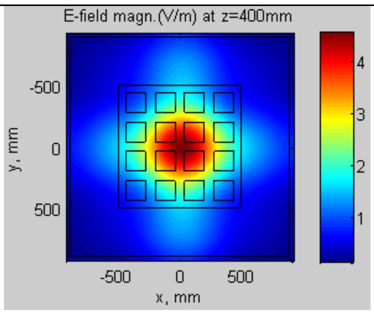
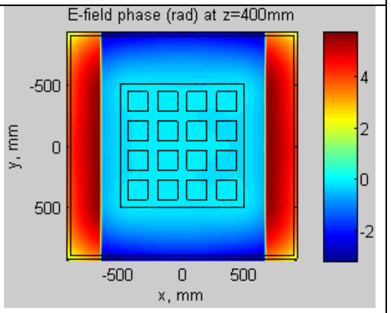
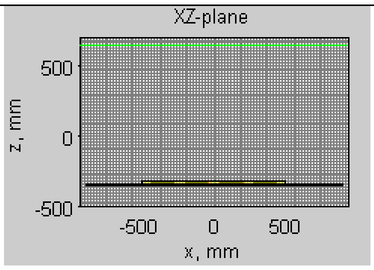
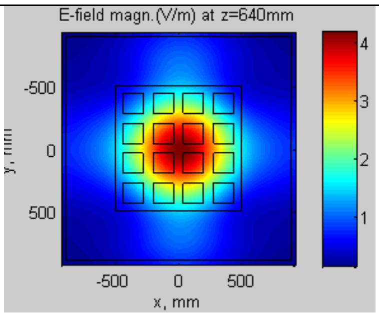
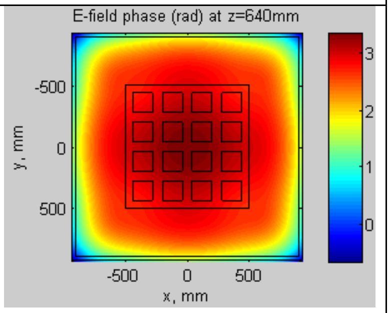
$$\frac{\lambda}{8},\quad \frac{\lambda}{4},\quad \frac{\lambda}{2},\quad \lambda,\quad \frac{3\lambda}{2},\quad 2\lambda \tag{9}$$

from the physical top of the antenna array. The array is simulated using the second-order Yee FDTD scheme and standard MATLAB® environment [26]. All radiators are terminated into an ideal sinusoidal generator voltage source in series with a 50Ω resistance.

Table 1 shows typical field distributions for the co-polar electric field at different distances from the array i.e. in the different observation planes. All elements have source amplitudes of 1V and equal phases.

Table 1 Electric field distributions in different observation planes (λ/2 spacing).

Plane height	Plane location vs. FDTD mesh	Co-polar electric field - magnitude	Co-polar electric field-phase
λ/8			
λ/4			

λ/2	 XZ-plane z, mm x, mm	 E-field magn.(V/m) at z=-80mm y, mm x, mm	 E-field phase (rad) at z=-80mm y, mm x, mm
λ	 XZ-plane z, mm x, mm	 E-field magn.(V/m) at z=160mm y, mm x, mm	 E-field phase (rad) at z=160mm y, mm x, mm
1.5λ	 XZ-plane z, mm x, mm	 E-field magn.(V/m) at z=400mm y, mm x, mm	 E-field phase (rad) at z=400mm y, mm x, mm
2.0λ	 XZ-plane z, mm x, mm	 E-field magn.(V/m) at z=640mm y, mm x, mm	 E-field phase (rad) at z=640mm y, mm x, mm

One can see that the fine structure of the fields close to individual patches is lost, as long as the distance from the array surface exceeds $\lambda/4$. However, the backpropagator will still be able to recover it, especially when the differential backpropagation is used.

Chapter 3: Near to Near/Far Field Transformations

The space surrounding an antenna is usually subdivided into three regions: the reactive near field; the radiating near-field, or "Fresnel region"; and the far field, or "Fraunhofer region". The boundaries between these regions are not unique and do not signify dramatic changes in the electromagnetic field occupying the regions; however, they provide a guideline of what portion of the field dominates in that region.

The reactive near field region is defined as "that portion of the near-field region immediately surrounding the antenna wherein the reactive field predominates." For most antennas, this region is described by the inequality

$$R < 0.62\sqrt{\frac{D^3}{\lambda}}$$

where λ is the wavelength and D is the largest dimension of the antenna.

The radiating near field region is defined as "that region of the field of an antenna between the reactive near field region and the far field region wherein radiation fields predominate and wherein the angular field distribution is dependent upon the distance from the antenna. If the antenna has a maximum dimension that is not large compared to the wavelength, this region may not exist. For an antenna focused at infinity, the radiating near field region is sometimes referred to as the Fresnel region on the basis of analogy to optical terminology. If the antenna has a maximum overall dimension which is very small compared to the wavelength, this field region may not exist." This region's boundary follows the inequality

$$0.62\sqrt{\frac{D^3}{\lambda}} < R < \frac{2D^2}{\lambda}$$

In this region, the fields' angular distribution changes considerably with distance from the antenna.

The far field region is defined as "that region of the field of an antenna where the angular friend distribution is essentially independent of the distance from the antenna. If the antenna has a maximum overall dimension D, the fat field region is commonly taken to exist at distances greater than $2D^2/\lambda$ from the antenna, λ being the wavelength. The far field patterns of certain antenna, such as multibeam reflector antennas, are sensitive to variations in phase over their apertures. For these antennas, $2D^2/\lambda$ may be inadequate. In physical media, if the antenna has a maximum overall dimension, D, which is large compared to $\pi/|\gamma|$, the far field region can be taken to begin approximately at a distance equal to $|\gamma|D^2/\pi$ from the antenna, $|\gamma|$ being the propagation constant in the medium. Fir an antenna focused at infinity, the far field region is sometimes referred to as the Fraunhofer region on the basis of analogy optical terminology." This region is defined by the inequality

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

In this domain, the fields' angular distribution is independent of the radial distance from the antenna. This is the field which governs the performance of an antenna, as its behavior is the product of design.

When the amplitude pattern of an antenna is measured, usually one (linear polarization) or two transverse (dual or circular polarization) components of $\mathbf{E}$ are taken. This may be done in the near or far zone; however, the near field range is a more accurate and reliable testing means. At this point, all measurements are assumed to be taken from the radiating near field.

An interesting problem to consider when taking measurements in the near field is how to produce the far field pattern with these measurements – after all, the antenna is designed for its far field characteristics. It turns out there are several methods of doing this when dealing in the sense of an aperture, that is to say, antennas which only radiate in the hemisphere covering them. This may be realized as a horn or as an antenna over a ground plane or parabolic reflector.

Near to Far Field Transformations

Fraunhofer Diffraction

One simple means of finding the far field pattern is by using the so called Fraunhofer diffraction equation. This is a very simple method of finding the field pattern at infinity using the same principles as studying diffraction of light through an aperture. Consider the diffraction of a wave in the z direction from the origin through an aperture onto another plane shown in Fig. 10

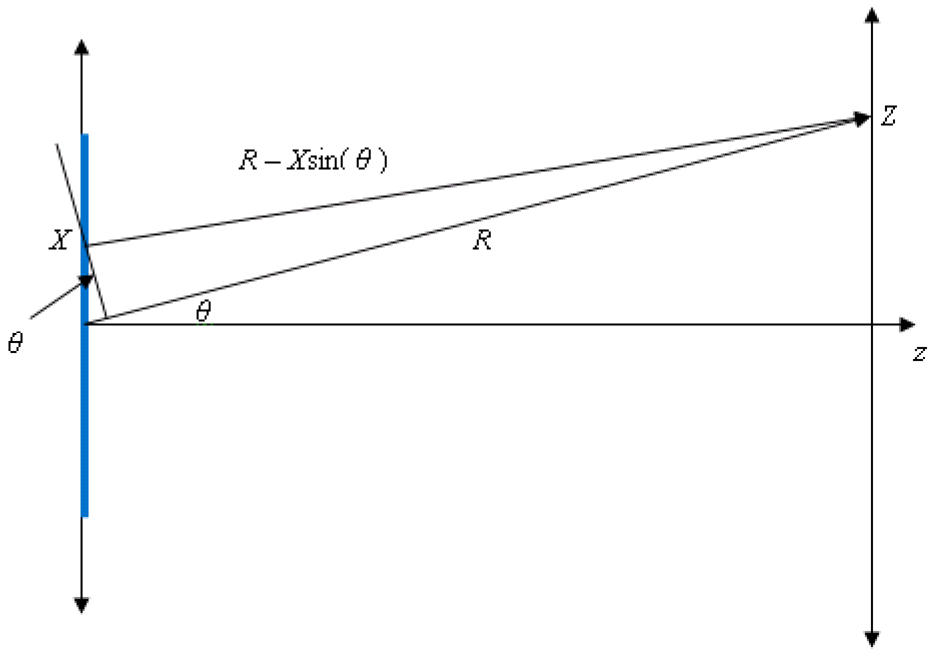


Figure 10 Geometry of Fraunhofer diffraction problem. The blue section is the aperture.

The electric field, of the point source in this case, takes the form

$$\mathbf{E}(r) = \hat{\mathbf{x}}\frac{E_0}{r}e^{-j(\omega t - kr)}$$

Incorporating every point source's effect on the plane implies an integral across the aperture, yielding the following expression

$$\mathbf{E}(r) = \int_{APERTURE} \hat{\mathbf{x}}\frac{E_0}{R - X\sin(\theta)}e^{-j(\omega t - k[R - X\sin(\theta)])}dX \tag{10}$$

For $R >> X$, we may make the approximation

$$\frac{1}{R - X\sin(\theta)} \approx \frac{1}{R}$$

Eq. (10) becomes

$$\mathbf{E}_\infty = \int_{-\infty}^{\infty} \hat{\mathbf{x}} A(X) \frac{E_0}{R} e^{-j\left(\omega t - k\left[R - X\sin(\theta)\right]\right)} dX$$

Where *A(X)* is the aperture distribution

Simply put, equation is derived by adjusting the phase of the E field according to the distance from the source to the surface of interest (in this case, infinity), and adjusting for the radial falloff of power of the field, thus

$$\mathbf{E}_\infty = \hat{\mathbf{x}} \frac{E_0}{R} e^{-j(\omega t - kR)} \int_{-\infty}^{\infty} A(X) e^{-jkX\sin(\theta)} dX$$

or

$$\mathbf{E}_\infty = C\Im\{\mathbf{E}_0\}$$

where C is some scaling constant (i.e. the result is proportional to the Fourier transform). This formula is mainly used as a quick way to find the normalized pattern, not absolute gain pattern.

Equivalent Magnetic Currents and MoM

For electrical engineers, the method of moments (MoM) may be the most reasonable means of calculating field quantities, as it deals in terms of currents, voltages, and impedances. In 1992, T. K. Sarkar and A. Taaghol [18]-[20] produced a simple yet effective method of calculating the far zone fields of an aperture using the MoM formulation. In order to present this idea, the field equivalence principle and inhomogeneous Helmholtz equations which follow from Maxwell's equations are used.

The following equations fully describe the behavior of electromagnetic fields in any medium.

Ampere's Law modified by displacement currents: $\varepsilon \frac{\partial \mathbf{E}}{\partial t} = \nabla \times \mathbf{H} - \mathbf{J}$ (a)

Faraday's Law: $\mu \frac{\partial \mathbf{H}}{\partial t} = -\nabla \times \mathbf{E}$ (b)

Gauss's Law for electric fields: $\nabla \cdot \varepsilon \mathbf{E} = \rho$ (c)

Gauss's Law for magnetic fields: $\nabla \cdot \mu \mathbf{H} = 0$ (d)

Continuity Equation: $\frac{\partial \rho}{\partial t} + \nabla \cdot \mathbf{J} = 0$ (e)

We would like to solve these equations for the driving sources, $\mathbf{J}$ and φ, which are easily applied by engineers, and, as such, are taken as givens along with the permittivity and permeability of the considered medium, and thus there are five equations with only two unknowns, $\mathbf{H}$ and $\mathbf{E}$, which we may solve for in terms of $\mathbf{J}$ and ρ.

The first step to take is to define the magnetic vector potential, $\mathbf{A}$, by recognizing that the curl of a vector field is divergence free in equation (d):

$$\nabla \cdot \mu \mathbf{H} = 0 \Rightarrow \nabla \cdot \mu \mathbf{H} = \nabla \times \mathbf{A} \tag{11}$$

Now, if we use equation (b) and plug in equation (11) for $\mu \mathbf{H}$ in a homogeneous medium (constant ε, μ), we get:

$$\mu \frac{\partial (\nabla \times \mathbf{A})}{\partial t} = -\nabla \times \mathbf{E}$$

$$\nabla \times \left(\mathbf{E} + \frac{\partial \mathbf{A}}{\partial t} \right) = 0$$

Since the curl of the divergence of a scalar field is zero, we may assume the solution takes the form of:

$$\mathbf{E} = -\nabla \varphi - \frac{\partial \mathbf{A}}{\partial t} \tag{12}$$

Now we have both unknown fields expressed through two potentials which we have yet to solve for in terms of their sources.

Taking the curl of both sides of (11), we get:

$$\mu\nabla\times\mathbf{H} = \nabla\times\nabla\times\mathbf{A} = \nabla(\nabla\cdot\mathbf{A}) - \nabla^2\mathbf{A}$$

plugging this into (a), we get:

$$\varepsilon\mu\frac{\partial\mathbf{E}}{\partial \mathrm{t}} = \nabla(\nabla\cdot\mathbf{A}) - \nabla^2\mathbf{A} - \mu\mathbf{J}$$

substituting in (12) gives us:

$$\nabla^2\mathbf{A} - \varepsilon\mu\frac{\partial^2\mathbf{A}}{\partial \mathrm{t}^2} = -\mu\mathbf{J} + \left[\nabla(\nabla\cdot\mathbf{A}) + \varepsilon\mu\nabla\frac{\partial\varphi}{\partial t}\right]$$

$$\nabla^2\mathbf{A} - \varepsilon\mu\frac{\partial^2\mathbf{A}}{\partial \mathrm{t}^2} = -\mu\mathbf{J} + \nabla\left[(\nabla\cdot\mathbf{A}) + \varepsilon\mu\frac{\partial\varphi}{\partial t}\right]$$

Leading us to the Lorentz gauge (since φ has yet to be defined):

$$\nabla\cdot\mathbf{A} + \varepsilon\mu\frac{\partial\varphi}{\partial t} = 0 \Rightarrow \frac{\partial\varphi}{\partial t} = -\frac{1}{\varepsilon\mu}\nabla\cdot\mathbf{A}$$

So we are left with:

$$\nabla^2\mathbf{A} - \varepsilon\mu\frac{\partial^2\mathbf{A}}{\partial t^2} = -\mu\mathbf{J} \tag{13}$$

$$\frac{\partial\varphi}{\partial t} = -\frac{1}{\varepsilon\mu}\nabla\cdot\mathbf{A} \tag{14}$$

leaving us only to solve for $\mathbf{A}$ in order to solve for φ.

In order to solve eq. (13), we will note that eq. (13) is indeed linear and may be expressed in phasor form when considering a time harmonic field. This may be done by introducing time dependence in the form of $e^{j\omega t}$, turning time derivatives into multiplication by $j\omega$, allowing us to then cancel the exponential terms and leave us with no time derivatives:

$$\nabla^2\mathbf{A} - k^2\mathbf{A} = -\mu\mathbf{J}, \quad k = \sqrt{\varepsilon\mu}\,\omega, \quad \text{or} \quad k = \frac{\omega}{c} \quad \text{for free space} \tag{15a}$$

$$\varphi = -\frac{1}{j\omega\varepsilon\mu}\nabla\cdot\mathbf{A} \tag{15b}$$

$$\mu\mathbf{H} = \nabla\times\mathbf{A} \tag{15c}$$

$$\mathbf{E} = -\nabla\varphi - j\omega\mathbf{A} \tag{15d}$$

In order to avoid taking two gradients of the magnetic vector potential, a numerical inconvenience, we can use the continuity equation, (e), in phasor form:

$$j\omega\rho + \nabla\cdot\mathbf{J} = 0 \tag{15e}$$

Taking the gradient of both sides of (14a) gives us:

$$\nabla^2(\nabla\cdot\mathbf{A}) - k^2(\nabla\cdot\mathbf{A}) = -\mu(\nabla\cdot\mathbf{J})$$

Using equations (14b) and (14e), we get:

$$\nabla^2\varphi - k^2\varphi = -\frac{\rho}{\varepsilon} \tag{16}$$

Thus, we are left with the following equations:

$$\nabla^2\mathbf{A} - k^2\mathbf{A} = -\mu\mathbf{J}$$

$$\nabla^2\varphi - k^2\varphi = -\frac{\rho}{\varepsilon}$$

$$j\omega\rho + \nabla\cdot\mathbf{J} = 0 \tag{17}$$

$$\mathbf{H} = \frac{1}{\mu}\nabla \times \mathbf{A}$$

$$\mathbf{E} = -\nabla\varphi - j\omega\mathbf{A}$$

The solutions to eqs. (17) may be expressed implicitly in the form of volume integrals using Green's functions, or the fundamental solution to the Helmholtz equation in an unbounded space. This solution is valid for any frequency:

$$\mathbf{A}(\mathbf{r}) = \mu \int_{\mathbf{R}^3} \mathbf{g}(\mathbf{r},\mathbf{r}')\mathbf{J}(\mathbf{r}')d\mathbf{r}$$

$$\varphi(\mathbf{r}) = \frac{1}{\varepsilon} \int_{\mathbf{R}^3} \mathbf{g}(\mathbf{r},\mathbf{r}')\rho(\mathbf{r}')d\mathbf{r}$$

where

$$g(\mathbf{r},\mathbf{r}') = \frac{e^{-jk_0|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|}$$

Similarly, if we come up with an imaginary source for the magnetic field, called the "magnetic current", $\mathbf{M}$, and we assume that $\mathbf{J} = \mathbf{0}$, we have $\nabla \cdot \mathbf{D} = 0$. This implies that the electric field is the curl of a different field potential. Thus, we may come up with an electric field potential, $\mathbf{F}$, and define it in the following way.

$$\mathbf{E} = \frac{1}{\varepsilon}\nabla \times \mathbf{F} \tag{18}$$

$$\mathbf{F}(\mathbf{r}) = \varepsilon \int_{\mathbf{R}^3} \mathbf{g}(\mathbf{r},\mathbf{r}')\mathbf{M}(\mathbf{r}')d\mathbf{r} \tag{19}$$

The field equivalence principle is a more rigorous formulation of the Huygens principle and is based on the uniqueness theorem that states "a field in a lossy region is uniquely specified by the sources within the region plus the tangential components of the electric field over the boundary, or the tangential components of the magnetic field over the boundary, or the former over the boundary and the latter over the rest of the boundary." [3] Using the equivalency principle, the fields outside an imaginary closed surface are obtained by placing suitable electric- and magnetic-current densities over the closed surface which satisfy the boundary conditions. The current densities may be selected such that the fields within the surface are zero and are equal to the radiation produced by the sources on the outside; thus the field equivalence principle may be used to obtain the fields radiated outside a closed surface by sources enclosed within it.

For example, if the region S is chosen with fields $\mathbf{E}_1$ and $\mathbf{H}_1$ enclosed, we may select the sources on the boundary to be $\mathbf{M}_s = -\hat{\mathbf{n}} \times \mathbf{E}_1$ and $\mathbf{J}_s = \hat{\mathbf{n}} \times \mathbf{H}_1$. Due to the uniqueness of Maxwell's equations, if the electric field is known, we may find the magnetic field; this means that we may choose the surface S to be made of a perfect electric conductor, leading us to the conditions $\mathbf{M}_s = -\hat{\mathbf{n}} \times \mathbf{E}_1$ and $\mathbf{J}_s = \hat{\mathbf{n}} \times \mathbf{H}_1 = 0$.

Now if we take an antenna and enclose it behind an infinite plane and define the magnetic current near an infinite conducting plane, using the method of images, we have the equivalent magnetic current $\mathbf{M} = -2\hat{\mathbf{n}} \times \mathbf{E}_1$. If E is measured, we may use it to define $\mathbf{M}$. Remembering that the new field $\mathbf{E}$ is equal to zero on the boundary of S, we can use eqs. (18)and (19) to find that

$$\mathbf{E}(\mathbf{r}) = -\iint_S \mathbf{M}(\mathbf{r}') \times \nabla' g(\mathbf{r},\mathbf{r}') ds'$$

where ∇' denotes the gradient operator with respect to the primed variable, and the Green's function is defined as

$$g(\mathbf{r},\mathbf{r}') = \frac{e^{-jk_0|\mathbf{r}-\mathbf{r}'|}}{4\pi|\mathbf{r}-\mathbf{r}'|}$$

This produces the two following sets of equations for both components of the electric field

$$E_x(\mathbf{r}) = -\iint_S \frac{\partial g(\mathbf{r},\mathbf{r}')}{\partial z'} M_y(\mathbf{r}') ds'$$

$$E_y(\mathbf{r}) = \iint_S \frac{\partial g(\mathbf{r},\mathbf{r}')}{\partial z'} M_x(\mathbf{r}')ds'$$

These equations are valid for finding the electric field at any distance from an aperture.

Rayleigh Diffraction Integral

Another means of propagating a near-field result to the far field is done by another optics tool known as the Rayleigh diffraction formula. Its meaning is as follows.

Assume the quantity V is a monochromatic (single frequency) scalar wave field in free space, i.e.

$$V = U(x,y,z)e^{j\omega t}$$

We also assume V satisfies the homogeneous (propagating) wave equation throughout the region of interest, and as such, satisfies the Helmholtz equation $\left(\nabla^2 + k^2\right)U = 0$. Generally, we may express such a quantity in the form of an angular spectrum of plane waves

$$U(x,y,z) = \iint_{\mathbf{R}^2} A(p,q)\exp\left(-jk\left[px + qy + mz\right]\right)dpdq$$

where

$$m = \sqrt{1 - p^2 - q^2} \quad \text{when } p^2 + q^2 \le 1$$

$$m = j\sqrt{p^2 + q^2 - 1} \text{ when } p^2 + q^2 > 1$$

The former value of m corresponds to homogeneous (propagating) waves, while the latter corresponds to evanescent waves which do not propagate, but decay exponentially to zero as the distance from the source becomes larger. If we consider U at the points $\left(x_0, y_0, z_0\right)$ and $\left(x_1, y_1, z_1\right)$ (Denoted by $U_k(x_k, y_k)$), we see that apart from scaling and proportionality constants, U is the two dimensional Fourier transform of the function $B_{1,2}(p,q) = A(p,q)\exp\left(jkmz_{1,2}\right)$, thus

$$A(p,q) = \left(\frac{k}{2\pi}\right)^2 \exp(jkmz_l)\iint_{\mathbf{R}^2} U_l(x_l, y_l)\exp(jk[px_l + qy_l])dx_l dy_l \text{ , } (l = 0,1)$$

This gives us

$$U_i(x_i, y_i) = \left(\frac{k}{2\pi}\right)^2 \iint_{\mathbf{R}^2} dpdq \exp(-jk[px_i + qy_i + mz_i]) \times \iint_{\mathbf{R}^2} dx_l dy_l U_l(x_l, y_l)\exp(jk[px_l + qy_l + mz_l])$$

If we choose $k = 1, l = 0$, we are able to propagate the wave at point (x_0, y_0, z_0) to (x_1, y_1, z_1). Changing the order of integration, we may define a function

$$K_{il}(x_i - x_l, y_i - y_l) \equiv K_{il}(x_i - x_l, y_i - y_l, z_i - z_l)$$
$$= \left(\frac{k}{2\pi}\right)^2 \iint_{\mathbf{R}^2} \exp(-jk[p(x_i - x_l) + q(y_i - y_l) + m(z_i - z_l)])dpdq$$

And therefore,

$$U_1(x_1, y_1) = \iint_{\mathbf{R}^2} U_0(x_0, y_0)K_{10}(x_1 - x_0, y_1 - y_0)dx_0 dy_0 \tag{20}$$

Using the plane wave expansion of a spherical wave and noting that $z_1 > z_0$,

$$\frac{\exp(-jk|\mathbf{r} - \mathbf{r}_0|)}{|\mathbf{r} - \mathbf{r}_0|} = -\frac{jk}{2\pi}\iint_{\mathbf{R}^2} \frac{\exp(-jk[p(x_1 - x_0) + q(y_1 - y_0) + m|z_1 - z_0|])}{m} dpdq$$

we may recognize that $K_{10}(x_1 - x_0, y_1 - y_0) = \frac{1}{2\pi}\frac{\partial}{\partial z_0}\left[\frac{\exp(-jk|\mathbf{r} - \mathbf{r}_0|)}{|\mathbf{r} - \mathbf{r}_0|}\right]$, so

$$U_1(x_1, y_1) = \frac{1}{2\pi}\iint_{\mathbf{R}^2} U_0(x_0, y_0)\frac{\partial}{\partial z_0}\left[\frac{\exp(-jk|\mathbf{r} - \mathbf{r}_0|)}{|\mathbf{r} - \mathbf{r}_0|}\right]dx_0 dy_0 \tag{21}$$

Eq. (21) is known as the Rayleigh diffraction formula, expressing eq. (20) in closed form.

Near to Nearer Field Transformations

All of the above equations are valid for transforming a near field result to the far field; however, in practice, it is also useful to solve the inverse problem, i.e. transform far/near field results back to the aperture plane (the face of the antenna). This back-transformed quantity is known as the *hologram* of the antenna. The hologram is especially needed when dealing with antenna arrays containing a multitude of elements, since all elements must be weighted properly. Unfortunately, none of these methods are completely accurate in solving the inverse problem.

We will start with the obvious Fraunhofer diffraction. The approximation made during the Fraunhofer diffraction derivation is that the measurement plane is located at infinity, this would allow us to convert far field results into the hologram; however, these results will still not be entirely accurate even excluding the proportionality constant mentioned above. To understand why the Fraunhofer diffraction equation should mainly be used as an approximation, we must first note the fundamental quantity being observed in the derivation: the Fraunhofer diffraction formula is derived solely from the notion of homogeneous waves, that is, those waves which are travelling in space. However, we may note that all antennas have a reactive near field in which the dominant energy is composed of inhomogeneous, or evanescent, waves that decay exponentially with distance from the aperture. Therefore, this quantity is ignored altogether and not reconstructed in the hologram.

Similarly, Narishman and Kumar [24] developed a method of calculating the hologram of an array by recognizing that the array pattern is nothing but the appropriately displaced sum of individually weighted element electric field patterns, i.e.

$$E_x(x, y, z_0) = \sum_{m=0}^{M-1} a_m E_{x_0}(x - x'_m; y - y'_m; z_0)$$

Taking the Fourier transform of both sides of the equation allows for the modeling of displacement as a phase shift in the frequency domain

$$\overline{E}_x(u, v) = \sum_{m=0}^{M-1} a_m \overline{E}_{x_0}(u, v) e^{-j(ux'_m + vy'_m)} = \overline{E}_{x_0}(u, v)[\Im\{J_x(x', y')\}]$$

As such, the hologram in this case is the inverse Fourier transform of the array far field pattern divided by the element pattern. This method fails to take into account the fact that mutual coupling creates a non-uniformity of element patterns in the array. This is the same issue that the array factor runs into when computing the array radiation pattern. Fig. 11 shows an example of how the antenna patterns of individual array elements can be different. A 4x4 array was simulated in Ansys HFSS. The top left image shows the active element pattern of one of the corner elements; the top right image shows the active element pattern of one of the inner elements; the bottom image shows the active element pattern of one of the edge elements. Note that these patterns are not the same shape as one another.

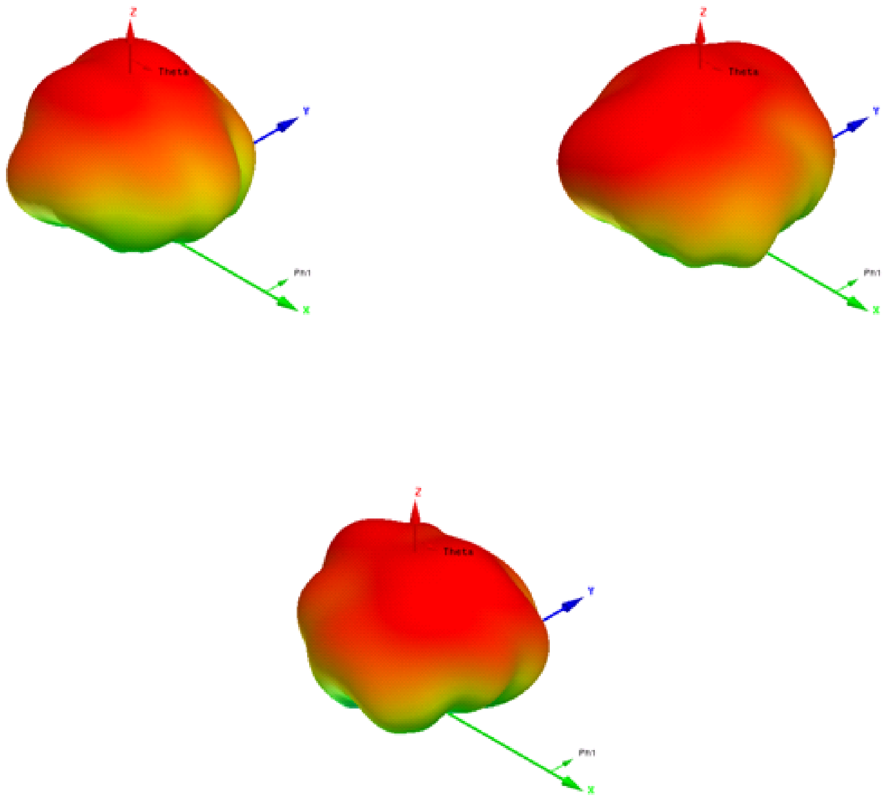


Figure 11 Example of non-uniform amplitude pattern of individual elements in a 4x4 array of patch antennas. The top left is a corner element, the top right is an inner element, and the bottom is an edge element.

This method also does not show the electric field distribution on the ground plane, but only the weight of each element, which may be desirable in some applications.

Since inverse Fraunhofer diffraction is not the ideal choice for calculating the hologram, we will move on to Sarkar and Taaghol's equivalent magnetic current approach. It can be seen that this method is inherently the same as the Rayleigh diffraction formula, and so we can observe the two together. Using this method may be shown to be valid for backpropagating near field results to the aperture plane; however, in 1968, Shewell and Wolf [14] showed that the inverse transform provided by this method is invalid for evanescent modes, and therefore we may say that it has a similar effect as the inverse Fraunhofer diffraction on results. Namely, Shewell and Wolf showed that the inverse Rayleigh diffraction formula is expressed as

$$U_1(x_1, y_1) = \frac{1}{2\pi} \iint_{\mathbf{R}^2} U_0(x_0, y_0) K_{01}(x_0 - x_1, y_0 - y_1) dx_0 dy_0$$

$$K_{01}(x_0 - x_1, y_0 - y_1) = \frac{\partial}{\partial z_0}\left[\frac{\exp(jk|\mathbf{r}-\mathbf{r}_0|)}{|\mathbf{r}-\mathbf{r}_0|}\right] - \frac{k^2}{\pi}\int_1^\infty \sinh\left[k(\rho^2-1)^{\frac{1}{2}}(z_1 - z_0)\right] J_0(k\sigma\rho)\rho d\rho \cdot$$

As such, this equation may be used to solve the inverse diffraction problem and obtain the hologram including the evanescent modes; however, the following means is an equivalent way of doing so, and is also derived as an exact solution to Maxwell's equations.

In the source free region, beyond the aperture plane, the field $\mathbf{E}$ of a monochromatic wave radiated by the aperture may be can be written as the superposition of plane waves in the form of a Fourier transform

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

Where k_x and k_y are the spectral frequencies which extend over the entire frequency spectrum $-\infty < k_x, k_y < \infty$, and $\mathbf{f}(k_x, k_y)$ is the vector amplitude of each plane wave. Since

$\mathbf{r} = \hat{\mathbf{a}}_x x + \hat{\mathbf{a}}_y y + \hat{\mathbf{a}}_z z$ and $\mathbf{k} = \hat{\mathbf{a}}_x k_x + \hat{\mathbf{a}}_y k_y + \hat{\mathbf{a}}_z k_z$,

FT can be written as

$$\mathbf{E}(x,y,z) = \frac{1}{4\pi^2} \iint_{\mathbf{R}^2} \left[\mathbf{f}(k_x,k_y)e^{-jk_z z}\right] e^{-j(k_x x + k_y y)} dk_x dk_y$$

But we can regard the portion in brackets as the transform of $\mathbf{E}$, thus

$$\mathbf{E}(x,y,z) = \frac{1}{4\pi^2} \iint_{\mathbf{R}^2} \widetilde{\mathbf{E}}(k_x,k_y,z) e^{-j(k_x x + k_y y)} dk_x dk_y$$

$$\widetilde{\mathbf{E}}(k_x,k_y,z) = \frac{1}{4\pi^2} \iint_{\mathbf{R}^2} \mathbf{E}(x,y,z) e^{+j(k_x x + k_y y)} dx dy$$

where

$$\widetilde{\mathbf{E}}(k_x,k_y,z) = \mathbf{f}(k_x,k_y) e^{-jk_z z}$$

This means we may find the field at any point z, provided we know its transform; however, in order to have the transform, we must know the electric field. We may end this dilemma by noticing that if we know the electric field at point zero

$$\widetilde{\mathbf{E}}(k_x,k_y,0) = \mathbf{f}(k_x,k_y),$$

We may use this to calculate the field at any point, which is exactly what we desire, but we must first do a bit more math. In general,

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

which can be expressed as

$$\mathbf{f}(k_x,k_y) = \mathbf{f}_t(k_x,k_y) + \hat{\mathbf{a}}_z f_z(k_x,k_y),$$

$$\mathbf{f}(k_x,k_y) = \hat{\mathbf{a}}_x f_x(k_x,k_y) + \hat{\mathbf{a}}_y f_y(k_x,k_y).$$

In order for $\mathbf{E}$ to satisfy the homogeneous wave equation in the source free region, the quantity k_z must be related to k_x and k_y in the following way

$$k_z^2 = k^2 - \left(k_x^2 + k_y^2\right).$$

or

$$k_z = \sqrt{k^2 - \left(k_x^2 + k_y^2\right)} \qquad \text{when} \quad k^2 \le k_x^2 + k_y^2$$

$$k_z = -j\sqrt{\left(k_x^2 + k_y^2\right) - k^2} \qquad \text{when} \quad k^2 > k_x^2 + k_y^2$$

where the two cases for k contribute to both homogeneous (former) and inhomogeneous (latter) waves.

Since $\mathbf{E}$ must obey Maxwell's equations in the source free region, we have

$$\nabla \cdot \mathbf{E}(x, y, z) = \nabla \cdot \left\{ \frac{1}{4\pi^2} \iint_{\mathbf{R}^2} \mathbf{f}\left(k_x, k_y\right) e^{-j\mathbf{k}\cdot\mathbf{r}} dk_x dk_y \right\} = 0$$

Using the chain rule, we have

$$\nabla \cdot \mathbf{E}(x, y, z) = \frac{1}{4\pi^2} \iint_{\mathbf{R}^2} \mathbf{f}\left(k_x, k_y\right) \cdot \nabla\left(e^{-j\mathbf{k}\cdot\mathbf{r}}\right) dk_x dk_y = 0$$

Since $\nabla \mathbf{f}\left(k_x, k_y\right) = \mathbf{0}$, as $\mathbf{E}$ is the superposition of $\mathbf{f}$

This equation may be satisfied provided that

$$\mathbf{f} \cdot \nabla\left(e^{-j\mathbf{k}\cdot\mathbf{r}}\right) = -j\mathbf{f} \cdot \mathbf{k} e^{-j\mathbf{k}\cdot\mathbf{r}} = 0$$

or

$$\mathbf{f} \cdot \mathbf{k} = \left(\mathbf{f}_t + \hat{\mathbf{a}}_z f_z\right) \cdot \mathbf{k} = 0$$

or

$$f_z = \frac{\mathbf{f}_t \cdot \mathbf{k}}{k_z} = -\frac{\left(f_x k_x + f_y k_y\right)}{k_z}$$

This means we can also find f_z provided we know the tangential components of $\mathbf{E}$ on some plane at distance z from the aperture.

This gives us our formulas to find the hologram given $E_x(x, y, z = 0)$ and $E_y(x, y, z = 0)$.

$$f_x(k_x, k_y) = \iint\limits_{APERTURE} E_x(x', y', z' = 0) e^{+j(k_x x' + k_y y')} dx'dy'$$

$$f_y(k_x, k_y) = \iint\limits_{APERTURE} E_y(x', y', z' = 0) e^{+j(k_x x' + k_y y')} dx'dy'$$

$$\mathbf{E}(x, y, z) =$$

$$\frac{1}{4\pi^2} \iint\limits_{\substack{k^2 \le k_x^2 + k_y^2 \\ k_z = \sqrt{k^2 - (k_x^2 + k_y^2)}}} \widetilde{\mathbf{E}}(k_x, k_y, z) e^{-j(k_x x + k_y y)} dk_x dk_y + \iint\limits_{\substack{k^2 > k_x^2 + k_y^2 \\ k_z = -j\sqrt{(k_x^2 + k_y^2) - k^2}}} \widetilde{\mathbf{E}}(k_x, k_y, z) e^{-j(k_x x + k_y y)} dk_x dk_y$$

$$\widetilde{\mathbf{E}}(k_x, k_y, z) = \left[\hat{\mathbf{a}}_x f_x(k_x, k_y) + \hat{\mathbf{a}}_y f_y(k_x, k_y) - \hat{\mathbf{a}}_z \left(\frac{f_x k_x + f_y k_y}{k_z} \right) \right] e^{-jk_z z}$$

Therefore, if we know the field at some point z_0, we simply phase the Fourier transform by $-z_0$ and we will find our hologram with all variables considered. This is an equivalent result to the four-fold integral simplified in the derivation of the Rayleigh diffraction formula. As mentioned above, the diffraction formula throws out evanescent modes in order to express the integral in closed form. An example of the implementation of this algorithm is shown in Appendix E.

Chapter 4: Idea of Differential Backpropagation

In this section, we present the standard direct and inverse propagator for horizontal observation planes based on Fourier transform in the *k*-space [13]-[16]. Note that other propagator models in the near field exist [17]-[25]. Only tangential electric fields E_x, E_y are included into considerations. The spatial Fourier transform over a finite plane aperture (*a*×*b*) reads

$$\begin{aligned} f_x &= \int_{-b/2}^{+b/2}\int_{-a/2}^{+a/2} E_x(x',y',z'=0)\exp\left(jk_x x' + jk_y y'\right)dx'dy' \\ f_y &= \int_{-b/2}^{+b/2}\int_{-a/2}^{+a/2} E_y(x',y',z'=0)\exp\left(jk_x x' + jk_y y'\right)dx'dy' \end{aligned} \tag{22}$$

Direct and inverse propagators (in the *z*-direction) have the form

$$\begin{aligned} E_x(x,y,z) = {} & \frac{1}{4\pi^2}\int_{k_x^2+k_y^2\le k^2} f_x(k_x,k_y)\exp\left(-jk_x x - jk_y y - j\sqrt{k^2 - k_x^2 + k_y^2}\,z\right)dk_x dk_y + \\ & \frac{1}{4\pi^2}\int_{k_x^2+k_y^2> k^2} f_x(k_x,k_y)\exp\left(-jk_x x - jk_y y - \sqrt{k_x^2 + k_y^2 - k^2}\,z\right)dk_x dk_y \\ E_y(x,y,z) = {} & \frac{1}{4\pi^2}\int_{k_x^2+k_y^2\le k^2} f_y(k_x,k_y)\exp\left(-jk_x x - jk_y y - j\sqrt{k^2 - k_x^2 + k_y^2}\,z\right)dk_x dk_y + \\ & \frac{1}{4\pi^2}\int_{k_x^2+k_y^2> k^2} f_y(k_x,k_y)\exp\left(-jk_x x - jk_y y - \sqrt{k_x^2 + k_y^2 - k^2}\,z\right)dk_x dk_y \end{aligned} \tag{23}$$

Positive values of *z* correspond to forward propagation; negative *z* – to back-propagation.

Dimensionless variables may further be introduced to quantify the effect of evanescent modes corresponding to $k_x^2 + k_y^2 > k$

$$\begin{aligned} & X = \frac{x}{\lambda}, Y = \frac{x}{\lambda}, Z = \frac{z}{\lambda}, \quad A = \frac{a}{\lambda}, B = \frac{b}{\lambda} \\ & K_x = \frac{k_x}{k}, K_y = \frac{k_y}{k}, k = \frac{2\pi}{\lambda} \end{aligned}$$

In the dimensionless form, the propagator reads

$$
\begin{aligned}
&f_{x,y} = \lambda^2 \int_{-B/2}^{+B/2}\int_{-A/2}^{+A/2} E_{x,y}(x',y',z'=0)\exp\left(2\pi jK_xX' + 2\pi jK_yY'\right)dX'dY' \\
&E_{x,y}(X,Y,Z) = \\
&\frac{1}{\lambda^2}\int_{K_x^2+K_y^2\le 1} f_{x,y}(k_x,k_y)\exp\left(-2\pi jK_xX - 2\pi jK_yY - 2\pi j\sqrt{1-K_x^2+K_y^2}\,Z\right)dK_xdK_y + \\
&\frac{1}{\lambda^2}\int_{K_x^2+K_y^2> 1} f_{x,y}(k_x,k_y)\exp\left(-2\pi jK_xX - 2\pi jK_yY - 2\pi\sqrt{K_x^2+K_y^2-1}\,Z\right)dK_xdK_y
\end{aligned}
\tag{24}
$$

Except for truncating the observation plane, direct and inverse Fourier propagators (constituting the angular spectrum method) satisfy Maxwell's equation precisely. The direct propagator is equivalent to Rayleigh (Rayleigh-Sommerfeld) diffraction integral as shown in the last section.

One reason for the resolution degradation in Table 1 is that the evanescent modes responsible for the fine distribution of the fields close to individual radiators decay very fast. However, a faulty element in an array contributes not only to evanescent modes, but also to propagating modes in Eq. (23). Its presence considerably changes the propagating part of the spatial Fourier spectrum of an antenna array as compared to the ideal case of a non-faulty array. Therefore, the precise location of such an element in an array, could in principle, be restored from the near field data computed or measured at distances λ-2λ from the aperture plane. The necessary condition for this operation is the availability of *two phase-synchronized field distributions*: that of the non faulty array and that of a faulty array with one (or more) malfunctioning elements.

Fig. 12 shows three numerically computed spatial Fourier spectra (spectrum magnitudes) in k_x, k_y space for the array from Fig. 9. All magnitude spectra are obtained from the co-polar electric field at a distance of λ/8 from the aperture plane. The first plot (top left) is the spectrum of a non faulty array with all radiators driven by identical generators; the second plot (top right) is the spectrum of a faulty array when the generator for radiator 22 is shorted out; the third plot (bottom) is the difference spectrum between first two, which shall be used for the identification of a faulty element.

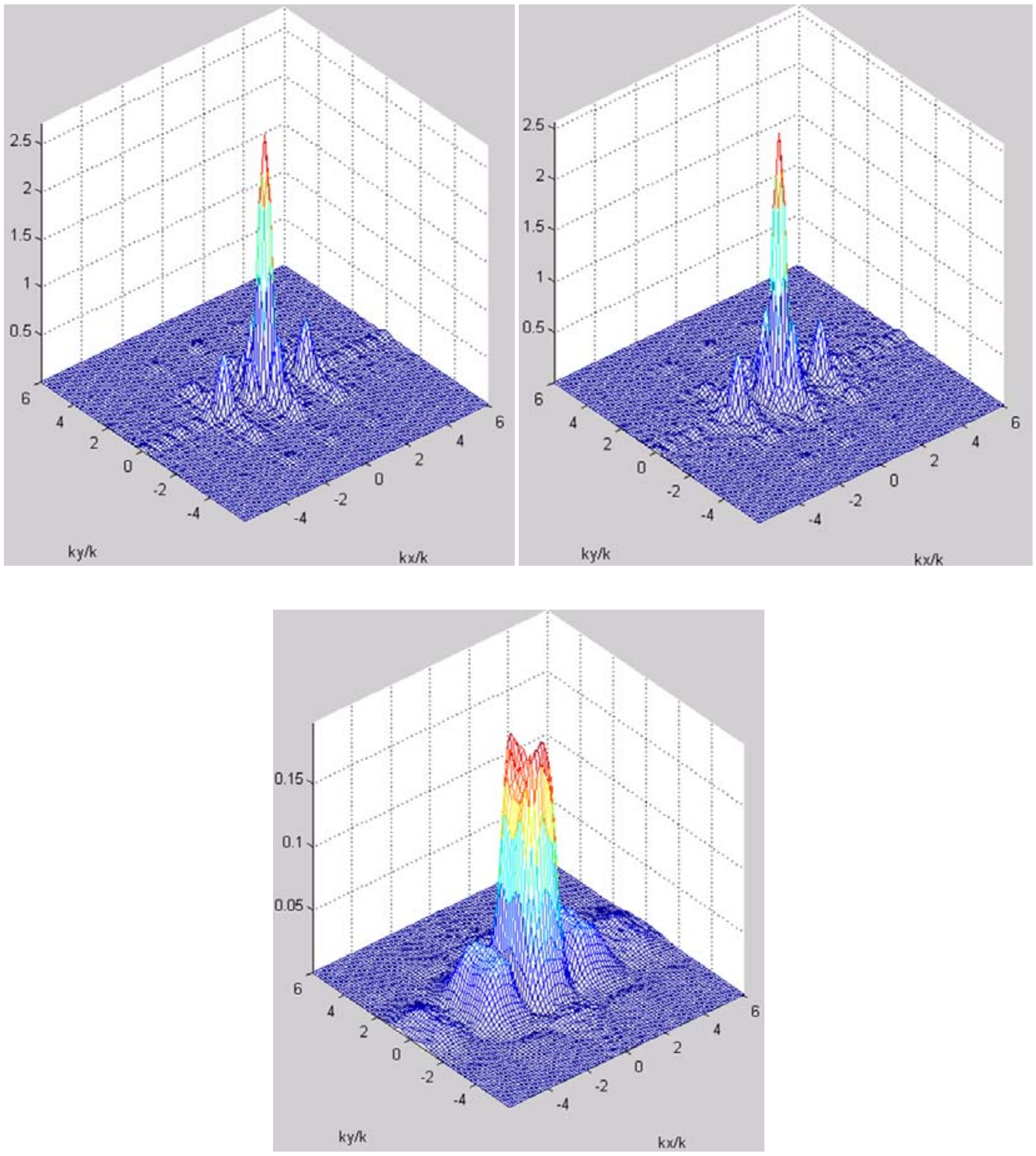


Figure 12 Three numerically computed spatial Fourier spectra (spectrum magnitudes) in *k*-space for a 4×4 array of λ/2 spaced patch antennas over a larger ground plane/reflector. All magnitude spectra are obtained for the co-polar electric field at the distance of λ/8 from the aperture plane. The observation plane is approximately twice as large as the array itself . The first plot (top left) is the spectrum for the non faulty array with all radiators driven by identical generators; the second plot (top right) is the spectrum for a faulty array where the generator for radiator 22 is shorted out; the third plot (bottom) is the difference spectrum between first two, which shall be used for the identification of a faulty element.

One can see from Fig. 12 that the difference spectrum carriers little power compared to the original array spectra. However, it clearly has a dominant power density peak within the unit circle,

$$k_x^2 + k_y^2 \leq k \tag{25}$$

which corresponds to *propagating* modes (plane waves). Whereas the evanescent part of the difference spectrum at $k_x^2 + k_y^2 > k$ will be quickly lost, the propagating part still remains at larger distances from the array. Therefore, it may potentially be recovered to determine the geometric location of a faulty element. It is thus suggested:

1. Form a difference between the co-polar fields of non faulty and faulty arrays at a distance of $\sim 2\lambda$ from the aperture plane.
2. Find the spatial Fourier spectrum of that difference and propagate it back to the array plane. The result is the error field magnitude, $|F_z|$.
3. The error field magnitude, $|F_z|$, presumably peaks at faulty elements (both amplitude and phase excitation fault).

Algorithm

Consider first a non-differential backpropagation. The backpropagation algorithm has two important parameters: window size in *k*-space and the roll-off coefficient of the raised cosine filter, *r*, applied to the restored field. The window in *k*-space must be applied in order to minimize errors caused by the exponential terms in the evanescent modes of the array. Table 2 lists the optimum values for these parameters found from the non-differential backpropagation.

The goal was to minimize the error between backpropagated and original fields. Parameters reported in Table 2 give a minimum restoration error percentage of the L^2 norm of the approximate solution as compared to the exact solution at $\lambda/8$. In the following sections, these same parameters will be used for differential backpropagation.

The L^2 norm was taken by minimizing the integral of the square of the resultant hologram. This numerical integration is conducted as follows: assume that the matrix representing the hologram is a surface composed of points with separation d, and let each point on the following grid represent each element of the matrix.

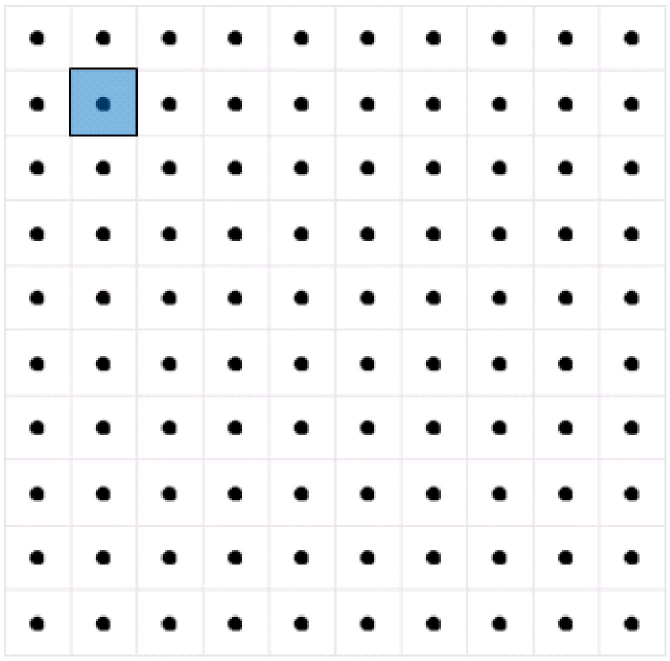


Figure 13 Lattice representation of surface with points stored within a matrix. Each square represents the differential area.

The surface area of this matrix may be calculated as each point multiplied by the differential area, d^2, however; this is not entirely accurate, as the edges of the matrix only represent an differential area of $d^2/2$. Even so, the corners also only contribute an incremental area of $d^2/4$.

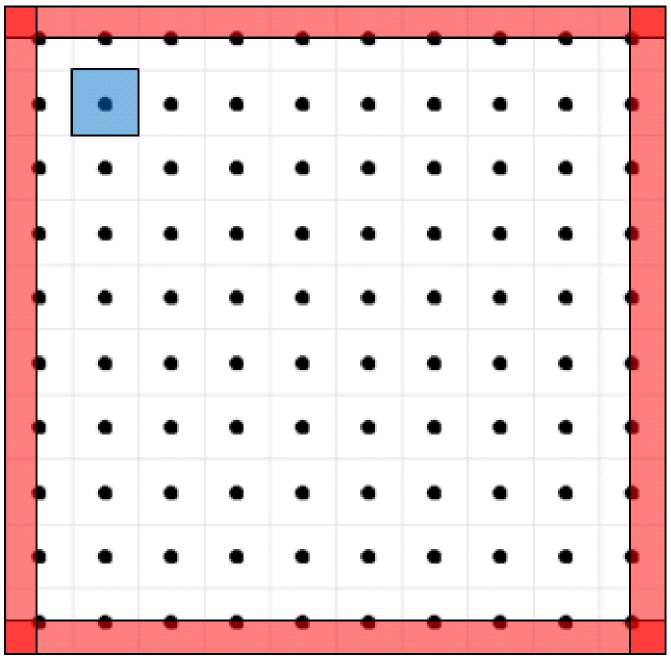


Figure 14 The differential area of matrix that should not be included in the surface area calculation is shown in red. Overlapping red regions must be subtracted twice.

Thus, the total area of the matrix in question is not simply equal to the sum of each element multiplied by the differential area. One must sum the matrix and subtract out the overlapping parts as shown in Fig. 14. An example of a surface area calculation for a matrix representing a surface is shown below.

```
%------------------------------------------------------------
for m = 1:length(surface_x)
    for n = 1:length(surface_y)
        temp     = surface_matrix;
        fxy(m, n) = d^2*(sum(sum(temp))-
0.5*(sum(temp(1,:))+sum(temp(end,:))+sum(temp(:,1))+sum(temp(:,end))));
    end
end
%------------------------------------------------------------
```

Table 2 Backpropagation parameters corresponding to a minimum restoration error of the co-polar *E*-field at the distance of λ/8 from the top of the antenna array in the near field.

Backpropagator (from to)	Minimum restoration error: percentage of L^2 norm compared to the exact solution at λ/8	Rectangular window in the *k*-space corresponding to minimum restoration error	Roll-off coefficient of the raised cosine filter, *r*, applied to the restored field	Integration step in the *k*-space
λ/8 to λ/8	0.3%	12*k*×12*k*	0.0	0.15*k*
λ/4 to λ/8	4.7%	3.75*k*×3.75*k*	0.2	0.15*k*
λ/2 to λ/8	17.3%	2.25*k*×2.25*k*	0.6	0.15*k*
λ to λ/8	31.1%	1.00*k*×1.00*k*	0.4	0.15*k*
1.5λ to λ/8	33.3%	0.85*k*×0.85*k*	0.5	0.15*k*
2.0λ to λ/8	35.8%	0.75*k*×0.75*k*	0.5	0.15*k*

The same parameters will be used for differential backpropagation as described in the following subsection.

Simulated Backpropagation Results

The results of a differential backpropagation are given in Table 3. The algorithm parameters are those from Table 2. Table 3 reports three backpropagated fields:

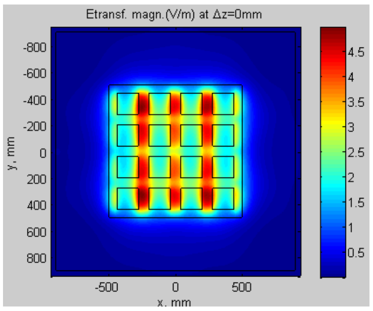
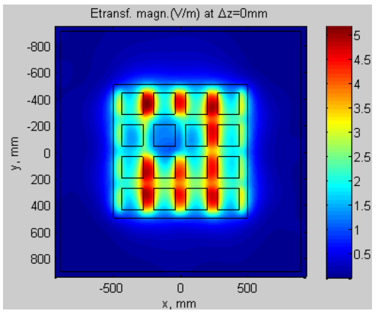
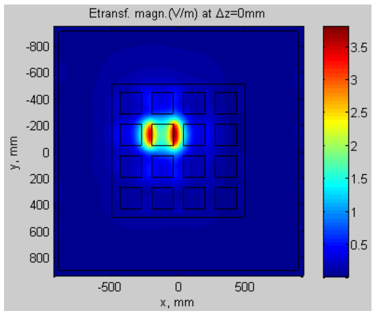
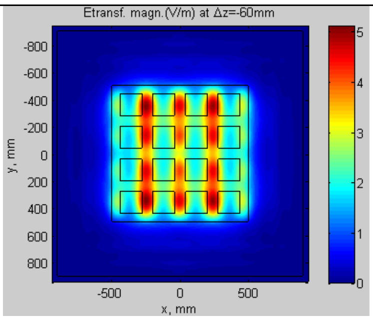
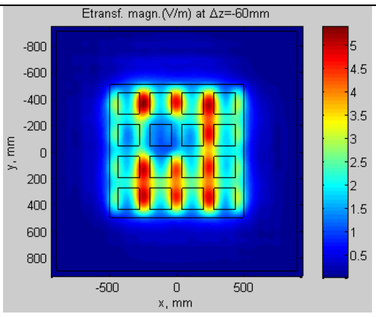
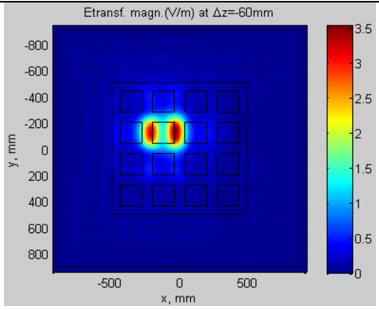
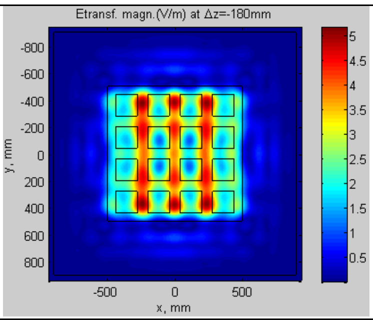
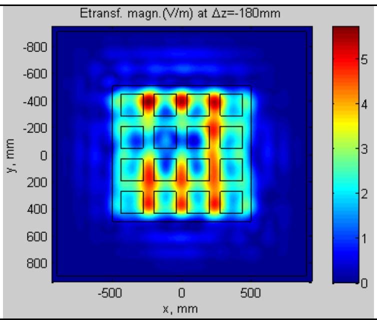
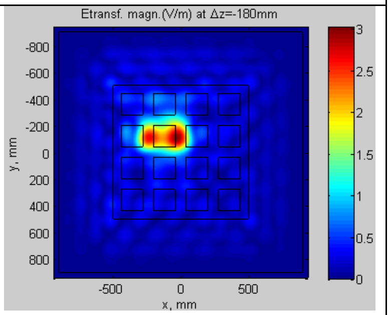
i. non faulty field;

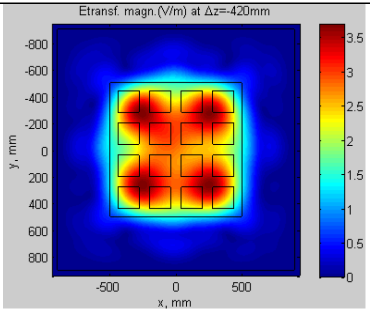
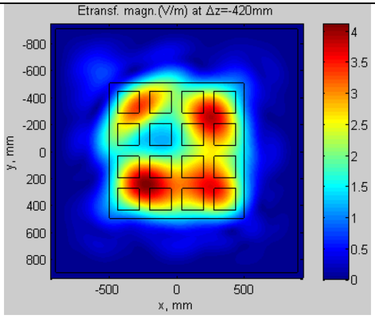
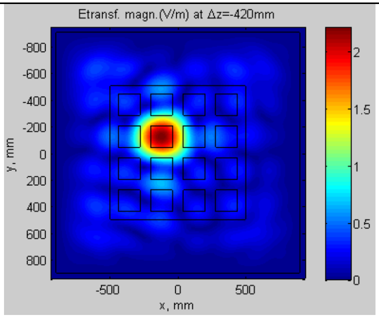
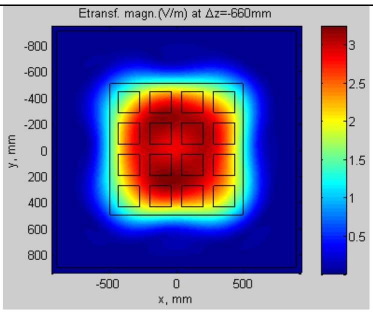
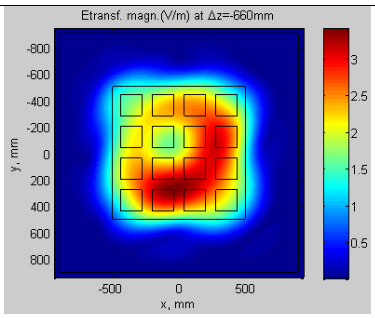
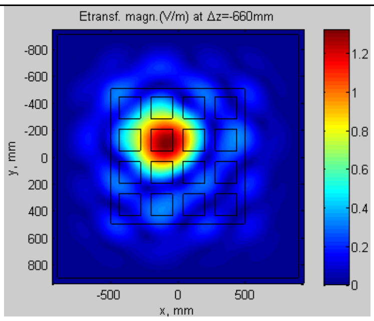
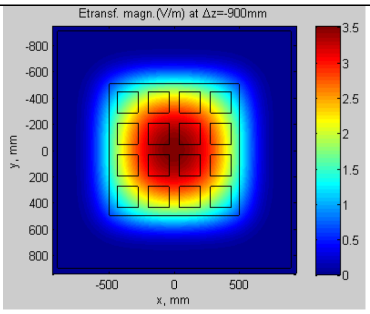
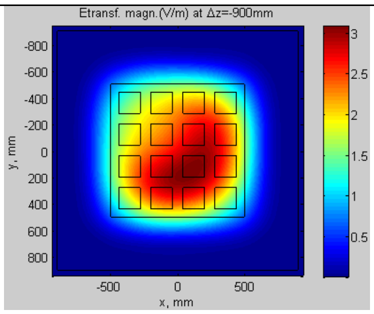
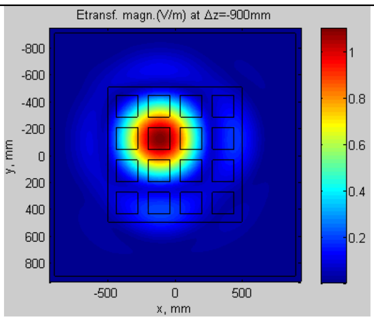
ii. faulty field – the generator of radiator 22 is shorted out, and;

iii. the (difference) error field, F_z .

One can see that the differential backpropagation uniquely determines the position of the faulty radiator, at any distance from the array, with a high degree of resolution.

Table 3 Backpropagated fields in three cases (λ/2 spacing). Element 22 is shorted out for a faulty array.

Backpropagator	Non-faulty array: all radiators are excited with equal generators at 1V amplitude	Faulty array: the generator of radiator 22 is shorted out	Difference between the fields of non-faulty and faulty arrays propagated back to λ/8
λ/8 to λ/8			
λ/4 to λ/8			
λ/2 to λ/8			

λ to λ/8			
1.5λ to λ/8			
2.0λ to λ/8			

Extensions

Appendix A gives the backpropagation results corresponding to the last row of Table 3 above, but for all possible faulty element locations. It is seen that the algorithm is working properly.

Appendix B gives the same backpropagation results, but corresponding to a partially attenuated element 12 (by -∞dB, -6dB, and -3dB). The algorithm is still working properly.

Appendix C gives the same backpropagation results corresponding to changes in phase of element 12. The algorithm is working properly for all phase variations.

Appendix D gives the same results as in Appendix A, but with element spacing equal to 0.32λ instead of 0.5λ . The algorithm is still working properly, which is a significant result with regard to backpropagation resolution at the lower frequency of the band.

Chapter 5: Measured Results

Antenna Measurements

Experimental results are needed to validate theoretical/numerical data in any case; therefore we should take pattern measurements of an antenna with faults and see if they are easily detectable with the differential hologram. Although antenna pattern measurements are usually taken with the test antenna receiving a signal from another radiator, the transmit pattern will be identical if the antenna is reciprocal. Under ideal conditions, the test antenna should be illuminated by plane waves with uniform amplitude and phase to find a far field pattern. Normally, the far field region is too great a distance to measure properly – this is another reason why our near field propagator and far field transforms are so important.

The testing and measurements of antennas is usually conducted in an antenna range, whether it is indoor or outdoor. The measurements conducted for this paper were performed in a rectangular anechoic chamber. The anechoic chamber is an isolated room whose walls are covered in RF absorbers. A rectangular anechoic chamber, as opposed to a tapered chamber, is typically designed to simulate free-space conditions and maximize the volume of the "quiet zone." This is the region around the test antenna which has minimal electromagnetic interference. The rectangular anechoic chamber takes into account the pattern, location of the source, frequency of operation, and assumes that the receiving antenna at the test point is isotropic. Although the reflected energy in the chamber is minimized using high quality RF absorber, the absorber's properties are frequency dependent and work better at some frequencies than others. Significant reflections may still occur at large angles of incidence.

Near field measurements may be taken in three different ways: planar, cylindrical, or spherical measurements. These measurements are taken along a grid with the transmitting antenna moving from point-to-point measuring data. The planar measurement takes place on an x, y grid, the cylindrical measurements take place on the z, φ cylindrical grid, while the spherical measurements take place on the θ, φ grid. Although the spherical and cylindrical methods allow for more field to be captured in the measurements, they are also more expensive and require more computations. The planar scan is also better suited for phased arrays.

When conducting a planar scan over the x, y grid, a spacing of $\Delta x, \Delta y < \lambda / 2$ must be used in order to avoid any aliasing. The probe transmitting the signal is normally an open ended waveguide or small horn with a relatively flat radiation pattern; this is for a special reason: as the probe moves along the x, y grid, its orientation relative to the test antenna changes. This directive property, along with the polarization of the horn, must be taken into account when observing measurements – this process is known as “probe compensation.” This method uses the Lorenz reciprocity theorem to couple the far fields of the test antenna to those of the probe. Note how in Fig. 15, the center of the array in relation to the center of the horn is different with regard to the horn pattern.

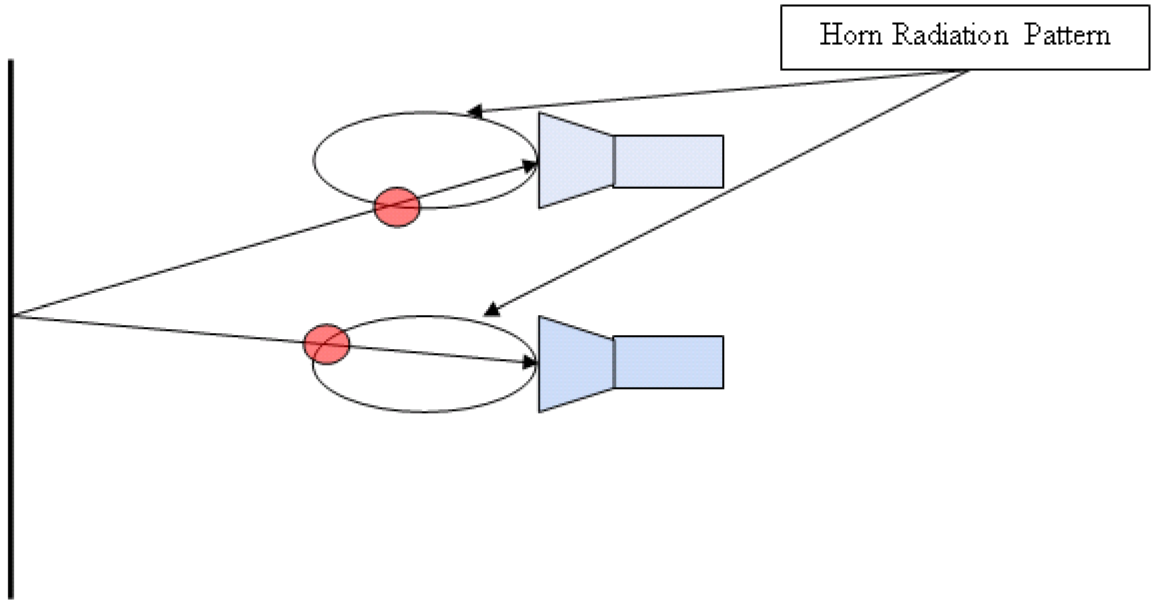


Figure 15 Example of how the placement of the transmitting antenna along the grid changes the orientation relative to the test antenna. This directive property, along with the polarization, must be taken into account when observing measurements. The red circles portray the different part of the horn pattern seen by the array when the horn is at different locations.

Hologram Error Sources

Although we are using a complete solution to Maxwell's equations to calculate the hologram, we still have errors caused by measurement techniques. These errors have been characterized by several authors [4]-[8] , and include

a. Noise;
b. Probe Relative pattern;
c. Probe polarization;
d. Probe gain;
e. Probe alignment;
f. Normalization constant;
g. Impedance mismatch;
h. AUT alignment;
i. Data point spacing (aliasing);
j. Measurement area truncation;
k. Probe X-Y position;
l. Probe Z-position;
m. Mutual coupling;
n. Receiver non-linearity;
o. Systematic phase errors;
p. Receiver dynamic range;
q. Room scattering;
r. Leakage;
s. Random errors

These errors, although large in number, add up to approximately 0.2dB in amplitude error, and a 2.5 degree phase error in the hologram [6]; even so, if we are to take the differential hologram of two arrays we will be able to eliminate some of this error. Consider the errors caused by b, d, f, q; these errors are linear in nature and, if repeatable, may either be measured and compensated for or minimized in a differential measurement. That is to say, any linear sources that cause changes to the amplitude of the electric field may be minimized with a differential measurement; however, this is only the case when the faulty array has amplitude errors only. Errors caused by positioning such as i, k, and l have been found to have a small but insignificant effect on measurements [7], especially in the case of i, which may be minimized if the measurement grid step is chosen carefully. Although errors e and h may be dramatic, it is easy to eliminate these by carefully aligning the probe and test array in the measurement setup.

On the other hand, the differential measurement will be unable to eliminate errors related to phase in any case. It is clear that the errors caused by g, n, o, p, r, and s also fall into the same category as phase errors; even so, these errors account for 0.1dB of the 0.2dB amplitude and 0.75 degree out of 2.5 degree phase error estimates in [6]. The errors created by random noise (Gaussian), Rochblatt and Rahmat-Samii [4] have found that unless dealing with a weak signal, or very small SNR, these errors do not have a large effect on results; however, those errors caused by other random errors may account for as much as 0.7dB and 0.5 degree phase errors.

One of the most noted errors in calculating the hologram is that of j, measurement plane truncation, studied in depth by Newell [8]. This is the notion that the electric field is measured on a finite rectangular grid at a distance z_0 from the aperture plane. As we know, this measurement plane must be infinite if the entire field were to be captured in such a measurement. Newell treats this finite measurement plane as a filter on the k-space spectrum. This filter alters the behavior of the backpropagator by smoothing the hologram discontinuities (i.e. convolving the result with a sinc function, see fig 16.)

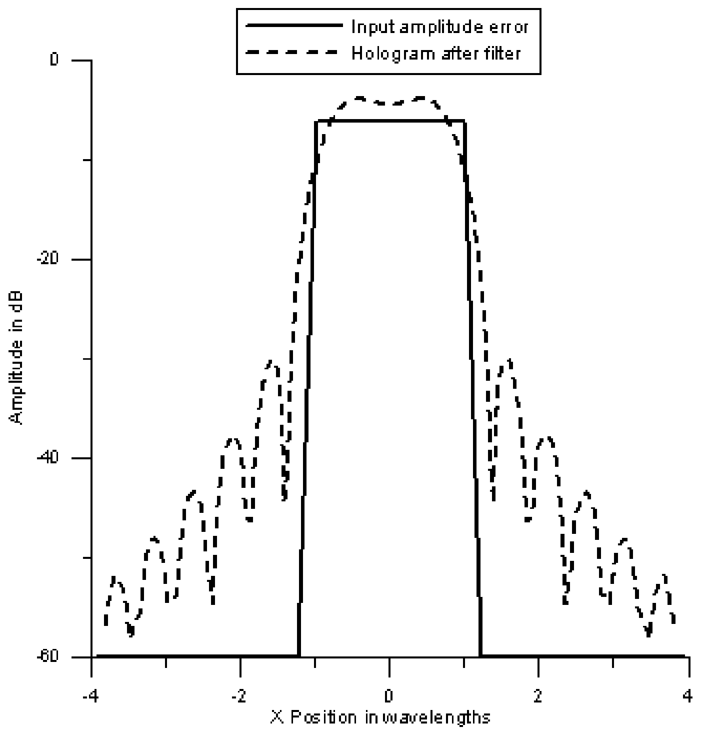


Figure 16 Example of smoothed hologram due to windowing in *k*-space, an effect of a finite measurement plane [8]

The severity of the smoothing uncertainties caused by this windowing is obviously dependent on the dimensions of the measurement plane, but is predictable. In most cases, this filtering removes a bit more of the spectrum than the portion just pertaining to evanescent modes. Although this information is necessary for reconstructing a perfect hologram, in reality, if the spectrum is not windowed properly according to its measurement plane, much of the calculated information is detrimental to reconstructing an accurate image: to clarify, any errors taken from the evanescent modes of the spectrum will be amplified exponentially in the reconstructed fields in the aperture plane, destroying the hologram altogether. Even though it leads to smoothing, It can be shown that this windowing in the k-space, to an extent, is a necessity when backpropagating any wave that is not expressed analytically, as even numerical errors may destroy the hologram if not filtered properly. This is due to the effect on evanescent modes provided by our backpropagator formulation. For example, in Fig. 18, five plots are shown for our 4x4 patch array in MATLAB with field data taken from 2λ and the following windows in k-space:

$k_x^2 + k_y^2 < 0.55k$	(42.5% Error)
$k_x^2 + k_y^2 < 0.65k$	(42.0% Error)
$k_x^2 + k_y^2 < 0.75k$	(Optimal, 39.9% Error)
$k_x^2 + k_y^2 < 0.85k$	(39.1% Error, fake)
$k_x^2 + k_y^2 < 0.95k$	(405% Error)

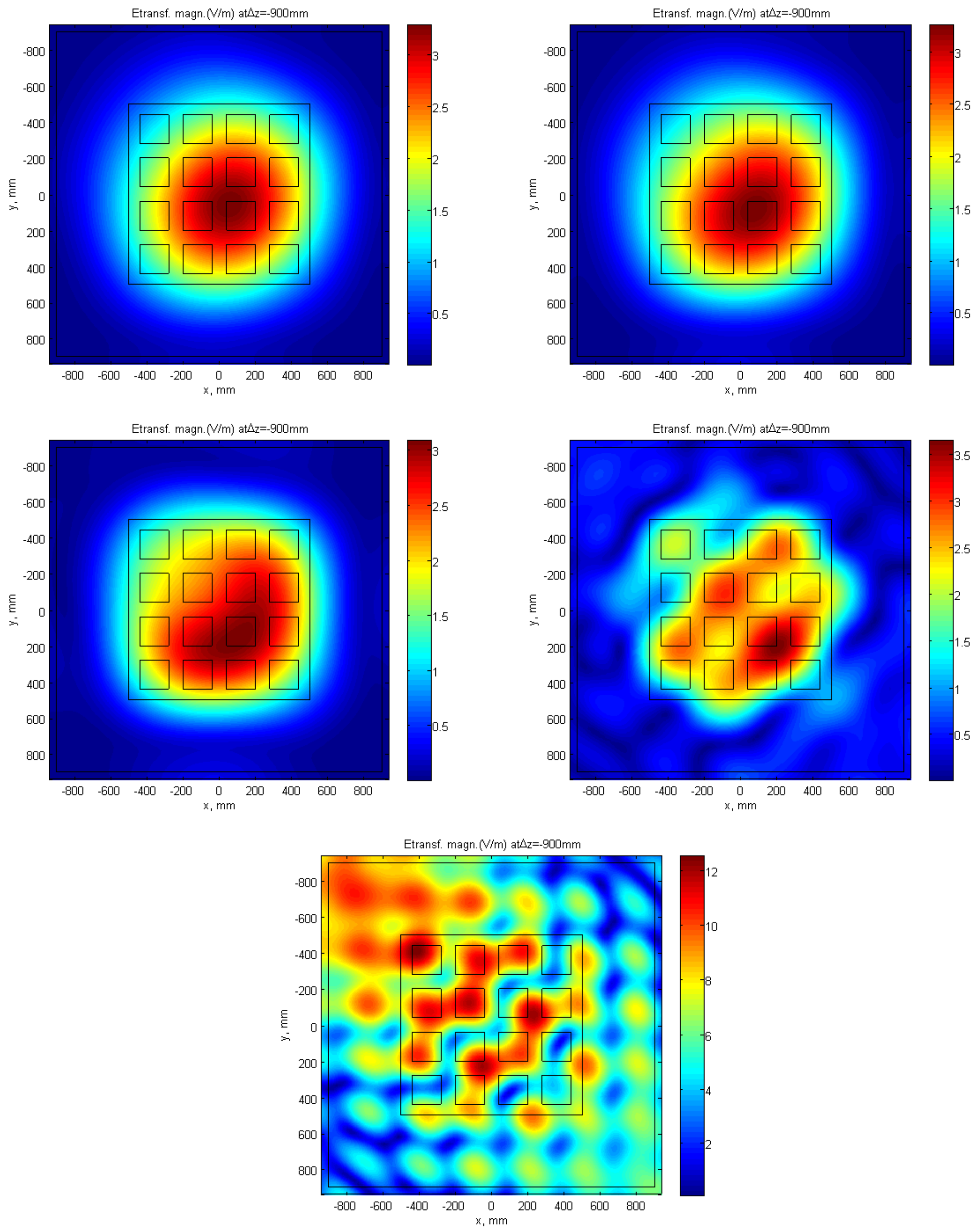


Figure 17 Hologram of 4x4 patch array modeled in MATLAB with different windows in k-space. Top left pertains to $k_x^2 + k_y^2 < 0.55k$ with a 42.5% error; Top right pertains to $k_x^2 + k_y^2 < 0.65k$ with a 42.0% error; Middle left pertains to $k_x^2 + k_y^2 < 0.75k$ with a 39.9% error; Middle right pertains to $k_x^2 + k_y^2 < 0.85k$ with a 39.1% error (fake); Bottom pertains to $k_x^2 + k_y^2 < 0.95k$ with a 405% error.

We can see that the measurements taken from a distance of 2λ, in this case, provide an unstable solution before the window in k-space even approaches $k_x^2 + k_y^2 = k$. This is because the window in k-space created by the finite scan plane has caused numerical errors when calculating values somewhere outside of the circle $k_x^2 + k_y^2 < 0.75k$; however, if we were to use a larger scan plane, it is possible to expand this optimal window in k-space to a larger number and obtain more accurate results. We may find the same result for measured fields as well. The window in k-space may be approximated by the following equations [8].

One distinct error uncharacterized fully by many is that of m, the mutual coupling of antenna elements in an array [5], [6]. This problem stems from the fact that, in an array, individual antenna element patterns are usually unique to one another, his non-uniqueness is seen as amplitude and phase ripples between elements [6]. This means that the *approximation* that the antenna array pattern is the array factor multiplied by the element pattern is not quite good enough. The errors caused by mutual coupling may only be seen as errors caused by approximations made in the backpropagator being used in the hologram calculation. In our case, the backpropagator is an exact solution to Maxwell's equations and is unaffected by this pattern variation – this means we may even see the effects of mutual coupling in our hologram if measurements are taken properly. The errors seen by mutual coupling may most likely be attributed to inverse propagation techniques similar to that of Narishman and Kumar [24], or incorrect array pattern simulation formulation (i.e. construction of the far field array pattern by multiplication of the element pattern with the array factor). This error is also usually mentioned alongside that of element pattern uncertainty [5], [6]; again, these errors may be seen as inaccuracies in the backpropagation formulation or simulation methodology.

Array Under Study

The array shown in Fig. 18 is a 4x4 passive array of patches with a corporate feed based on T-power dividers. The array is etched on a 60 mil Rogers4003 substrate ($\varepsilon = 3.45$, $\tan\delta = 0.0027$). The array is 148mmx148mm in size and radiates at 4.00 GHz.

An array with the corporate non-isolated feeding network provides more challenges than the array where every radiator has an isolated feed. The reason is that a faulty (detuned in this case) radiator changes local termination to a corporate feeding network, which results in a different performance of a group of nearby radiators. As an example, Fig. 19 shows the current distribution on the array ground plane when one of the radiators is detuned by a copper strip (capacitive loading) versus the non-faulty current distribution. Nevertheless, even in this case, convincing results may be obtained.

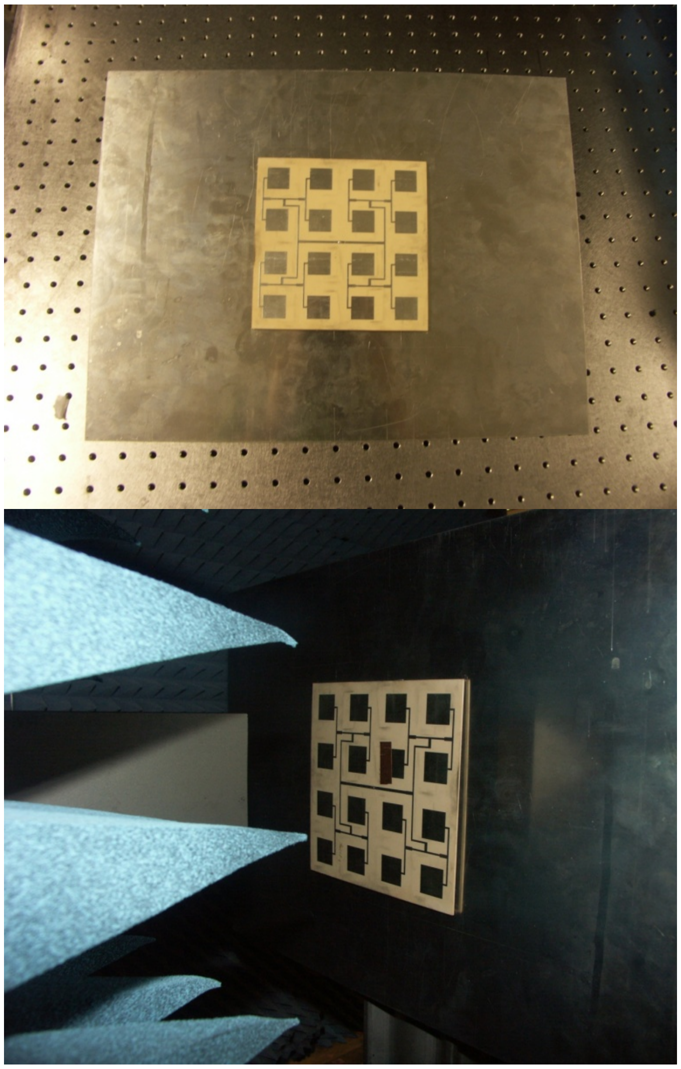


Figure 18 Top – A 4x4 array of patches with a corporate feed and posterior-fastened aluminum ground plane; bottom – the same array in the near-field range.

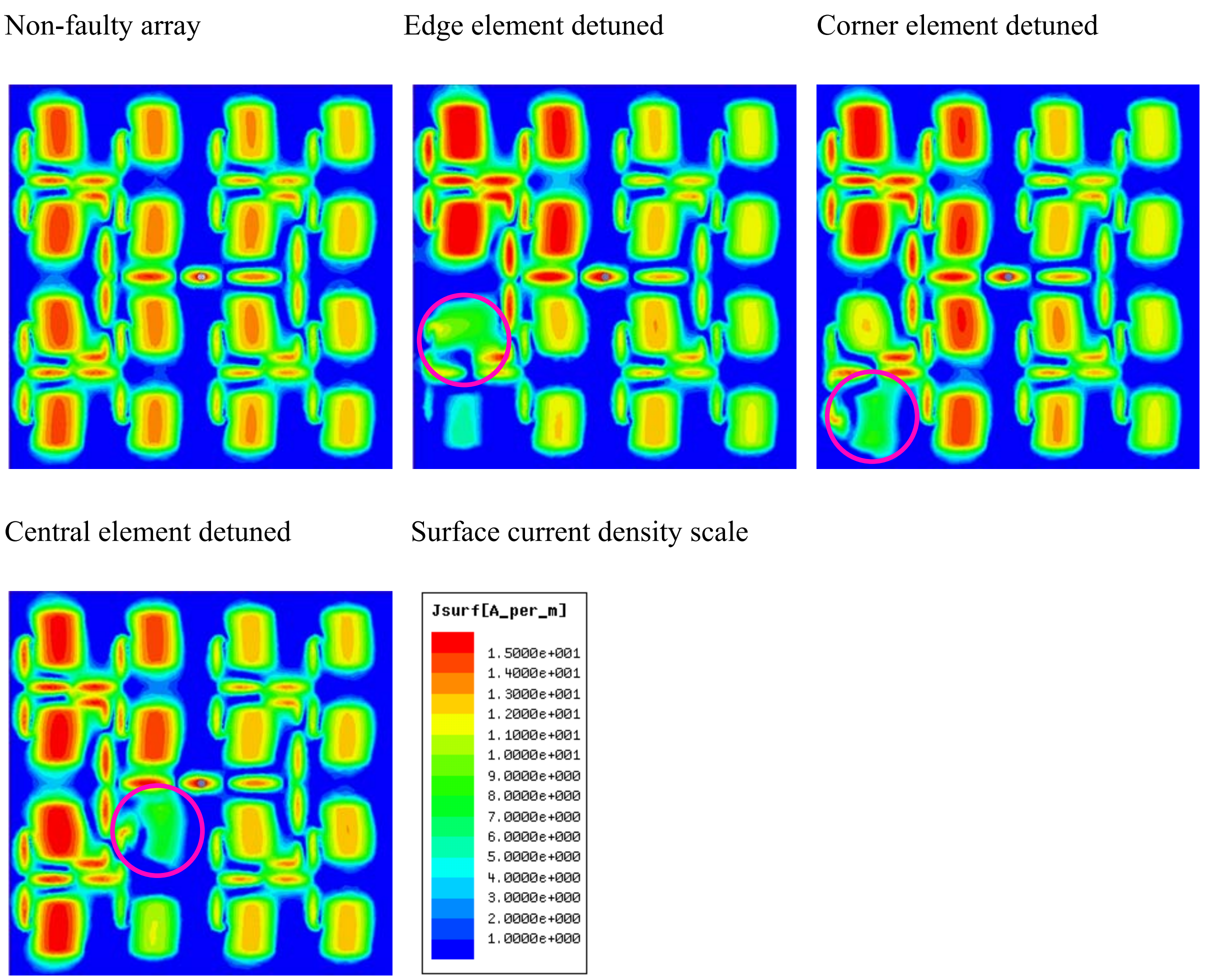


Figure 19 Simulated (Ansoft/ANSYS HFSS) current distribution on the ground plane when one of the array elements is detuned.

Numerical vs. experimental backpropagation

In order to simulate the presence of a faulty element, a piece of copper tape (25mm×10mm) was used to detune an individual patch. Measurements were taken at a distance of 2.0λ, as shown in Fig. 18. The copol component of the electric field has been sampled over a 20'×16" observation plane with a rectangular horn, with a sampling interval of 0.5". All simulations in this section have been done with Ansoft/ANSYS HFSS.

Table 4 provides the results for every array element. The first row in every page shows the array geometry; the second row gives the current distribution on the ground plane. The last row gives two results:

i. The error field magnitude, $|F_z|$ when the numerical solution for a faulty array is subtracted from the numerical template for the regular array - left.

ii. The error field magnitude, $|F_z|$ when the experimental data for a faulty array is subtracted from the experimental data for the regular array - right.

Table 4 Backpropagation results (num. to num. and experiment to experiment) – last row.

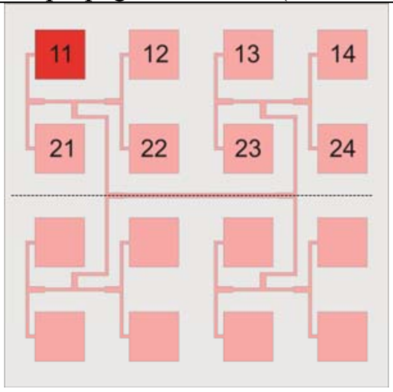
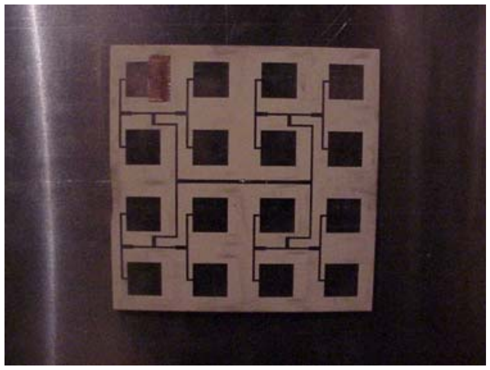
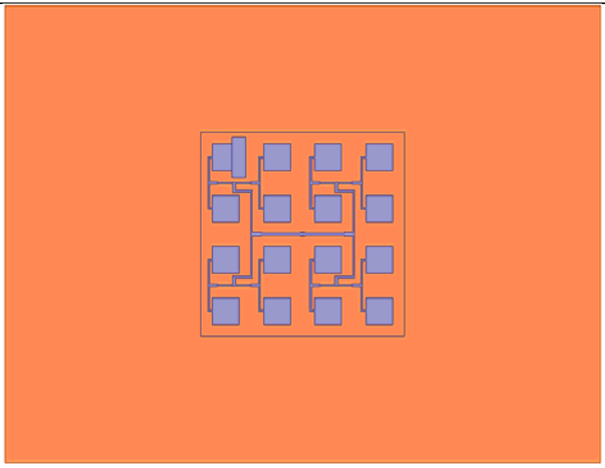
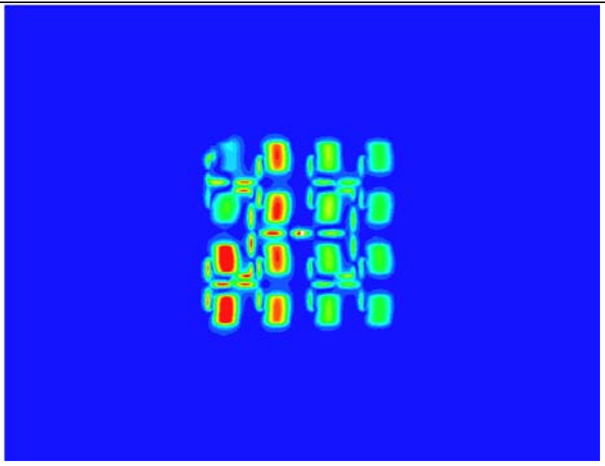
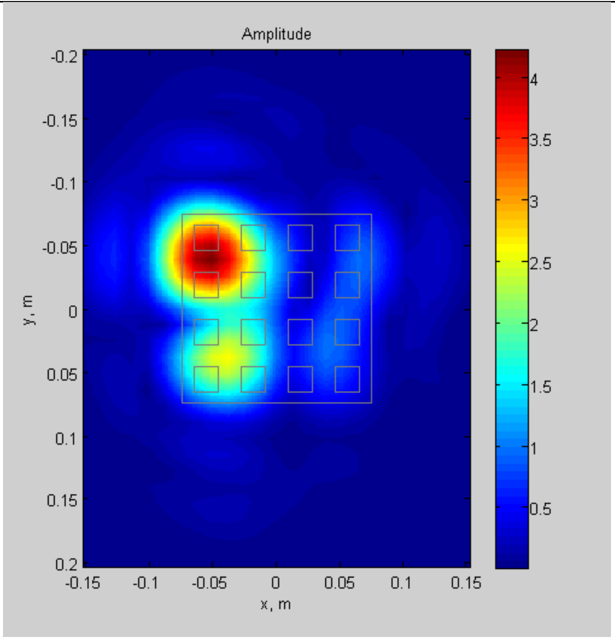
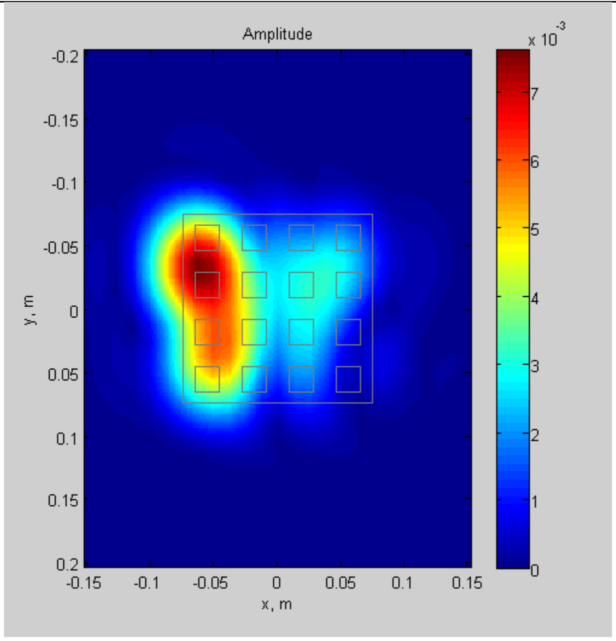
	
	
	

Table 4 (cont.)

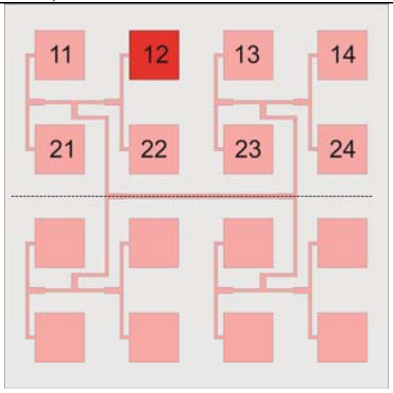
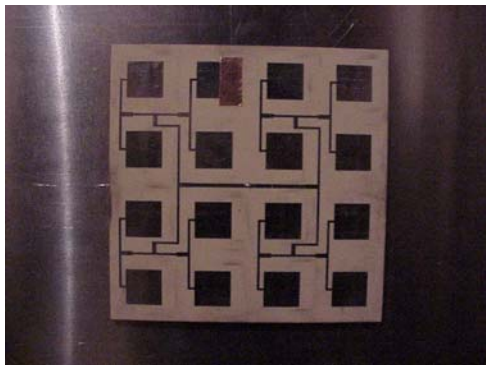
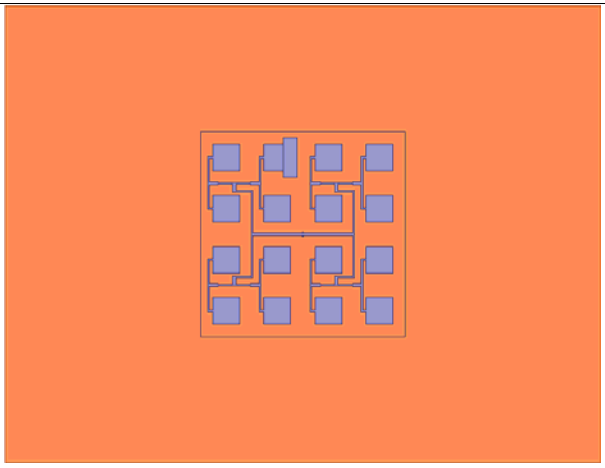
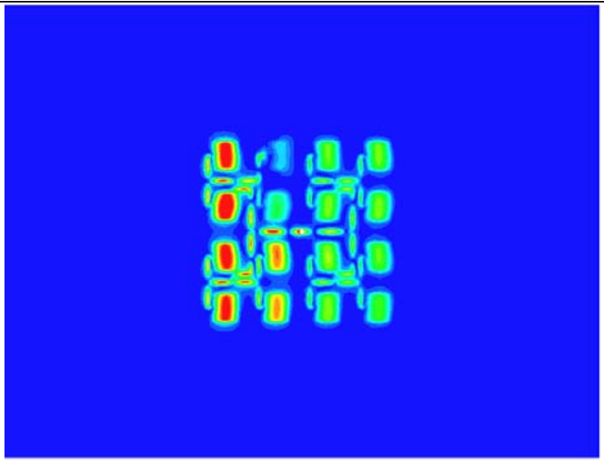
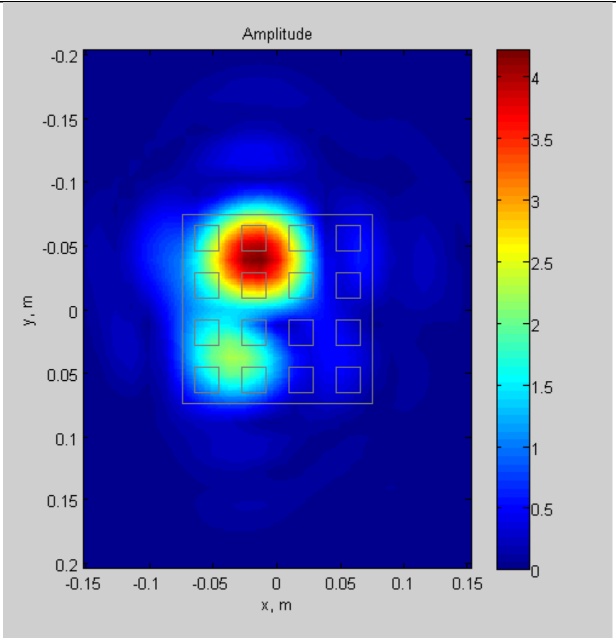
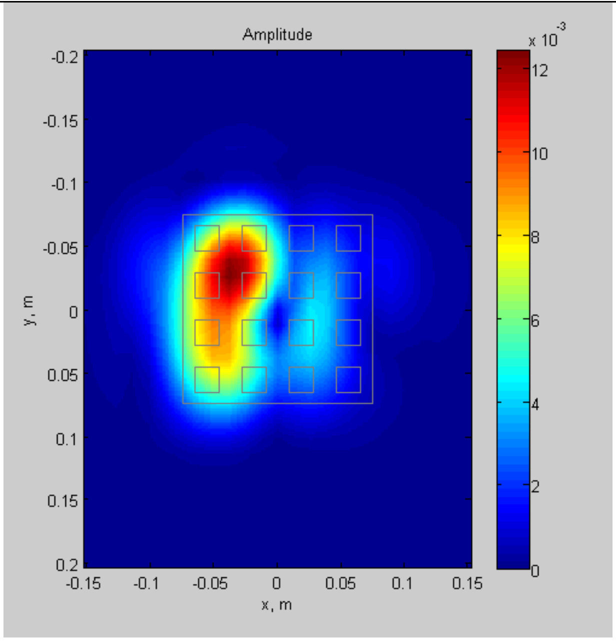
11 12 13 14 21 22 23 24	
	
Amplitude y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 0.5 1 1.5 2 2.5 3 3.5 4	Amplitude x 10^-3 y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 2 4 6 8 10 12

Table 4 (cont.)

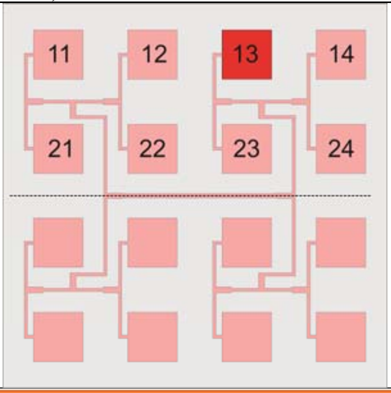
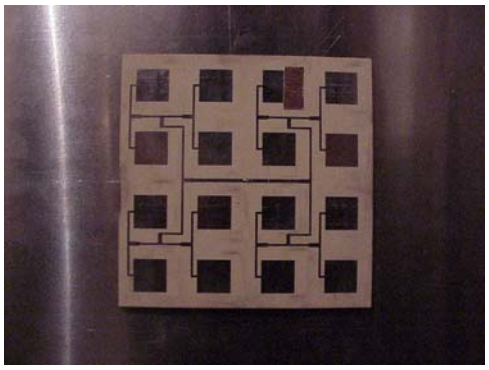
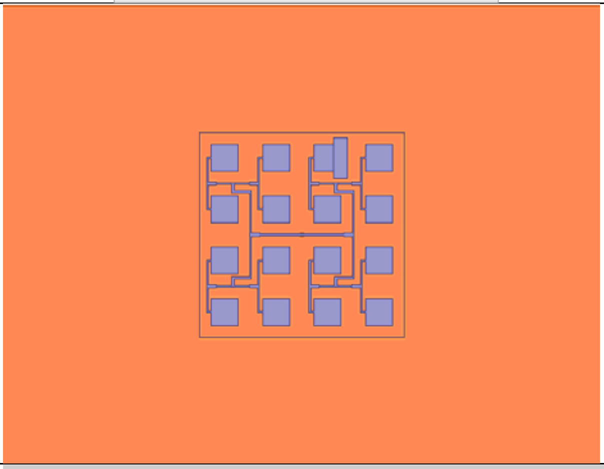
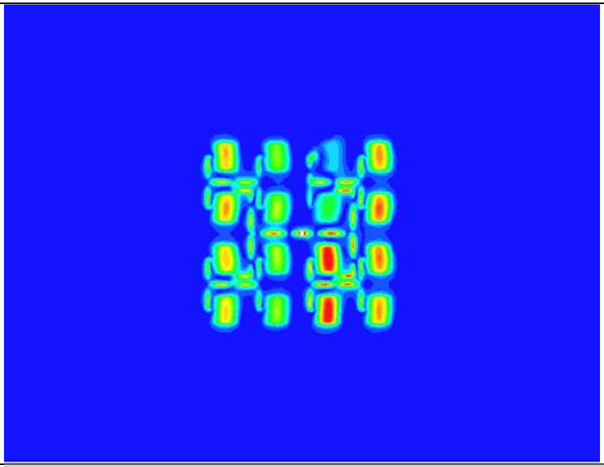
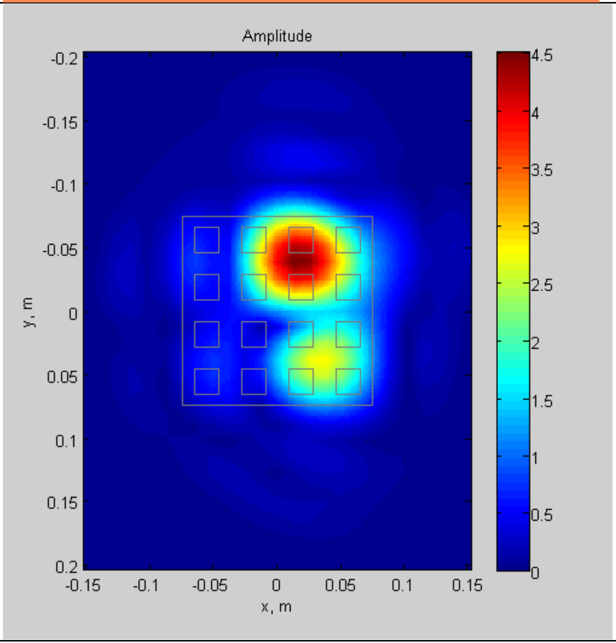
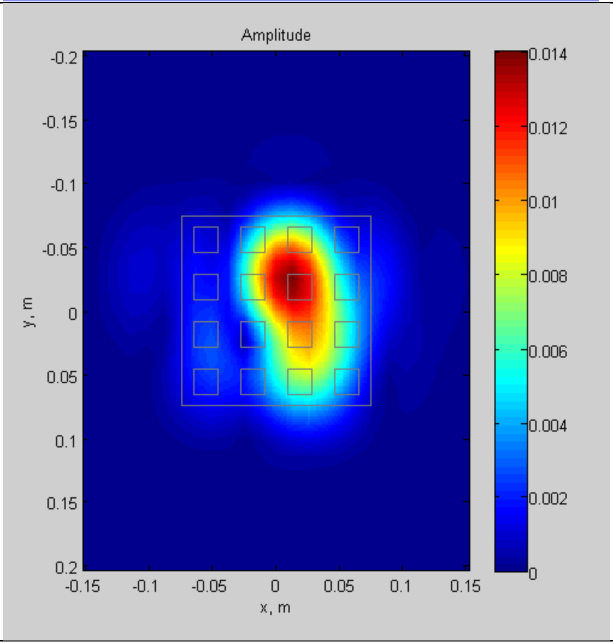
	
	
	

Table 4 (cont.)

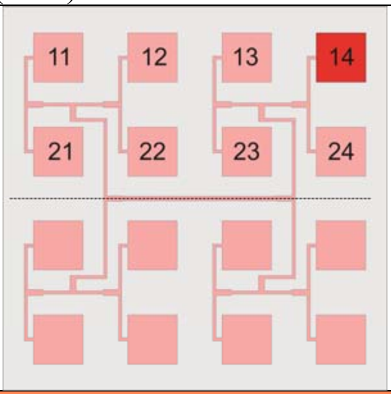
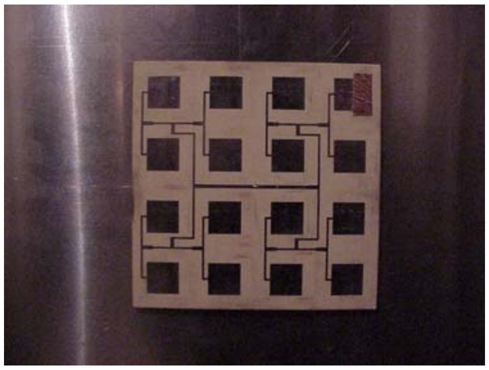
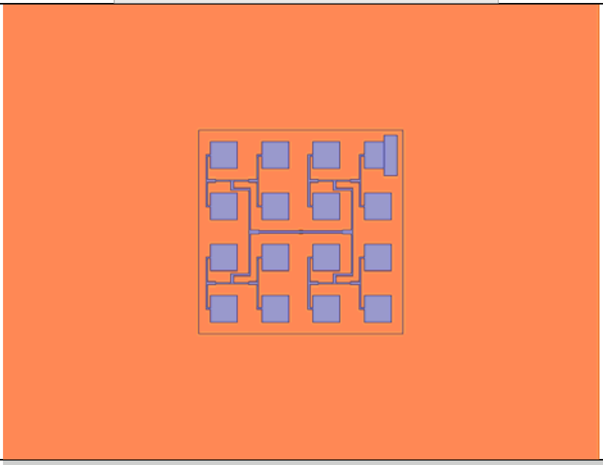
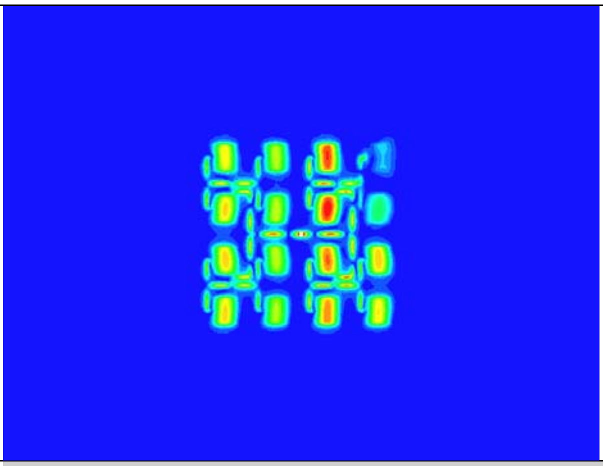
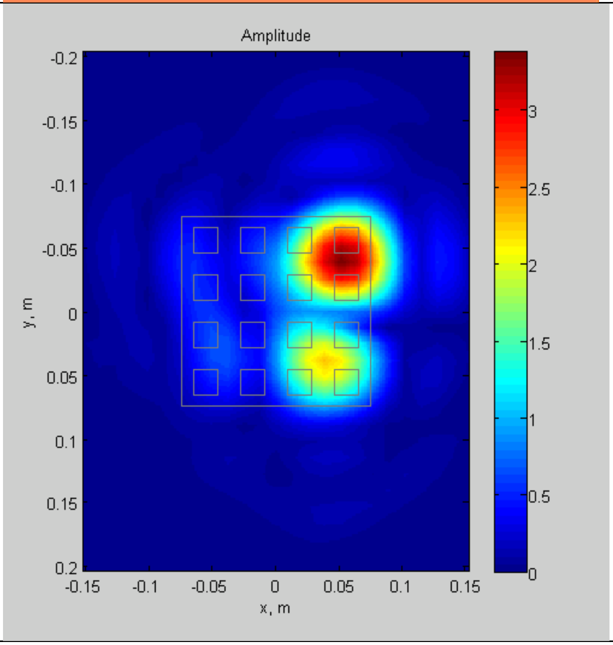
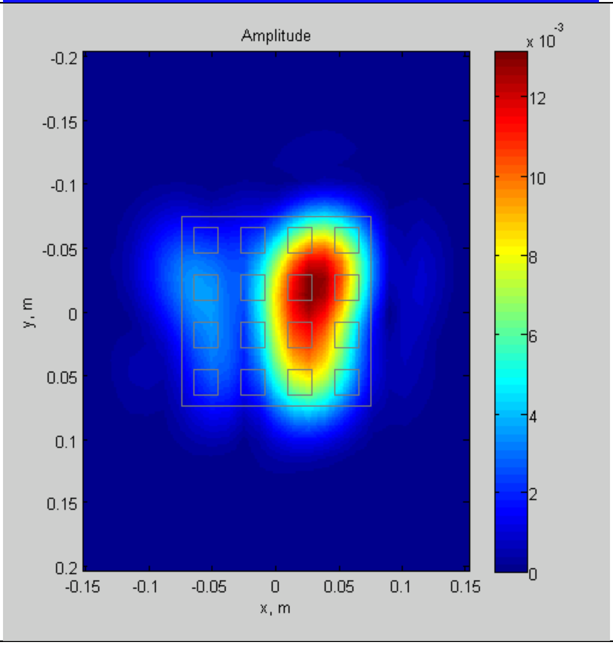
 11 12 13 14 21 22 23 24	
	
 Amplitude y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 3 2.5 2 1.5 1 0.5	 Amplitude x 10^-3 y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 12 10 8 6 4 2

Table 4 (cont.)

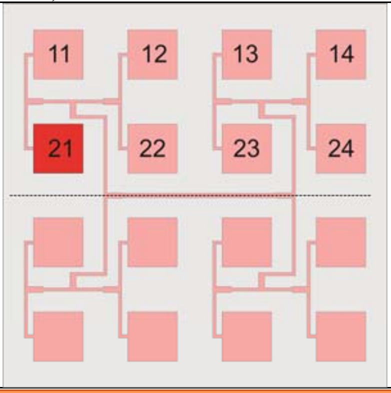
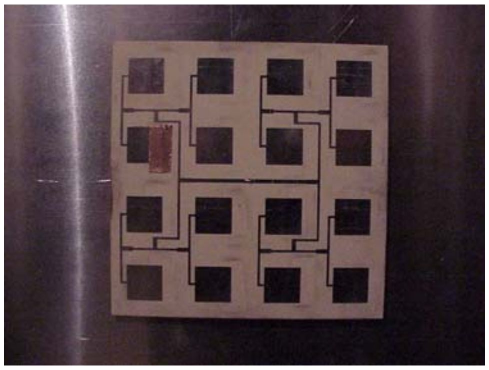
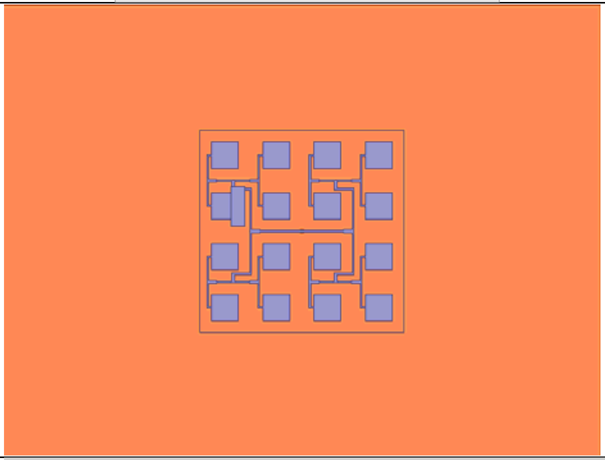
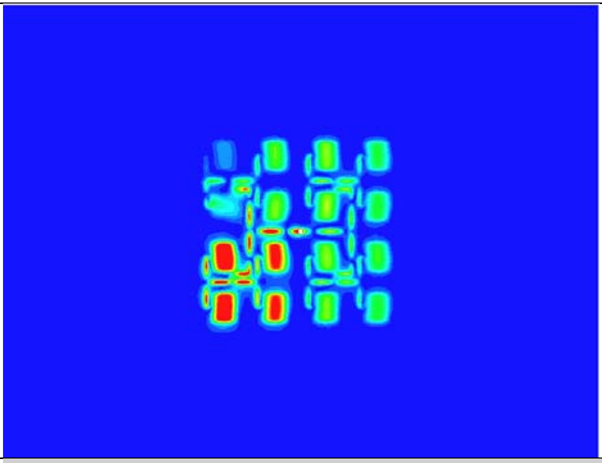
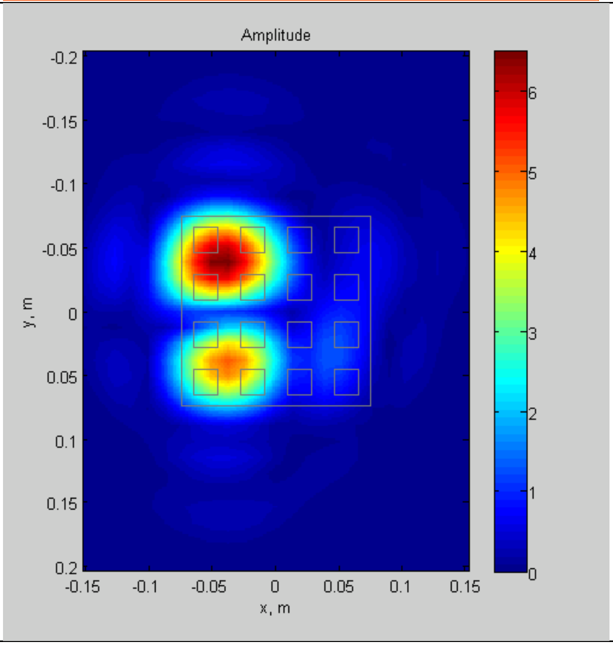
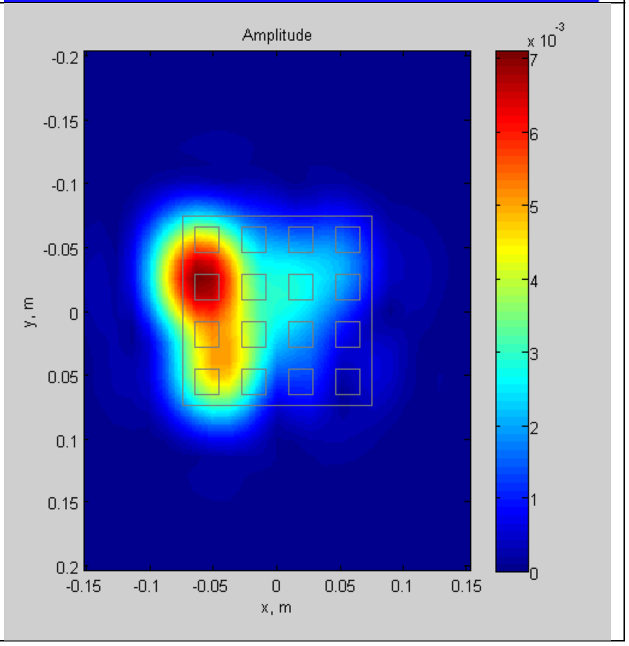
	
	
	

Table 4 (cont.)

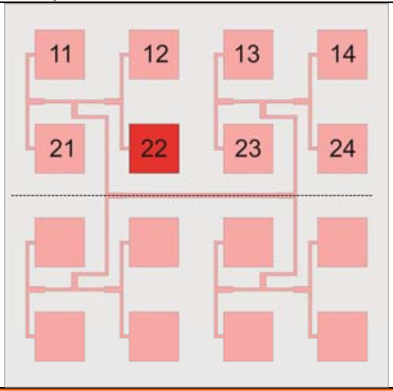
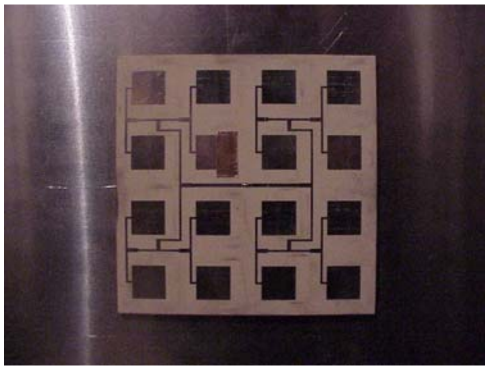
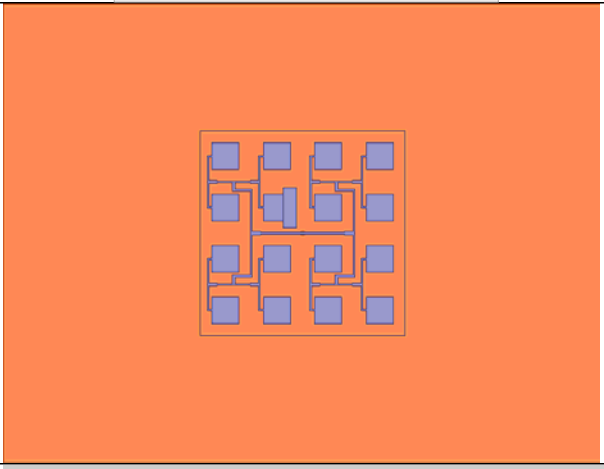
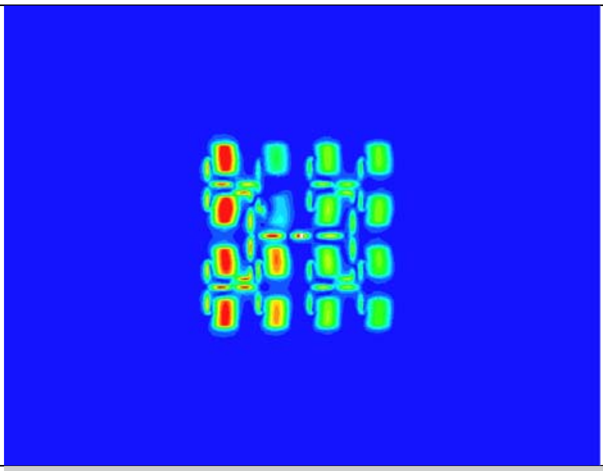
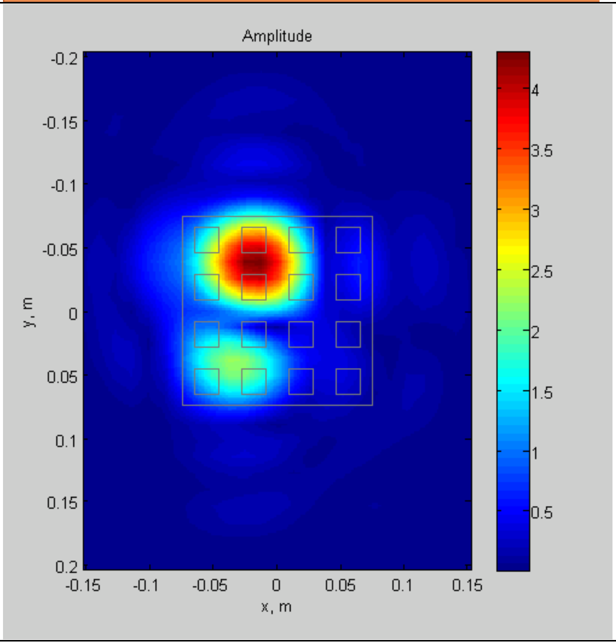
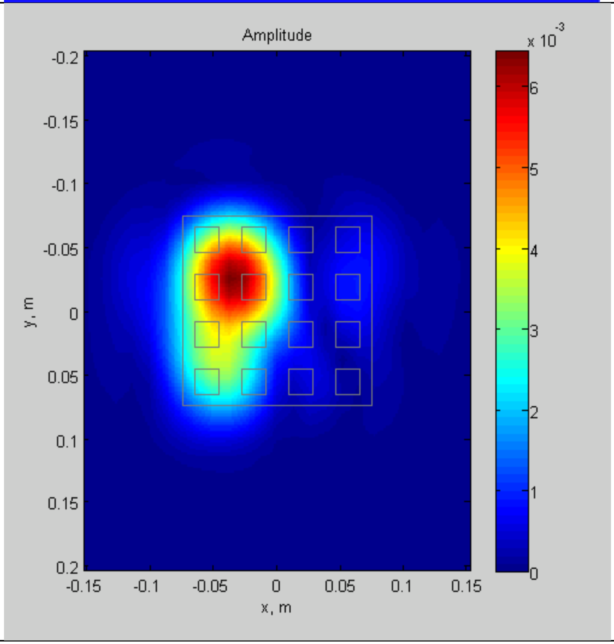
	
	
	

Table 4 (cont.)

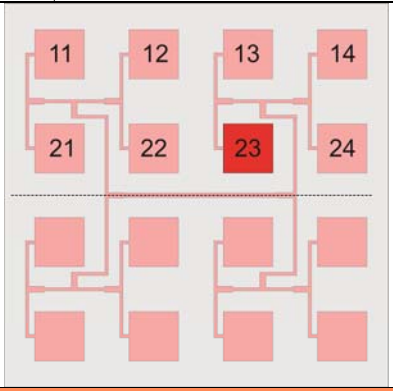
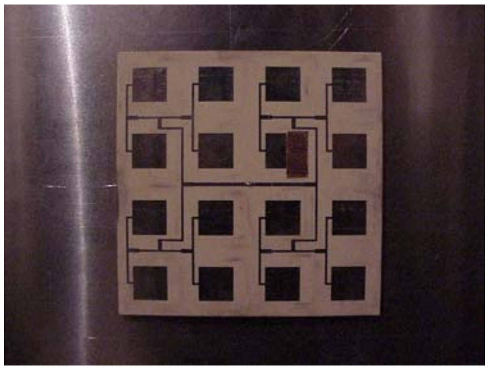
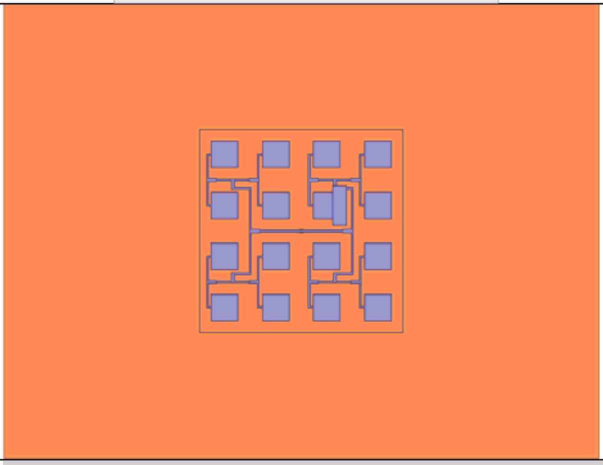
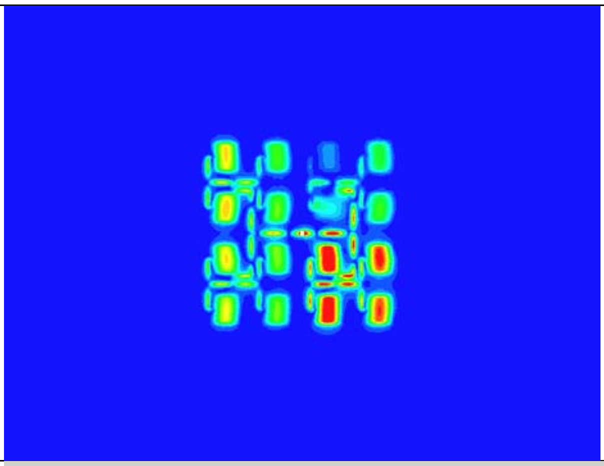
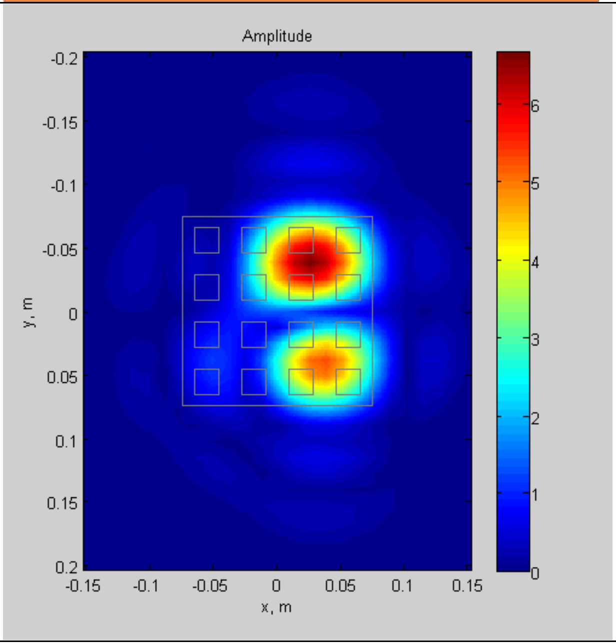
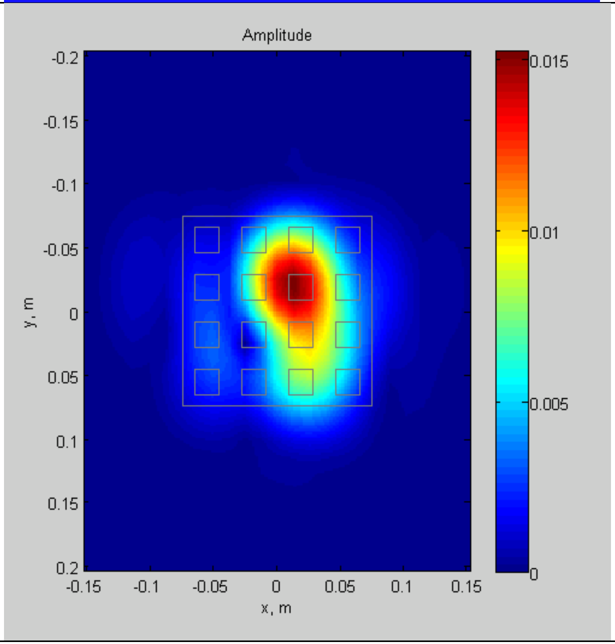
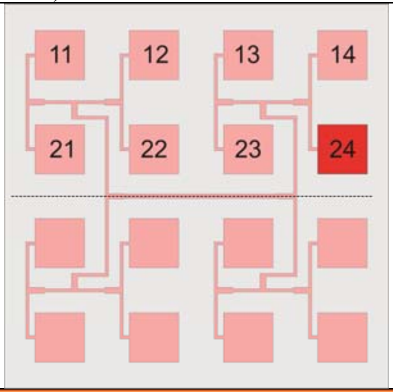
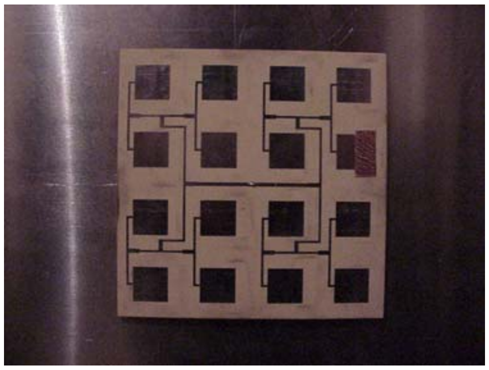
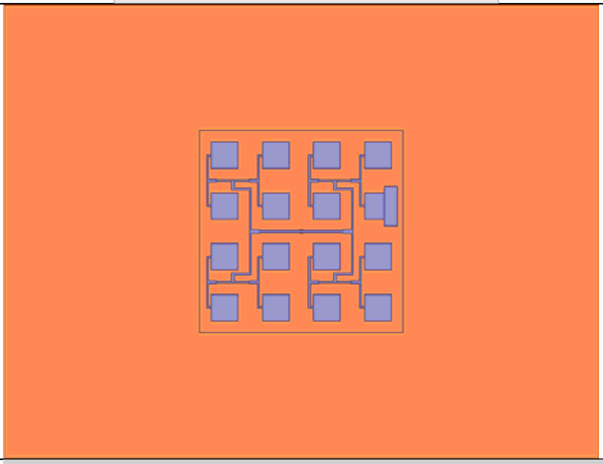
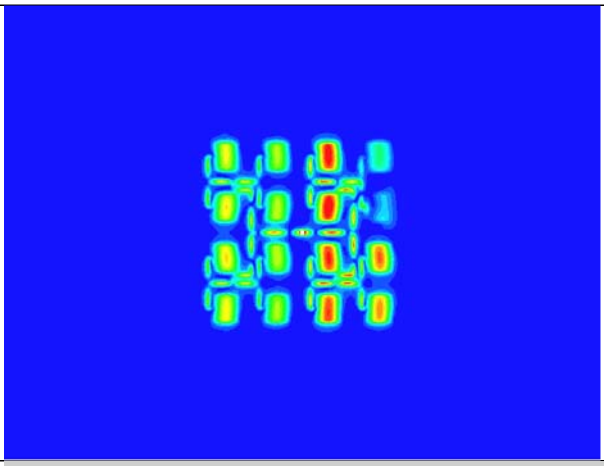
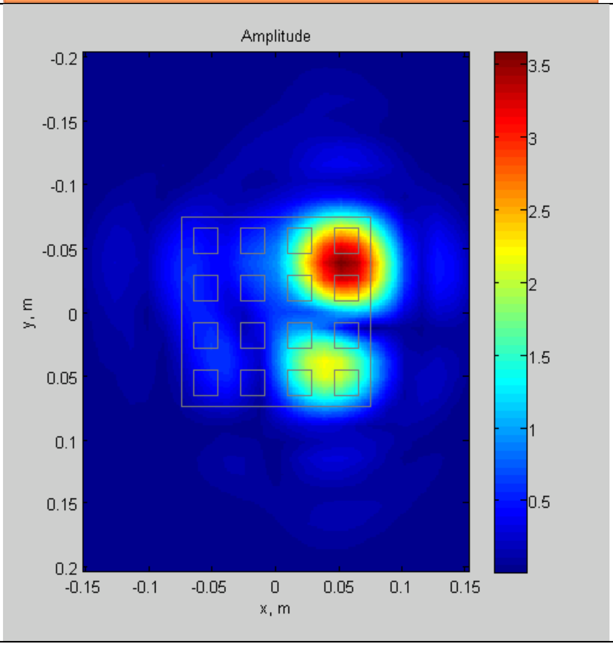
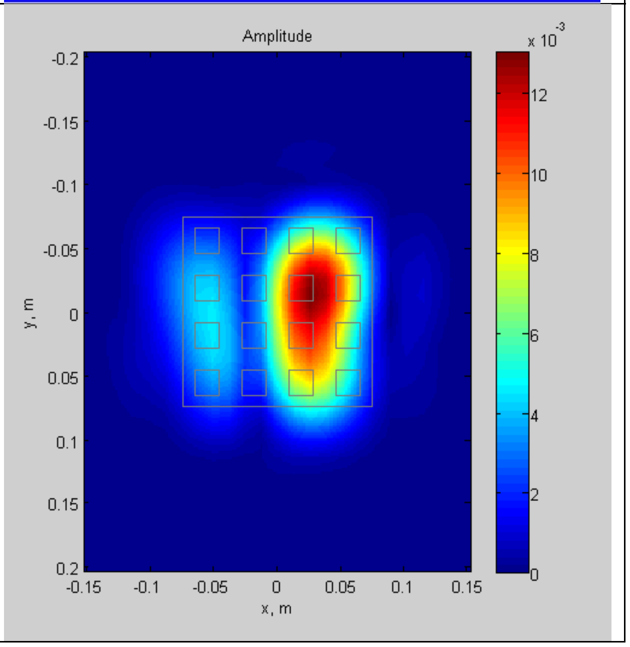
11 12 13 14 21 22 23 24	
	
Amplitude y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 6 5 4 3 2 1 0	Amplitude y, m x, m -0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.2 -0.15 -0.1 -0.05 0 0.05 0.1 0.15 0.015 0.01 0.005 0

Table 4 (cont.)

Despite the non-isolated feeding network, it is seen from Table 4 that:

i. The differential backpropagation of *measured* faulty- and non-faulty data identifies the faulty (detuned) element with accuracy better than or equal to element spacing. In some cases the agreement is exact.

ii. The differential backpropagation of *simulated* faulty- and non-faulty data identifies the faulty (detuned) element with accuracy better than or equal to element spacing. The simulated data provides only slightly better resolution accuracy. This means that the experimental procedure does not introduce much error.

Hybrid (Numerical – Experimental) Backpropagation

Sometimes the template array with no faulty elements may not be available for measurements. Only an "unknown" array is thus available. In this case, the numerical data for the template array may be generated and used in the differential algorithm.

However, the data should be aligned in phase. A way to this is to find the complex radiation pattern and align the phases of the main beam. Yet another, perhaps less accurate, way is to find maximum near field magnitude, in both measurements and simulations, and to assign phase zero to the field at that location. In order to synchronize the phase of the plots in Table 5, the L^2 norm of the difference between the measured array data and a phased version of the HFSS data (multiplied by the constant matrix $e^{j\theta}$), both of non-faulty arrays, was minimized. Fig. 21. shows how the phase synchronization affects this minimization - the ideal version of the hologram in Fig. 21 should be zero.

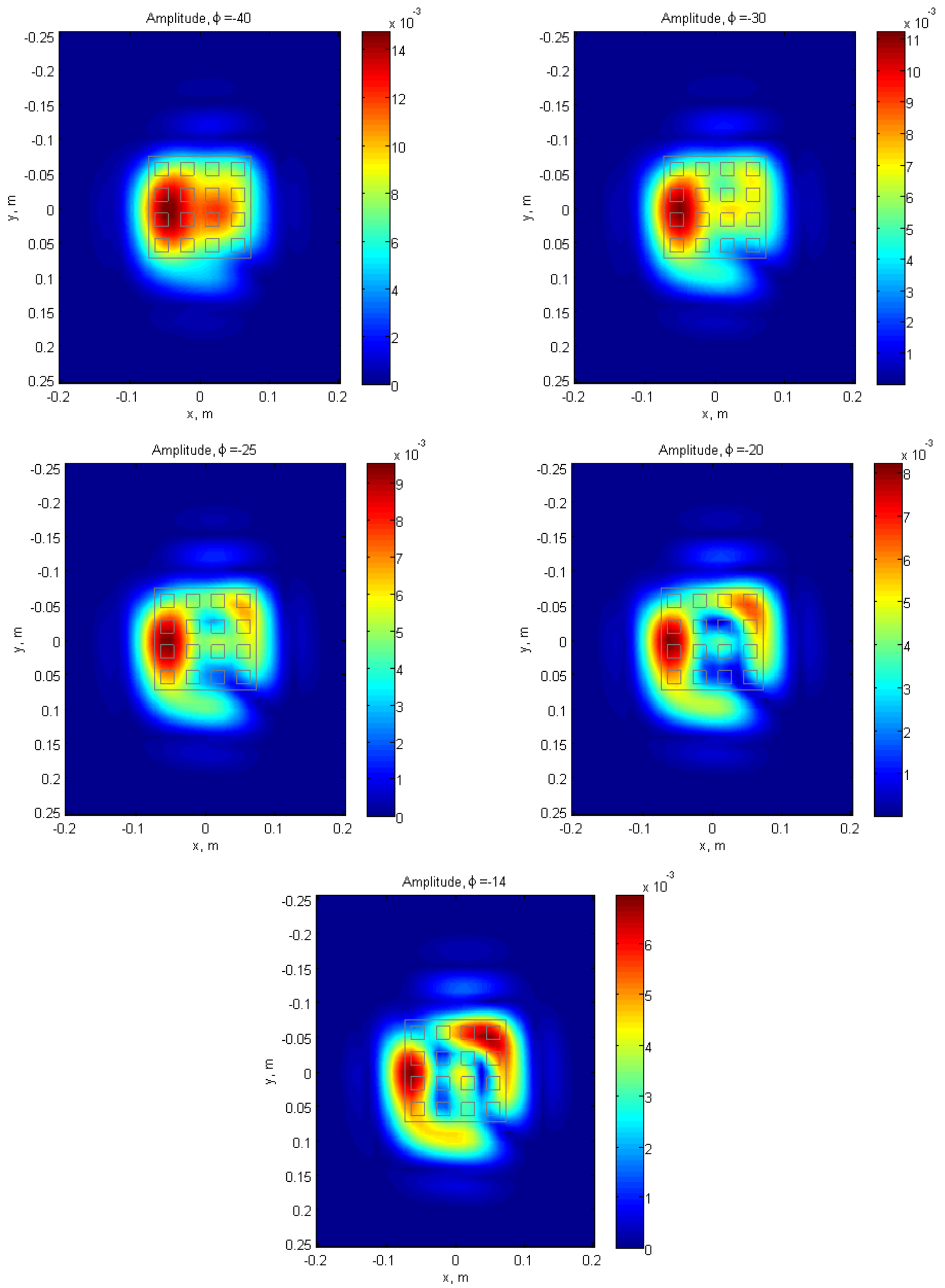
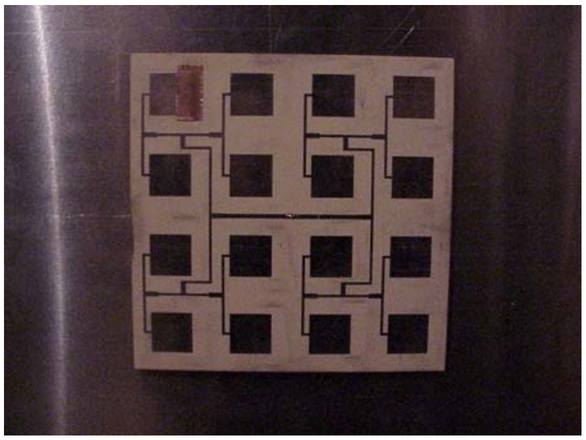
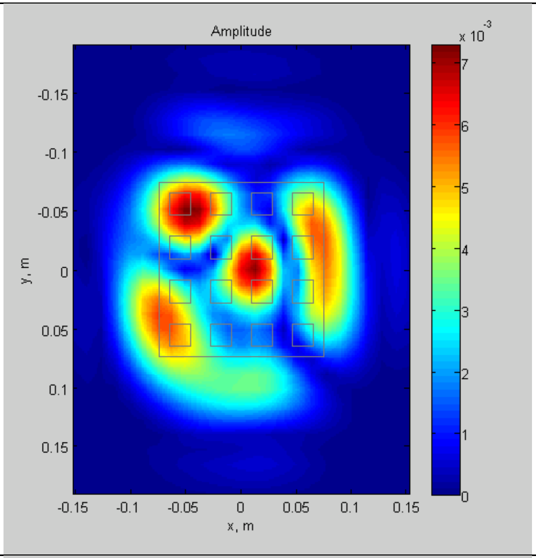
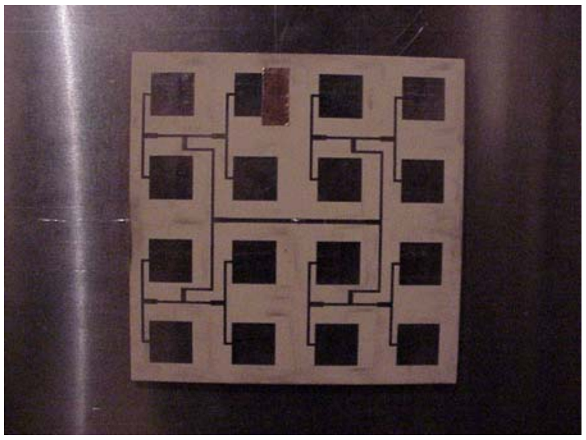
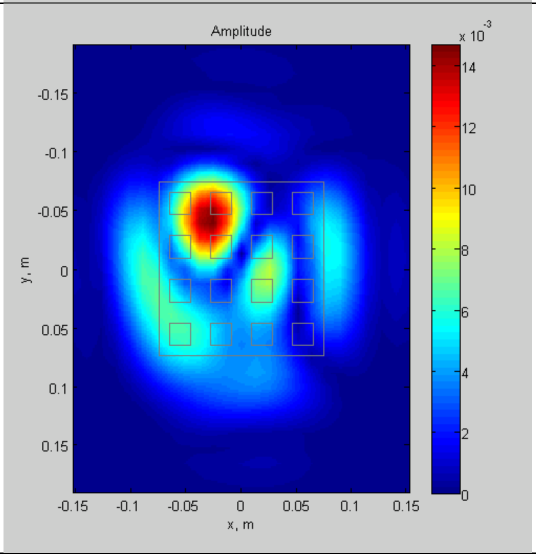


Figure 20 Squared differential hologram of non-faulty measured and simulated (HFSS) data with the simulated data multiplied by a variable phase. From top left, the phases are -40°, -30°, -25°, -20°, -14° (optimal). It can be seen that the measured array may not have been completely parallel with the plane of elevation.

Table 5 shows two cases of differential backpropagation when the non-faulty array was modeled in Ansoft/ANSYS and the faulty array field was measured. The phase alignment was done using the existing measured solution for the non-faulty array. One can see that the method works, but needs improvement.

Table 5 Hybrid backpropagation results (numerical to experiment) – last row.

Array Geometry	Error Field
	
	

Chapter 6: Potential Extension to Array Calibration & Conclusions

Phased arrays must be calibrated/aligned after they are build, as the attenuators and phase shifters used in the design rarely exhibit identical characteristics to one another. Another problem which may arise is faulty solder joints, or geometrical errors which may alter the radiation pattern of an array. For example, if an element is attenuated by 3dB, we see a significant difference in second null of the array factor; the second side lobe amplitude is also affected. This may be unacceptable in some cases, especially if an array has a taper applied to it with specified side lobe levels.

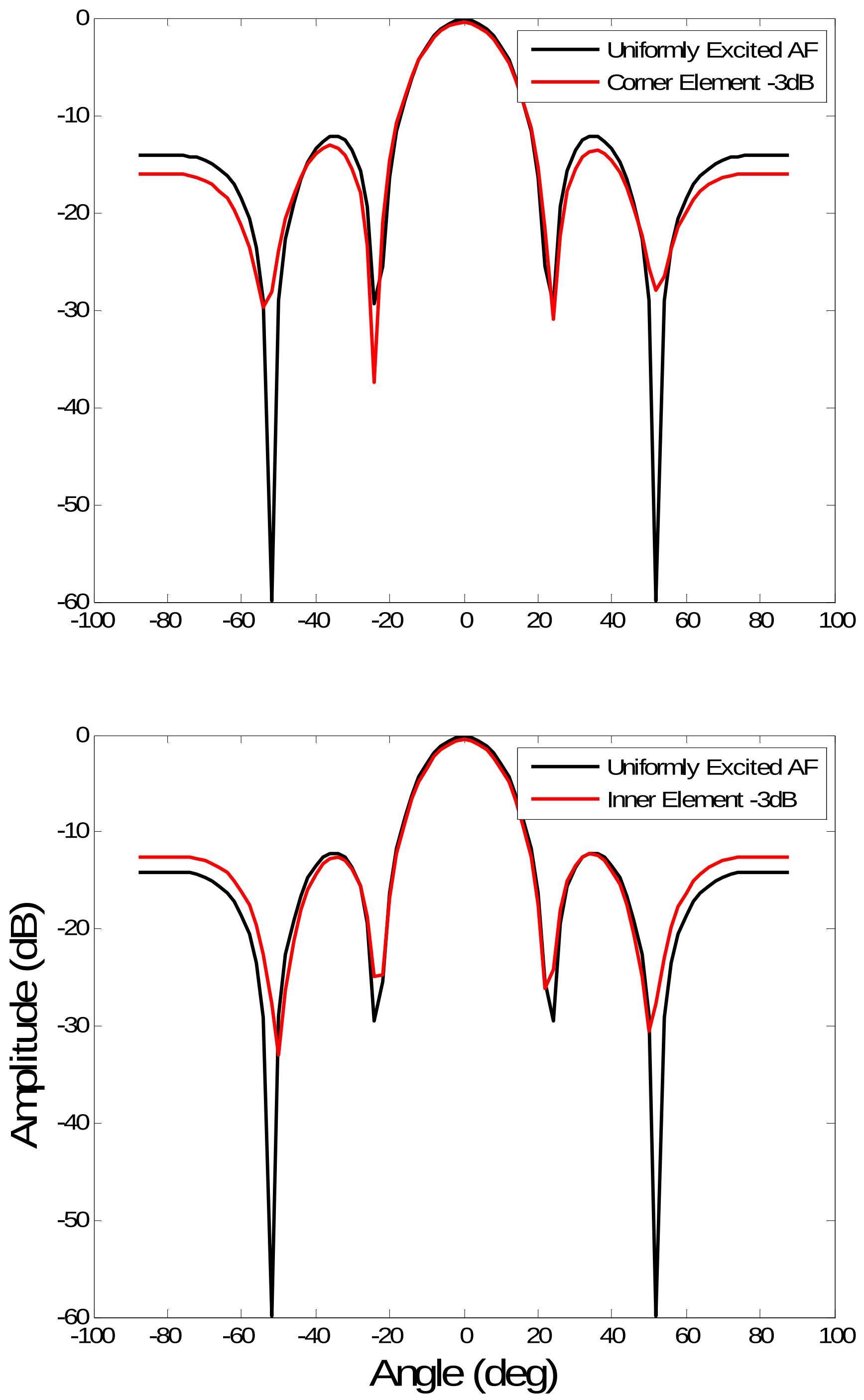


Figure 21 Top - Array factor of uniformly excited 4x4 array with $\lambda / 2$ spacing (black) and array with corner element attenuated by 3dB (red). Bottom - Array factor of uniformly excited 4x4 array with $\lambda / 2$ spacing (black) and array with inner element attenuated by 3dB (red).

The most common calibration techniques consist of measuring the individual settings of each element of an array [11]-[13]. When considering these methods, additional RF components must be included in the array design, such as circulators, detectors, and filters, which may not be desirable in some cases.

Although holography will always remain less accurate compared to element-by-element testing, many of its common errors (standing waves, room scattering, and cable losses) are reduced when the difference between the electric field distributions is utilized. This makes the differential hologram a more reliable method to calibrate faulty elements in an array and an array as a whole. There are also some cases in which element-by-element testing is undesirable altogether, making holography one of the only options. For instance, when aligning broadband arrays that rely on mutual coupling to remain matched at lower frequencies – it is impractical to turn off elements in this case, as this can alter the active impedance of elements and lead to a faulty calibration.

Conclusions

1. We have shown via numerical simulations that differential backpropagation produces convincing results for a small 4×4 patch array of individually fed radiators including all effects of mutual coupling and the effect of a finite measurement plane.
2. These results allow us to uniquely determine a malfunctioning array element with a different excitation phase and/or amplitude at any position in the array.
3. The array spacing may be as low as 0.32λ as long as the measurement plane is located at 2λ or a smaller distance.
4. The experimental results have been given for a 4×4 patch array with a corporate feed. This case is more challenging due to low element isolation. Despite the low isolation, differential backpropagation of measured faulty and non-faulty data identifies the detuned element with accuracy better than or equal to element spacing. In some cases the agreement is exact. The corresponding numerical simulation has confirmed this conclusion.
5. The hybrid approach (numerical data for the template array vs. measured data for a presumably faulty array) shows promise, but needs extra work with phase synchronization.

The above means that with the use of differential microwave holography one may quickly assess the performance of a small patch array. When considering the calibration of arrays, less RF components are necessary when using holography. Although element-by-element techniques are more accurate, they may sometimes be detrimental to array performance when dealing with broadband arrays.

Some challenges of the current method include:

1. Amplitude errors reflected in the error field are simply indicators, not correct values unless the phase of the error field is constant
2. Phase errors are identifiable, but not correct values in the error field in any case.
3. The provision of a calibrated array near-field pattern, either experimentally or numerically, is a must.
4. Phase synchronization between the template pattern and that of the AUT is required.
5. A larger ground/reflector plane is needed.
6. Effects of positioning errors when testing separate units may have a severe influence.

References

[1]. P. L. Ransom and R. Mitra, “A method of locating defective elements in large phased arrays,” *Proceedings of the IEEE-Letters ,* pp. 1029-1030, 1971.

[2]. P. L. Ransom and R. Mitra, “A method of locating defective elements in large phased arrays,” *Phased Array Antennas.* Dedham, MA: Artech House, 1972.

[3]. C.A. Balanis, "Fundamental Parameters of Antennas," *Antenna Theory 3rd edition*. Hoboken, NJ: John Wiley & Sons, Inc., 2005

[4]. D.J. Rochblatt, Y.R.-S.“Effects of measurement errors on microwave antenna holography,” *IEEE Transactions of Antennas and Prop.,* vol 39, no. 7, pp. 933-942, Jul. 1991

[5]. C. A. Rose, “Accuracy estimation of microwave holography from planar near-field measurements,” Microwave Instrumentation Technologies, www.mi-technologies.com

[6]. G.F. Masters, “Hologram accuracy estimation,” AMTA Conference, Nov. 13-17, 1995

[7]. P. K. Agrawal, “A method to compensate for probe positioning errors in an antenna near field test facility,” *in Antenna Propagat. Soc. Symp. Dig., Albuquerque, NM,* May 1982, vol. 1, pp. 218–221.

[8]. A.C. Newell, “Estimating the uncertainties due to truncation in planar near-field holograms,” AMTA Conference, 2004.

[9]. D. J. Rochblatt, B.L. Seidel, “Microwave antenna holography,” *IEEE Transations on Microwave Theory and Techniques*, vol. 40, no. 6, pp. 1294-1300, Jun. 1992.

[10]. W.J. Krzystofik, “Microwave holography,” *Microwaves, Radar and Wireless Communications 2000. MIKON-2000 13th International Conference on.*, pp. 597-600, May 2000.

[11]. R. Sorace, “Phased array calibration,” *IEEE Transactions of Antennas and Prop.,* vol 49, no. 4, pp. 517-525, Apr. 2001.

[12]. W. T. Patton, “Phased array alignment with planar near-field data,” *in Proc. Antenna Applicat. Symp.,* Urbana, IL, Sept. 1981.

[13]. D. K. Alexander and R. P. Gray, Jr., “Computer-aided fault determination for an advanced phased array antenna,” *in Proc. Antenna Applicat. Symp., Urbana, IL,* Sept. 1979.

[14]. J.R. Shewell and E. Wolf, “Inverse diffraction and new reciprocity theorem,” *Journal of the Optical Society of America*, vol. 58, no. 12, pp.1596-1603, Dec. 1968.

[15]. G.C. Sherman, “Integral-transform of diffraction theory,” *Journal of the Optical Society of America*, vol. 57, no. 12, pp. 1490-1498, Dec. 1967.

[16]. F. Shen and A. Wang, “Fast-Fourier-transform based numerical integration method for the Rayleigh-Sommerfeld diffraction formula,” *APPLIED OPTICS*, vol. 45, no. 6, pp. 1102-1110, Feb. 2006.

[17]. A.C. Newell, R.J. Davis, “Holographic projection to an arbitrary plane from spherical near-field measurements,” AMTA Conference, 2002

[18]. Taaghol and T. K. Sarkar, "Near-field to near/far-field transformation for arbitrary field geometry, utilizing an equivalent magnetic current," *IEEE Transactions On Electromagnetic Compatibility*, vol. 38, no. 3, pp. 536-542, Aug. 1996.

[19]. T. K. Sarkar and A. Taaghol, "Near-field to near/far-field transformation for arbitrary near-field geometry utilizing an equivalent electric current and MoM," *IEEE Transactions on Antennas and Propagation,* vol. 47, no. 3, pp. 566-573, Mar. 1999.

[20]. P. Petre and T. K. Sarkar, "Planar near-field to far-field transformation using an equivalent magnetic current approach," *IEEE Transactions of Antennas and Prop.,* vol 40, no. 11, pp. 1348-1356, Nov. 1992.

[21]. L. Klinkenbusch, "Spherical-multipole based time-domain near-field to near-field transformation," *2010 URSI International Symposium on Electromagnetic Theory,* pp. 737-740, 2010.

[22]. K. L. Shlager and G. S. Smith, "Comparison of two FDTD near-field to near-field transformations applied to pulsed antenna problems," *Electronic Letters,* vol. 31, no. 12, pp. 936-938, Jun. 2010.

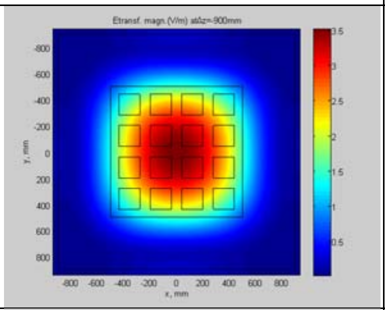
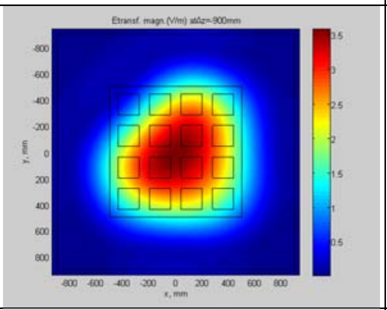
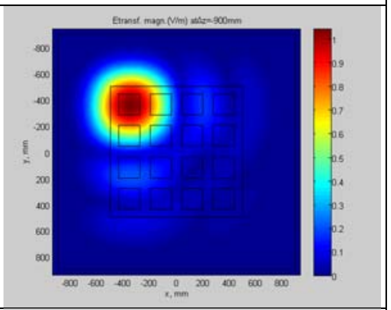
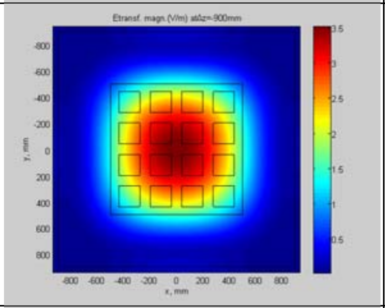
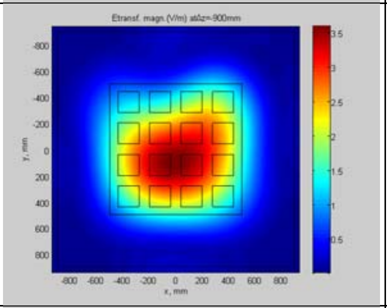
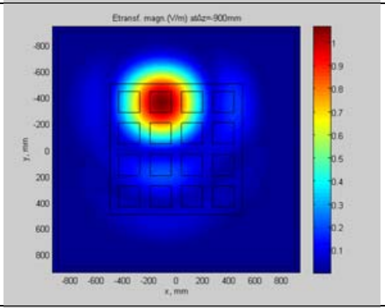
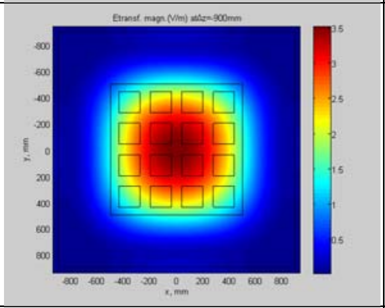
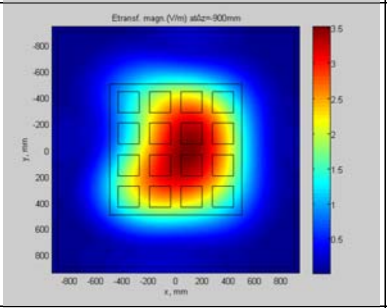
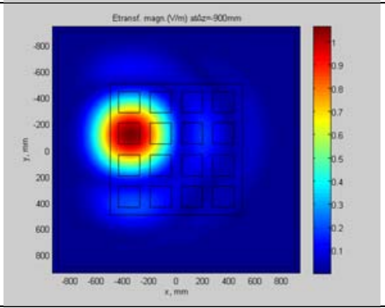
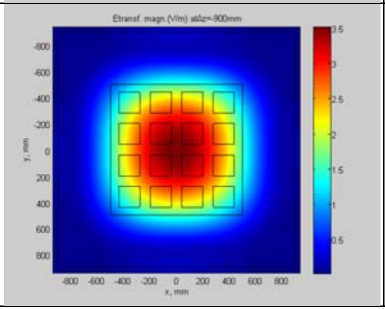
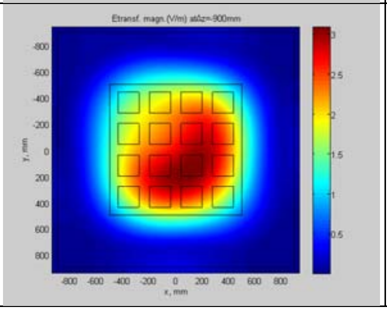
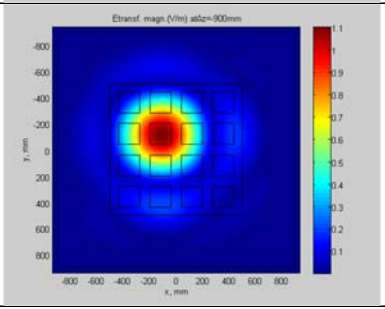
[23]. K. F. Razavi and Y. Rahmat-Samii, "To phase or not to phase in planar near field measurements: overcoming large probe positioning error," *IEEE Antennas and Propagation Society, AP-S International Symposium (Digest)*, 2010.

[24]. M. S. Narasimhan and B. Preetham Kumar, "A technique of synthesizing the excitation currents of planar arrays or apertures," *IEEE Transactions on Antennas and Prop.*, vol. 38, no. 9, pp. 1326-1332, Sep. 1990.

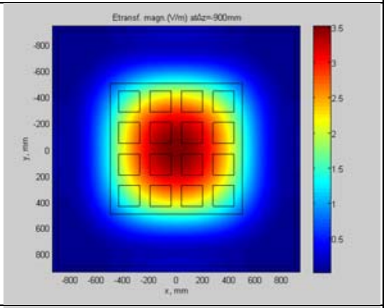
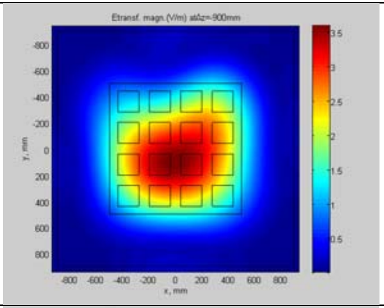
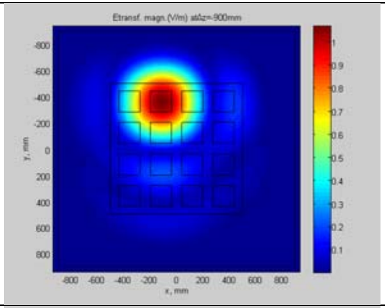
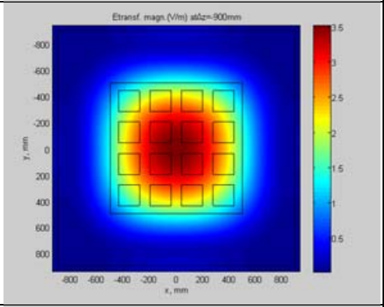
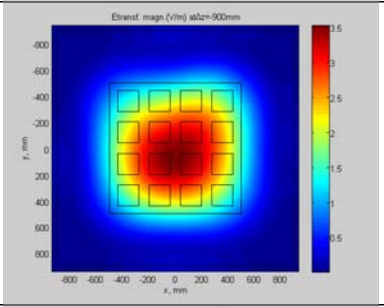
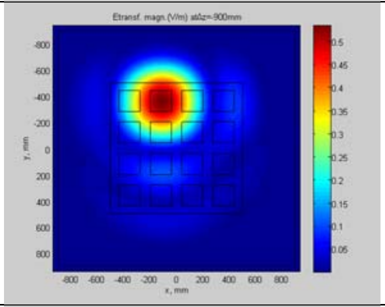
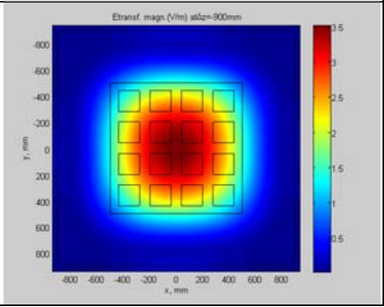
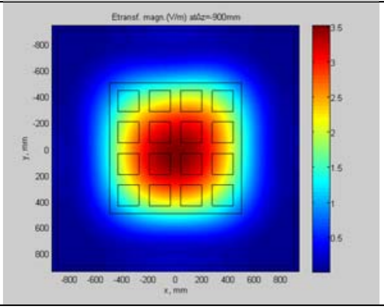
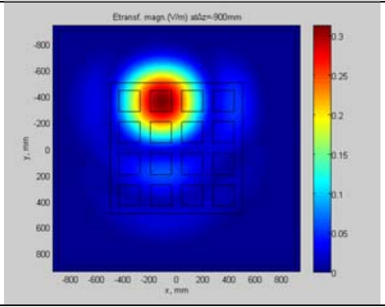
[25]. N.S. Karnik, R. Taulpule, M. Shah, P.s. Verma, C.Y. Huang, J.Y. Cha, A. Pandya, S. Usman, V. Pulipati, P. Pagadala, and B.P. Kumar, "Design, simulation and experimental study of near-field beam forming techniques using conformal waveguide arrays," *IET Microw. Antennas Propag.,* vol. 4, iss. 2, pp. 162-174, 2010.

[26]. G. Noetscher, N. Chow, and S. Makarov, "Modeling accuracy and features of body area networks with out-of-body antennas at 402 MHz, " *IEEE Antennas and Propagation Magazine*, vol. 53, Sep.-Oct. 2011, to appear.

Appendix A: Backpropagation for all array elements.

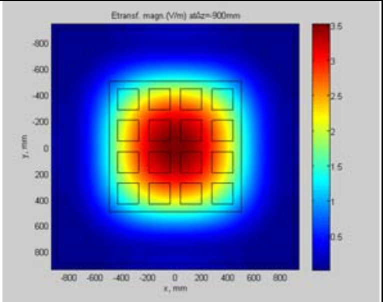
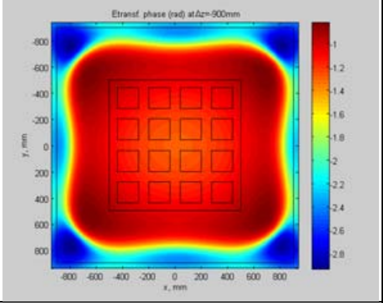
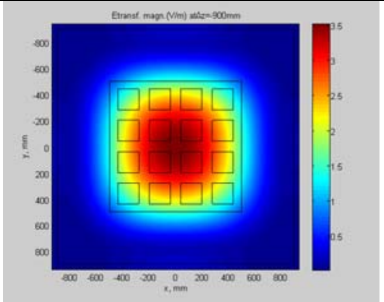
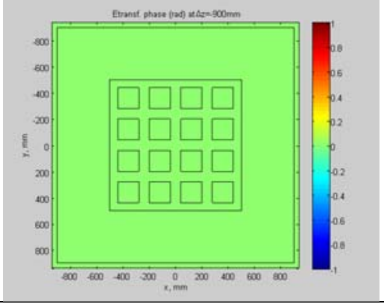
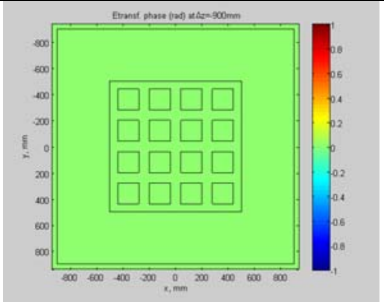
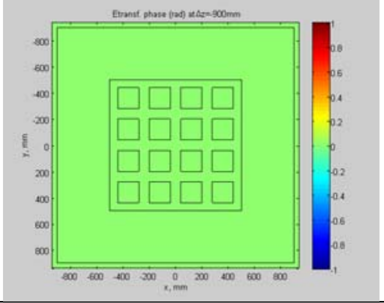
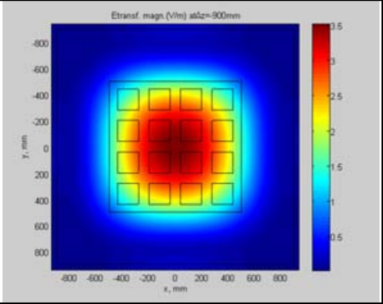
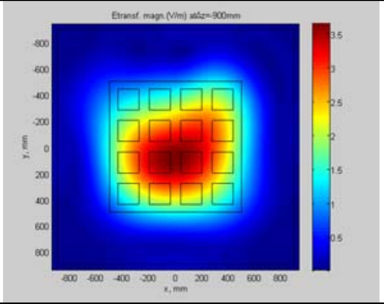
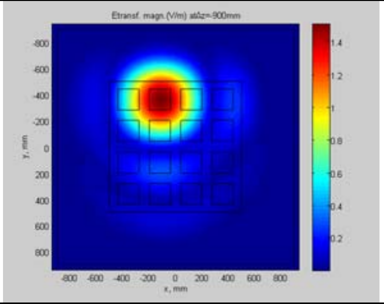
Backpropa-gator	Non-faulty array backpropagated near field	Faulty array backpropagated near field	Difference between the fields of non-faulty and faulty arrays propagated back to λ/8
2λ to λ/8			
2λ to λ/8			
2λ to λ/8			
2λ to λ/8			

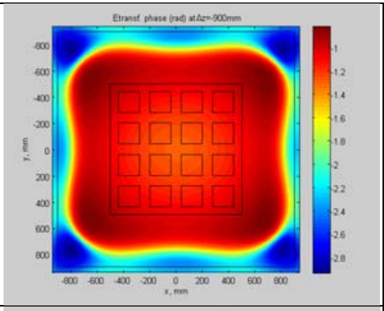
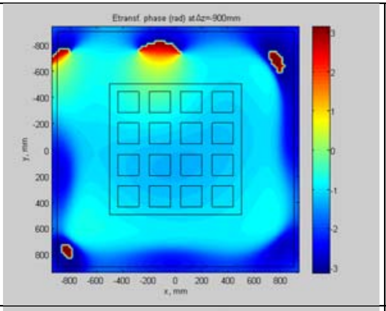
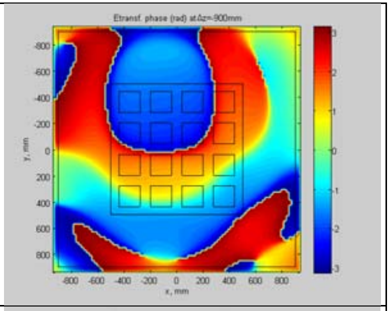
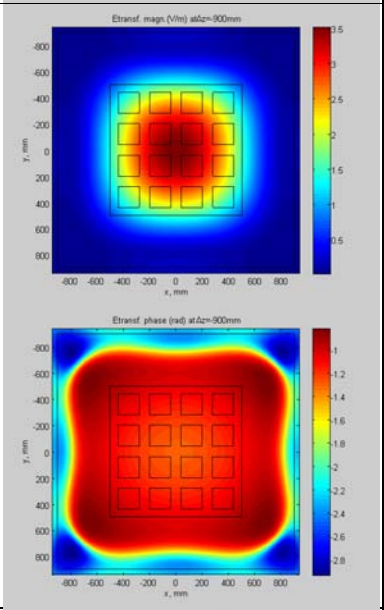
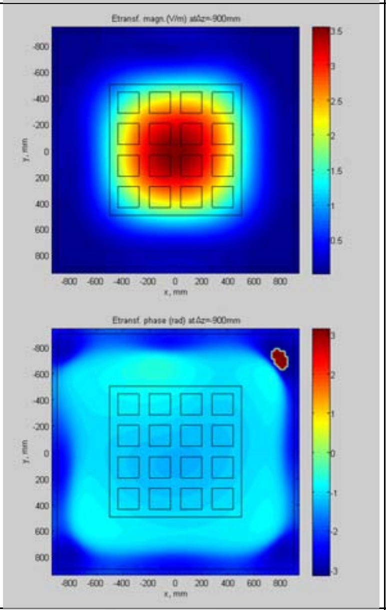
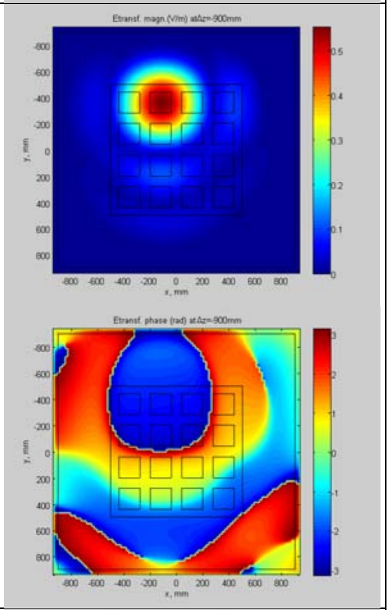
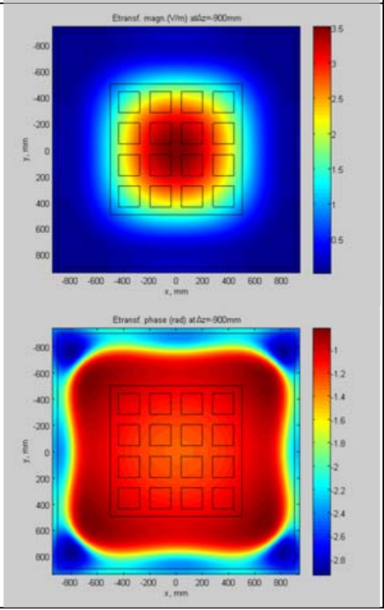
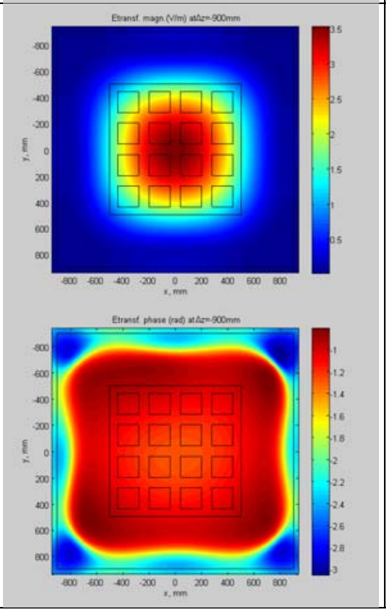
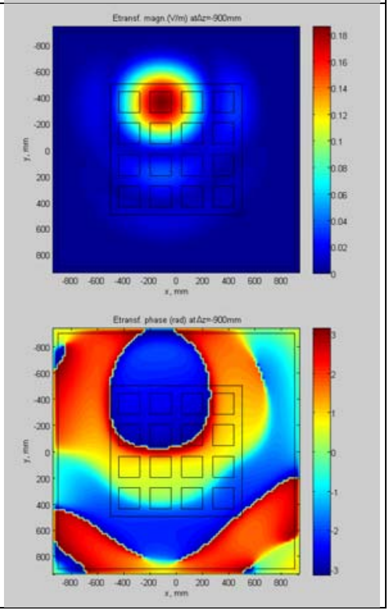
Appendix B: Backpropagation for partially attenuated element.

Attenuation	Non-faulty array backpropagated near field	Faulty array backpropagated near field	Difference between the fields of non-faulty and faulty arrays propagated back to λ/8
-∞dB			
-6dB			
-3dB			

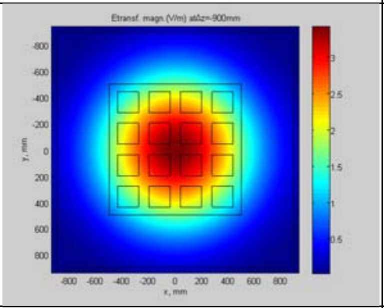
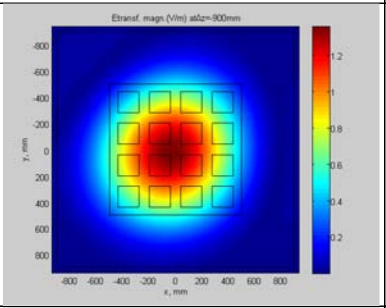
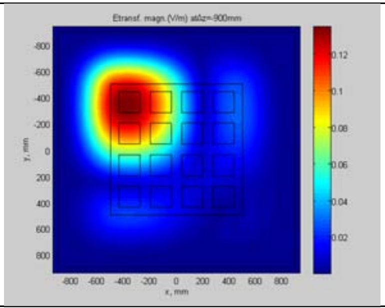
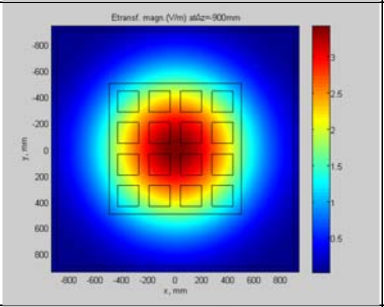
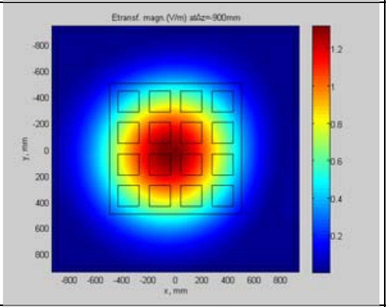
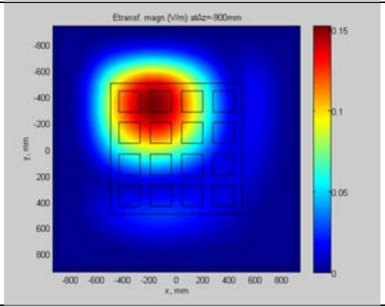
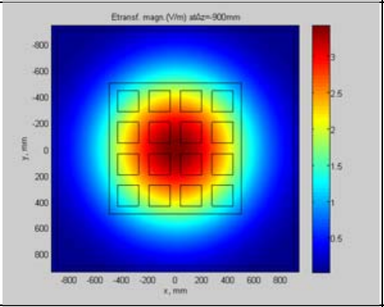
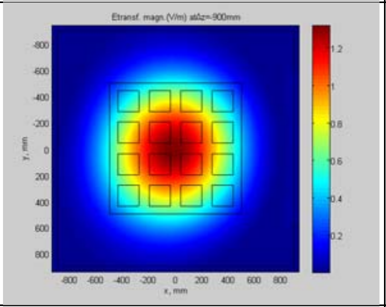
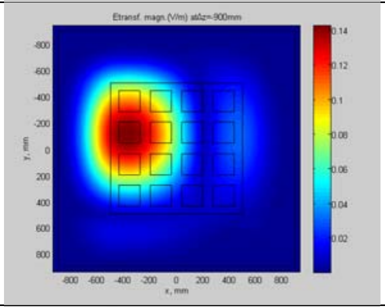
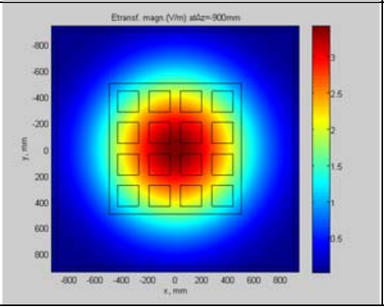
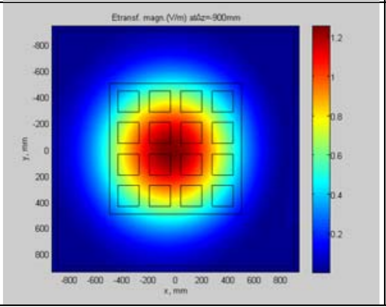
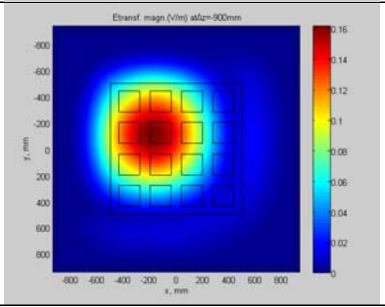
Appendix C: Backpropagation for element partially out of phase elements

(In every cell corresponding to a given phase, the first row corresponds to amplitude and the second row corresponds to phase).

Phase Shift	Non-faulty array backpropagated near field	Faulty array backpropagated near field	Difference between the fields of non-faulty and faulty arrays propagated back to λ/8
0°	 	 	 
90°			

	Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. phase (rad) at Δz=-900mm y, mm x, mm
30°	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm
10°	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm	Etransf. magn (V/m) at Δz=-900mm Etransf. phase (rad) at Δz=-900mm y, mm x, mm

Appendix D: Backpropagation for all array elements with 0.32λ spacing.

Backpropagator	Non-faulty array backpropagated near field	Faulty array backpropagated near field	Difference between the fields of non-faulty and faulty arrays propagated back to λ/8
2λ to λ/8			
2λ to λ/8			
2λ to λ/8			
2λ to λ/8			

Appendix E: FDTD MATLAB Codes

main.m

```
%   FDTD MATLAB antenna/array solver (preliminary version)
%   Copyright SNM Spring 2011
%   MAIN SCRIPT
clear all;

%%  Read project file from subfolder PROJECTS (see file format) and visialize
project geometry
[FileName,PathName,FilterIndex] = uigetfile('projects/*.m','Select the MATLAB
project file');
run(strcat(PathName, FileName));

constructor;
save input;

%%  Execute FDTD script (either as a script or as a function)
fdtd;
if ~isempty(custom)
    return;
else
    close all;
end
%   Its output includes:
%   A. Currents/voltages and self/mutual impedances  of the ports (for total
M ports)
%   This data is given for any excitation type (pulse or CW):
%   AntI    = zeros(M, M, length(t));          %   antenna currents for all
ports
%   AntV    = zeros(M, M, length(t));          %   antenna voltages for all
ports
%   Z       = zeros(length(findex), M, M);     %   port impedance matrix in
frequency domain (at all frequencies)
%   B. Fields on all boundaries and in the observation plane
%   This data is only given for CW excitation (complex Fourier coefficients
%   of the fields are given):
%   Extop = zeros(Nx  , Ny+1);                 %    Ex in the xy-observation
plane
%   Eytop = zeros(Nx+1, Ny);                   %    Ey in the xy-observation
plane
%   Hxtop = zeros(Nx+1, Ny);                   %    Hx in the xy-observation
plane
%   Hytop = zeros(Nx  , Ny+1);                 %    Hy in the xy-observation
plane

load output;

%%  Interpolate, plot, and save port impedances (total M ports) into Zout
if (length(f)>1)&(f(end)>f(1))  %   frequency sweep - pulse excitation
    for m = 1:M
        Zout(:, m, m) = interp1(ftemp, Z(:, m, m), f);  %   still a row here
    end
    scrsz = get(0,'ScreenSize');
    figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
    if M<=4     FontSize = 10;   end;
    if M<=16    FontSize = 8;    end;
    if M>16     FontSize = 7;    end;
    for m = 1:RowNo(end)
        for n = 1:ColNo(end)
            index = n+(m-1)*ColNo(end);
            subplot(RowNo(end), ColNo(end), index);
            temp = squeeze(Zout(:, index, index));
            plot(f/1e6, real(temp), 'r', f/1e6, imag(temp), 'b'); grid on;
            string = strcat('Port', num2str(m), num2str(n));
            title (strcat(string, ': R-red, X-blue,
\Omega'),'FontSize',FontSize);
            xlabel('freq, MHz','FontSize',FontSize);
            set(gca,'FontSize',FontSize);
            axis square; axis tight;
        end
    end
    scrsz = get(0,'ScreenSize');
    figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
    if M<=4     FontSize = 10;   end;
    if M<=16    FontSize = 8;    end;
    if M>16     FontSize = 7;    end;
    for m = 1:RowNo(end)
        for n = 1:ColNo(end)
            index = n+(m-1)*ColNo(end);
            subplot(RowNo(end), ColNo(end), index);
            temp = squeeze(Zout(:, index, index));
            temp = 20*log10(abs((temp-R0)./(temp+R0)));
            plot(f/1e6, real(temp), 'r', f/1e6, imag(temp), 'b'); grid on;
            string = strcat('S', num2str(m), num2str(n));
            title (strcat(string, ': Refl. coeff., dB'),'FontSize',
FontSize);
            xlabel('freq, MHz','FontSize', FontSize);
            set(gca, 'FontSize', FontSize);
            axis square; axis tight;
        end
    end
    scrsz = get(0,'ScreenSize');
    figure('Position', [1 0.3*scrsz(4) 0.3*scrsz(3) 0.3*scrsz(4)]);
    RealizedG = zeros(length(temp),1);
    for m = 1:RowNo(end)
        for n = 1:ColNo(end)
            index = n+(m-1)*ColNo(end);
            temp = squeeze(Zout(:, index, index));
            GAMMA =(temp-R0)./(temp+R0);
            RealizedG = RealizedG + (1 - abs(GAMMA).^2);
        end
    end
    %RealizedG(find(RealizedG<0)) =0.001;
    RealizedG = 10*log10(RealizedG/index);
    plot(f/1e6, RealizedG, 'k'); grid on;
    title ('Difference between realized gain and ideal gain, dB');
    xlabel('freq, MHz');
    axis square; axis tight;
else                            %   single frequency - CW
   for m = 1:M
       Zout(RowNo(m), ColNo(m)) = Z(:, m, m);   %   a matrix here
   end
end

%%  Plot radiation patterns, near/far fields, and impedances at a single
frequency
if length(f)==1
   if plane
        plot_nearfield;
   end
end

%%  Save project data
%%   Data output into file
ftemp = f; Ztemp = Zout;
save(strcat(PathName, strcat(FileName(1:end-2),'.mat')));
```

array_4x4_cw.m

```
%   FDTD MATLAB antenna/array solver (preliminary version)
%   Copyright SNM Spring 2011
%   PROJECT FILE: geometry/FDTD domain/frequency/timing/terminations
%%  Geometry
%   Geometry construction is based on individual objects. The number of
objects is arbitrary.
%   Template for an individual object includes a top row with 15 numbers
separated by commas.
%   Any object may be
%       i. a brick material (diel, metal, etc.);
%       ii. a rectangular sheet (metal);
%       iii. or a line (ports only).
%   Template for the top row:
```

```
%   3,                        %   brick width, mm   (x)
%   3,                        %   brick length, mm  (y)
%   0.5,                      %   brick height, mm  (z)
%   0.0,                      %   brick center, mm (x)
%   0.0,                      %   brick center, mm (y)
%   0.0,                      %   brick center, mm (z)
%   0,                        %   is metal (0 or 1)
%   0,                        %   is port (0-not a port; 1- port with Ex; 2 -
port with Ey; 3 - port with Ez)
%   1,                        %   relative permittivity
%   0.0,                      %   electric conductivity, S/m
%   1,                        %   relative permeability
%   0.0,                      %   magnetic conductivity, S/m
%   0,                                  %   object color(R)
%   1,                                  %   object color(G)
%   0,                                  %   object color(B)
%   Template for an object also includes three remaining rows: xyz-positions
for
%   object translational copies - clones. Template for three remaining rows:
%   0, 20, 40,      % x-translations for copies (use 0, if the object is not
cloned)
%   0, 20, 40,      % y-translations for copies (use 0, if the object is not
cloned)
%   0, 20, 40,      % z-translations for copies (use 0, if the object is not
cloned)
custom = [];

%   This is the dielectric brick
objectd{1} = [1000, 1000, 20, +000, 0,  -320,  0,  0,  2.0, 0, 1, 0,  1.0,
1.0, 0.0];
objectx{1} = [0];
objecty{1} = [0];
objectz{1} = [0];

%   This is the metal ground plane
```

```
objectd{2} = [1800, 1800, 0,  +000, 0,  -340, 1, 0,  1,  0, 1, 0, 0.5, 0.5,
0.5];
objectx{2} = [0];
objecty{2} = [0];
objectz{2} = [0];

%   This is the patch feed (for a square 4x4 array)
objectd{3} = [0,  0,  20,  60,  0, -320,  0, 3,  1, 0, 1, 0, 1.0, 0.0, 0.0];
objectx{3} = [-360, -120, +120, +360, -360, -120, +120, +360, -360, -120,
+120, +360, -360, -120, +120, +360];
objecty{3} = [-360, -360, -360, -360, -120, -120, -120, -120, +120, +120,
+120, +120, +360, +360, +360, +360];
objectz{3} = zeros(1, 16);

%   This is the top patch (for a square 4x4 array)
objectd{4} = [160, 160, 0,  0,  0, -320, 1, 0, 1, 0, 1, 0,  0.5, 0.5, 0.5];
objectx{4} = [-360, -120, +120, +360, -360, -120, +120, +360, -360, -120,
+120, +360, -360, -120, +120, +360];
objecty{4} = [-360, -360, -360, -360, -120, -120, -120, -120, +120, +120,
+120, +120, +360, +360, +360, +360];
objectz{4} = zeros(1, 16);

%%   FDTD domain data
W0  = 1880;                           %   volume width, mm   (x)
L0  = 1880;                           %   volume length, mm  (y)
H0  = 1200;                           %   volume height, mm  (z)
XC0 = 0;                              %   volume center, mm   (x)
YC0 = 0;                              %   volume center, mm  (y)
ZC0 = 100;                            %   volume center, mm  (z)
D   = 20;                             %   cell size, mm (in all directions)

%%   Frequency sweep data/FDTD timing data
fstart  = 625e6;                      %  start frequency of the frequency
loop, Hz
fstop   = 625e6;                      %  stop frequency of the frequency
loop, Hz
```

```
steps   = 1;                          %  # of frequency steps
TO      = 15e-9;                      %  maximum total FDTD running time
(per port), s
R0      = 50;                         %  generator/load impedance, Ohm
NoP     = 16;                         %  number of ports (total)
fields  = 0;                          %  plot (1) or not (0) instantaneous
fields
component = 5;                        %  component to plot if any (1-Ex, 2-
Ey, 3-Ez, 4-Hx, 5-Hy, 6-Ez)
power    = 0.25;                      %  power factor
(abs(Ez)^power*sign(Ez) is plotted)
scale    = 0.6;                       %  scale factor to plot (to
abs(Ez)^power)
plane   = 1;                          %  plot (1) or not (0) near fields

planeZ  = -260;                        %  absolute height of the observation
plane for near fields, mm (0.125*lambda)
planeZ  = -200;                        %  absolute height of the observation
plane for near fields, mm (0.250*lambda)
planeZ  = -080;                        %  absolute height of the observation
plane for near fields, mm (0.500*lambda)
planeZ  = +160;                        %  absolute height of the observation
plane for near fields, mm (1.000*lambda)
planeZ  = +400;                        %  absolute height of the observation
plane for near fields, mm (1.500*lambda)
planeZ  = +640;                        %  absolute height of the observation
plane for near fields, mm (2.000*lambda)

%%   Port termination data/Scan data (only for antenna arrays)
%   TERMINATION DATA
%   Termination data is given by two matrixes: TerminationV (voltage),
TerminationY (admittance)
%   Both matrixes must have the size of Nrow by Ncol where Nrow is the number
of port rows
%   (along the x-axis) and  Ncol is the number of port columns (along the y-
axis)
```

```
%   TERMINATION TYPES
%   Three port termination types may be used:
%   1. Terminations for the standard impedance matrix
%   (one element excited with all other open circuited)
%   2. Terminations for the active impedance of an array
%   (one element excited with all other match terminated)
%   3. Terminations for the scan impedance of an array
%   (all element are excited with proper phases)
%   TerminationV = 0; TerminationY = 0;     %   for impedance matrix (simple
antenna)
%   TerminationV = 0; TerminationY = 1;     %   for active impedance matrix
(antenna array)
%   TerminationV = 1; TerminationY = 1;     %   for scan impedance matrix
(antenna array)
%   TerminationV may be complex - complex excitations weights)
TerminationV = [1,1,1,1;...
                1,0,1,1;...
                1,1,1,1;...
                1,1,1,1];

TerminationY = [1,1,1,1;...
                1,1,1,1;...
                1,1,1,1;...
                1,1,1,1];
%   Scan angles in deg; (default values are zeros)
xscan       = 0;      %  progressive phase shift along the x-axis, deg (for
antenna array only)
yscan       = 0;      %  progressive phase shift along the y-axis, deg (for
antenna array only)
```

constructor.m

```
%   FDTD MATLAB antenna/array geometry constructor (preliminary version)
%   Copyright SNM Spring 2011
%   GEOMETRY CONSTRUCTION SCRIPT
%   Domain/grid construction from geometry data
```

```
%%  Constructor 1

x   = [-W0/2+XC0:D:W0/2+XC0];           %   x-grid
y   = [-L0/2+YC0:D:L0/2+YC0];           %   y-grid
z   = [-H0/2+ZC0:D:H0/2+ZC0];           %   z-grid

Nx = length(x)-1;                       %   number of cells in the x-
direction
Ny = length(y)-1;                       %   number of cells in the y-
direction
Nz = length(z)-1;                       %   number of cells in the z-
direction

%   Grid patches in the xy-plane
XYX = zeros(5, Nx*Ny);
XYY = zeros(5, Nx*Ny);
for m = 1:Nx
    xmin = x(m);
    for n = 1:Ny
        ymin = y(n);
        index = n + (m-1)*Ny;
        XYX(1:5, index) = [xmin xmin xmin+D xmin+D xmin];
        XYY(1:5, index) = [ymin ymin+D ymin+D ymin ymin];
    end
end
%   Grid patches in the xz-plane
XZX = zeros(5, Nx*Nz);
XZZ = zeros(5, Nx*Nz);
for m = 1:Nx
    xmin = x(m);
    for n = 1:Nz
        zmin = z(n);
        index = n + (m-1)*Nz;
        XZX(1:5, index) = [xmin xmin xmin+D xmin+D xmin];
        XZZ(1:5, index) = [zmin zmin+D zmin+D zmin zmin];
    end
```

```
end
%   Grid patches in the yz-plane
YZY = zeros(5, Ny*Nz);
YZZ = zeros(5, Ny*Nz);
for m = 1:Ny
    ymin = y(m);
    for n = 1:Nz
        zmin = z(n);
        index = n + (m-1)*Nz;
        YZY(1:5, index) = [ymin ymin ymin+D ymin+D ymin];
        YZZ(1:5, index) = [zmin zmin+D zmin+D zmin zmin];
    end
end


%%  Constructor2
%   Convert initial geometry data to arrays, snap to grid all geometry
objects
%   Convert thin bodies to faces, convert narrow faces to lines
%   Read initial geometry/composition data
index = 0;
for m = 1:size(objectd, 2)
    temp  = cell2mat(objectd(m));
    tempx = cell2mat(objectx(m));
    tempy = cell2mat(objecty(m));
    tempz = cell2mat(objectz(m));
    for n = 1:length(tempx)
        index = index + 1;
        Objects(index)  = index;
        W(index)        = temp(1);
        L(index)        = temp(2);
        H(index)        = temp(3);
        XC(index)       = temp(4) + tempx(n);
        YC(index)       = temp(5) + tempy(n);
        ZC(index)       = temp(6) + tempz(n);
        Ismet(index)    = temp(7);
        Isport(index)   = temp(8);
        Eps(index)      = temp(9);
        SigE(index)     = temp(10);
        Mu(index)       = temp(11);
        SigM(index)     = temp(12);
        Color(index, :) = [temp(13) temp(14) temp(15)];
    end
end

%   Internal variables
Wtemp = W;
Ltemp = L;
Htemp = H;
%   Convert all narrow faces/bricks to lines/faces (internally only)
threshold = 1.5*D;
for m = 1:length(Objects)
    if ~Isport(m)&(min([W(m) L(m) H(m)])==0)     %  face found
        if (W(m)<threshold)&(W(m)>0)
            Wtemp(m) = 0;
        end
        if (L(m)<threshold)&(L(m)>0)
            Ltemp(m) = 0;
        end
        if (H(m)<threshold)&(H(m)>0)
            Htemp(m) = 0;
        end
    end
end
%   Introduce line indicator
for m = 1:length(Objects)
    Isline(m) = 0;
    if (W(m)==0)&(L(m)==0)
        Isline(m) = 3;
    end
    if (W(m)==0)&(H(m)==0)
        Isline(m) = 2;
    end
    if (L(m)==0)&(H(m)==0)
        Isline(m) = 1;
    end
end

%   Snap all objects (port lines, faces, bricks) to grid:
%   Replace physical boundaries by indexes into nearest integer cell nodes
for m = 1:length(Objects)
    [dummy, N1X(m)] = min(abs(XC(m)-Wtemp(m)/2-x-eps)); %  left boundary
index
    [dummy, N2X(m)] = min(abs(XC(m)+Wtemp(m)/2-x-eps)); %  right boundary
index
    [dummy, N1Y(m)] = min(abs(YC(m)-Ltemp(m)/2-y-eps)); %  front boundary
index
    [dummy, N2Y(m)] = min(abs(YC(m)+Ltemp(m)/2-y-eps)); %  rear boundary
index
    [dummy, N1Z(m)] = min(abs(ZC(m)-Htemp(m)/2-z-eps)); %  bottom boundary
index
    [dummy, N2Z(m)] = min(abs(ZC(m)+Htemp(m)/2-z-eps)); %  top boundary index

    XCtemp(m) = (x(N1X(m))+ x(N2X(m)))/2;          %  center alignment
    YCtemp(m) = (y(N1Y(m))+ y(N2Y(m)))/2;          %  center alignment
    ZCtemp(m) = (z(N1Z(m))+ z(N2Z(m)))/2;          %  center alignment
end

[dummy, planeNZ] = min(abs(planeZ-z-eps));

for m = 1:length(Objects)
     if (N1X(m)<=1)|(N1X(m)>=Nx) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
     if (N2X(m)<=1)|(N2X(m)>=Nx) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
     if (N1Y(m)<=1)|(N1Y(m)>=Ny) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
     if (N2Y(m)<=1)|(N2Y(m)>=Ny) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
```

```
     if (N1Z(m)<=1)|(N1Z(m)>=Nz) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
     if (N2Z(m)<=1)|(N2Z(m)>=Nz) error('Antenna/array structure is not within
the ABC boundary (check FDTD domain)!'); end;
end

%%  Plotter1
%   Visualize antenna/array geometry
%   Display resulting objects and FDTD mesh alignment (XY, XZ, YZ planes)
% %   Plot a 3D figure first
scrsz = get(0,'ScreenSize');
figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);

for m = 1:length(W)
    Transp = 0.5;
    wtemp = W(m);
    ltemp = L(m);
    htemp = H(m);
    if ~Isport(m)&(min([W(m) L(m) H(m)])==0)     %  face found
        Transp = 1;
        if (wtemp==0)
            wtemp = 0.001*max([ltemp htemp]);
        end
        if (ltemp==0)
            ltemp = 0.001*max([wtemp htemp]);
        end
        if (htemp==0)
            htemp = 0.001*max([ltemp wtemp]);
        end
    end
    if ~Isline(m)
        viewer(wtemp, ltemp, htemp, XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), Transp, 1, 'k');
    else
        viewer(Wtemp(m), Ltemp(m), Htemp(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), Transp, 0.25, 'k');
```

```
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
viewer(W0, L0, H0, XC0, YC0, ZC0, [1 1 1], 0, 1, 'k');
axis('equal'); axis('tight');  view(-63, 40);
xlabel('x, mm'); ylabel('y, mm'); zlabel('z, mm');
grid on; title('Project geometry - hit ENTER', 'FontWeight', 'bold');
pause; close gcf;

%   Plot three grid projections next
scrsz = get(0,'ScreenSize');
a=figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
POSF = get(a, 'Position');
sp1 = subplot(1,3,1);
    patch(XYX, XYY, zeros(size(XYX)), 'w', 'EdgeColor', [0.5 0.5 0.5]);
    for m = 1:length(W)
        if Ismet(m)
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 1.5, 'k');
        elseif Isline(m)
            viewer(Wtemp(m), Ltemp(m), Htemp(m), XCtemp(m), YCtemp(m),
ZCtemp(m), Color(m, :), 0.5, 0.25, 'k');
        else
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 0.5, 'k');
        end
    end
    if ~isempty(custom)
        patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5],
'FaceAlpha', 0.5);
    end
    viewer(W0, L0, H0, XC0, YC0, ZC0, [1 1 1], 0, 1, 'k');
    axis('equal'); axis('tight');  view(0, 90);
    title('XY-plane'); xlabel('x, mm'); ylabel('y, mm');
sp2 = subplot(1,3,2);
    patch(XZX, zeros(size(XZX)), XZZ, 'w', 'EdgeColor', [0.5 0.5 0.5]);
    for m = 1:length(W)
        if Ismet(m)
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 1.5, 'k');
        elseif Isline(m)
            viewer(Wtemp(m), Ltemp(m), Htemp(m), XCtemp(m), YCtemp(m),
ZCtemp(m), Color(m, :), 0.5, 0.25, 'k');
        else
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 0.5, 'k');
        end
    end
    if ~isempty(custom)
        patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5],
'FaceAlpha', 0.5);
    end
    viewer(W0, L0, H0, XC0, YC0, ZC0,    [1 1 1],  0, 1, 'k');
    if plane
        viewer(W0, L0, 0,  XC0, YC0, planeZ, [1 0 1],  0, 1, 'g');
    end
    axis('equal'); axis('tight');  view(0, 0);
    title('XZ-plane'); xlabel('x, mm'); zlabel('z, mm');
    text(-0.5, 1.75, 'Three projections and FDTD grid-hit Enter',
'FontWeight', 'bold', 'Units', 'normalized')
sp3 = subplot(1,3,3);
    patch(zeros(size(YZY)), YZY, YZZ, 'w', 'EdgeColor', [0.5 0.5 0.5]);
    for m = 1:length(W)
        if Ismet(m)
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 1.5, 'k');
        elseif Isline(m)
            viewer(Wtemp(m), Ltemp(m), Htemp(m), XCtemp(m), YCtemp(m),
ZCtemp(m), Color(m, :), 0.5, 0.25, 'k');
```

```
        else
            viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m),
Color(m, :), 0.5, 0.5, 'k');
        end
    end
    if ~isempty(custom)
        patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5],
'FaceAlpha', 0.5);
    end
    viewer(W0, L0, H0, XC0, YC0, ZC0, [1 1 1],  0, 1, 'k');
    if plane
        viewer(W0, L0, 0,  XC0, YC0, planeZ, [1 0 1],  0, 1, 'g');
    end
    axis('equal'); axis('tight');  view(-90, 0);
    title('YZ-plane'); ylabel('y, mm'); zlabel('z, mm');
pause; close gcf;
```

viewer.m

```
function [] = viewer(W, L, H, XC, YC, ZC, Color, Transparency, LineWidth,
EdgeColor)
%   FDTD MATLAB antenna/array geometry constructor (preliminary version)
%   Copyright SNM Spring 2011
%   PLOT OF A RECTANGULAR OBJECT(a plane, a brick, or a line)
%   __________W__________
%   |         |y         |
%   |L        |          |
%   |         *(XC,YC,ZC)
%   |         |          |
%   |_________|_________|x
%

    if (W>0) & (L>0)
        hr = patch([-W/2 -W/2 +W/2 +W/2]+XC, [-L/2 +L/2 +L/2 -L/2]+YC, [-H/2
-H/2 -H/2 -H/2]+ZC, ...
```

```
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   bottom
        hr = patch([-W/2 -W/2 +W/2 +W/2]+XC, [-L/2 +L/2 +L/2 -L/2]+YC, [+H/2
+H/2 +H/2 +H/2]+ZC, ...
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   top
    end
    if (L>0) & (H>0)
        hr = patch([+W/2 +W/2 +W/2 +W/2]+XC, [-L/2 -L/2 +L/2 +L/2]+YC, [-H/2
+H/2 +H/2 -H/2]+ZC, ...
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   right
        hr = patch([-W/2 -W/2 -W/2 -W/2]+XC, [-L/2 -L/2 +L/2 +L/2]+YC, [-H/2
+H/2 +H/2 -H/2]+ZC, ...
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   left
    end
    if (W>0) & (H>0)
        hr = patch([-W/2 -W/2 +W/2 +W/2]+XC, [+L/2 +L/2 +L/2 +L/2]+YC, [-H/2
+H/2 +H/2 -H/2]+ZC, ...
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   back
        hr = patch([-W/2 -W/2 +W/2 +W/2]+XC, [-L/2 -L/2 -L/2 -L/2]+YC, [-H/2
+H/2 +H/2 -H/2]+ZC, ...
            Color, 'FaceAlpha', Transparency, 'LineWidth', LineWidth,
'EdgeColor', EdgeColor);   %   front
    end
    if (L==0)&(H==0)    %  x-line
        hrl = line([-W/2+XC W/2+XC], [YC YC], [ZC ZC], 'LineWidth', 3,
'Color', Color);
    end
    if (W==0)&(H==0)    %  y-line
        hrl = line([XC XC], [-L/2+YC L/2+YC], [ZC ZC], 'LineWidth', 3,
'Color', Color);
    end
    if (W==0)&(L==0)    %  z-line
```

```
        hrl = line([XC XC], [YC YC], [-H/2+ZC H/2+ZC], 'LineWidth', 3,
'Color', Color);
    end
    h = 1;

end
```

fdtd.m

```
%   FDTD MATLAB antenna/array solver (preliminary version)
%   Copyright SNM Spring 2011
%   FDTD SCRIPT
%%  External and internal inputs
load input;
f           = linspace(fstart, fstop, steps);   %  port frequency, Hz
P           = 10;                               %  maximum number of cells
along the port line (internal)
factorMU    = 0.8;
factorEPS   = 0.8;

%%  Constructor 1
%   Define FDTD parameters and network parameters
eps0      = 8.85418782e-012;                %    dielectric permittivity of
vacuum(~air)
mu0       = 1.25663706e-006;                %    magnetic permeability of
vacuum(~air)
c0        = 1/sqrt(eps0*mu0);               %    speed of light in
vacuum(~air)
eta0      = sqrt(mu0/eps0);                 %    vacuum impedance, Ohm

%   Time stepping information
d       = D/1000;                               %   Cell size in m
dt      = 0.9/(c0*sqrt(1/d^2 + 1/d^2 +1/d^2));  %   Magic time step reduced
by a factor of 0.9
KT      = round(TO/dt); t = [0: dt: KT*dt];     %   Number of time steps
```

```
%   DFT(FFT) information
NFFT        = 2^14;                                   %   number of sample
frequencies
Fs          = 1/dt;                                   %   sample frequency
fFFT        = Fs/2*linspace(0, 1, NFFT/2+1);          %   all positive FFT
frequencies
findex      = zeros(1, length(f));                    %   index into DFT
harmonics

findex(1) = floor(f(1)*(dt*NFFT));                    %   either exact or
to the left
for n = 2:length(f)
    findex(n) = ceil(f(n)*(dt*NFFT))+1;               %   either exact or
to the right
end
findex  = unique(findex);
ftemp   = fFFT(findex);                               %   vector of FFT
frequencies for the band

%   Define ports and port matrixes
%   All port lines are defined exactly on the main grid
m   = 0;     %    port number
    for n = 1:length(Objects)
        if Isport(n)>0   % port found
            m = m + 1;
            PortNumber(m)       = m;        %   port number
            PortDir(m)          = Isport(n);%   port direction (along x, y,
or z)
            PortImpedance(m)    = R0;       %   port impedance, Ohm
            RowNo(m)            = floor((m-1)/size(TerminationV, 2)) + 1;
            ColNo(m)            = m - size(TerminationV, 2)*floor((m-
1)/size(TerminationV, 2));
            PortVoltage(m)      = TerminationV(RowNo(m), ColNo(m)); %
generator voltage, V (may be complex)

            indX = N1X(n):N2X(n);           %   integer grid
            indY = N1Y(n):N2Y(n);           %   integer grid
            indZ = N1Z(n):N2Z(n);           %   Integer grid
            if length(indX)>1 indX = indX(1:end-1); end;     %   port in x-
direction: indexes into Ex (on half grid)
            if length(indY)>1 indY = indY(1:end-1); end;     %   port in y-
direction: indexes into Ey (on half grid)
            if length(indZ)>1 indZ = indZ(1:end-1); end;     %   port in z-
direction: indexes into Ez (on half grid)

            PortIndX(m, 1:P) = 0;
            PortIndY(m, 1:P) = 0;
            PortIndZ(m, 1:P) = 0;
            PortIndX(m, 1:length(indX)) = indX;     %   indexes of nodes
beloning to the port
            PortIndY(m, 1:length(indY)) = indY;     %   indexes of nodes
beloning to the port
            PortIndZ(m, 1:length(indZ)) = indZ;     %   indexes of nodes
beloning to the port

            PortLength(m) = max([length(indX) length(indY) length(indZ)]);
        end
    end
    M = length(PortNumber); %   # of ports

    %%  Constructor2
    %   Define metal-specific arrays (impose boundary conditions) for all
objects
    %   Define all metal edges
    MetalX  = ones(Nx, Ny+1, Nz+1);         %   3D Metal indicator array for
Ex (on half grid in x and integer grid in y,z)
    MetalY  = ones(Nx+1, Ny, Nz+1);         %   3D Metal indicator array for
Ey (on half grid in y and integer grid in x,z)
    MetalZ  = ones(Nx+1, Ny+1, Nz);         %   3D Metal indicator array for
Ez (on half grid in z and integer grid in x,y)

    IndM     = [];   %   all nodes at exactly metal edges (mu)
    IndE     = [];   %   all cube centers around metal edges (eps)

    for m = 1:length(Objects)
        if Ismet(m)
            indX = N1X(m):N2X(m);  % inside object and on the boundary
            indY = N1Y(m):N2Y(m);  % inside object and on the boundary
            indZ = N1Z(m):N2Z(m);  % inside object and on the boundary

            %   Boundary conditions of zero tangential electric field
component
            if length(indX)>1 % non-trivial x-dimension
                MetalX(indX(1:end-1), indY, indZ) = 0;   % zero tangential x-
field on the boundary
            end
            if length(indY)>1 % non-trivial y-dimension
                MetalY(indX, indY(1:end-1), indZ) = 0;   % zero tangential y-
field on the boundary
            end
            if length(indZ)>1 % non-trivial z-dimension
                MetalZ(indX, indY, indZ(1:end-1)) = 0;   % zero tangential z-
field on the boundary
            end

            %   Accumulate all nodes for the metal edges
            if (length(indX)>1)&(length(indY)>1)       %   face in the xy-
plane
               for mx = 1:length(indX)                 %   edge in the x-
direction
                    row = [indX(mx)  indY(1)  indZ];
                    IndM = [IndM;  row];
                    row = [indX(mx)  indY(end) indZ];
                    IndM = [IndM;  row];
                    row1 = [indX(mx)  indY(1)      indZ];
                    row2 = [indX(mx)  indY(1)-1    indZ];
                    row3 = [indX(mx)  indY(1)      indZ-1];
                    row4 = [indX(mx)  indY(1)-1    indZ-1];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX(mx)  indY(end)    indZ];
                    row2 = [indX(mx)  indY(end)-1  indZ];
                    row3 = [indX(mx)  indY(end)    indZ-1];
                    row4 = [indX(mx)  indY(end)-1  indZ-1];
                    IndE = [IndE; row1; row2; row3; row4];
               end
               for my = 1:length(indY)                  %   edge in the y-
   direction
                    row = [indX(1)  indY(my)  indZ];
                    IndM = [IndM;  row];
                    row = [indX(end)  indY(my)  indZ];
                    IndM = [IndM;  row];
                    row1 = [indX(1)    indY(my)  indZ];
                    row2 = [indX(1)-1  indY(my)  indZ];
                    row3 = [indX(1)    indY(my)  indZ-1];
                    row4 = [indX(1)-1  indY(my)  indZ-1];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX(end)    indY(my)  indZ];
                    row2 = [indX(end)-1  indY(my)  indZ];
                    row3 = [indX(end)    indY(my)  indZ-1];
                    row4 = [indX(end)-1  indY(my)  indZ-1];
                    IndE = [IndE; row1; row2; row3; row4];
               end
            end
            if (length(indX)>1)&(length(indZ)>1)         %   face in the xz-
   plane
               for mx = 1:length(indX)                  %   edge in the x-
   direction
                    row = [indX(mx)  indY  indZ(1)];
                    IndM = [IndM;  row];
                    row = [indX(mx)  indY  indZ(end)];
                    IndM = [IndM;  row];
                    row1 = [indX(mx)  indY    indZ(1)];
                    row2 = [indX(mx)  indY-1  indZ(1)];
                    row3 = [indX(mx)  indY    indZ(1)-1];
                    row4 = [indX(mx)  indY-1  indZ(1)-1];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX(mx)  indY    indZ(end)];
                    row2 = [indX(mx)  indY-1  indZ(end)];
                    row3 = [indX(mx)  indY    indZ(end)-1];
                    row4 = [indX(mx)  indY-1  indZ(end)-1];
                    IndE = [IndE; row1; row2; row3; row4];
               end
               for mz = 1:length(indZ)                  %   edge in the z-
    direction
                    row = [indX(1)  indY  indZ(mz)];
                    IndM = [IndM;  row];
                    row = [indX(end)  indY  indZ(mz)];
                    IndM = [IndM;  row];
                    row1 = [indX(1)    indY  indZ(mz)];
                    row2 = [indX(1)-1  indY  indZ(mz)];
                    row3 = [indX(1)    indY-1  indZ(mz)];
                    row4 = [indX(1)-1  indY-1  indZ(mz)];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX(end)    indY  indZ(mz)];
                    row2 = [indX(end)-1  indY  indZ(mz)];
                    row3 = [indX(end)    indY-1  indZ(mz)];
                    row4 = [indX(end)-1  indY-1  indZ(mz)];
                    IndE = [IndE; row1; row2; row3; row4];
               end
            end

            if (length(indY)>1)&(length(indZ)>1)        %   face in the yz-
    plane
               for my = 1:length(indY)                  %   edge in the y-
    direction
                    row = [indX  indY(my)  indZ(1)];
                    IndM = [IndM;  row];
                    row = [indX  indY(my)  indZ(end)];
                    IndM = [IndM;  row];
                    row1 = [indX    indY(my)  indZ(1)];
```

```
                    row2 = [indX-1  indY(my)  indZ(1)];
                    row3 = [indX    indY(my)  indZ(1)-1];
                    row4 = [indX-1  indY(my)  indZ(1)-1];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX    indY(my)  indZ(end)];
                    row2 = [indX-1  indY(my)  indZ(end)];
                    row3 = [indX    indY(my)  indZ(end)-1];
                    row4 = [indX-1  indY(my)  indZ(end)-1];
                    IndE = [IndE; row1; row2; row3; row4];
               end
               for mz = 1:length(indZ)                 %   edge in the z-
direction
                    row = [indX  indY(1)  indZ(mz)];
                    IndM = [IndM;  row];
                    row = [indX  indY(end)  indZ(mz)];
                    IndM = [IndM;  row];
                    row1 = [indX   indY(1)  indZ(mz)];
                    row2 = [indX-1 indY(1)  indZ(mz)];
                    row3 = [indX   indY(1)-1  indZ(mz)];
                    row4 = [indX-1 indY(1)-1  indZ(mz)];
                    IndE = [IndE; row1; row2; row3; row4];
                    row1 = [indX   indY(end)  indZ(mz)];
                    row2 = [indX-1 indY(end)  indZ(mz)];
                    row3 = [indX   indY(end)-1  indZ(mz)];
                    row4 = [indX-1 indY(end)-1  indZ(mz)];
                    IndE = [IndE; row1; row2; row3; row4];
               end
            end
        end
    end

    %%  Constructor3
    %   Fill out material-specific arrays
    %   Fill out difference coefficients for FDTD difference equations
    DIELC    = ones(Nx, Ny, Nz);            %   3D Permittivity array  on
half grid (cube centers)
```

```
    MAGNC    = ones(Nx+1, Ny+1, Nz+1);      %   3D Permeability array on
integer grid (cube nodes)
    SIGMAEC  = zeros(Nx, Ny, Nz);           %   3D Electric conductivity
array on half grid (cube centers)
    SIGMAMC  = zeros(Nx+1, Ny+1, Nz+1);     %   3D Magnetic conductivity
array on integer grid (cube nodes)

    if ~isempty(custom)
        eps_body = 50;
        sig_body  = 0.5;
        DIELC       = DIELC +       (eps_body-1)*DIELext;
        SIGMAEC     = SIGMAEC +     sig_body*DIELext;
    end

    for m = 1:length(Objects)
       if (~Ismet(m))&(~Isport(m))
            indX = N1X(m):N2X(m);
            indY = N1Y(m):N2Y(m);
            indZ = N1Z(m):N2Z(m);
            DIELC(indX(1:end-1), indY(1:end-1), indZ(1:end-1))  = Eps(m);
            MAGNC(indX(1:end-1), indY(1:end-1), indZ(1:end-1))  = Mu(m);
            SIGMAEC(indX(1:end-1), indY(1:end-1), indZ(1:end-1))= SigE(m);
            SIGMAMC(indX(1:end-1), indY(1:end-1), indZ(1:end-1))= SigM(m);
       end
    end

    IndM = unique(IndM, 'rows');
    IndE = unique(IndE, 'rows');

   for m = 1:length(IndM)
        i1 = IndM(m, 1);
        i2 = IndM(m, 2);
        i3 = IndM(m, 3);
        MAGNC(i1, i2, i3) =  factorMU*MAGNC(i1, i2, i3);
    end
    for m = 1:length(IndE)
```

```
        i1 = IndE(m, 1);
        i2 = IndE(m, 2);
        i3 = IndE(m, 3);
        DIELC(i1, i2, i3) =  factorEPS*DIELC(i1, i2, i3);
    end

    DIELC = DIELC*eps0; MAGNC = MAGNC*mu0;

    %   Fill out dielectric field arrays for every brick with taking into
    %   account boundary smoothing: epsilon/sigma for Ex, Ey, Ez on the
    %   edge of the grid are averaged over four bricks sharing the same
    %   edge
    %   Arrays for Ex
    Dtemp = eps0*ones(Nx, Ny+1, Nz+1); Stemp = zeros(Nx, Ny+1, Nz+1);
    nx = 1:Nx; ny = 2:Ny; nz = 2:Nz;
    Dtemp(:, ny, nz) = (DIELC(:, ny, nz)   + DIELC(:, ny-1, nz)   + DIELC(:,
ny, nz-1)   + DIELC(:, ny-1, nz-1))/4;
    Stemp(:, ny, nz) = (SIGMAEC(:, ny, nz) + SIGMAEC(:, ny-1, nz) +
SIGMAEC(:, ny, nz-1) + SIGMAEC(:, ny-1, nz-1))/4;

    Ex1     = (1 - dt*Stemp./(2*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ex2     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ex3     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ex1     = Ex1(nx, ny, nz);
    Ex2     = Ex2(nx, ny, nz);
    Ex3     = Ex3(nx, ny, nz);

    %   Arrays for Ey
    Dtemp = eps0*ones(Nx+1, Ny, Nz+1);
    Stemp = zeros(Nx+1, Ny, Nz+1);
    nx = 2:Nx; ny = 1:Ny; nz = 2:Nz;
    Dtemp(nx, :, nz) = (DIELC(nx, :, nz)   + DIELC(nx-1, :, nz)   + DIELC(nx,
:, nz-1)   + DIELC(nx-1, :, nz-1))/4;
    Stemp(nx, :, nz) = (SIGMAEC(nx, :, nz) + SIGMAEC(nx-1, :, nz) +
SIGMAEC(nx, :, nz-1) + SIGMAEC(nx-1, :, nz-1))/4;
```

```
    Ey1     = (1 - dt*Stemp./(2*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ey2     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ey3     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ey1     = Ey1(nx, ny, nz);
    Ey2     = Ey2(nx, ny, nz);
    Ey3     = Ey3(nx, ny, nz);

    %   Arrays for Ez
    Dtemp = eps0*ones(Nx+1, Ny+1, Nz);
    Stemp = zeros(Nx+1, Ny+1, Nz);
    nx = 2:Nx; ny = 2:Ny; nz = 1:Nz;
    Dtemp(nx, ny, :) = (DIELC(nx, ny, :)   + DIELC(nx-1, ny, :)   + DIELC(nx,
ny-1, :)   + DIELC(nx-1, ny-1, :))/4;
    Stemp(nx, ny, :) = (SIGMAEC(nx, ny, :) + SIGMAEC(nx-1, ny, :) +
SIGMAEC(nx, ny-1, :) + SIGMAEC(nx-1, ny-1, :))/4;

    Ez1     = (1 - dt*Stemp./(2*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ez2     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ez3     = (dt./(d*Dtemp))./(1 + dt*Stemp./(2*Dtemp));
    Ez1     = Ez1(nx, ny, nz);
    Ez2     = Ez2(nx, ny, nz);
    Ez3     = Ez3(nx, ny, nz);

    %   Fill out magnetic field arrays for every brick with taking into
    %   account boundary smoothing: mu/sigmam for Hx, Hy, Hz on the
    %   face of the grid are averaged over four face nodes
    %   Arrays for Hx
    Htemp = mu0*ones(Nx+1, Ny, Nz); Stemp = zeros(Nx+1, Ny, Nz);
    ny = 1:Ny; nz = 1:Nz;
    Htemp(:, ny, nz) = (MAGNC(:, ny, nz)   + MAGNC(:, ny+1, nz)   + MAGNC(:,
ny, nz+1)   + MAGNC(:, ny+1, nz+1))/4;
    Stemp(:, ny, nz) = (SIGMAMC(:, ny, nz) + SIGMAMC(:, ny+1, nz) +
SIGMAMC(:, ny, nz+1) + SIGMAMC(:, ny+1, nz+1))/4;

    Hx1  = (1 - dt*Stemp./(2*Htemp))./(1 + dt*Stemp./(2*Htemp));
    Hx2  = (dt./(d*Htemp))./(1 + dt*Stemp./(2*Htemp));
    %   Arrays for Hy
    Htemp = mu0*ones(Nx, Ny+1, Nz); Stemp = zeros(Nx, Ny+1, Nz);
    nx = 1:Nx; nz = 1:Nz;
    Htemp(nx, :, nz) = (MAGNC(nx, :, nz)   + MAGNC(nx+1, :, nz)   + MAGNC(nx, 
:, nz+1)   + MAGNC(nx+1, :, nz+1))/4;
    Stemp(nx, :, nz) = (SIGMAMC(nx, :, nz) + SIGMAMC(nx+1, :, nz) + 
SIGMAMC(nx, :, nz+1) + SIGMAMC(nx+1, :, nz+1))/4;

    Hy1  = (1 - dt*Stemp./(2*Htemp))./(1 + dt*Stemp./(2*Htemp));
    Hy2  = (dt./(d*Htemp))./(1 + dt*Stemp./(2*Htemp));

    %   Arrays for Hz
    Htemp = mu0*ones(Nx, Ny, Nz+1); Stemp = zeros(Nx, Ny, Nz+1);
    nx = 1:Nx; ny = 1:Ny;
    Htemp(nx, ny, :)  = (MAGNC(nx, ny, :)   + MAGNC(nx+1, ny, :)   + 
MAGNC(nx, ny+1, :)   + MAGNC(nx+1, ny+1, :))/4;
    Stemp(nx, ny, :) = (SIGMAMC(nx, ny, :) + SIGMAMC(nx+1, ny, :) + 
SIGMAMC(nx, ny+1, :) + SIGMAMC(nx+1, ny+1, :))/4;

    Hz1  = (1 - dt*Stemp./(2*Htemp))./(1 + dt*Stemp./(2*Htemp));
    Hz2  = (dt./(d*Htemp))./(1 + dt*Stemp./(2*Htemp));

    clear Dtemp Htemp Stemp;

    %   Difference coefficients for ports (at load terminations)
    %   Averaging follows the E-field scheme
    for m = 1:M
        if PortDir(m)==1
            diel(m)  =  (DIELC( PortIndX(m,1), PortIndY(m,1), PortIndZ(m,1)) 
+ ...
                         DIELC( PortIndX(m,1), PortIndY(m,1)-1, 
PortIndZ(m,1)) + ...
                         DIELC( PortIndX(m,1), PortIndY(m,1), PortIndZ(m,1)-
1) + ...
```

```
                         DIELC( PortIndX(m,1), PortIndY(m,1)-1,
PortIndZ(m,1)-1))/4;
        end
        if PortDir(m)==2
            diel(m)  =  (DIELC( PortIndX(m,1), PortIndY(m,1), PortIndZ(m,1))
+ ...
                         DIELC( PortIndX(m,1)-1, PortIndY(m,1),
PortIndZ(m,1)) + ...
                         DIELC( PortIndX(m,1), PortIndY(m,1), PortIndZ(m,1)-
1) + ...
                         DIELC( PortIndX(m,1)-1, PortIndY(m,1),
PortIndZ(m,1)-1))/4;
        end
        if PortDir(m)==3
            diel(m)  =  (DIELC( PortIndX(m,1), PortIndY(m,1), PortIndZ(m,1))
+ ...
                         DIELC( PortIndX(m,1)-1, PortIndY(m,1),
PortIndZ(m,1)) + ...
                         DIELC( PortIndX(m,1), PortIndY(m,1)-1,
PortIndZ(m,1)) + ...
                         DIELC( PortIndX(m,1), PortIndY(m,1)-1,
PortIndZ(m,1)))/4;
        end
        sigma(m)   = PortLength(m)*d/(d*d*PortImpedance(m));
    end

%%  Constructor4
AntI    = zeros(M, M, length(t));          %   antenna currents for all ports
AntV    = zeros(M, M, length(t));          %   antenna voltages for all ports
Z       = zeros(length(findex), M, M);     %   port impedance matrix in
frequency domain (at all frequencies)
T       = linspace(0, 1/ftemp(1), 100);    %   Fourier integration for CW
excitation

%   Port voltages and currents in time/frequency domain
VG      = zeros(M, length(t));
```

```
if length(f) >1     %   Port voltages for pulse excitation
    for m = 1:M
        Xscan                  = (ColNo(m)-1)*xscan/180*pi;
        Yscan                  = (RowNo(m)-1)*yscan/180*pi;
        for n = 1:length(findex)
            tdelay       = (Xscan + Yscan)/(2*pi*ftemp(n));
            VoltageAmpl  = abs(PortVoltage(m));
            VoltagePhase = angle(PortVoltage(m));
            VGdelay      = VoltageAmpl*sin(2*pi*ftemp(n)*t + VoltagePhase -
Xscan - Yscan);
            VGdelay(find(t<tdelay)) = 0;
            taper       = exp(-4*((t-1/ftemp(1)-tdelay)*ftemp(1)).^2);     %
amplitude taper based on the lowest band frequency; variations are possible
            taper(find(t<tdelay)) = 0;
            VG(m, :)   = VG(m, :) + VGdelay.*taper;     %   generator
voltages for every port
        end
    end
    VG = VG/length(findex);                                %   normalize
else    %   port voltages for CW excitation
    for m = 1:M
        Xscan                  = (ColNo(m)-1)*xscan/180*pi;
        Yscan                  = (RowNo(m)-1)*yscan/180*pi;
        tdelay                 = (Xscan + Yscan)/(2*pi*ftemp(1));
        VoltageAmpl            = abs(PortVoltage(m));
        VoltagePhase           = angle(PortVoltage(m));
        VGdelay                = VoltageAmpl*sin(2*pi*ftemp(1)*t + VoltagePhase
+ Xscan + Yscan);
        VGdelay(find(t<tdelay)) = 0;
        taper       = 1- exp(-4*(t-tdelay).^2*f(1)^2);
        taper(find(t<tdelay)) = 0;
        VG(m, :)   = VGdelay.*taper;                        %   generator
voltages for every port
    end
end
```

```
%%  FDTD-marching
%   FDTD method - marching on in time
close gcf;
if fields
    scrsz = get(0,'ScreenSize');
    a=figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
    POSF = get(a, 'Position');
    colormap(jet(128));
end

% Allocate (clear) field matrices
ExP = zeros(Nx  , Ny+1, Nz+1);
EyP = zeros(Nx+1, Ny  , Nz+1);
EzP = zeros(Nx+1, Ny+1, Nz  );
HxP = zeros(Nx+1, Ny  , Nz  );
HyP = zeros(Nx  , Ny+1, Nz  );
HzP = zeros(Nx  , Ny  , Nz+1);
ExN = zeros(Nx  , Ny+1, Nz+1);
EyN = zeros(Nx+1, Ny  , Nz+1);
EzN = zeros(Nx+1, Ny+1, Nz  );
HxN = zeros(Nx+1, Ny  , Nz  );
HyN = zeros(Nx  , Ny+1, Nz  );
HzN = zeros(Nx  , Ny  , Nz+1);

ExPP = zeros(Nx  , Ny+1, Nz+1);
EyPP = zeros(Nx+1, Ny  , Nz+1);
EzPP = zeros(Nx+1, Ny+1, Nz  );

%   Allocate (clear) boundary matrixes
if length(ftemp)==1
    KTrack = floor(1/dt/ftemp(1))+ 2;    %   one last period (a bit over)

    Eyleft = zeros(KTrack,  Ny  , Nz+1);
    Ezleft = zeros(KTrack,  Ny+1, Nz);
    Hyleft = zeros(KTrack,  Ny+1, Nz);
    Hzleft = zeros(KTrack,  Ny  , Nz+1);

    Eyright = zeros(KTrack,  Ny  , Nz+1);
    Ezright = zeros(KTrack,  Ny+1, Nz);
    Hyright = zeros(KTrack,  Ny+1, Nz);
    Hzright = zeros(KTrack,  Ny  , Nz+1);

    Exfront = zeros(KTrack,  Nx  , Nz+1);
    Ezfront = zeros(KTrack,  Nx+1, Nz);
    Hxfront = zeros(KTrack,  Nx+1, Nz);
    Hzfront = zeros(KTrack,  Nx  , Nz+1);

    Exback = zeros(KTrack,  Nx  , Nz+1);
    Ezback = zeros(KTrack,  Nx+1, Nz);
    Hxback = zeros(KTrack,  Nx+1, Nz);
    Hzback = zeros(KTrack,  Nx  , Nz+1);

    Exbottom = zeros(KTrack,  Nx  , Ny+1);
    Eybottom = zeros(KTrack,  Nx+1, Ny);
    Hxbottom = zeros(KTrack,  Nx+1, Ny);
    Hybottom = zeros(KTrack,  Nx  , Ny+1);

    Extop = zeros(KTrack,  Nx  , Ny+1);
    Eytop = zeros(KTrack,  Nx+1, Ny);
    Hxtop = zeros(KTrack,  Nx+1, Ny);
    Hytop = zeros(KTrack,  Nx  , Ny+1);

    ExtopP = zeros(KTrack,  Nx  , Ny+1);
    EytopP = zeros(KTrack,  Nx+1, Ny);
    HxtopP = zeros(KTrack,  Nx+1, Ny);
    HytopP = zeros(KTrack,  Nx  , Ny+1);
end

%   Main FDTD loop - "bootstrapping" (initial conditions are zeros)
```

```
kt = 2; s = 1;
E  = zeros(1, KT);
tic
while kt <= KT
    if kt/KT>=0.05*s
        s = s +1;
        disp(strcat('FDTD is running - ', num2str(100*kt/KT, '%4.0f'), '%
done'));
    end
    %%  E-field update (everywhere except on the boundary; (45% of time))
    ExN(:,2:Ny,2:Nz) = Ex1.*ExP(:,2:Ny,2:Nz) + Ex2.*(diff(HzP(:,:,2:Nz),1,2)
- diff(HyP(:,2:Ny,:),1,3));
    EyN(2:Nx,:,2:Nz) = Ey1.*EyP(2:Nx,:,2:Nz) + Ey2.*(diff(HxP(2:Nx,:,:),1,3)
- diff(HzP(:,:,2:Nz),1,1));
    EzN(2:Nx,2:Ny,:) = Ez1.*EzP(2:Nx,2:Ny,:) + Ez2.*(diff(HyP(:,2:Ny,:),1,1)
- diff(HxP(2:Nx,:,:),1,2));

    %%  FDTD_abc1: radiation BCs (Mur 1981, first order, homogeneous
material)
      abc_murfirst;
    % abc_mursecond;

    %%  FDTD-terminal
    %   Port updates (through the E-field)
    %   Terminal model (fills network arrays [mm, mm, kt])
    for m = 1:M
        cond       = sigma(m)*TerminationY(RowNo(m), ColNo(m));
        es1(m)     = (1 - dt*cond/(2*diel(m)))/(1 + dt*cond/(2*diel(m)));
        es2(m)     = (dt/(d*diel(m)))/(1 + dt*cond/(2*diel(m)));
        es3(m)     = (dt*cond/(d*diel(m)))/(1 + dt*cond/(2*diel(m)));
        if PortDir(m) == 1  %   port along the x-axis
            temp = nonzeros(PortIndX(m, :));
            CellsPerPort = length(temp);
            for k = 1:CellsPerPort
                k_e = PortIndX(m, k);
                m_e = PortIndY(m, 1);
                p_e = PortIndZ(m, 1);
                ExN(k_e, m_e, p_e) = es1(m) * ExP(k_e, m_e, p_e)+ ...
                                     es2(m) *(HzN(k_e, m_e, p_e) - HzN(k_e,
m_e-1, p_e) - HyN(k_e, m_e, p_e) + HyN(k_e, m_e, p_e-1))- ...
                                     es3(m) *(VG(m, kt) + VG(m, kt-
1))/(2*CellsPerPort);
            end
            AntV(m, m, kt) =  -d*sum(ExN(PortIndX(m, 1:CellsPerPort), m_e,
p_e));
        end
        if PortDir(m) == 2  %   port along the y-axis
            temp = nonzeros(PortIndY(m, :));
            CellsPerPort = length(temp);
            for k = 1:CellsPerPort
                k_e = PortIndX(m, 1);
                m_e = PortIndY(m, k);
                p_e = PortIndZ(m, 1);
                EyN(k_e, m_e, p_e) = es1(m) * EyP(k_e, m_e, p_e)+ ...
                                     es2(m) *(HxN(k_e, m_e, p_e) - HxN(k_e,
m_e, p_e-1) - HzN(k_e, m_e, p_e) + HzN(k_e-1, m_e, p_e))- ...
                                     es3(m) *(VG(m, kt) + VG(m, kt-
1))/(2*CellsPerPort);
            end
            AntV(m, m, kt) =  -d*sum(EyN(k_e, PortIndY(m, 1:CellsPerPort),
p_e));
        end
        if PortDir(m) == 3  %   port along the z-axis
            temp = nonzeros(PortIndZ(m, :));
            CellsPerPort = length(temp);
            for k = 1:CellsPerPort
                k_e = PortIndX(m, 1);
                m_e = PortIndY(m, 1);
                p_e = PortIndZ(m, k);
                EzN(k_e, m_e, p_e)  = es1(m) * EzP(k_e, m_e, p_e)+ ...
                                      es2(m) *(HyN(k_e, m_e, p_e) - HyN(k_e-
1, m_e, p_e) - HxN(k_e, m_e, p_e) + HxN(k_e, m_e-1, p_e))- ...
```

```
                                      es3(m) *(VG(m, kt) + VG(m, kt-
1))/(2*CellsPerPort);
            end
            AntV(m, m, kt) =  -d*sum(EzN(k_e, m_e, PortIndZ(m,
1:CellsPerPort)));
        end
        AntI(m, m, kt) =  -(AntV(m, m, kt)- VG(m,
kt))/PortImpedance(m)*TerminationY(RowNo(m), ColNo(m));
        AntV(m, m, kt) =  (AntV(m, m, kt) + AntV(m, m, kt-1))/2; % at half
grid in time - optional
        AntI(m, m, kt) =  (AntI(m, m, kt) + AntI(m, m, kt-1))/2; % at half
grid in time - optional
    end

    %%   Boundary conditions -metal
            ExN = MetalX.*ExN;
            EyN = MetalY.*EyN;
            EzN = MetalZ.*EzN;

    %%  H-field update (everywhere, 35% of time)
            HxN =  Hx1.*HxP + Hx2.*(diff(EyN,1,3)- diff(EzN,1,2));
            HyN =  Hy1.*HyP + Hy2.*(diff(EzN,1,1)- diff(ExN,1,3));
            HzN =  Hz1.*HzP + Hz2.*(diff(ExN,1,2)- diff(EyN,1,1));

    %%  Superabsorption
    abc_super;

    %% Impedance matrix
    if length(findex)>1
        %  FDTD-network - pulse
        %   Phasor calculations for pulse excitation
        for m = 1:M
            if ((kt-2)*dt)> (1/f(1))                              %
at least one period detected  at the lowest frequency
                AntennaV    = squeeze(AntV(m, m, :));             %
voltage in time domain
```

```
                if TerminationY(RowNo(m), ColNo(m))==0                  %
standard impedance or active impedance
                    mm = find(abs(PortVoltage)>0);
                    AntennaI    = squeeze(AntI(mm, mm, :));             %
current in time domain
                else
                    AntennaI    = squeeze(AntI(m, m, :));               %
current in time domain
                end
                AntennaVFFT = fft(AntennaV, NFFT);                      %
FFT of voltage
                AntennaIFFT = fft(AntennaI, NFFT);                      %
FFT of current
                for n = 1:length(findex)
                    Z(n, m, m) =
AntennaVFFT(findex(n))/AntennaIFFT(findex(n));
                end
            end
        end
    else
        %  FDTD-network-CW
        %   Phasor calculations (using Fourier coefficients for fundamental
frequency)
        %   mm - driven port number (from fdtd_marching)
        %   m  - other port number (local)
        for m = 1:M
            if ((kt-2)*dt)> (1/ftemp)                                    %
at least one period detected
                index       = kt-round(1/dt/f(1))-1:1:kt;               %
window covering one period
                tempt       = t(index)-t(index(1));
                omega       = 2*pi*ftemp;                                   %
radian frequency
                %   Voltage phasor
                AntennaV            = squeeze(AntV(m, m, index));
                AntennaV            = interp1(tempt, AntennaV, T,'linear');
                func                = AntennaV.*exp(-j*omega*T);    %
integrand for Fourier coefficients
                VPhasor(m, m)       = (1/length(T))*(sum(func)-0.5*func(1)-
0.5*func(end));
                                                                  %
trapezoidal
                                                                  %
integration (1/T*int(exp(-j*omega*t)*f))
                %   Current phasor
                AntennaI            = squeeze(AntI(m, m, index));
                AntennaI            = interp1(tempt, AntennaI, T,'linear');
                func                = AntennaI.*exp(-j*omega*T);    %
integrand for Fourier coefficients
                IPhasor(m, m)       = (1/length(T))*(sum(func)-0.5*func(1)-
0.5*func(end));
                                                                  %
trapezoidal
                                                                  %
integration  (1/T*int(exp(-j*omega*t)*f))
            end
        end
        %   Once all currents/voltages have been found->Z
        for m = 1:M
            if ((kt-2)*dt)> (1/ftemp)                                  %   at
least one period detected
                if TerminationY(RowNo(m), ColNo(m))==0                 %
standard impedance or active impedance
                    mm = find(abs(PortVoltage)>0);
                    Z(1, m, m) = VPhasor(m, m)/IPhasor(mm, mm);
                else
                    Z(1, m, m) = VPhasor(m, m)/IPhasor(m, m);
                end
            end
        end
    end
```

```
%%  Plotter1
%   Scale/plot fields and port voltages
if fields
    %   Real-time figure
    %   Subdivide the window (a plot for each port)
    %   Fields in the xz-plane (or in any other plane)
    if M<=9
        sp1 = subplot(M, 2, [1:2:2*M-1]);
        POSS1   = get(sp1, 'Position');
    end
    if component == 1 % plot Ex
       string0 = 'Electric field E_x at t=';
       if rem(Ny, 2)>0     %   Ny is odd
            output = 0.25*( ExN(:, (Ny+1)/2, 1:end-1)   + ...
                            ExN(:, (Ny+1)/2+1, 1:end-1) + ...
                            ExN(:, (Ny+1)/2, 2:end)     + ...
                            ExN(:, (Ny+1)/2+1, 2:end) );
        else                %   Ny is even
            output = 0.50*( ExN(:, Ny/2+1, 1:end-1)     + ...
                            ExN(:, Ny/2+1, 2:end) );
       end
    end
    if component == 2 % plot Ey
       string0 = 'Electric field E_y at t=';
       if rem(Ny, 2)>0     %   Ny is odd
            output = 0.25*( EyN(:, (Ny+1)/2, 1:end-1)   + ...
                            EyN(:, (Ny+1)/2+1, 1:end-1) + ...
                            EyN(:, (Ny+1)/2, 2:end)     + ...
                            EyN(:, (Ny+1)/2+1, 2:end) );
        else                %   Ny is even
            output = 0.50*( EyN(:, Ny/2+1, 1:end-1)     + ...
                            EyN(:, Ny/2+1, 2:end) );
       end
       output = 0.50*( output(1:end-1, :,:) + output(2:end, :,:));
    end
    if component == 3 % plot Ez
```

```
   string0 = 'Electric field E_z at t=';
   if rem(Ny, 2)>0     %   Ny is odd
        output = 0.25*( EzN(1:end-1, (Ny+1)/2, :)   + ...
                        EzN(1:end-1, (Ny+1)/2+1, :) + ...
                        EzN(2:end, (Ny+1)/2, :)     + ...
                        EzN(2:end, (Ny+1)/2+1, :) );
    else                %   Ny is even
        output = 0.50*( EzN(1:end-1, Ny/2+1, :)     + ...
                        EzN(2:end, Ny/2+1, :) );
   end
end
if component == 4 % plot Hx
   string0 = 'Magnetic field H_x at t=';
   if rem(Ny, 2)>0     %   Ny is odd
        output = 0.5* ( HxN(:, (Ny+1)/2,  :) + ...
                        HxN(:, (Ny+1)/2+1,:));
    else                %   Ny is even
        output = HxN(:, Ny/2+1, :);
   end
   output = 0.50*( output(1:end-1, :,:) + output(2:end, :,:));
end
if component == 5 % plot Hy
   string0 = 'Magnetic field H_y at t=';
   if rem(Ny, 2)>0     %   Ny is odd
        output = 0.5* ( HyN(:, (Ny+1)/2,  :) + ...
                        HyN(:, (Ny+1)/2+1,:));
    else                %   Ny is even
        output = HyN(:, Ny/2+1, :);
   end
end
if component == 6 % plot Hz
   string0 = 'Magnetic field H_z at t=';
   if rem(Ny, 2)>0     %   Ny is odd
        output = 0.5* ( HzN(:, (Ny+1)/2,  :) + ...
                        HzN(:, (Ny+1)/2+1,:));
    else                %   Ny is even
```

```
                output = HzN(:, Ny/2+1, :);
             end
             output = 0.50*( output(:, :,1:end-1) + output(:, :,2:end));
          end
          output = squeeze(output); output = abs(output).^power.*sign(output);
          output = interp2(output, 3); output(1) = +scale; output(end)=-scale;
          imagesc( [x(1)+D/2 x(end)-D/2], [z(1)+D/2 z(end)-D/2], output'); %
this function removes all old patches
          for m = 1:length(W)
              Wtemp = W;
              if Ismet(m)
                  viewer(W(m), H(m), L(m),            XCtemp(m), ZCtemp(m),
YCtemp(m), Color(m, :), 0.5, 1.5, 'k');
              elseif Isline(m)
                  viewer(W(m), H(m), L(m),            XCtemp(m), ZCtemp(m),
YCtemp(m), Color(m, :), 0.5, 0.25, 'k');
              else
                  viewer(W(m), H(m), L(m),            XCtemp(m), ZCtemp(m),
YCtemp(m), Color(m, :), 0.5, 0.5, 'k');
              end
          end
          if ~isempty(custom)
              patch(Xshape, Zshape, Yshape, 'w','EdgeColor',[0.75 0.75 0.75],
'FaceAlpha', 0.1);
          end
          string      =   strcat(num2str(1e9*t(kt)), ' ns');
          axis 'equal';    axis 'tight', set(gca,'YDir','normal');
          xlabel('x, mm'); ylabel('z, mm');
          title(strcat(string0, string));

          %   Port voltages
          for m = 1:M
             if M<=9
                 subplot(M, 2, 2*m);
                 time    = t(1:kt);
                 GeneratorV      = VG(m, 1:kt);
```

```
               AntennaV        = squeeze(AntV(m, m, 1:kt));
               string1          =   strcat('Port#', num2str(RowNo(m)),
num2str(ColNo(m)));
               string2 = num2str(mean(Z(:, m, m)), '%5.1f');
               plot(time*1e9, GeneratorV, 'b', time*1e9, AntennaV, 'r'); grid
on;
               xlabel('time, ns', 'FontSize', 7); ylabel('volts', 'FontSize',
7);  set(gca,'FontSize',7);
               title (strcat(string1, ': Vg-blue; Va-red, Z=', string2),
'FontSize', 7);
            end
         end
         drawnow;
    end
    %% Accumulate fields on the faces (half a step offsetted from the
boundaries) over one last period
    if length(ftemp)==1
        if kt>KT-KTrack
            index = kt -(KT-KTrack);

            Eyleft(index, :, :)= (EyN(1, :, :)+EyN(2, :, :))/2;
            Ezleft(index, :, :)= (EzN(1, :, :)+EzN(2, :, :))/2;
            Hyleft(index, :, :)=  HyN(1, :, :);
            Hzleft(index, :, :)=  HzN(1, :, :);

            Eyright(index, :, :)= (EyN(end, :, :)+EyN(end-1, :, :))/2;
            Ezright(index, :, :)= (EzN(end, :, :)+EzN(end-1, :, :))/2;
            Hyright(index, :, :)=  HyN(end, :, :);
            Hzright(index, :, :)=  HzN(end, :, :);

            Exfront(index, :, :)= permute((ExN(:, 1, :)+ExN(:, 2, :))/2, [2 1
3]);
            Ezfront(index, :, :)= permute((EzN(:, 1, :)+EzN(:, 2, :))/2, [2 1
3]);
            Hxfront(index, :, :)= permute(HxN(:, 1, :), [2 1 3]);
            Hzfront(index, :, :)= permute(HzN(:, 1, :), [2 1 3]);
            Exback(index, :, :)= permute((ExN(:, end, :)+ExN(:, end-1, :))/2,
[2 1 3]);
            Ezback(index, :, :)= permute((EzN(:, end, :)+EzN(:, end-1, :))/2,
[2 1 3]);
            Hxback(index, :, :)= permute(HxN(:, end, :), [2 1 3]);
            Hzback(index, :, :)= permute(HzN(:, end, :), [2 1 3]);

            Exbottom(index, :, :)= permute((ExN(:, :, 1)+ExN(:, :, 2))/2, [3
1 2]);
            Eybottom(index, :, :)= permute((EyN(:, :, 1)+EyN(:, :, 2))/2, [3
1 2]);
            Hxbottom(index, :, :)= permute(HxN(:, :, 1), [3 1 2]);
            Hybottom(index, :, :)= permute(HyN(:, :, 1), [3 1 2]);

            Extop(index, :, :)= permute((ExN(:, :, end)+ExN(:, :, end-1))/2,
[3 1 2]);
            Eytop(index, :, :)= permute((EyN(:, :, end)+EyN(:, :, end-1))/2,
[3 1 2]);
            Hxtop(index, :, :)= permute(HxN(:, :, end), [3 1 2]);
            Hytop(index, :, :)= permute(HyN(:, :, end), [3 1 2]);

            ExtopP(index, :, :)= permute((ExN(:, :, planeNZ)+ExN(:, :,
planeNZ-1))/2, [3 1 2]);
            EytopP(index, :, :)= permute((EyN(:, :, planeNZ)+EyN(:, :,
planeNZ-1))/2, [3 1 2]);
            HxtopP(index, :, :)= permute(HxN(:, :, planeNZ-1), [3 1 2]);
            HytopP(index, :, :)= permute(HyN(:, :, planeNZ-1), [3 1 2]);
        end
    end
    %%  Prepare for the next step
    kt   = kt + 1;
    ExPP = ExP; EyPP = EyP; EzPP = EzP;
    ExP = ExN; EyP = EyN; EzP = EzN; HxP = HxN; HyP = HyN; HzP = HzN;
end     %   end of fdtd marching loop
```

```
%%  Fields phasors on the boundary
if length(ftemp)==1
    tempt       = t(1:KTrack) - t(1);
    omega       = 2*pi*ftemp;                    %   radian frequency
    %   left/right
    [Eyleft]    = fc(tempt, T, omega, Eyleft);
    [Hyleft]    = fc(tempt, T, omega, Hyleft);
    [Ezleft]    = fc(tempt, T, omega, Ezleft);
    [Hzleft]    = fc(tempt, T, omega, Hzleft);
    [Eyright]   = fc(tempt, T, omega, Eyright);
    [Hyright]   = fc(tempt, T, omega, Hyright);
    [Ezright]   = fc(tempt, T, omega, Ezright);
    [Hzright]   = fc(tempt, T, omega, Hzright);
    %   front/back
    [Exfront]    = fc(tempt, T, omega, Exfront);
    [Hxfront]    = fc(tempt, T, omega, Hxfront);
    [Ezfront]    = fc(tempt, T, omega, Ezfront);
    [Hzfront]    = fc(tempt, T, omega, Hzfront);
    [Exback]    = fc(tempt, T, omega, Exback);
    [Hxback]    = fc(tempt, T, omega, Hxback);
    [Ezback]    = fc(tempt, T, omega, Ezback);
    [Hzback]    = fc(tempt, T, omega, Hzback);
    %   top/bottom
    [Exbottom]    = fc(tempt, T, omega, Exbottom);
    [Hxbottom]    = fc(tempt, T, omega, Hxbottom);
    [Eybottom]    = fc(tempt, T, omega, Eybottom);
    [Hybottom]    = fc(tempt, T, omega, Hybottom);
    [Extop]    = fc(tempt, T, omega, Extop);
    [Hxtop]    = fc(tempt, T, omega, Hxtop);
    [Eytop]    = fc(tempt, T, omega, Eytop);
    [Hytop]    = fc(tempt, T, omega, Hytop);
    %   observation plane
    [ExtopP]    = fc(tempt, T, omega, ExtopP);
    [HxtopP]    = fc(tempt, T, omega, HxtopP);
    [EytopP]    = fc(tempt, T, omega, EytopP);
    [HytopP]    = fc(tempt, T, omega, HytopP);
```

```
end

clear ExPP EyPP EzPP ExP EyP EzP HxP HyP HzP ExN EyN EzN HxN HyN HzN;
clear Ex1 Ey1 Ez1 Ex2 Ey2 Ez2 Ex3 Ey3 Ez3 Hx1 Hy1 Hz1 Hx2 Hy2 Hz2 DIELC MAGNC
SIGMAEC SIGMAMC MetalX MetalY MetalZ;
clear AntennaIFFT AntennaVFFT fFFT

toc
save output;
```

abc_murfirst.m

```
%% Radiation BCs (Mur 1981, first order)
%   Copyright Greg Noetscher/SNM Spring 2011
%   Mur 1st order ABCs are implemented on each face of the domain

m1      = (c0*dt - d)/(c0*dt + d);
%   Left
EyN(1, :,:)   =  EyP(2,:,:)  + m1*(EyN(2,:,:) - EyP(1,:,:));        %  left -
Ey;
EzN(1, :,:)   =  EzP(2,:,:)  + m1*(EzN(2,:,:) - EzP(1,:,:));        %  left -
Ez;
%   Right
EyN(Nx+1, :,:)=  EyP(Nx,:,:) + m1*(EyN(Nx, :,:) - EyP(Nx+1,:,:));   %   right
- Ey;
EzN(Nx+1, :,:)=  EzP(Nx,:,:) + m1*(EzN(Nx, :,:) - EzP(Nx+1,:,:));   %   right
- Ez;
%   Front
ExN(:, 1,:)   =  ExP(:,2,:)  + m1*(ExN(:,2,:) - ExP(:,1,:));        %   front
- Ex;
EzN(:, 1,:)   =  EzP(:,2,:)  + m1*(EzN(:,2,:) - EzP(:,1,:));        %   front
- Ez;
%   Rear
ExN(:, Ny+1,:)=  ExP(:,Ny,:) + m1*(ExN(:,Ny,:) - ExP(:,Ny+1,:));    %   rear
- Ex;
```

```
EzN(:, Ny+1,:)=  EzP(:,Ny,:) + m1*(EzN(:,Ny,:) - EzP(:,Ny+1,:));    %   rear
- Ey;
%   Bottom
ExN(:, :,1)   =  ExP(:, :,2)  + m1*(ExN(:,:,2) - ExP(:,:,1));       %
bottom - Ex;
EyN(:, :,1)   =  EyP(:, :,2)  + m1*(EyN(:,:,2) - EyP(:,:,1));       %
bottom - Ey;
%   Top
ExN(:, :, Nz+1)=  ExP(:,:,Nz) + m1*(ExN(:,:,Nz) - ExP(:,:,Nz+1)); %    top -
Ex;
EyN(:, :, Nz+1)=  EyP(:,:,Nz) + m1*(EyN(:,:,Nz) - EyP(:,:,Nz+1)); %    top -
Ex;
```

abc_super.m

```
%%  FDTD-superabsorption
%   Super ABCs for H (Mei 1992, for Mur's first order BCs only)
%   Copyright SNM Spring 2011
coeff1  = (c0*dt - d)/(c0*dt + d);
rho     = c0*dt/d; RHO = 1 + rho;
%   Left
HyN(1,:,:) = (HyN(1,:,:) + rho*(HyP(2,:,:) + coeff1*(HyN(2,:,:) -
HyP(1,:,:))))/RHO; %  left - Hy;
HzN(1,:,:) = (HzN(1,:,:) + rho*(HzP(2,:,:) + coeff1*(HzN(2,:,:) -
HzP(1,:,:))))/RHO; %  left - Hz;
%  Right
HyN(Nx,:,:) = (HyN(Nx,:,:) + rho*(HyP(Nx-1,:,:) + coeff1*(HyN(Nx-1,:,:) -
HyP(Nx,:,:))))/RHO; %  right - Hy;
HzN(Nx,:,:) = (HzN(Nx,:,:) + rho*(HzP(Nx-1,:,:) + coeff1*(HzN(Nx-1,:,:) -
HzP(Nx,:,:))))/RHO; %  right - Hz;
%   Front
HxN(:,1,:) = (HxN(:,1,:) + rho*(HxP(:,2,:) + coeff1*(HxN(:,2,:) -
HxP(:,1,:))))/RHO; %  front - Hx;
HzN(:,1,:) = (HzN(:,1,:) + rho*(HzP(:,2,:) + coeff1*(HzN(:,2,:) -
HzP(:,1,:))))/RHO; %  right - Hz;
%   Rear
```

```
HxN(:,Ny,:) = (HxN(:,Ny,:) + rho*(HxP(:,Ny-1,:) + coeff1*(HxN(:,Ny-1,:) -
HxP(:,Ny,:))))/RHO; %  rear - Hx;
HzN(:,Ny,:) = (HzN(:,Ny,:) + rho*(HzP(:,Ny-1,:) + coeff1*(HzN(:,Ny-1,:) -
HzP(:,Ny,:))))/RHO; %  rear - Hz;
%   Bottom
HxN(:,:,1) = (HxN(:,:,1) + rho*(HxP(:,:,2) + coeff1*(HxN(:,:,2) -
HxP(:,:,1))))/RHO; %  bottom - Hx;
HyN(:,:,1) = (HyN(:,:,1) + rho*(HyP(:,:,2) + coeff1*(HyN(:,:,2) -
HyP(:,:,1))))/RHO; %  bottom - Hy;
%   Top
HxN(:,:,Nz) = (HxN(:,:,Nz) + rho*(HxP(:,:,Nz-1) + coeff1*(HxN(:,:,Nz-1) -
HxP(:,:,Nz))))/RHO; %  top - Hx;
HyN(:,:,Nz) = (HyN(:,:,Nz) + rho*(HyP(:,:,Nz-1) + coeff1*(HyN(:,:,Nz-1) -
HyP(:,:,Nz))))/RHO; %  top - Hy;
```

fc.m

```
%   FDTD MATLAB antenna/array solver (preliminary version)
%   Copyright SNM Spring 2011
%   FOURIER EXPANSION OF A PERIODIC SIGNAL ON A BOUNDARY
function [FIELD1] = FC(tempt, T, omega, FIELD)

    FIELD      = interp1(tempt, FIELD, T,'linear');
    func       = FIELD.*repmat(exp(-j*omega*T'), [1, size(FIELD, 2),
size(FIELD, 3)]);      %   integrand for Fourier coefficients
    FIELD1    = (1/length(T))*squeeze(sum(func, 1)-0.5*func(1,:,:)-
0.5*func(end,:,:));      %   trapezoidal integration

end
```

plot_nearfield.m

```
%   FDTD MATLAB antenna/array solver (preliminary version)
%   Copyright SNM Spring 2011
%   NEAR FIELD/POYNTING VECTOR PLOT
```

```
%%  Add common phase factor (to better resolve the phases without +/-pi
jumps)
factor = exp(j*pi);
ExtopP = ExtopP*factor;
EytopP = EytopP*factor;
HxtopP = HxtopP*factor;
HytopP = HytopP*factor;

%%  Determine which component to plot (dominant component is selected)
tempx = max(max(abs(ExtopP)));
tempy = max(max(abs(EytopP)));
CompE = zeros(Nx, Ny);
CompH = zeros(Nx, Ny);
if tempx>tempy
    for m = 1:Nx
        for n = 1:Ny
            CompE(m ,n) = (ExtopP(m, n)+ExtopP(m, n+1))/2;
        end
    end
else
    for m = 1:Nx
        for n = 1:Ny
             CompE(m, n) = (EytopP(m, n)+EytopP(m+1, n))/2;
        end
    end
end
tempx = max(max(abs(HxtopP)));
tempy = max(max(abs(HytopP)));
if tempx>tempy
    for m = 1:Nx
        for n = 1:Ny
            CompH(m ,n) = (HxtopP(m, n)+HxtopP(m+1, n))/2;
        end
    end
else
    for m = 1:Nx
```

```
        for n = 1:Ny
             CompH(m, n) = (HytopP(m, n)+HytopP(m, n+1))/2;
        end
    end
end
if tempx>tempy %    Hx dominates
    PoyntingP = -0.5*real(CompE.*conj(CompH));
    PoyntingQ = -0.5*imag(CompE.*conj(CompH));
else
    PoyntingP = +0.5*real(CompE.*conj(CompH));
    PoyntingQ = +0.5*imag(CompE.*conj(CompH));
end

%%  Plot magnitudes and phases of the near field on the top face (E and H)
scrsz = get(0,'ScreenSize');
figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
%   E-field in the xy-plane
subplot(2, 2, 1);
output = abs(CompE)';
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
Dx = (max(x)-min(x))/ColNo(end);
Dy = (max(y)-min(y))/RowNo(end);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
```

```
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('E-field magn.(V/m) at z=', num2str(planeZ), 'mm'));
colorbar;

%   H-field in the xy-plane
subplot(2, 2, 2);
output = abs(CompH)';
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('H-field magn.(A/m) at z=', num2str(planeZ), 'mm'));
colorbar;

%   E-field phase in the xy-plane
subplot(2, 2, 3);
output = unwrap(angle(CompE)');
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
```

```
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('E-field phase (rad) at z=', num2str(planeZ), 'mm'));
colorbar;

%   H-field phase in the xy-plane
subplot(2, 2, 4);
output = unwrap(angle(CompH)');
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('H-field phase (rad) at z=', num2str(planeZ), 'mm'));
```

```
colorbar;

%%  Plot magnitudes of the near field on the top face (P and Q)
scrsz = get(0,'ScreenSize');
figure('Position', [1 0.3*scrsz(4) 0.6*scrsz(3) 0.6*scrsz(4)]);
%   Poynting vector in the xy-plane (P)
subplot(1, 2, 1);
output = PoyntingP';
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('P=re(ExH*)/2 [W] at z=', num2str(planeZ), 'mm'))
colorbar;

%   Poynting vector in the xy-plane (Q)
subplot(1, 2, 2);
output = PoyntingQ';
output = interp2(output, 3);
imagesc( [x(1)+D/2 x(end)-D/2], [y(1)+D/2 y(end)-D/2], output);
for m = 1:length(Objects)
    Transparency = 0.0;
    if Ismet(m)
```

```
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    else
        viewer(W(m), L(m), H(m), XCtemp(m), YCtemp(m), ZCtemp(m), Color(m,
:), 0.0, 1, 'k');
    end
end
if ~isempty(custom)
    patch(Xshape, Yshape, Zshape, 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha',
0.5);
end
xlabel('x, mm'); ylabel('y, mm'); axis equal; axis tight;
title(strcat('Q=im(ExH*)/2 [W] at z=', num2str(planeZ), 'mm'))
colorbar;
```

Appendix F: Backpropagation MATLAB Code

```
clear all

ftemp   = 4e9;              %% Hz
c0      = 3e8;              %% m/s
Domain  = 0.75;             %% unitless
step    = 0.10;             %% unitless
d       = 0.1*25.4e-3;            %% m
deltaZ = -130e-3;         %% m

x = (-8:0.5:8)*25.4e-3;   %% m
y= (-10:0.5:10)*25.4e-3;  %% m

CompEfalse = dlmread('11','\ ',2,0);                                    %%
V/m
CompEfalse = reshape(CompEfalse(:,6)+1i*CompEfalse(:,7),length(x),length(y));
%% V/m

CompEtrue = dlmread('perfect','\ ',2,0);
%% V/m
CompEtrue = reshape(CompEtrue(:,6)+1i*CompEtrue(:,7),length(x),length(y));
%% V/m

CompE=CompEtrue - CompEfalse;
r1 = 0.0;                     %% Raised cosine window ratio for near field
r2 = 0;                       %% Raised cosine window ratio for k-space
r3 = 0.5;

%%  Define the propagator and window the E-field - raised cosine (r = 0 -
rect)

Nx = size(CompE, 1);
Ny = size(CompE, 2);
```

```
xw = tukeywin(Nx, r1);
yw = tukeywin(Ny, r1);
Window = xw*yw';
CompEW = CompE.*Window;

%%   Introduce the observation plane and domain of integration
[X1, Y1]    = meshgrid(x, y);     % grid

%   spatial harmonics
k   = 2*pi*ftemp/c0;
kx  = [-Domain*k:step*k:Domain*k];
ky  = [-Domain*k:step*k:Domain*k];
dkx = kx(2) - kx(1);
dky = ky(2) - ky(1);
fx = zeros(length(kx), length(ky));
fy = zeros(length(kx), length(ky));
KX_ = kx/k;
KY_ = ky/k;

%%  Fourier spectrum
for m = 1:length(kx)
    for n = 1:length(ky)
        argument = kx(m)*X1' + ky(n)*Y1';
        exponent = exp(j*argument);
        temp     = CompEW.*exponent;    % Ex
        fxy(m, n) = d^2*(sum(sum(temp))-
0.5*(sum(temp(1,:))+sum(temp(end,:))+sum(temp(:,1))+sum(temp(:,end))));
    end
end

%%  Windowing the spectrum - raised cosine (r = 0 - rect)
xw = tukeywin(length(kx), r2);
yw = tukeywin(length(ky), r2);
Window  = xw*yw';
fxyW    = fxy.*Window;
```

```
%%  Back propagation and inverse DFT
KX          = zeros(length(kx), length(ky));
KY          = zeros(length(kx), length(ky));
KZ          = zeros(length(kx), length(ky));
for m = 1:length(kx)
    for n = 1:length(ky)
        KX(m, n) = kx(m);
        KY(m, n) = ky(n);
        temp = kx(m)^2 + ky(n)^2;
        if temp<=k^2
            KZ(m, n) = sqrt(k^2 - temp);
        else
            KZ(m, n) = -j*sqrt(temp - k^2);
        end
    end
end

for m = 1:length(x)
    for n = 1:length(y)
        xtemp = x(m);
        ytemp = y(n);
        ztemp = deltaZ;
        argument = xtemp*KX + ytemp*KY + ztemp*KZ;
        exponent = exp(-j*argument);
        temp     = fxyW.*exponent;
        CompEprop(m, n) = 1/(4*pi^2)*dkx^2*(sum(sum(temp))-
0.5*(sum(temp(1,:))+sum(temp(end,:))+sum(temp(:,1))+sum(temp(:,end))));
    end
end

xw = tukeywin(Nx, r3);
yw = tukeywin(Ny, r3);
Window       = xw*yw';
CompEprop    = CompEprop.*Window;

%% PLOTS
```

```
scrsz = get(0,'ScreenSize');
figure('Position', [1 0.3*scrsz(4) 0.3*scrsz(3) 0.5*scrsz(4)]);
subplot(1, 2, 1);
imagesc(x,y,interp2((abs(CompEprop')),3));
xlim([-6*25.4e-3 6*25.4e-3]); ylim([-8*25.4e-3 8*25.4e-3]);
patch([-148e-3 -148e-3 148e-3 148e-3]/2,[-148e-3 148e-3 148e-3 -148e-3]/2,[0
0 0 0], 'w','EdgeColor',[0.5 0.5 0.5], 'FaceAlpha', 0.);
patch_spacing = 37e-3;
X_patchi = [-19.4e-3 -19.4e-3 19.4e-3 19.4e-3]/2-3*patch_spacing/2;
Y_patchi = [-19.4e-3 19.4e-3 19.4e-3 -19.4e-3]/2+3*patch_spacing/2;
for k = 0:3;
    Y_patchi = [-19.4e-3 19.4e-3 19.4e-3 -19.4e-3]/2+3*patch_spacing/2;
    for m = 0:3;
        patch(X_patchi,Y_patchi,zeros(1,4), 'w','EdgeColor',[0.5 0.5 0.5],
'FaceAlpha', 0.);
        Y_patchi=Y_patchi-2*patch_spacing/2;
    end
    X_patchi=X_patchi+2*patch_spacing/2;
end

title('Amplitude'); xlabel('x, m'); ylabel('y, m');
colorbar;
ylim([-200 200]);
subplot(1, 2, 2);
imagesc(x,y,interp2(flipud(180/pi*angle(CompEprop')),3));
patch([-148 -148 148 148],[-148 148 148 -148],[0 0 0 0], 'w','EdgeColor',[0.5
0.5 0.5], 'FaceAlpha', 0.);
patch_spacing = 37;
X_patchi = [-19.4 -19.4 19.4 19.4]-3*patch_spacing;
Y_patchi = [-19.4 19.4 19.4 -19.4]+3*patch_spacing;
for k = 0:3;
    Y_patchi = [-19.4 19.4 19.4 -19.4]+3*patch_spacing;
    for m = 0:3;
        patch(X_patchi,Y_patchi,zeros(1,4), 'w','EdgeColor',[0.5 0.5 0.5],
'FaceAlpha', 0.);
        Y_patchi=Y_patchi-2*patch_spacing;
    end
    X_patchi=X_patchi+2*patch_spacing;
end
title('Phase'); xlabel('x, mm'); ylabel('y, m'); axis equal; axis tight;
colorbar;
ylim([-200 200]);
```