NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GENETIC ALGORITHMS AS A TOOL FOR PHASED ARRAY RADAR DESIGN

by

Jon A. Bartee

June, 2002

Thesis Advisor:	Michael Melich
Co-Advisor:	David Jenn
Co-Advisor:	Rodney Johnson

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2002	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: Genetic Algorithms as a Tool for Phased Array Radar Design	**5. FUNDING NUMBERS**
6. AUTHOR(S) Jon A. Bartee	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(maximum 200 words)*

The United States Navy needs creative ways to design multi-function phased array radars. This thesis proposes that Genetic Algorithms, computer programs that mimic natural selection to arrive at innovative solutions to complex problems, would be particularly well suited to this task. The ability of a Genetic Algorithm to properly predict the behavior of an array antenna with randomly located elements was examined with encouraging results through the construction and measurement of a test array. Comparison of test data to Genetic Algorithm and Method of Moments calculations showed significant qualitative agreement in the antenna test patterns of a thin, randomly distributed array. Areas of disagreement between the test article pattern and the calculated ones were traced to systematic errors in the anechoic chamber and alignment error during antenna positioning. The final experiment to demonstrate beam steering was not completed due to lack of time and poor response of mechanical phase shifters. Despite the inability to demonstrate beam steering, the early experiments demonstrate the significant potential for using Genetic Algorithms for complex shipboard phased array radar antenna design.

14. SUBJECT TERMS Phased Array, Antenna, Radar, Radar Design, Air Search Radar, Evolutionary Computation, Genetic Programming, Genetic Algorithms, Theater Ballistic Missile Defense (TBMD), Area Air Defense.	**15. NUMBER OF PAGES** 91
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

GENETIC ALGORITHMS AS A TOOL FOR PHASED ARRAY RADAR DESIGN

Jon A. Bartee
Lieutenant, United States Navy
B.S., Texas A&M University, 1995

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED PHYSICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2002

Author:	Jon A. Bartee
Approved by:	Michael Melich, Thesis Advisor
	David Jenn, Co-Advisor
	Rodney Johnson, Co-Advisor
	William B. Maier II , Chairman Department of Physics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The United States Navy needs creative ways to design multi-function phased array radars. This thesis proposes that Genetic Algorithms, computer programs that mimic natural selection to arrive at innovative solutions to complex problems, would be particularly well suited to this task. The ability of a Genetic Algorithm to properly predict the behavior of an array antenna with randomly located elements was examined with encouraging results through the construction and measurement of a test array. Comparison of test data to Genetic Algorithm and Method of Moments calculations showed significant qualitative agreement in the antenna test patterns of a thin, randomly distributed array. Areas of disagreement between the test article pattern and the calculated ones were traced to systematic errors in the anechoic chamber and alignment error during antenna positioning. The final experiment to demonstrate beam steering was not completed due to lack of time and poor response of mechanical phase shifters. Despite the inability to demonstrate beam steering, the early experiments demonstrate the significant potential for using Genetic Algorithms for complex shipboard phased array radar antenna design.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

The author shares his deepest thanks to his wife, Jennifer, for her support, love and patience throughout both his academic and naval careers, and to my parents, who taught me that the value of education lies in its application to daily life.

Thanks go to my advisors, Mike Melich, Dave Jenn and Rod Johnson, as well as the colorful and only slightly retired Nick Willis for their patient mentoring. I also extend my deepest appreciation to the Deputy Assistant Secretary of the Navy for Theater Combat Systems, RADM Altwegg, for providing the funds for my research. Thanks to the Physics Department master-machinist, George Jaksha, whose work in fabricating precision components from innovative and inexpensive materials was critical. Thanks to Lieutenant Commander Brian Herdlick for his skill with electronics which proved invaluable in constructing the device and carrying out measurements. Lieutenant Commander Lance Esswein receives my gratitude for carrying on along the path in making this work a valuable tool for the defense of our country.

Thanks to my *alma mater*, Texas A&M University, for the excellent preparation given me, both academically and in life experience.

I also extend my gratitude to the men and women currently deployed around the globe, taking the fight to a vicious and fanatic enemy in some of the world's darkest places. This thesis is dedicated to them and should serve as a reminder that the US Naval Postgraduate School exists to give them the tools they need to win that fight for us.

"Greater love hath no man than this, that a man lay down his life for his friends."

John 15:13

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Naval warfare has changed dramatically in the last hundred years. The great close range gunfights of the battle line gave way to duels to the death between fleets that never got close enough to see one another, except through the eyes of young pilots. The revolutionary advances of the electronic age brought a whole new dimension to naval warfare. Warships must now contend with a broad range of threats, from inexpensive mines to small high speed and high lethality missiles. The range of threats has grown exponentially, and as the threat has grown so has the realization that resources to counter them are limited and must be chosen wisely. Consequently, new means to ensure the warfighter has the information he needs to survive are constantly in demand.

1. New Needs for New Warships

Naval architecture is in the midst of a major revolution. Warships are now being designed for minimal radar cross section (RCS) and minimal manning requirements. Systems put on board must be low maintenance and able to function in all environments, both natural and man made. Indeed, the Navy is currently looking for a new generation of advanced multi-function array radar (MFAR) systems to equip the next generation of surface combatants, the DD(X) class. The engineering plant must be capable of both fast speeds and efficient operation. Communications and information systems must be seamless. Combat systems must be reliable and lethal. The problem lies in developing innovative methods of meeting all of these requirements simultaneously. Figure 1 shows an artist's conception of DD-21, the now defunct predecessor of the DD(X) program. DD-21 was cancelled for several reasons, not the least of which was the lack of a viable radar design. A method needs to be found that would allow high volume search radars suitable for such missions as Theater Ballistic Missile Defense (TBMD) and Fleet Air Defense (FAD) as well as surface surveillance for Surface Warfare (SUW) to be constructed more reliably and cheaply, and be able to be put on smaller ships with no loss of capability. Unfortunately, no such process exists today. Much of the design of array antennas is based on traditional approaches. Modern array radar antennas are large planar arrays, heavy, maintenance intensive, easily recognizable and vulnerable. In fact, these arrays and the support structure they require are often the driving factor in the construction of a ship.

Figure 1. DD-21 Artist's Conception, Predecessor to DD(X) (From: January 99 *All Hands*)[1]

2. A Solution to the Need

What if another option existed? The ability to design the ships structure from the standpoint of survivability, sea-keeping and RCS reduction, with the ship's sensors and communications added to the design later and placed wherever convenient for power and reducing topside weight. The radars designed this way would also have the advantage of having extremely good resolution, as the angular resolution is proportional to the ratio of the wavelength, λ, over the aperture width of the array, D. Instead of the compact width of a conventional planar array, the aperture width might conceivably be approximately the overall length of the ship, with consequent dramatic improvement in beam width, or the ability to use lower frequencies and maintain the same currently available resolution. Such a ship wide distributed array is possible through the use of a branch of computer programs known as Genetic Algorithms (GA) that use logic modeled on natural selection in nature to achieve a variety of tasks, from optimizing a system to searching large databases or even adaptively controlling electronics.

[1] Lockheed-Martin artist's conception of DD-21 taken from the Navy Media Center Website, *All Hands* publication section. Retrieved June 8, 2002 from the World Wide Web: http://www.mediacen.navy.mil/pubs/allhands/Jan99/janpg48.htm.

The purpose of this thesis is to show that Genetic Algorithms have the potential to become a powerful design tool for ship-based phased array antennas.

Planar arrays of regularly spaced elements are currently in use because of their simplicity in design and construction. However, the benefit of using a Genetic Algorithm is its ability to adapt to meet complex criteria over a series of several generations. The program effectively “evolves“ a solution over the course of time. Such evolved solutions are rarely intuitively obvious, even to experienced professionals in the field, because the algorithm is not tied to any preconceived ideas of how to proceed. The GA therefore has the capacity for innovation that would be difficult for a human engineer.

The process of using a Genetic Algorithm for antenna design would begin with determining the required specifications for the antenna. A series of GA program runs would be made to determine the evolved array configuration that optimally fits those criteria. Constraints can be placed on the program that limit it to determining solutions that are workable in reality, such as denying the algorithm the choice of placing elements on the hull near the waterline or on the superstructure in the same locations as engine intakes and access doors. Allowing the algorithm to choose from any vacant location is what provides the benefit of the GA over conventional methods. The GA has no loyalty or inclination toward previous solutions. Preference is given only to optimum performance. Furthermore, instead of designing a ship from scratch around the sensors and communications suite, RF elements can be distributed throughout available locations on any ship design, wherever service connections are convenient. A distributed ship-wide array approach would minimize impact on seakeeping and stability, reduce vulnerability to damage and allow for significant improvement in angular resolution. As long as the GA has been programmed to accurately reflect the hardware that will be used in the array, any type of Transmit/ Receive (TR) elements may be used to construct the array. Inexpensive commercially available TR elements made from mass produced semiconductors could become a viable option for making a distributed ship-wide array a real possibility. Cheap, low weight and with almost no maintenance required, semiconductor elements might only need a power feed and connections for injecting and receiving a digital signal. Other Commercial Off The Shelf (COTS) components could also be integrated into radar systems using Genetic Algorithms for optimization.

GA’s are not limited to any specific constraint beyond the imagination of the programmer and the limits of what the current level of technology can translate into hardware. For example, Genetic Algorithms could be used to provide tactically realistic means for utilizing bistatic radar configurations, both on a single platform and between several ships. GA’s could also provide a method for providing optimized settings for currently existing arrays specific to the propagation environment.

A good example of a future vessel that might benefit from the use of a GA-based design tool is the Littoral Combat Ship (LCS). The LCS is a part of the upcoming DD(X) family of ships and will concentrate primarily on wresting control of near shore areas from hostile forces. Because this ship’s inshore mission requires a vessel of small size, no air search capability is currently planned, however the use of a Genetic Algorithm to place array elements in various topside locations might allow the introduction of an air surveillance capability, without compromising displacement, lethality, reduced signature or speed. Indeed, this capability would seem necessary to provide the ability to control unmanned air and surface vehicles, another capability envisioned for the LCS.

Genetic Algorithms therefore have the potential for the design of advanced radar antennas with complex geometries, but do they maintain the necessary fidelity to reality for this concept to be truly applicable to the design process?

B. SCOPE AND ORGANIZATION

1. Scope

This thesis will test a portion of a specific GA program, Athin.m, with the performance of a physical array antenna through experimental measurement, but will not attempt to leap directly into validating the design capabilities of the program.

The portion of the Athin.m code to be tested is that which builds the antenna patterns for the program and thereby allows it to calculate the effectiveness of a given solution. The parameters for the test array will be determined without the assistance of the GA.

2. Primary Research Question

The cornerstone for any GA program is the fitness function that quantifies the effectiveness of a potential solution. Given that the Athin.m Genetic Algorithm code derives this claculation by first generating an antenna pattern, does the Athin.m Pattern Builder function sufficiently reflect reality to justify using it as a design tool for future phased array radar antennas?

3. Organization

Chapter II provides a functional overview of Genetic Algorithms in general and the Athin.m program in particular. A description of what defines a GA is followed by discussion of the logical structure and mechanisms used in genetic programming. The Athin code is then presented with actual values used to illustrate the programmer's decision making process.

Chapter III begins with a discussion of the experimental objectives. After a brief discussion of the instrumentation used for measurements, a detailed description of the construction of both variants of the test array is presented. The three steps in the measurement process are then discussed, in the order they were taken.

Chapter IV provides a summary of the experimental results followed by suggestions for further research into the use of GA's for phased array design.

Appendix A provides a glossary of terms and abbreviations used throughout this thesis.

Appendix B provides a complete program listing of the Athin.m script file for MATLAB.

Appendix C provides the random element locations in the X-Y plane used for all the experimental computational purposes.

THIS PAGE INTENTIONALLY LEFT BLANK

II. GENETIC ALGORITHMS

A. NATURAL SELECTION IN THE COMPUTER AGE

The concept of natural selection has been accepted in the biological sciences for some time. Biological organisms improve their ability to survive through a generational process of optimization. Evolutionary processes occur in biological systems when the organism has the ability to reproduce itself, there is a population of such organisms, there is variety among the members of the population and that variety can be related to the ability to survive in the environment.[2] Traits which assist in survival are continued and improved through the mechanism of reproduction. Individuals in a population that are the best suited to survive generally receive preferential treatment in the procreation process. However, it is important to note that not only the best-suited individuals produce offspring. The success and health of a population depends as much upon its genetic diversity as upon the best individual examples of its membership. Without variety in survival characteristics there would be no means of improvement.

Genetic Algorithms are written along similar lines in order to optimize non-biological systems at a much faster rate and at much less expense than could otherwise occur through trial and error manufacturing or simulation. The computer's ability to do tedious repetitive calculations at high speed allows it to perform iterative processes hundreds of thousands or millions of times within a useable timescale for problem solving. The difficult part is often not the program run itself, but the determination of what the problem is and how it can be put into a program in a form useable by a digital processor.

A Genetic Algorithm proceeds through a succession of generations. Each generation is composed of a number of individual population members. These are vectors consisting of characterized traits that form a potential complete solution of the problem. Figure 2 shows the logical flow of solving a problem using a Genetic Algorithm. Note in particular the loop structure. Each iteration of the loop is considered a generation. The relatively simple structure potentially allows several thousand generations of evolution to

[2] Koza, J. (1992). *Genetic Programming.* Cambridge, MA: MIT Press. p. 17.

be completed in only a matter of hours with even modest computing resources. However, as a practical matter, the time it takes to complete a loop depends heavily upon how long it takes to do the fitness evaluation. Consequently, interesting problems that apply to real world needs often have a tendency to get bottlenecked computationally on the evaluation step. It should also be clear that the loop structure as shown will continue indefinitely until stopped in some way. The question of when to halt the evolutionary process is another consideration for the programmer.

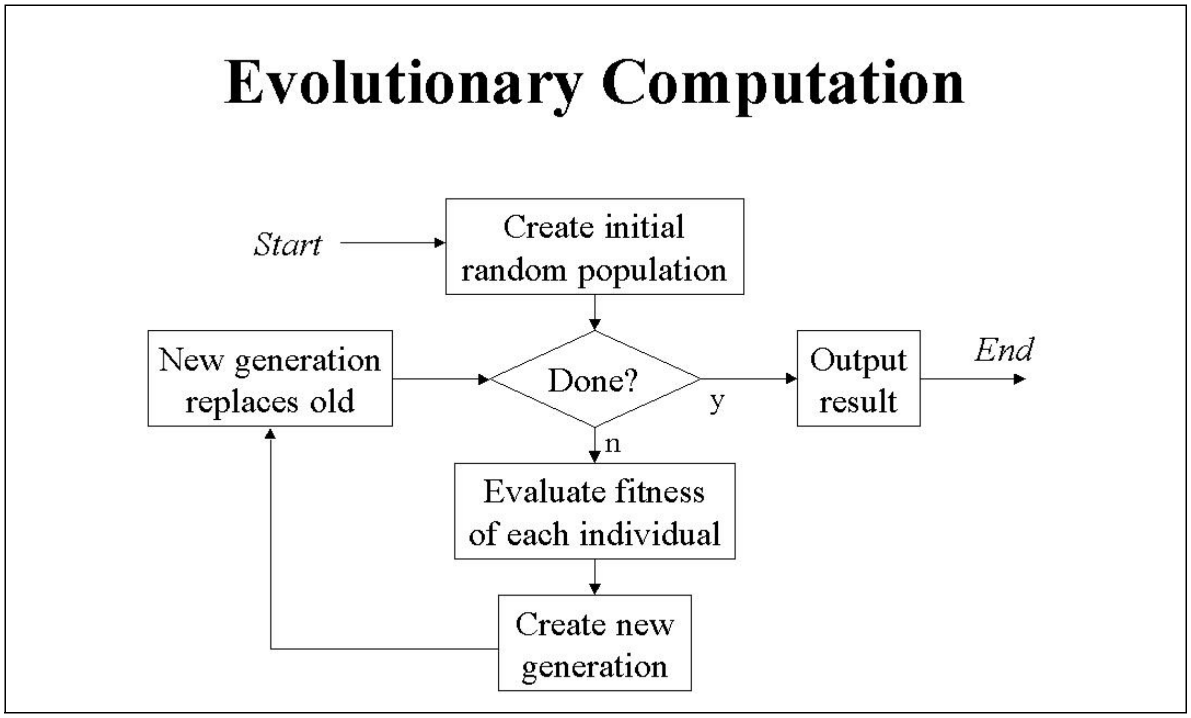

Evolutionary Computation
Start
Create initial random population
New generation replaces old
Done?
y
n
Output result
End
Evaluate fitness of each individual
Create new generation

Figure 2. Genetic Algorithm Logic Flow (From Johnson, 15 August 2001[3])

"Fitness" is the criterion to be optimized and is the basis for selection of individuals from the population of each generation. The method used to evaluate the fitness of individual population members, as well as the four commonly used reproduction methods for determining the population of each generation will be covered in more detail.

[3] From a slide presentation given by Dr. Rodney Johnson developed for the Institute for Joint Warfare Analysis (IJWA) at the US Naval Postgraduate School, "Phased-Array Antenna Patterns via Genetic Algorithms", on 15 August 2001.

B. MECHANISMS IN GENETIC ALGORITHMS

Genetic algorithms use several mechanisms to evolve a system over the course of the run. Selection of which mechanisms to use, and the probability each will have, is a critical decision for the programmer. Since Genetic Algorithms are an iterative process, modeling either the fitness criteria or individual incorrectly can cause the program to diverge from answering the problem of interest to the programmer. However, by self-correcting over the course of a large numbers of these iterations, using assumptions and criteria based on proven theory, the algorithm has the capability of finding a set of most favorable solutions to highly complex problems that might otherwise take years of measurement and data collection.

1. Fitness Measurement

The initial item the programmer must address is how to evaluate the traits of an individual population member against the desired outcome. This is known as a "fitness measurement," and it must be performed for each individual during each consecutive generation of the run in order to be able to faithfully rank the individuals' suitability to deliver the desired result. Defining the fitness function is the most critical step in the process. A failure to effectively shape the question at hand in a form that can be translated through a programming language into a measurement of fitness for each individual relative to each other prevents the preferential treatment of the best suited individuals to the next generation. This allows too many of the weaker members to move on and the population will continue to be characterized by randomness. Also important is the shaping of the fitness criteria based on the reality of the problem. In using a Genetic Algorithm to design and optimize an electrical or mechanical system for example, the actual performance parameters and physical limits of such a system must be faithfully reflected in the mathematics used to determine the relative ranking of the individuals. Without this fidelity to the physical world you might very well get an optimized solution to the problem posed, but that problem might not reflect the true complexity of the environment it must exist in and is therefore useless as anything other than an academic exercise. The fitness calculation may therefore be of a very complex nature.

The process is twofold. The fitness of each individual must first be evaluated and compared to its peers, then individuals must be selected for a new population. The traits for each population member are evaluated using the fitness criteria, usually involving an analytic or numerical evaluation of a mathematical formula. The criterion must provide enough resolution that two individuals will usually have different values in order to be able to rank the entire population. For example, in the case of a simple radar antenna design problem, the fitness assessment might involve determining the antenna gain for each individual. The traits of the individual might be those elements of gain which are under the control of the designer: radar frequency, *f*, the antenna aperture efficiency, ρ_a, which is controlled by antenna shape and the physical area of the antenna, *A*. The fitness assessment for each member would involve using the member's traits to calculate antenna gain in decibels, G_{dB}.[4]

$$G_{dB} = 10\log\left[\frac{4\pi\rho_a A f^2}{c^2}\right]. \tag{1}$$

Where *c* is the speed of light. The resulting gain would allow the individuals to be ranked from most fit, meaning highest gain, to least fit. Note that the fitness criterion is not expressed as a set binary limit, such as "above 30 dB," as this limits the ranking of individuals to only two categories.

2. Population Selection

The population of any GA is composed of individuals. Each individual has discrete traits that characterize the individual and are directly applicable to the mathematics involved in determining fitness. Generally speaking, the initial population is determined randomly for the first generation, and by the fitness assessment, selection, and creation of new individuals in subsequent generations.

a. Seeding

A refinement to population selection is the concept of "seeding" the initial population of a Genetic Algorithm with the results of a previous run or predetermined configurations that represent probable solutions based on known facts, problem solver experience or even the best ranked results from previous runs. Not only does the

[4] Skolnik, M. (2001). *Introduction to RadarSystems, 3rd edition*. New York: McGraw-Hill. p 6.

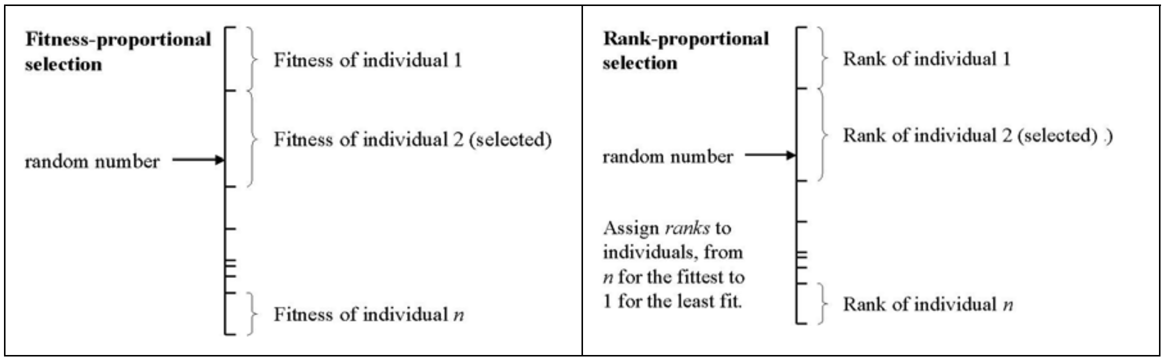
introduction of evolved, known or probable solutions cause more rapid convergence, but it also allows the GA to be adjusted between runs in order to better track toward the desired goal.

b. Fitness- and Rank-Proportional Selection

Once the fitness of each individual has been determined the following step in formation of a new generation involves selecting the individuals that will be allowed to contribute genetic material to the next population through one of the genetic operators, described in detail later. Individual population members are selected for membership in the next generation by their relative fitness ranking with one of several methods, two of which are described below. Sometimes, a probability of selecting less fit individuals over more fit ones is included to retain some of the diversity of the original population, but the more fit ones must always have a higher probability of selection in order for a solution to emerge. A broader population diversity will result in slower convergence to the solutions of a problem and will require more computational resources and time, but has a greater chance of arriving at a better and perhaps unanticipated solution.

Fitness-proportional selection means that the probability of an individual being selected for continuation is weighted based on its performance during the fitness evaluation. One common method for fitness-proportional selection involves the creation of “bins,” one for each individual present in the population. The size, or length, of an individual’s bin is proportional to its assessed fitness. A random number is generated within the value range of all the bins, and the individual in whose bin this number falls is selected for inclusion in the next generation, after the application of a genetic operator. Like pitching pennies into cups of differing sizes, there is a finite probability of selection for any individual but the individuals with better fitness scores have a higher probability of selection.

A problem can arise with fitness-proportional selection when the raw fitness score values of most of the individuals are close to each other. This is particularly evident in later generations of a run, where all the individuals have begun to converge on a narrow range of solutions. The method for fixing this problem is to use rank-proportional selection. In rank-proportional selection, individuals are ranked with an integer value based on their raw fitness score, from 1 for the least fit to the population size, *n*, for the most fit. They are again placed in bins, but the bin size is now proportional to the integer ranking. The highest ranked individuals have the largest bins and therefore the higher probability of selection. Figure 3 illustrates this difference.



Fitness-proportional selection
Fitness of individual 1
Fitness of individual 2 (selected)
random number
Fitness of individual n
Rank-proportional selection
Rank of individual 1
Rank of individual 2 (selected))
random number
Assign ranks to individuals, from n for the fittest to 1 for the least fit.
Rank of individual n

Figure 3. Fitness-Proportional vs. Rank-Proportional Selection (From Johnson, 15 August 2001[5])

3. Generalized Genetic Operators

Another item of concern to the writer of a genetic algorithm is the method by which individual population members will be used in creation of the next generation once their fitness has been ranked. There are four methods usually used for this process: reproduction, crossover, inversion and mutation, with each of the four having potentially independent mechanisms for determining if they occur. Each is normally assigned a probability of occurring, with this probability weighted in favor the individual's fitness ranking, thereby giving traits of the most fit the best chance of survival into later generations. Using these four mechanisms, a new generation is formed and the fitness test is applied once again.

a. Reproduction

Reproduction, or the inclusion of an unaltered individual population member in the next generation, is the simplest process of promotion for any Genetic Algorithm. Figure 4 shows the reproduction to a new generation graphically. When an individual population member is selected for reproduction the traits of the member, denoted as the vector a_1 through a_n in Figure 4, are copied directly into an available slot for a population member in the new generation. No changes or rearrangements are made.

[5] Johnson; "Phased-Array Antenna Patterns via Genetic Algorithms"; 15 August 2001

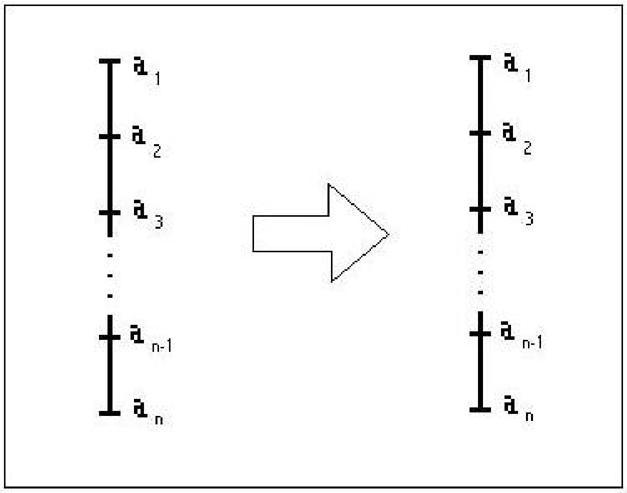


Figure 4. Illustration of Reproduction

The biological equivalent of this process could be seen as either survival of the individual or procreation through asexual reproduction, producing an identical copy of a single parent in the new generation. This mechanism is often assigned a significant probability of occurrence, although not as high as crossover. This same process is alternatively called copying or promotion.

b. Crossover

Crossover mimics sexual recombination in biological organisms. Starting with two parent members of the population, a number of distinct traits are swapped between mated pairs to produce two offspring, each different from the other and also from their parents, but with "genetic material" common to the family line. Crossover is also controlled by probability, again usually weighted in favor of selecting more fit individuals as parents. Like reproduction, crossover is usually assigned a relatively high probability of occurring, usually exceeding the proportion assigned the other operators. The programmer must decide on values for some specific parameters that are not required for simple reproduction. Two parents must be selected based on fitness, rather than one. The number of traits that will be crossed between mated pairs must be determined, and then a process must be included to determine which specific traits this will be. The specific traits are often selected randomly to further promote innovative results. Figure 5 shows an example of the crossover operation at work.

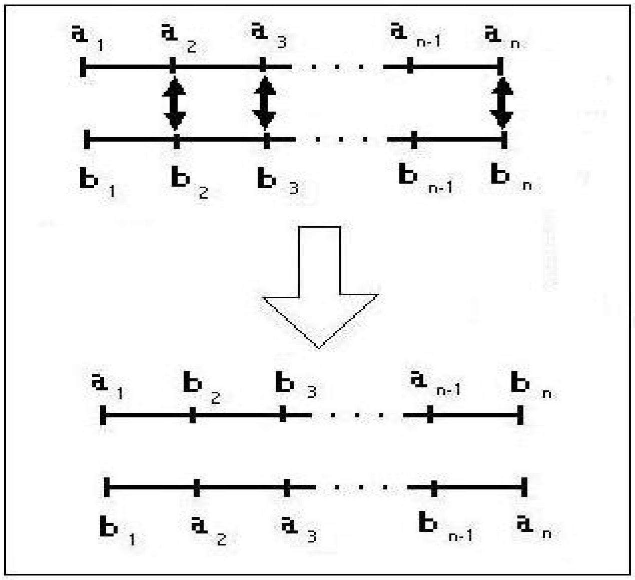

a_1 a_2 a_3 a_{n-1} a_n
b_1 b_2 b_3 b_{n-1} b_n
a_1 b_2 b_3 a_{n-1} b_n
b_1 a_2 a_3 b_{n-1} a_n

Figure 5. Illustration of Crossover

In the case shown in Figure 5, an individual with traits a_1 through a_n has been selected to mate with another member with traits b_1 through b_n. The programmer has chosen to use a three point crossover, and traits with indices 2, 3 and n have been chosen to be swapped between the parents. The resulting offspring are uniquely different from each other, and each is also different from both parents. Yet both share traits with parents who were more than likely to have been ranked higher than the average in fitness. Population members whose vectors schemes have few traits may be effected little by crossover operations.[6]

c. *Inversion*

Inversion is an unusual reproductive mechanism in Genetic Algorithms, both for its effect on the selected population entity and because it really has no corollary in biological systems. If used at all, the probability of this type of genetic operation is often set very low compared to the previously mentioned cases. A single population member is chosen at random, again weighted toward the most fit individuals. Again, an even number of random indices are chosen, usually two. There is no specific reason why only two points must be used for the process, but it keeps the operation simple and avoids unnecessary randomization of the traits. The vector is then effectively folded between

[6] Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. Cambridge, MA: MIT Press. p. 97-106.

traits with these two indices. This process is simplest to understand in illustration. Figure 6 shows a typical inversion operation.

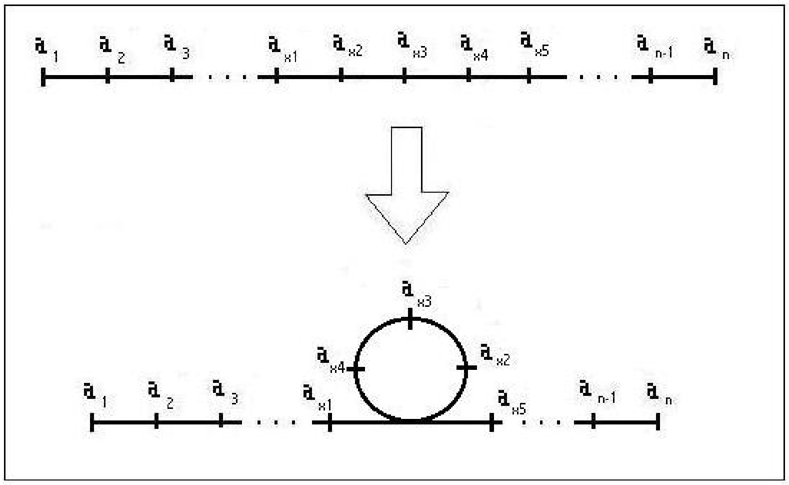

a_1 a_2 a_3 a_{x1} a_{x2} a_{x3} a_{x4} a_{x5} a_{n-1} a_n
a_{x3}
a_{x4}
a_{x2}
a_1 a_2 a_3 a_{x1} a_{x5} a_{n-1} a_n

Figure 6. Illustration of Inversion (After Holland p. 107.)

Any possible rearrangement of traits can be accomplished by successive inversions. Also, like crossover, inversion has little effect on populations with individuals that have only a small number of character traits.[7]

d. Mutation

Mutation is another probability controlled process which introduces random changes to the characteristics of an individual member, encouraging diversity in the population as a whole and therefore increasing the chances of a unique and otherwise unexpected solution. Directly analogous to the biological process of the same name, mutation can be easily applied in two distinct ways by the designer of a Genetic Algorithm. Either the probability mechanism can be applied the same way as with the other operators, with each individual having a set probability of a mutation somewhere in the individual's traits, with another random process determining which trait is effected, or in a more complex manner which can have a dramatically different effect. Unlike previous operators, arguably the most effective way to apply this genetic operator is by allowing an extremely small but non-zero probability of a random change occurring for each trait in all individuals, rather than for the individual as a whole. The probability

[7] Ibid. p .107.

must be independent of whether mutation occurred in adjacent individuals, or even in an adjacent characteristic. This second method is undeniably more computationally intense.

Traits selected for mutation are replaced with a random value within the designed limits of the attribute. Figure 7 shows the mutation process.

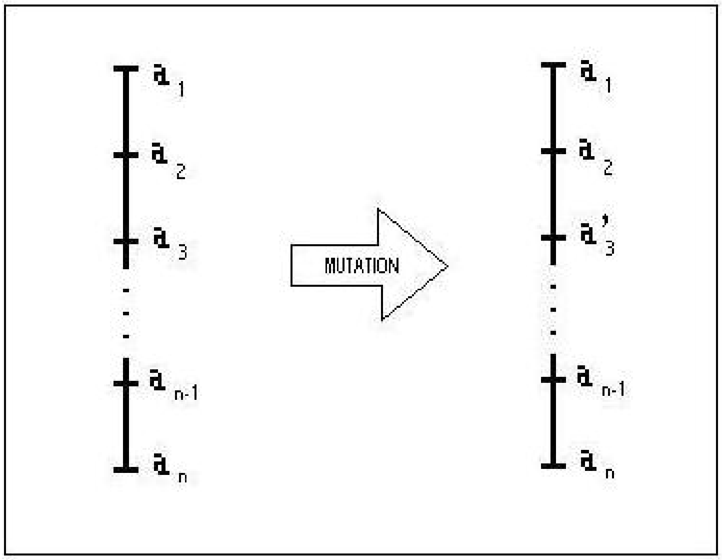


Figure 7. Illustration of Mutation

All the characteristic traits for this individual, as well as all other individuals in the population, will independently be tested against a small operator-selected value for the probability of mutation. When a_3 is selected for mutation, a new trait value, a_3', is chosen at random to replace it in the new generation.

4. Stopping the Process

The final question that must be answered by the programmer is determining when to stop the process. If an exact solution is known, the code can be designed to stop on its own when it is achieved. However the use of a Genetic Algorithm in this case would be unnecessary! Computational constraints, such as programming language and hardware, are not only factors that limit population size, individual member complexity and fitness calculation intricacy, but also may limit the allowed run time on scarce computing resources. Genetic Algorithms will rarely arrive at an exact solution anyway, regardless of the amount of time allotted, due to embedded encouragement in the process for continued population diversity. The code may include a process by which an operator or the program itself may siphon off and observe the results of the process every few generations in order to determine if the algorithm is tracking in the desired direction or has achieved a solution that is good enough to be within set error limits for the task required. Either the programmer can then interrupt the process, or it may be programmed to jump out of the iterative loop. Other limiting factors may exist to curtail the run before the optimum solution is achieved, such as cost of constructing a physical device based on a Genetic Algorithm solution. By far the simplest way to end the run is to set a counter and run a pre-specified number of generations. At the end of these runs, the final output is all or part of the population of the final generation. Records may also be generated of previous generations, so that earlier sub-optimal solutions may be used that keep materials, labor and complexity within the available budget.

C. THE ATHIN.M CODE

A brief description of the code used in our experiment is in order, so that the thought processes involved in formatting a Genetic Algorithm to solve the particular problem of designing phased array radar antennas can be fully explored. Named Athin.m, the code was written by Dr. Rodney Johnson of the US Naval Postgraduate School in the programming language used with the commercially available mathematical and graphing program MATLAB.[8] Dr. Johnson tailored the code for the purpose of designing answers to the complex but known problem of phase-only beam-shaping and steering of an array of randomly positioned elements. A complete program listing of the actual MATLAB code is provided in Appendix B.

1. Population Design

The first step in solving the random array problem is to determine what parameters are physically controlled by the antenna engineers and list these as the traits that characterize an individual population member. The Athin.m Code was originally designed to show that similar or superior antenna patterns could be realized using fewer elements by evolving antenna patterns optimized for maximum peak main lobe to average or maximum side lobe power levels with a thinly-populated array. A first round of experimental computations used 100 elements randomly placed in a 10λ by 10λ planar array. A fully populated array of this size would have four hundred elements, assuming

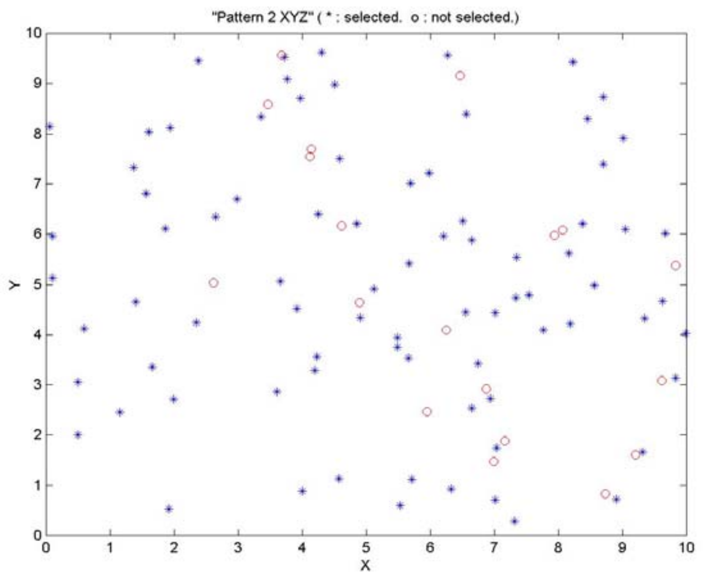
[8] Copyright, Mathworks, Inc.

half wavelength ($\lambda/2$) spacing. In this case each individual population member had two hundred traits: each element location had an “on-off” bit that describes whether a particular location is turned on or not and a phase shift setting. For comparison, an antenna pattern was calculated for a broadside beam, with all phases set to zero and all elements turned on. As Figure 8 shows, the Athin GA code was able to thin the array by a further twenty percent while maintaining comparable if not slightly better performance for the ratio of peak main lobe to maximum side lobe by turning unnecessary elements off and compensating for them with phase differences at other elements. The unusual selection of which elements to turn off is a good illustration of the original nature of many GA solutions. It is in fact a difficult problem to determine which elements to turn on without using a GA. Currently accepted models for constructing beam patterns such as Method of Moments, have no inherent capability to do this. The antenna engineer or physicist with long experience working with arrays may be able to intuitively narrow the required number of trials, but it would still involve large amounts of time spent in trial and error and there would be no guarantee that the final result was the best possible solution without trying every possible permutation.

For the purposes of experimental verification, however, the code required modification to fit the relatively modest means available for measurement. Cost and size restrictions limited the number of elements to twenty four. The locations of the elements were determined randomly, external to the GA. The twenty four elements could be selected either “on” or “off.’ Each element could also again be assigned a relative phase value that allows beam-shaping and steering. Therefore, each population member consisted of forty eight distinct pieces of information that formed a complete configuration for the array.

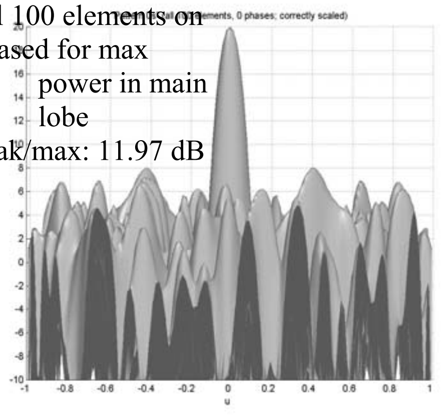
Three runs were made, with somewhat different values for some parameters of the GA. There were 5000 members in each population for all three runs. The initial population of the first run was randomly determined, while the subsequent two runs were seeded with the results from the previous run.

*** - selected (81 elements)   o - not selected (19 elements)**



Non-Evolved Example

All 100 elements on
Phased for max
power in main
lobe
Peak/max: 11.97 dB



Evolved Example

Pop: 3000
200 points used for
fitness evaluation
300 generations
81 elements on
Peak/max: 12.23 dB

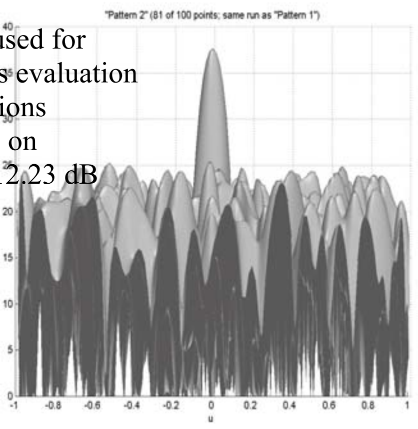


Figure 8. GA vs. Conventional Solution for 100 Element Random Array (Johnson, 13 March 2002[9])

[9] Jenn, David and Johnson, Rodney; "New Design Codes and Methods for Random-Element Phased-Array Antennas"; Presentation and slide show given at the US Naval Postgraduate School for representatives of the Office of Naval Research (ONR); 13 March 2002.

2. Fitness Criterion

The measure of fitness used was the ratio of the peak power in the main lobe to an estimate of the average value for power in the side lobes. The anechoic chamber that was available for testing was nineteen feet in length from the passive device under test to the feed horn that transmitted the microwave signal. Our computations assumed a near-field approximation for a main lobe focused at this distance. The average side lobe level was estimated in the GA by taking the square magnitude of the electric field strength, approximately equivalent to power, over a set of points on the hemisphere in front of the antenna with a radius of nineteen feet. The number of sampled side lobe points varied from run to run, but generally increased from the initial run to later ones. Typical values started in the hundreds and increased by the final run into the thousands to ensure the solution converged with sufficient resolution to have confidence that there were no large hidden lobes in the result. The locations of these points were randomly selected from a uniform distribution upon the hemisphere. The number of points was user-controlled for each run. The first run looked at 500 points, the second 1000 and the third run examined 2400 points. The ever-present and relatively high shoulders that are very near the main lobe were excluded from being valid side lobe locations on the hemisphere, and points selected in this region were discarded. It was considered unnecessary to make up additional points to replace the ones that were thrown out.

In order to determine what these power values are, the Athin code was equipped with a pattern builder to generate a magnitude versus azimuth and elevation plot for the expected signal. In order to do this, the array factor needed to be calculated for each individual in the population as a part of the fitness criteria. Array factor provides the effective pattern if the array were made up of isotropic radiating elements. The array factor, G_a, in terms of field amplitudes for a randomly distributed array is given by:

$$G_a = \left(\frac{3}{2\ N}\right)^{1/2} R \left| \sum_{j=1}^{N} \frac{a_j \exp 2i\pi\left(R - \rho_j + \Phi_j\right)}{\rho_j} \right|^2 . \qquad (2)$$

Where N is the number of elements in the array, in this case 24. The variable ρ_j refers to the X, Y and Z coordinates of the jth element, while R is the coordinate of the observation point. The "on/off" bit is represented by a_j, and is simply 1 if j is on, and 0 if it is off. The phase setting of the element, in cycles, is given by Φ_j. The value of Z for all elements was zero, i.e. the surface of the ground plane was taken to be the X-Y plane. This was a constraint of the physical antennas we were capable of building, however, rather than one of the GA. The ability to include three dimensional locations in space for element locations demonstrates the engineering flexibility made available by using the GA concept as a design tool.

In practice, the dipoles are placed over a ground plane to increase directivity. The effect of an infinite ground plane can be accounted for by using images. For each element above the X-Y plane an image dipole is introduced and the ground plane removed. The currents on the image dipoles are opposite of those on the source dipoles. Thus the equivalent problem for the array over an infinite ground plane is a two layer array in free space.[10]

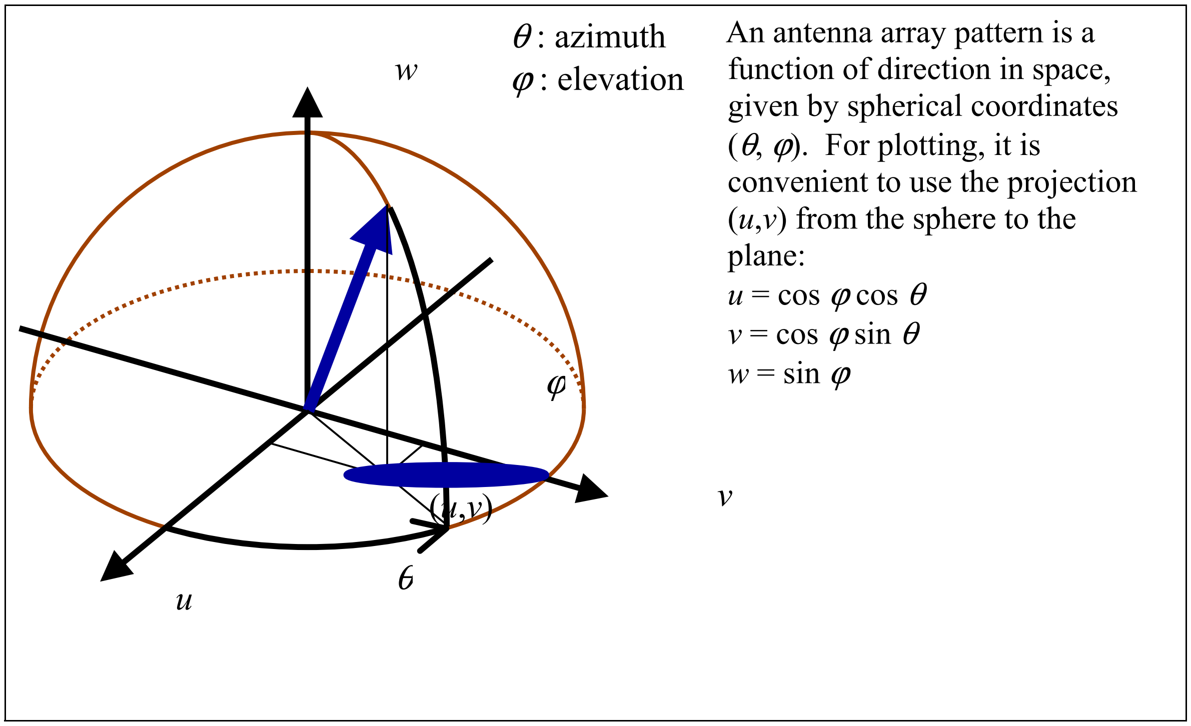
No antenna actually has a perfectly isotropic pattern, and therefore the array factor alone does not produce an entirely precise pattern without also multiplying it with an element factor, G_e, that represents the pattern created by an individual radiating element. The original 100 element computations used the array factor alone, but for comparison with experiment a more realistic calculation was necessary. A cosine theta ($\cos \theta_j$) element factor was included for completeness, and the effects of the back plane were modeled by replicating each element, once with a positive Z value of $\lambda/4$ and once with a negative Z value of $-\lambda/4$ and an additional phase shift of half a cycle. Further detail could also have been incorporated by taking mutual coupling and manufacturing dissimilarities between elements into account, but these effects are small enough that the product of the array and element factors is a widely used approximation.[11] The resulting pattern is then sampled at the chosen number of points to determine fitness.

3. Coordinate System

Spherical coordinates were the most convenient for expressing antenna patterns. Figure 9 breaks down the coordinate system used throughout the rest of this thesis.

[10] Griffiths. (1999) *Introduction to Electrodynamics, 3rd Edition*. p. 121-125.

[11] Skolnik; p. 562-3.



An antenna array pattern is a function of direction in space, given by spherical coordinates (θ, φ). For plotting, it is convenient to use the projection (u,v) from the sphere to the plane:

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

Figure 9. Coordinate System (From Johnson, 15 August 2001)

4. Genetic Operations

Reproduction, mutation and crossover were all used in the Athin code. Inversion was not used, as changing the order of elements' phase and on/off bits would have an overly randomizing effect on the results that made little sense in promoting convergence on an optimum configuration. The proportions of each operator changed with each run as well in order to promote the introduction of "random innovation" as the population members began to converge and lose diversity. The ratio of operations for the three runs, in the order of (reproduction: crossover: mutation) was: 20:79:1, 18:80:2, and 15:80:5, respectively. All of the operations were performed using rank-proportional selection methods.

The crossover operation was performed using two points. Two parents were selected proportional to rank. Then, two randomly chosen indices, i and j, were selected such that they fell within the range of 1 to N, with i always less than j. The values for the on/off bit, a_k, were swapped between these two parents for k in the range $i \le k \le j$, as was the phase setting, Φ_k. Furthermore, the phases at the two ends were perturbed by a random amount proportional to the difference between the values for the two parents. The motivation for doing this was to roughly approximate the behavior that would have been obtained if the phases had been represented in fixed point binary, bitwise two point crossover had been used, and the endpoints of the interval had fallen somewhere within the representations of the *i*th and *j*th phases.

The mutation operation was fairly straightforward. Indices *i* and *j* were chosen in the same fashion as for crossover. The elements between these two indices had their on/off bits replaced with a randomly selected value of either 0 or 1. Each had equal probability. Likewise, the phase values for elements between the endpoints were replaced with a new phase value, a random floating-point number between 0.0 and 1.0.

5. Figures of Merit

In order to determine the relative degree of success for the results of a series of Genetic Algorithm runs, as well as get an idea of the computational and time resources required to achieve the end results, a look at some of the more important parameters and performance characteristics of the Athin code is in order. Particular attention should be paid to which factors changed between runs and which did not. Table 1 provides the Figures of Merit (FoM) for the three runs described.

FoM	Run Number		
	1	2	3
No. of Generations	36	36	15
Population Size	5000		
Selection Method	Rank-Proportional		
Rep:Cross:Mutate	20:79:1	18:80:2	15:80:5
Fitness Sample Size	500 points	1000 points	2400 points
Fitness Criteria	Maximize Main Lobe to Average Side Lobe Ratio		
Run Time (minutes)	400	1124	1126
Computing resources	Pentium III 500-MHz Desktop Computer		

Table 1. Athin Code Figures of Merit (FoMs)

III. EXPERIMENTAL EVALUATION OF THE GA

A. INVESTIGATION OF GENETIC ALGORITHM EFFECTIVENESS

In order to answer the broad question of the whether Genetic Algorithms are suitable as design tools for advanced shipboard radars, a careful, methodical approach is necessary. In fact, although methods were developed to model antennas in the 1950s and 60s, the computational resources have only become available to do so relatively recently. Even the smallest of these systems can be many millions of dollars each to construct, therefore any applied design program must be shown to reliably demonstrate the complex way array antennas transmit and receive radiated energy. Without prior work to expand upon, it was necessary to carefully document the basic functions of the Athin GA code and determine its suitability based on measurements of an actual physical antenna identical to that modeled in the program. Initially, this measurement program was designed to have two steps.

B. MEASUREMENT OBJECTIVES

The first step was designed to verify the process used to build the antenna pattern, an integral part of fitness assessment and therefore the validity of the final solution, both against experimental measurement and against rigorous numerical methods such as the Method of Moments (MoM). The second step was intended to demonstrate the effectiveness of the GA to perform beam steering. The first step was entirely successful, the second was not, although not as a result of any flaw in the GA. A third unanticipated measurement was added mid-way through the test regime that successfully explained minor anomalies in data from the first step. These steps will be covered in detail, in the order they were performed.

C. ARRAY DESIGN AND CONSTRUCTION

The driving factors in selecting a test frequency were compatibility with the anechoic chamber at the Naval Postgraduate School and the availability of an existing radiating element design. Twenty four dipole elements were custom fabricated on printed circuits at a nominal operating frequency of 7.5 GHz. Testing on a Hewlett-Packard 8510C Network Analyzer showed that all the elements were relatively uniform, but had the best return loss (input match) near 7.6 GHz (λ of 3.95cm or 1.56inches), so that was adopted as our operating frequency for all testing. A square aluminum plate, fifteen inches (38.1 cm) on a side, was manufactured as a ground plane providing the 10λ by 10λ ground plane the Athin code was written for. The random locations generated for the elements are given in Appendix C. The origin for these coordinates was taken as the geographic center of the array ground plane. Figure 10 shows this arrangement graphically. The printed circuit elements were then soldered to sub-millimeter array (SMA) coaxial cable female connectors for integration with the rest of the array.

Received signals were combined using a parallel feed beamforming network as shown in Figure 11. Each individual element were routed through a precisely measured (accurate within 1mm in length or 2.5% of λ) copper sheath rigid coaxial cable to one of three Anaren Model 04-0287 eight-way power dividers where it was joined by the signal from seven other elements. The eight-way power dividers then fed into a single Anaren Model 04-3040 three-way power divider that merged them into a single output.

Since the original array plan did not include phase shifters, all phase control was accomplished by attempting to precisely control path length. In order to preserve as much uniformity in phase as possible, the rigid coaxial cable was all hand cut to precisely equal lengths. The ability to make the array free-standing, as well as support for the power dividers, was provided by a frame constructed out of thick aluminum borders from surplus chalk boards. Since the locations of the power dividers determined the length of cabling that would be required to reach the elements, a wooden frame was first constructed before the aluminum one was cut and the farthest distance from an element to an eight-way power divider, and from an eight-way power divider to a three-way divider was determined using spare rigid cabling. Including a small extra length required for the minimum bending radius for this type of cable, the required lengths were determined to be nine inches and six inches respectively. The most time consuming aspect of the antenna's construction was fabrication and custom fitting of the cabling. Twenty four nine-inch and three six-inch lengths were precisely hand cut from new copper-sheathed sections of cable using a jeweler's saw and Vernier scale calipers to ensure quality control. After cutting, each section had SMA-male connectors soldered to the ends. The cables then had to be bent into place, using a special purpose bending tool to restrict the curvature to acceptable limits. Since the element locations were random, the cable length

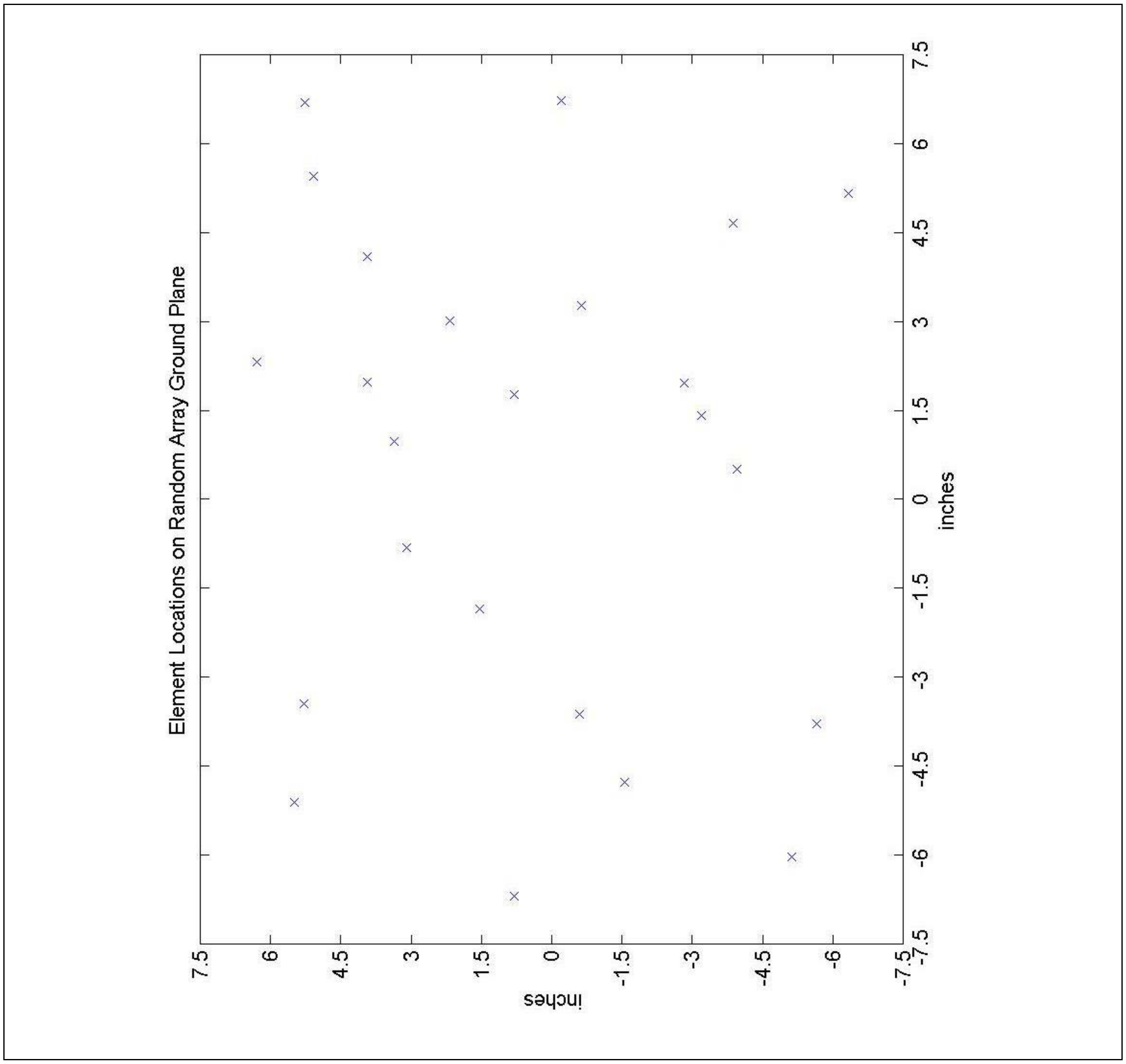


Figure 10. Random Array Element X-Y Plot

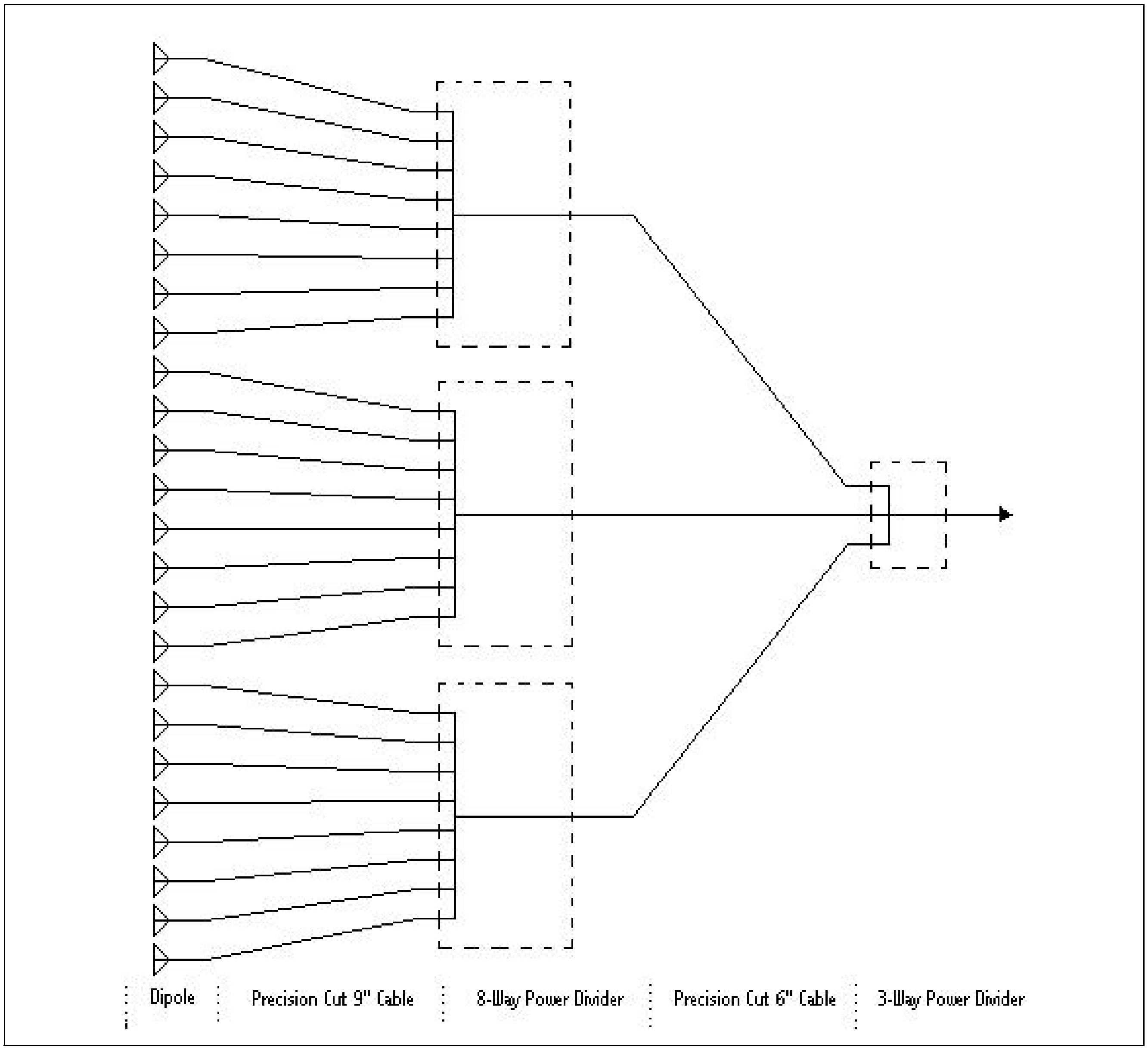


Figure 11. Random Array Schematic Diagram

was determined using the maximum distance that would be required. All the cables needed to be exactly the same length to form a broadside beam and minimize phase error. Some cables required significantly more bending than others. The phase errors this introduced were unavoidable, but of significantly smaller effect than if the cable length had been adjusted to fit the length to each element. The transverse electro-magnetic (TEM) waves that propagate through the rigid coaxial cable experience no significant phase shift as long as the minimum bending radius is observed, as the waves then remain perpendicular to the axis of the cable. Conversely, variance in length (*ΔL*) that is not exactly equal to the wavelength, λ, immediately translates into phase error:

$$\Phi_j^{error} = \frac{2\pi(\Delta L)}{\lambda}. \tag{3}$$

The most complicated aspect of designing the antenna was developing a system for securing the antenna's printed circuit elements in place. Not only did it have to hold the elements in place firmly during transport of the antenna, but it had to keep the T-shaped dipole as close to a quarter wavelength above the ground plane as possible for the image charge on the plane to provide the best element factor. The securing system had to ensure that the element position could be adjusted and then held in place to guarantee the dipole was physically in the center of the hole in the ground plane, and thereby at the required X-Y coordinates. The system that eventually was adopted utilized a pair of L-shaped brackets, with one side containing a groove for the printed circuit and the other a having drilled hole for attachment to the ground plane with Teflon screws and nuts. Plastic was preferred for fabricating the brackets, but this material and the facilities to work it were not available, so aluminum was used. In order to ensure that the elements were electrically isolated from the ground plane, a layer of sheet rubber was sandwiched between the ground plane and each bracket. The elements also received two or more layers of electrical tape along the sides where they would contact the brackets, providing the added benefit of increasing the friction and tightening the fit between brackets and element. Figure 12 shows the arrangement.

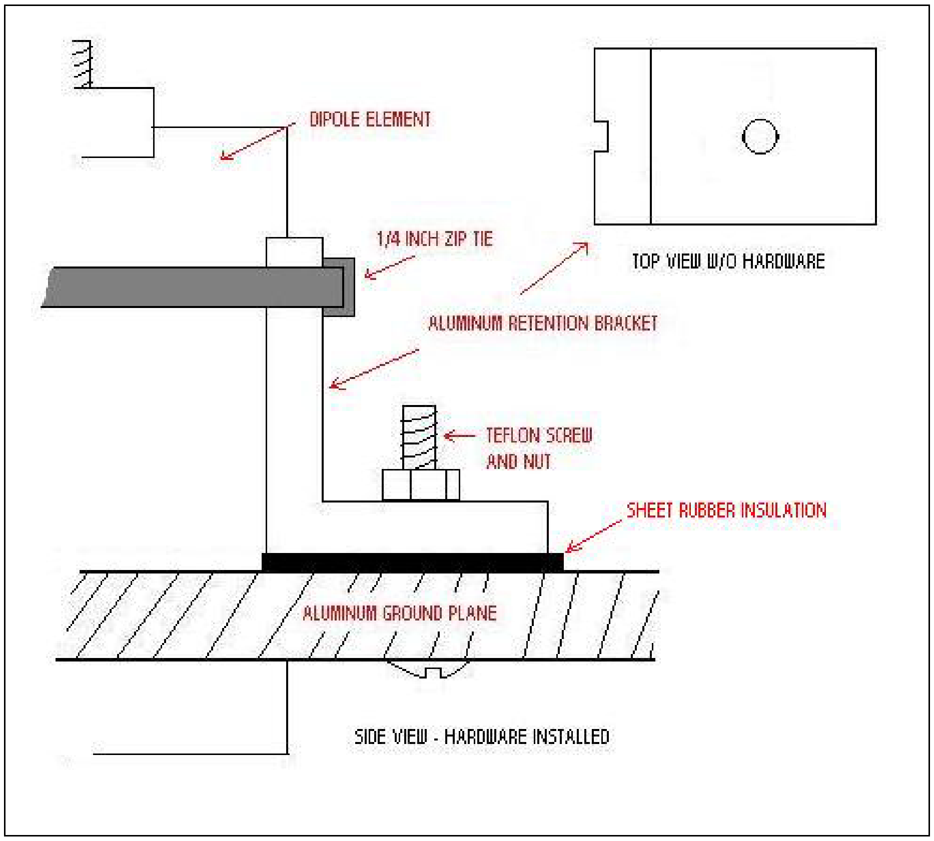


Figure 12. Element Retention Method

In order to secure the elements in place better, a ¼-inch zip tie was added to the top of the bracket. A specially machined aluminum gauge was fabricated to ensure the elements could be set just prior to measurement to within .0025cm (or 0.06% of λ) of the correct X-Y position on the ground plane and within 0.05cm, or about 5%, of the required λ/4 height above the plane. Tightening the bracket screws and zip tie then locked the element firmly in place. Figure 13 shows the final product, including phase shifters and a graduated elevation mounting which were added at a later date and will be discussed later. Comparison with the X-Y plot and schematic drawing, Figures 10 and 11 respectively, are particularly useful in conjunction with these pictures. The point (-7.5, -7.5) in Figure 10 is the lower left hand corner of the array in the front view in Figure 13. Note the configuration of the power dividers, in blue, in the back view.

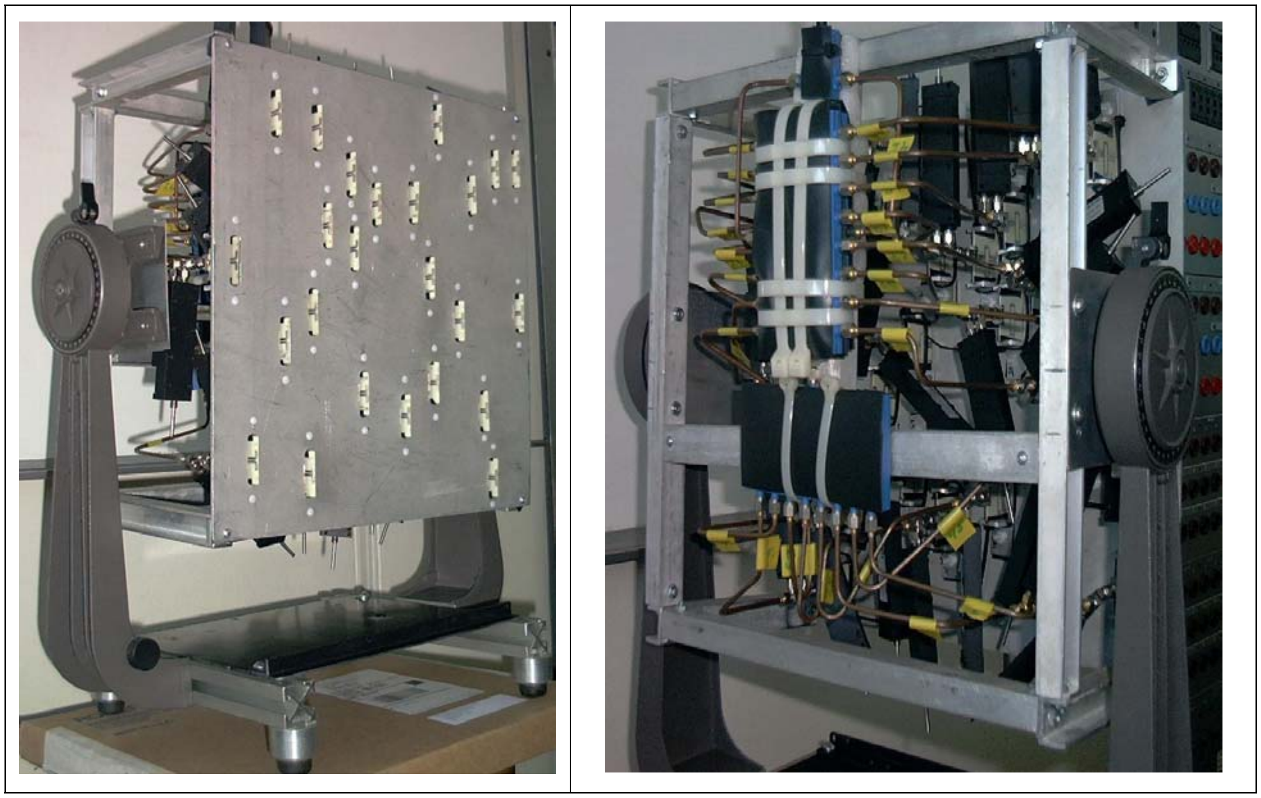


Figure 13. Front and Back View of Random Array

1. Modifications to the Array

In order to accomplish the third experimental objective, some modifications had to be made to the previously described configuration, hereafter referred to as Mod 0. In order to conduct evolved beam steering measurements, the array required the ability to control phase shift throughout the full range of the waveform, from 0º to 360º, and to tilt the array in elevation in order to construct a full three dimensional measured antenna pattern. This new configuration, Mod 1, has mechanically adjustable phase shifters at each element, and is the one shown in Figure 13 above. Controlling phase shift was accomplished by inserting Sage Labs Model 6708 DC-8GHz phase shifters into the transmission line just behind the dipole element. The actual 4-inch long phase shifters can be seen as the black rectangular boxes situated just behind each element in the back view of Figure 13. The aluminum frame supporting the transmission lines had to be extended in order to accommodate the increased line length of the new configuration. Figure 14 shows a schematic comparison of the Mod 0 and Mod 1 transmission line configurations.

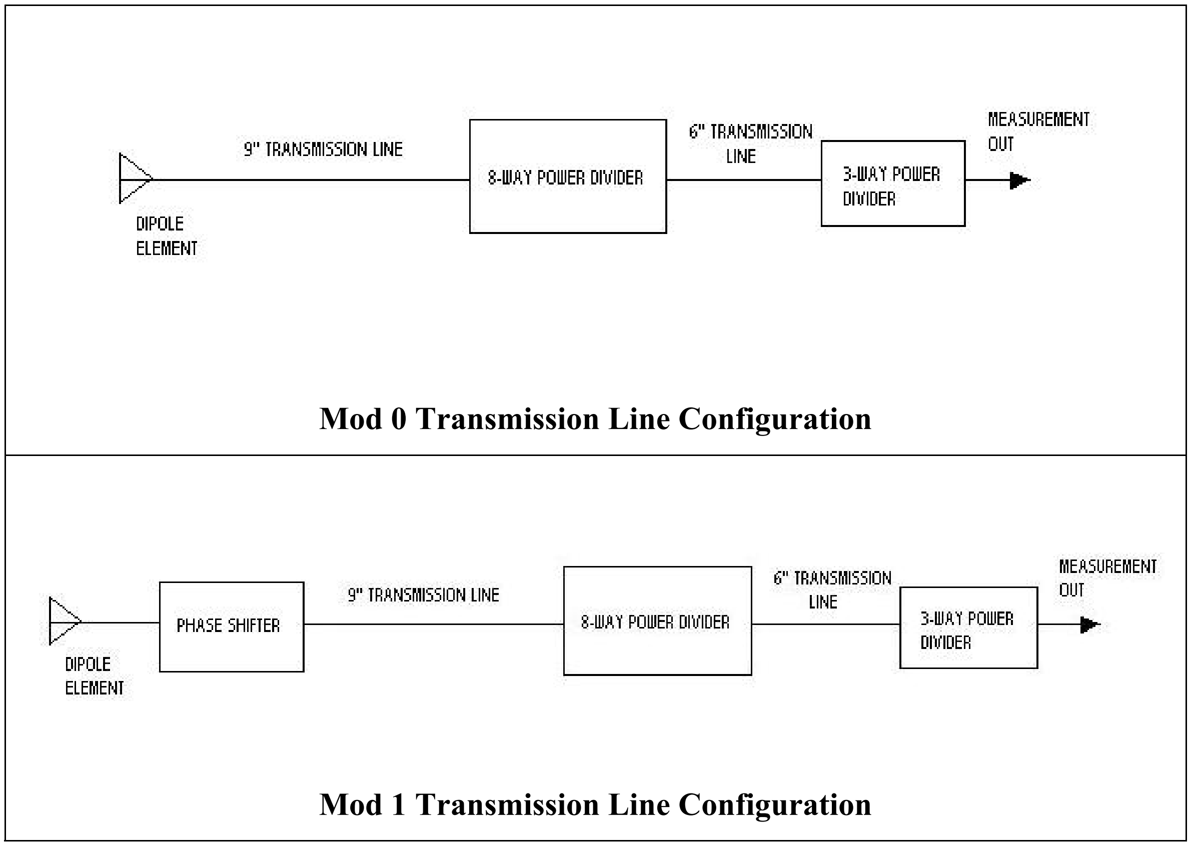


Figure 14. Schematic Comparison of Mod 0 and Mod 1 Transmission Line Configurations

The antenna had to be tilted backward for measurement so that the scanned beam cut through the horizontal (0 elevation) measurement plane, since the pedestal is only capable of rotating about a single axis. The elevation adjustment was accomplished by rigidly attaching the aluminum frame to a surplus telescope armature. This allowed for both a more stable platform and the control of antenna elevation accurate to within one degree.

D. INSTRUMENTATION IN THE LAB AND ANECHOIC CHAMBER

Transmission line measurements made in the lab during construction of the Mod 0 and Mod 1 arrays and phase setting with the Mod 1 variant were accomplished by using an HP 8510C Network Analyzer connected to a two-port HP 8517A S-Parameter Test Set. Signals were generated using an HP 83651A Synthesizer-Sweeper.

The instrumentation in the anechoic chamber was nearly identical. The signal generator was used to feed a Narda XC460 5.3-8.2 GHz feed horn at one end of the chamber that could be rotated 90º by hand in order to measure patterns for cross-polarizations on the device under test. The test antenna was secured to a rotating pedestal that sat directly in front of the transmitting feed horn at a distance of 19 feet. The pedestal sweeps from -90 degrees to +90 degrees using a stepper motor. The chamber is lined with microwave absorbent foam and has a narrow walkway that is also constructed of absorbent material, though probably not as effective as that used in the walls, floor and ceiling. The rectangular transmission feed horn can be rotated 90 degrees by hand and checked with a level in order to provide measurements in both cross polarizations. At 0 degrees tilt the dipoles are vertical, perpendicular to the floor. Horizontal polarization is measured with the long axis of the feed horn perpendicular to the floor. Vertical polarization is measured with the feed horn parallel to the floor. The received signal from the device was sent to the chamber's network analyzer which in turn passed it on to a desktop computer equipped with a Labview program with a graphic user interface for control of the chamber pedestal and compilation of the antenna pattern. Measured patterns could be exported to text or spreadsheet files, with a maximum of resolution of 1º. Figure 15 shows the Mod 1 antenna in place on the pedestal during a test run. The blue material around the antenna and throughout the chamber is the microwave absorbent foam designed to help minimize multipath interference.

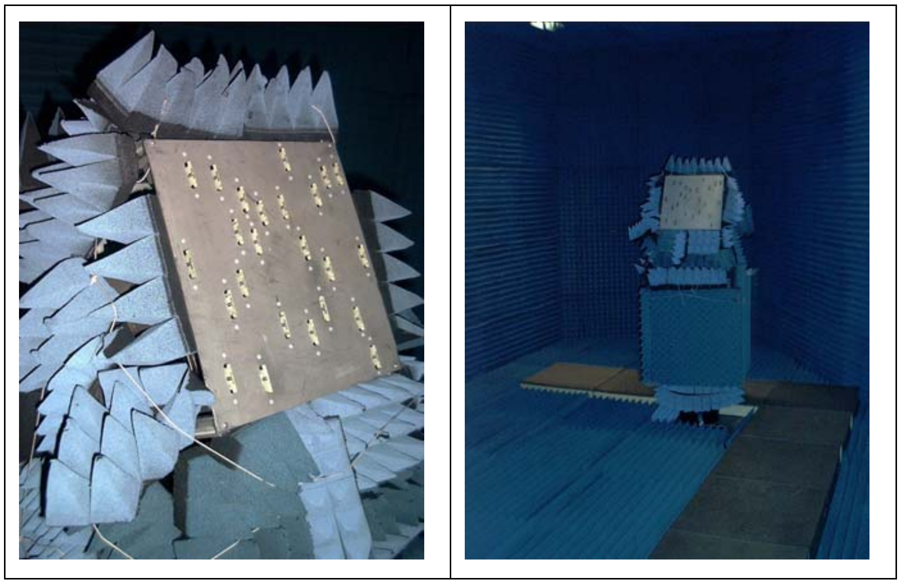


Figure 15. Mod 1 Array in Anechoic Chamber

It should be noted that the chamber was designed for classroom teaching rather than precise scientific measurements and is almost exclusively used to measure the relatively simple patterns with broad beams and high sidelobes.

E. STEP 1: PATTERN BUILDER EXPERIMENTAL VERIFICATION

Perhaps the most important aspect of using Genetic Algorithms for optimizing or designing complex systems is the formulation of fitness criteria that accurately reflect reality. In the unique problem of using GAs for designing phased array radar antenna systems this is an essential issue. This first experiment conducted in evaluating the suitability of GAs as a design code was to qualitatively measure the code's ability to accurately compare population members to real world RF propagation. In order to rank the fitness of individual population members, the Athin Code constructed by Dr. Johnson builds an antenna pattern for determining the ratio of main lobe power to average side lobe power. The focus of this experiment is to measure this GA Pattern Builder's results for a known array configuration with that of both the widely accepted means for building phased array patterns, Method of Moments (MoM), and with a measured pattern of a passive test array built to the same parameters. Method of Moments is a technique for solving the electric field integral equation (EFIE). The antenna surfaces are separated into small subdomains and the current on all subdomains is solved for simultaneously using matrix techniques. The subdomains are assumed short if their length is small compared to the incident wavelength. This ensures that the current is uniform throughout the subdomain. The flat plates of the antenna are also assumed to be dividable into patches that are small compared to the wavelength. The computed currents on the surface can then be added to determine the electric field at any point in space using the principle of superposition.[12]

The parameters for the array were constrained by budget, available test facilities and time, and are therefore relatively unsophisticated. In fact, almost universally the constraints were based on hardware limits rather than any inherent requirements of the GA. However, in order to more fully show the flexibility of the GA as a naval architect's tool, the locations of the array's elements were selected by pseudo-random generation,

[12] SETH Corporation (2002); "EMI/EMC Modeling Techniques – Method of Moments"; Retrieved April 9, 2002 from the World Wide Web; http://www.sethcorp.com/seth4.html.

similar to the problem of a ship being primarily designed for sea keeping and low radar cross section, and having the array added later. The Athin code was therefore artificially constrained to these X-Y coordinates, but is in fact capable of handling much more complex three dimensional arrays.

1. Phase Error Measurements

The last step in construction, and the first in taking measurements with the Mod 0 Random Array, was to determine the inherent phase difference induced by the small differences in length of the cables. Since the design goal was to have the relative phases of all elements equal to zero, this measured experimental phase difference is considered a phase error that must be accounted for in later measurements. The ground plane and elements were removed and the device was connected to the HP 8510C Network Analyzer. One port of the analyzer was connected to cable from the element while the other was connected to the output so that the signal traveled the full length of the system transmission line with the exception of the dipole element itself. Table 3 provides the measured phase errors for each element. As can be seen, the error in each of the two measured cable lengths in a single transmission line during the construction phase was rarely more than the nominal 2mm cumulative change in path length, which correlates to approximately 18º of phase error. This value is exceeded in two of the lines, perhaps due to permanent heat expansion of the cable’s dielectric material during solder attachment of the cables’ end connectors or other human error in the fabrication process. These measured phase errors were inserted into both the Pattern Builder and Method of Moments for later comparison with measured antenna patterns.

Element Index j	Phase Error Φ_j^{error} [degrees]	Phase Error Φ_j^{error} [cycles]
1	-1.49	0.9959
2	-14.76	0.9590
3	-9.52	0.9736
4	-10.00	0.9722
5	-2.15	0.9940
6	-15.70	0.9564
7	-1.73	0.9952
8	-5.59	0.9845
9	1.94	0.0054
10	0.71	0.0020
11	-15.28	0.9576
12	-12.54	0.9652
13	-0.22	0.9994
14	16.64	0.0462
15	-5.29	0.9853
16	-16.73	0.9535
17	22.64	0.0629
18	-8.10	0.9775
19	0.85	0.0024
20	-2.50	0.9931
21	0.69	0.0019
22	8.17	0.0227
23	-10.00	0.9722
24	-23.75	0.9340

Table 2. Measured Phase Error in Transmission Line

Once the phase errors inherent in the transmission line were recorded, the array face was carefully reconnected element by element to the cabling and the complete antenna was transported to the US Naval Postgraduate School (NPS) anechoic chamber for pattern measurement. The response of the device under test is supplied to a PC-based graphic interface that provides a low resolution view of the antenna pattern as it is being measured as well as the option of exporting the full resolution data to a file. Three pattern measurements were taken for the Random Array, all for the full 180 degrees of sweep available, from endfire through broadside to endfire. All measurements had one degree resolution. The three measured patterns were so similar, within 0.01 dB, that only one is presented.

2. Pattern Measurement and Analysis

Prior to measuring the pattern, the measured phase errors (Φ_j^{error})in the transmission line were included in the Athin code's Pattern Builder using equation (2) for the Array Factor. Multiplying by the cosine theta element factor, a pattern was computed for the element locations and measured phases of the random array. Note that this did not require any GA runs. A pattern was also computed using Method of Moments software for comparison. Figures 16 and 17 show the results of the measured pattern from the NPS anechoic chamber for the E- and H- planes respectively plotted alongside the results from MoM. In both, the dashed green curve is a computed pattern for an ideal array, all phases set to zero, with elements at the Random Array locations. The blue curve is also a MoM computation, with the actual measured phase errors included. The red curve is measured data from the chamber. Figures 18 and 19 show the same measured E- and H- plane patterns as a green curve alongside the GA Pattern Builder for the Athin code in dashed blue. In all graphs, the value of the measured data point is marked with a red or green cross matching the fitted curve. With the exception of two regions, from -30 degrees to -60 degrees on the E-plane plots and the vicinity of +30 degrees on the H-plane plots, the measured data appears to agree very well with both the GA Pattern Builder and Method of Moments. The agreement is also apparent between the GA and MoM. This is encouraging because it gives the programmer confidence that the results of GA optimization will reliably reflect the real world concerns of the antenna engineer.

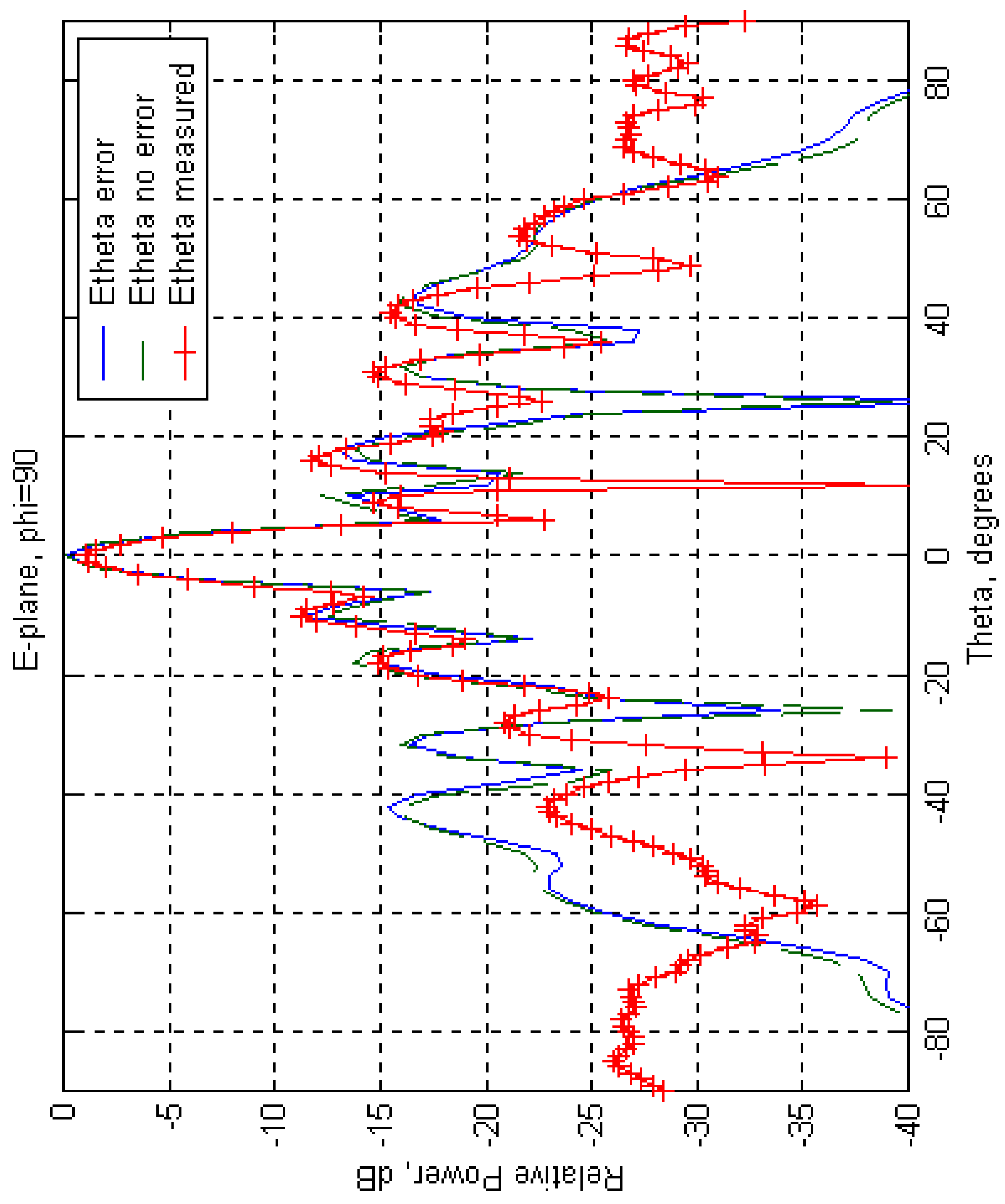


Figure 16. Method of Moments vs. Measured Data (E-Plane)

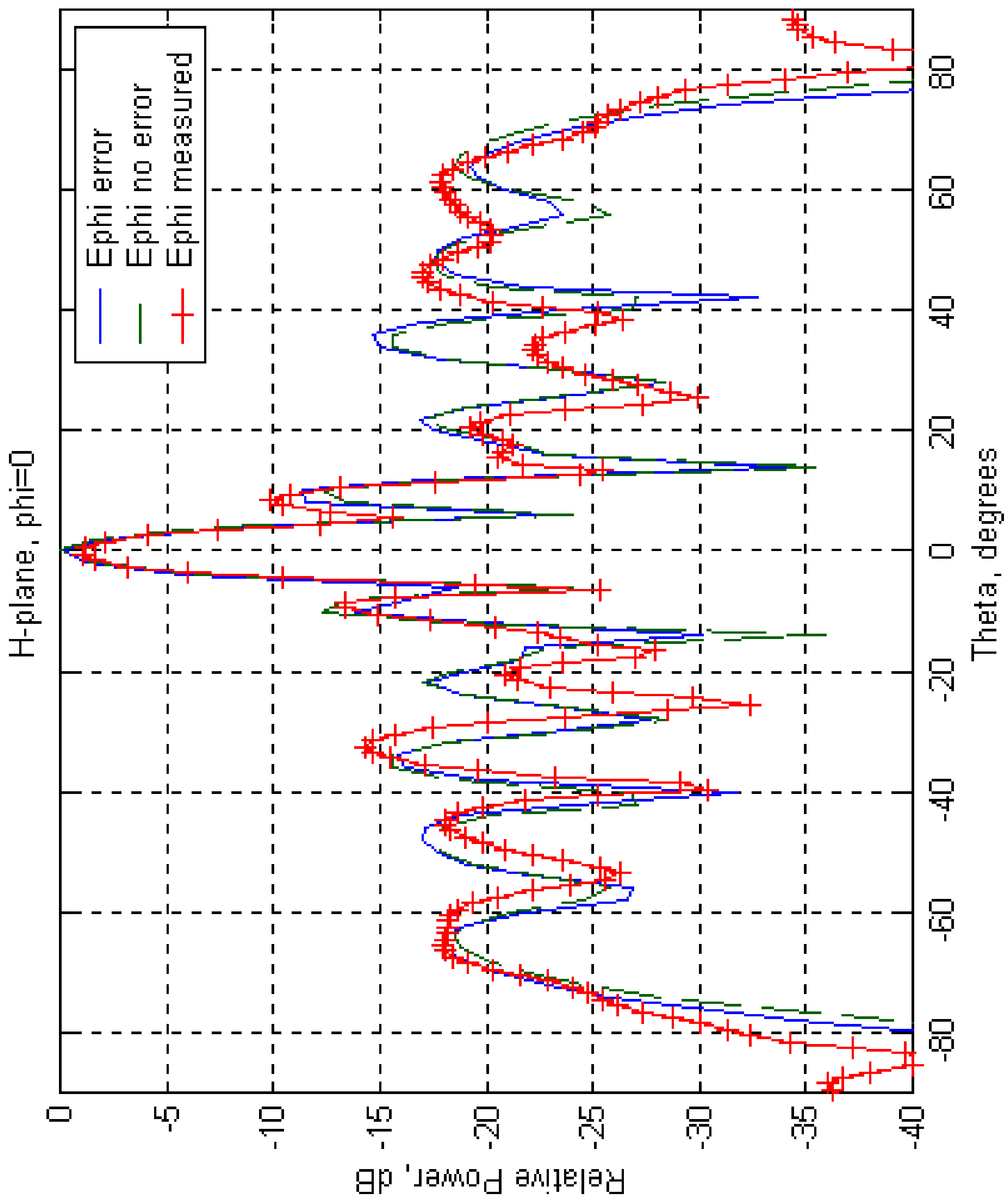


Figure 17. Method of Moments vs. Measured Data (H-Plane)

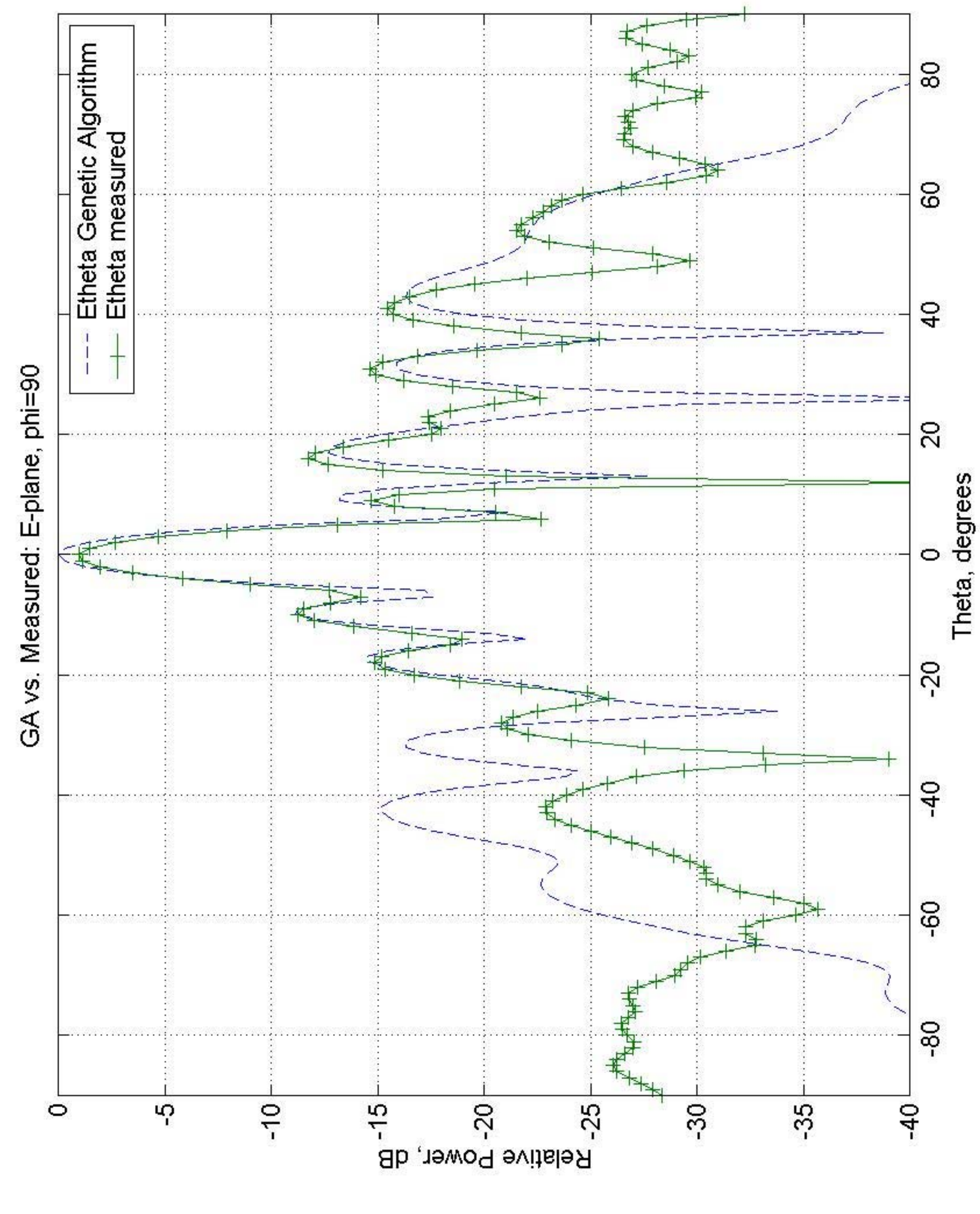


Figure 18. GA Pattern Builder vs. Measured Data (E-Plane)

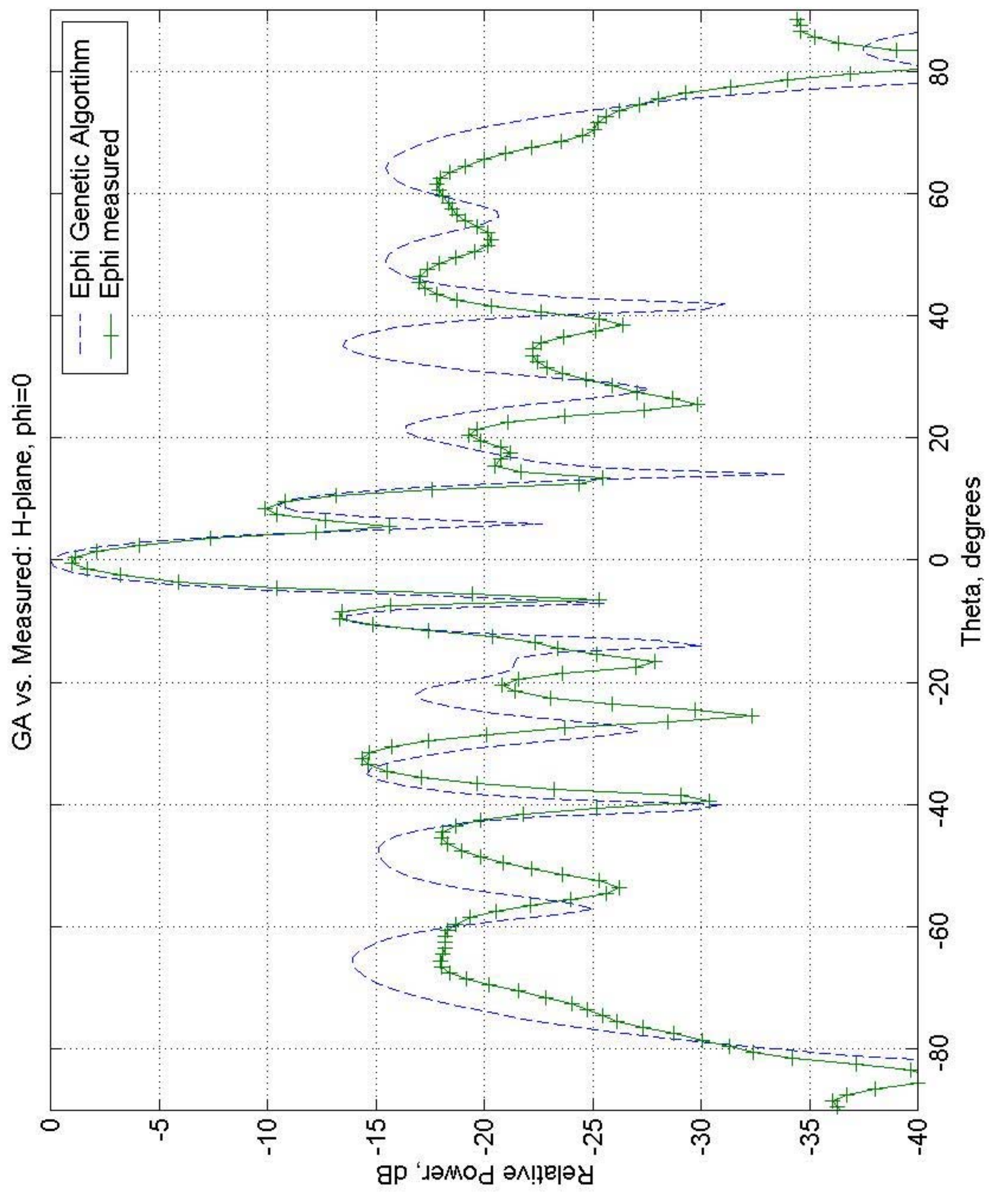


Figure 19. GA Pattern Builder vs. Measured Data (H-Plane)

3. Assessment of Measured Pattern Results

There are several regions where agreement is not as good as it could be, however, and these must be addressed in detail. From the previous graphs, it is obvious there is a slight misalignment in the direction of the main lobe between the measured and computed patterns. This is the result of human inaccuracy in aligning the antenna for measurement in the chamber, as well as possibly the result of errors in the calibration of the stepper motor in the test pedestal. Antennas that are installed for measurement are simply placed upon the pedestal and secured with string and zip ties. Furthermore, the pedestal stows at its +90 degrees position when not in use so it can be accessed from the walkway next to it. This means that, without a benchmark, angular and positional alignment of the antenna with the pedestal centerline must be "eyeballed" for correctness. A carpenter's square and laser pointer provided only limited assistance in proper positioning. With the tools available, this error is unlikely to be entirely correctable, but doesn't effect agreement between MoM and the GA, and doesn't significantly degrade confidence in the quality of the measured pattern. Figures 20 and 21 show the differences between measured curves and MoM, measured and GA and GA and MoM for E- and H-polarizations respectively. The high difference at wide angles can be explained as a known problem of the anechoic chamber used. The measurement accuracy is known to become highly suspect at small values below -25 dB down.[13] The narrow spikes in both graphs are explained by the angular positioning problem just described. Subtracting the values of two slightly misaligned lobes creates a narrow spike which crosses the 0 dB axis then peaks sharply again on the opposite side. Of the great interest is the difference between the GA and MoM. In the E-Plane there is complete agreement (effectively 0 dB difference). However, with the electric field now in the same direction as the dipole, the cross-polarization shows a quadratic shape. The MoM calculations are exact, while the GA only approximates the element pattern and assumes it is identical for all dipoles and in both polarizations, using a cosine theta factor for all. In fact, the best fit to the measured pattern shows that the cosine theta element factor fits the E-plane cut nicely, while a square root cosine theta [$(\cos \theta_j)^{1/2}$] would have fit the measured data for the H-

[13] Jenn, David; Written feedback on the results of first chamber measurement, 17 November, 2001.

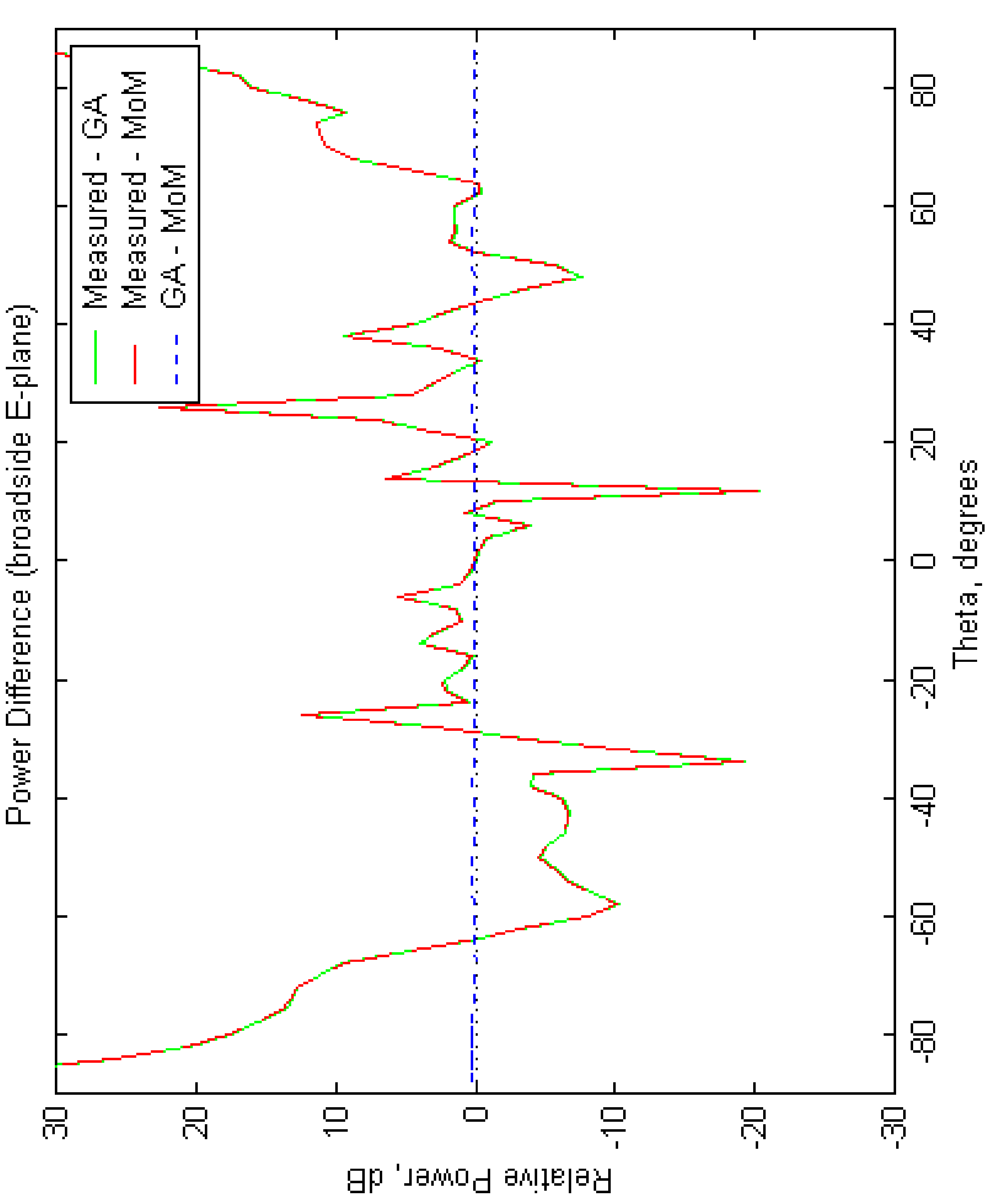


Figure 20. Power Difference for E-Plane

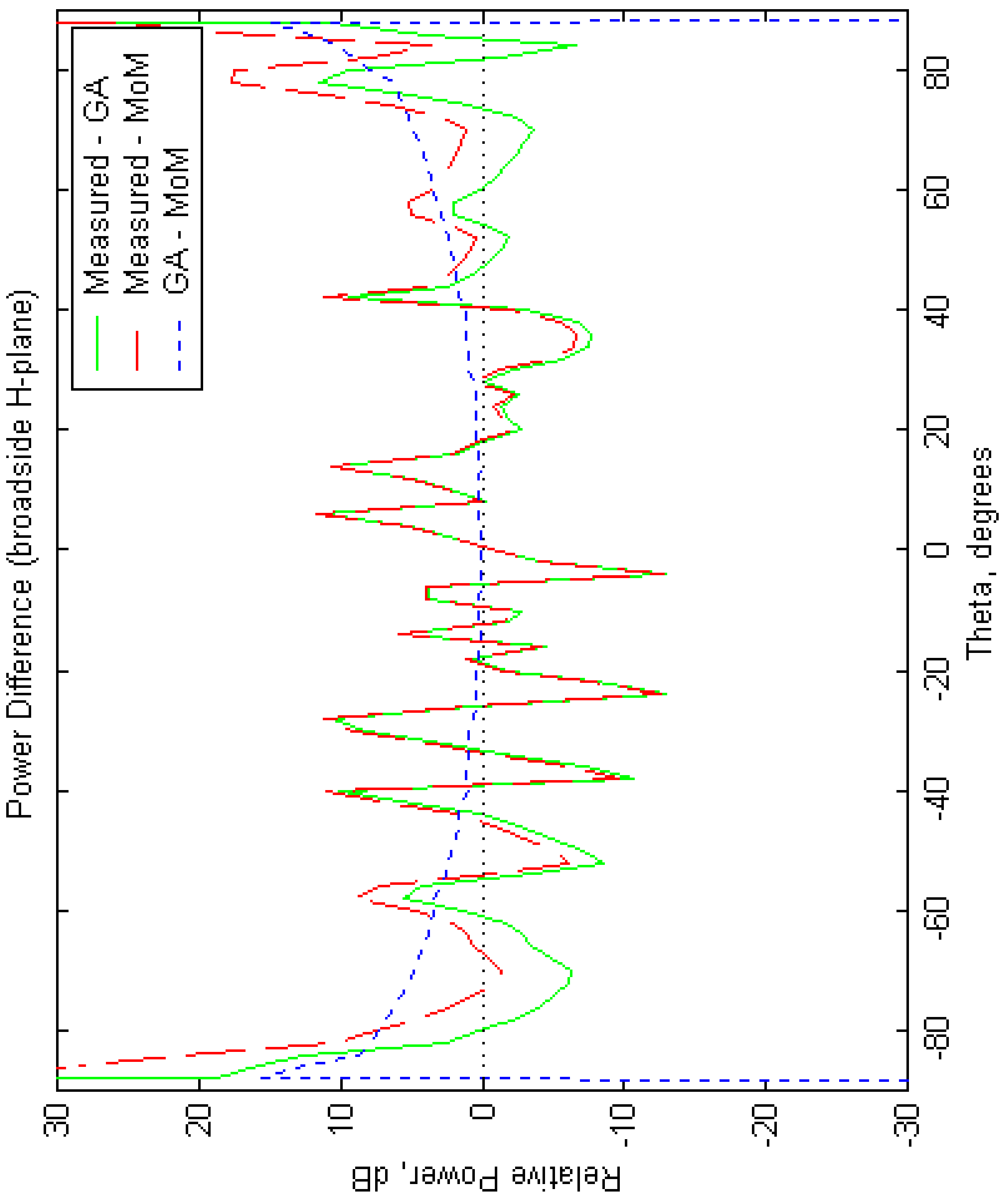


Figure 21. Power Difference for H-Plane

plane much better. As a result of this measurement, the new element factor was included in the Athin code for H-plane cuts.

The two previously mentioned regions of attenuation are more troubling. At +30 degrees in the H-plane and around -30 to -60 in the E-plane there is significant loss, but the shape of the side lobe is preserved in the pattern measurement. Previous experiences with the NPS anechoic chamber have shown that at extremely low power, around -25 dB below the main lobe, the equipment becomes less accurate as the signal becomes more embedded in noise. Another theory was that these attenuation regions were a result of multipath returns from areas within the chamber that were worn or poorly designed. It was the quest to examine this effect more closely that created an intermediate step that had not been anticipated.

F. STEP 2: EXPERIMENTAL EXAMINATION OF CHAMBER EFFECTS

In order to examine the possibility of multipath interference causing the two anomalous regions in the measured patterns, a thorough examination of the chamber was made. This was deemed important because of the high probability that such effects would also effect future work. Sources of multipath interference were indeed discovered, although not quantified definitively, that seemed to explain the attenuation. Figure 22 shows the basic layout of the chamber, which may make explanation of the multipath scattering centers more clear. Areas of suspected multipath scattering are circled in red.

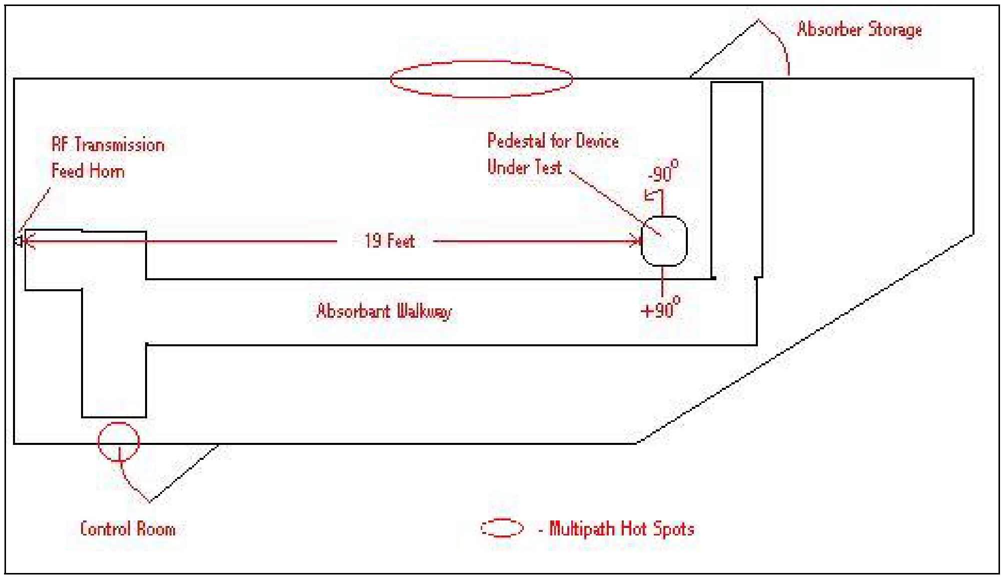
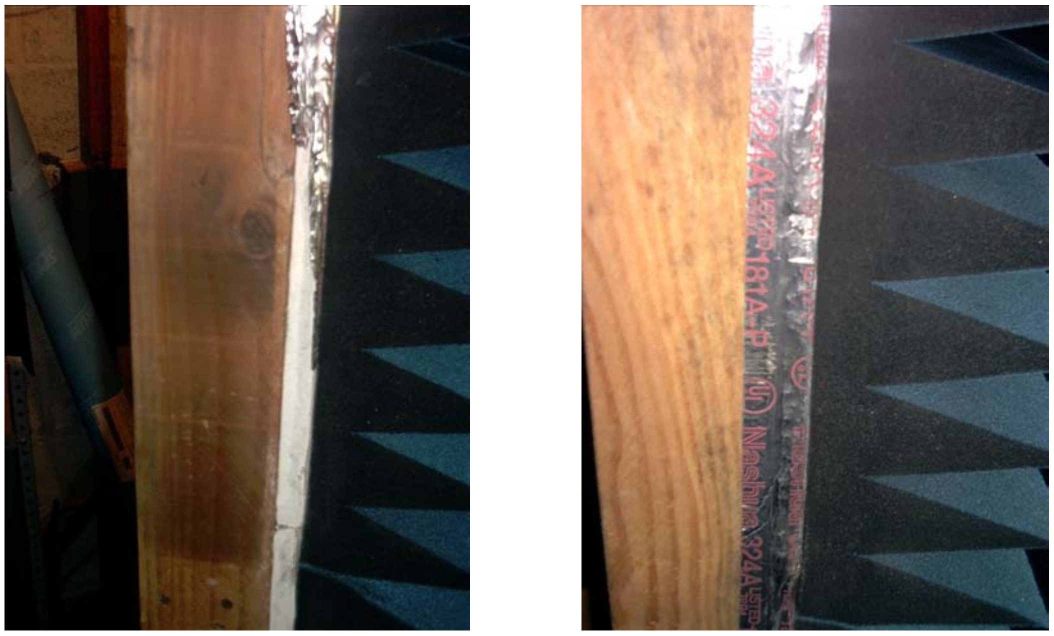


Figure 22. Plan Diagram for Layout of NPS Anechoic Chamber

1. The H-Plane Attenuation

The probable source of the +30 degree region of attenuation in the H-plane pattern, which drops 7 dB below expected, was relatively easy to find. Using a laser pointer and protractor narrowed to the region in question. The laser's beam was immediately noticed to fall on the seam of the door to the chamber's control room at an azimuth of +30 degrees. Placing a flat hand mirror on the walkway at this angle also caused the laser beam to reflect directly at the door seam. Since this door is heavily traveled, it was inspected for damage. There is a conductive coating that lines the walls of the chamber beneath the absorbent material in order to ground any stray induced currents rather than reflecting the RF. This metallic covering is exposed in the seams of doors where no absorbent material can be placed. Not surprisingly, it was severely worn around the control room door. Suspicions that this is the cause of the attenuation region in the H-plane patterns would be very well founded. Figure 23 is a photograph of the damaged material around the control room door. A photo of the seam to the much less used back door into the storage space is included for comparison.



Control room door showing damage to conductive coating

Storage door showing intact conductive coating

Figure 23. Damage to Chamber Control Room Door

2. The E-Plane Attenuation

The E-plane attenuation was of concern because of the broad range of angles it covered and because a visual survey of the chamber yielded no definitive flaws or clues as to the cause. The only physical feature that correlated with the angular spread was the junction of the ceiling and the wall. Such a basic design flaw would have been noticed by the students who use the chamber for class experiments. At this point a novel idea presented itself: the destructive interference might be dependent on the reflection of the array face itself, as well as the wall-ceiling junction. The thin population of the array would exacerbate this condition, leaving more of the ground plane open for reflection. Figure 24 shows the results of a novel approach to determining the cause.

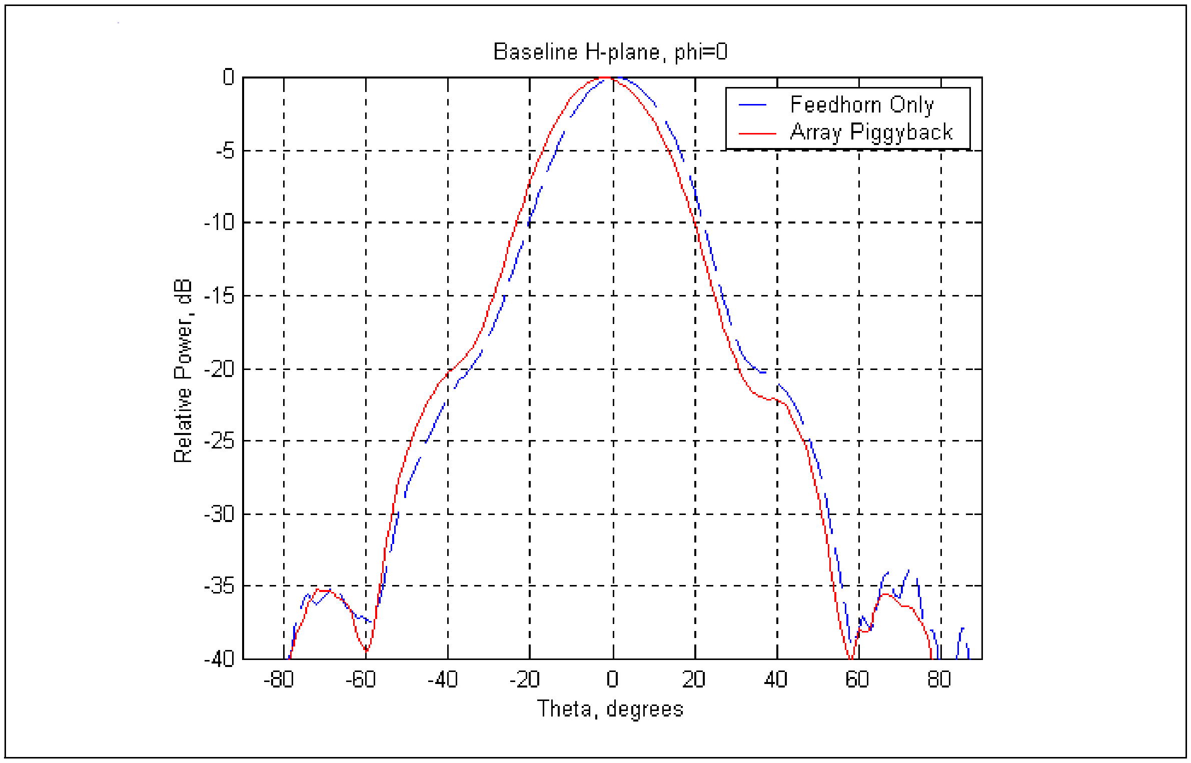


Figure 24. Attenuation in Baseline Feed horn Measurement

The standard Narda feed horn was mounted on the receive side and three E-plane patterns were measured. All were virtually identical and showed no visible distortion. The main lobe was basically symmetrical. A fourth measurement was made with the Random Array sitting "piggy back" on top of the receiving feed horn. The array was not connected to the measurement system and was not in physical contact with the receiving feed horn. This measurement yielded a telltale distortion and power loss of the main lobe shoulder at between 35º and 55º, in the same general region as the attenuation found in previous measurements with the array. The region in question from Figure 24 is magnified in Figure 25 below.

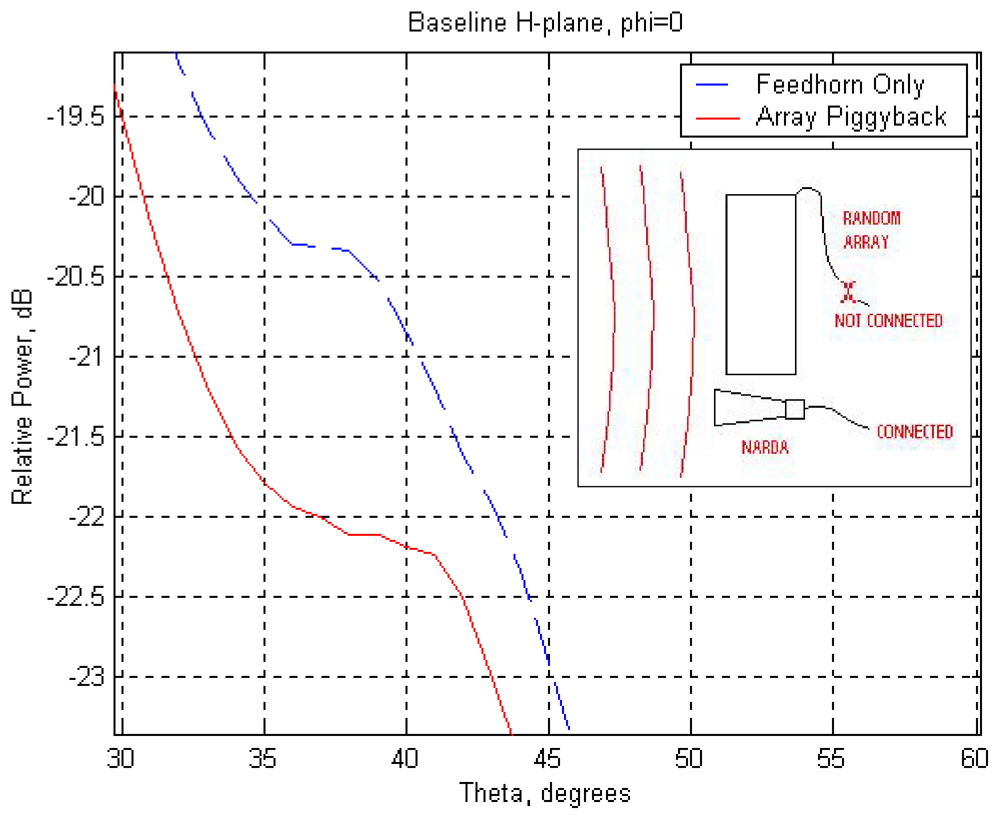


Figure 25. Expanded View of Attenuation Region

Although not as dramatic as the loss seen in the Random Array measurements, this can be explained by the differences between horn and array antenna patterns. In effect, the attenuation of the feed horn occurs in the main lobe, due to the significantly broader pattern of the horn. The interference may also be stronger because reflection off the array face is occurring right next to the measuring elements, vice several wavelengths away in the case of the piggyback configuration, in effect exacerbated by image charges on the face of the ground plane in the same way the dipole elements themselves use an image dipole to amplify gain. However computing this notch would require complete mapping of the field everywhere in the chamber in order to determine exactly where all the reflections originate and be able to calculate the net image charge for a given location on the ground plane as a result.

G. STEP 3: BEAM-STEERING WITH THE GA

The final measurements that remained within the scope of this effort to verify the use of Genetic Algorithms as design tools was to test the code's ability to optimize the array for steering angles off of broadside. Several changes had to be made to the array in order to do this. This necessitated the Mod 1 changes to the array in order to set the solutions determined by the Athin Genetic Algorithm into the antenna and measure a three dimensional beam pattern. The new knowledge about the limitations of the chamber provided guidance on where not to steer the beam in order to avoid attenuation induces by multipath propagation. The representative Figures of Merit given in Chapter II were for the runs resulting in a solution for this beam steering problem. However, although the GA worked smoothly, the hardware side proved less cooperative and no reliable measured data was available for comparison.

1. Alterations to the Array with Disappointing Results

Due to both a limited budget and finite time to graduation that prohibited learning complicated digital control software, mechanical analog phase shifters were selected for installation. Initial testing of the phase shifters alone on the network analyzer showed what would eventually serve as the show stopper for collecting measured data during this step. The phase response was non-linear. When the adjustment screw was turned, the phase moved linearly along a saw toothed wave as expected, but would reach a point where the saw tooth wave would begin to distort. Continuing to turn the screw caused the measured phase to peak and finally return back in the original direction. This rendered whole ranges of phases unreachable. Figure 25 shows the expected and actual waveforms as seen on the network analyzer. To make matters worse, the point where the saw-toothed wave distortion was centered was slightly different for each individual phase shifter by itself, but once connected to the transmission line it appeared to become highly dependent on the incoming phase of the line it was connected to. Consequently, three of the required phase settings fell directly on top of the non-linearity, and swapping out the phase shifter with one from another element's line had little or no effect on the phase response. This was not expected as the manufacturer's data sheet described a linear relationship that should have given 547.2° of phase shift at the operating frequency of 7.6 GHz.

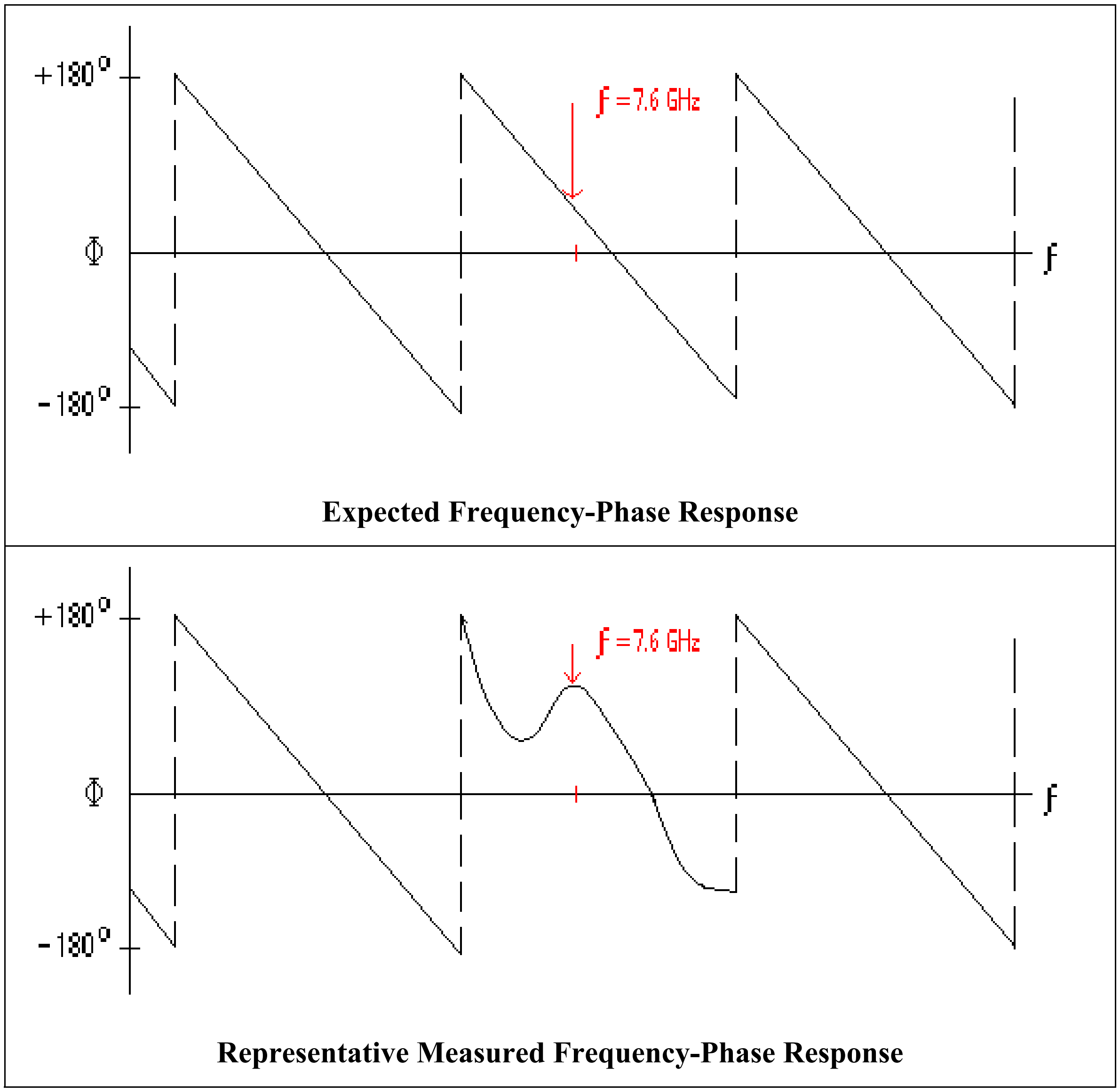


Figure 26. Frequency-Phase Response Characteristics of Sage Phase Shifters

Also, once installed the shifters were extremely sensitive to vibration. Despite mechanical attempts to minimize phase error, such as using Loc-Tite on connectors and the adjustment screw, the slightest perturbation caused phase swings in excess of 30 degrees and these variations rarely settled out to their preset value. This effect had not been observed with the previous Mod 0 configuration. Phase settings that were in the vicinity of the non-linear point obviously were effected the most.

The technique necessary for setting the phases was highly intrusive. Like the initial phase error measurements, the array face was removed, along with the elements, and the phase shift through all the components of the transmission line except the actual dipoles was measured using the network analyzer. The phases were set using the adjustment screw and then frozen in place with a drop of glue. Once this process had been completed for all twenty four elements, the ground plane and elements had to be reconnected. Perhaps the largest source of error came from the reconnection of the elements and ground plane to the coaxial plumbing after setting the phases. Although the work was done methodically and with great care, it still involved putting large hands into the confines of the cabling behind the ground plane. Unlike previous experience with the Mod 0 array, this area was now full of bulky phase shifters and errors may have been introduced at this stage as a result. Consequently, confidence going into the measurement process was not as high as for the first Random Array measurement. A series of ten patterns were taken but no discernable main lobe could be found. The measured antenna pattern was effectively noise. Time did not permit any troubleshooting to determine the exact problem.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS

A. EXPERIMENTAL SUMMATION

Although the final experiment was not completed, due to mechanical problems and time limitations, the earlier results of the experiment lend credence to the idea of using Genetic Algorithms as a tool for advanced phased array design work. The pattern builder was shown to have good agreement with rigorous computational methods (Method of Moments) as well as with the real world measurements of a test array built to the same parameters. There were two anomalous regions that could not be explained as errors in the device or with either the GA or MoM software. The conformity in shape for all three patterns in these regions caused the need to explore the possibility of a cause of attenuation in the measurement chamber itself. Such causes were found in the form of multipath interference. Damage was found in a portion of the chamber directly aligned with the azimuth angle that showed attenuation in one region. The other region could be recreated with a different test device.

The end result of these measurements is the validation of the GA's Pattern Builder function. This ensures that the fitness assessment of the Athin code is firmly grounded in both theory and reality, and consequently solutions to complex antenna design problems that result out of this code can be seen as fundamentally serviceable in reality.

However, like the development of any design tool for complex, expensive systems, validation of a single component is not enough to place the tool unequivocally in service. Further work needs to be done to develop the full potential of GA's to design antennas, both in terms of building confidence in the code's fidelity and in it's superiority over or in conjunction with current methods.

B. RECOMMENDATIONS FOR FUTURE WORK

1. Beam Steering and Shaping

The aborted attempt of this thesis to demonstrate the flexibility of the GA when applied to the relatively mundane task of steering the main beam should be a first step in future work. A more interesting and less trivial natural follow on to steering the beam would be shaping it for a specific task. The GA could be used to generate a null at a desired location for example, in order to demonstrate the ability to counteract jamming or interference from that direction.

2. Dual Frequency Array

An even more exciting concept would be the creation of a dual frequency array, that contains elements of two separate wavelengths on the same ground plane. The placement of the second set of elements could either be random or evolved by the GA from the space left available around the first Random Array. This might simulate the restrictions placed on putting a new array on an existing ship design. The Athin code will definitely have to be expanded to include the effects of mutual coupling between elements, but the effort will provide solutions to a non-trivial problem that is currently avoided altogether by industry due to its complexity. The idea that a ship might have her communications and sensor needs scattered throughout the hull and superstructure in a distributed array is attractive for several of the reasons enumerated in the introduction.

Work is already underway at Naval Postgraduate School pursuing a dual frequency array with receive elements of the COTS semiconductor variety. Successful testing of a GA designed array that can successfully overcome the problems associated with mutual coupling would be significant progress toward a truly multi-function array, possibly including communications as well as radar in the capabilities of future array-based systems.

3. Conformal Array

The ability to be able to make up an array from elements that are dispersed in more than two dimensions is inherent to the distributed ship wide array concept. As well as the advantages of having the sensor's elements widely dispersed about the ship, the ability to group elements in non-planar configurations would pay dividends in reduction of radar cross section and enhanced surveillance capabilities for small vessels where topside area is limited.

4. USS Spanagel : The Building Wide Array

Perhaps one of the more innovative and interesting ideas is the implementation of the multi-frequency distributed array on the facade of Spanagel Hall, where the Physics and Electrical Engineering departments are housed at the Naval Postgraduate School. Elements would be placed at either random or evolved locations in the windows and on the roof of the building, simulating the hull and superstructure of a ship, in a distributed array. The array would be used to track surface vessels in Monterey Bay and aircraft over flying the building as part of the routine traffic through Monterey Airport. I would recommend the use of at least two different RF frequencies to show the advantages in terms of removing blind ranges and speeds. The full length of the building should be used as well in order to demonstrate the advantage in angular resolution achieved with a large aperture, even at low frequencies. A third frequency for voice communications would reinforce the benefits of such a configuration to the Navy. The problem of real time mensuration of the elements' location may also need to be addressed in this step, since weather and other factors will affect such a large array, and would even more so on a ship at sea.

C. A TOOL FOR THE FUTURE

The need for a robust and flexible design tool for shipboard sensors is there now. Genetic Algorithms have the potential to become that tool. As the requirements of the radar designer change, the Genetic Algorithm is easily adapted to meet them. The successful validation of a significant aspect of the fitness assessment process presented in this thesis is the first step in making such a design tool available to the architects of future sensor systems.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A: GLOSSARY OF TERMINOLOGY

COTS	Commercial Off The Shelf
dB	Decibels
DD(X)	US Navy Next Generation Surface Combatant – Destroyer
f	Frequency
FAD	Fleet Air Defense
fitness	Computed quantifiable score of the effectiveness of a population member as a solution to the given problem
GA	Genetic Algorithm
GHz	Gigahertz (10^9 cycles/second)
generation	A complete GA reproductive cycle including evaluation of fitness, selection and the formulation of a new population for the next generation.
HP	Hewlett-Packard
individual	A discrete set of bit strings and/or vectors that forms a complete solution to the given problem as evaluated by the fitness function.
IEEE	Institute of Electrical and Electronics Engineers
λ	Lambda – Wavelength
LCS	Littoral Combat Ship
MFAR	Multi-Function Array Radar
MoM	Method of Moments
Φ	Phi – Phase
population	All the individuals in a given GA run.
RCS	Radar Cross Section

RF	Radio Frequency
SUW	Surface Warfare
TBMD	Theater Ballistic Missile Defense
TR	Transmit/Receive

APPENDIX B: THE ATHIN.M CODE LISTING

```
%
%    The main function is:
%         athin
%
% and the following 10 additional functions are defined further down in
% the file:
%         apattern        arect           cross           makepattern
%         mutate          randpop         rect_sph        roulette
%         rslav           sqcirca
%
%     The function "apattern" computes values of an array pattern given
% (1) locations and phases for the "on" elements, and
% (2) points on the unit sphere (direction vectors) at which to sample
% the pattern.  It is invoked once for each individual in the
population
% in each generation.
%     "arect" is a plotting function called once each generation to
% display a 3-D plot of the current putative best-of-generation
pattern.
%     "cross" is a genetic operator, ("crossover") called for a
% specified fraction of the individuals in each generation to help
% create a new generation.
%     "makepattern" is a cover function for "apattern".
%     "mutate" is another genetic operator, called for a specified
% fraction of the individuals in each generation to help create a new
% generation.
%     "randpop" is called once initially to create an initial
% population, "generation 0".
%     "rect_sph" computes rectangular coordinates from spherical
% coordinates; it is called once during initialization.
%     "roulette" is used for fitness-proportional (or rank-
proportional)
% selection.  It is called once each generation with a vector of
% fitness values (or ranks) after these have been computed for the
% entire current generation.  It is used in selecting the members of
the
% current generation that will participate (via copying, mutation, or
% crossover) in the creation of the next generation.
%     "rslav" ("relative sidelobe level---average") computes a fitness
% function given a pattern.  It is invoked once for each individual in
% the population in each generation.
%     "sqcirca" computes a grid of points for plotting purposes.  It is
% used once per generation in conjunction with "makepattern" and
% "arect" to display a plot of the current best-of-generation pattern.
%
%     Probably a large proportion of the total computer time expended
is
% accounted for by the single statement
%         pat = sum(exp((2i*pi*x) * u), 1);
% on line 16 of "apattern".  This is a simplification of the code that
% is actually run, as the matrix product
%         (2i*pi*x) * u
% can be large enough to exceed memory limitations on the PC that I am
% using.  The actual code is a loop that breaks the computation into
% several chunks.
%
%     The dimensions of "x" are <number of array elements> by 4; the
% dimensions of "u" are 4 by <number of points at which to sample the
% pattern>.
%
%     The following Matlab functions are used in the code.  Some are
% built-in primitives, and some are defined functions supplied with
% Matlab:
%         abs             cat             ceil            clock
%         colormap        complex         conj            cos
%         cumsum          diff            exp             figure
%         gray            imag            length          log10
%         max             meshgrid        min             mod
%         nargin          pause           permute         plot
%         prod            rand            randint         randperm
%         real            rep             repmat          reshape
%         round           set             shading         sign
%         sin             size            sort            sqrt
%         sum             surfl           zeros
%
%======================================================================
=

function [on, ph, fitness] = athin(xyz, npop, ngen, rep, seed)
    % xyz [<number of elements>, 3]:    element locations (units of
    %                                       lambda)
    % npop:                             population size
    %   or [npop, ncases], where
    %   npop:                           population size
    %   ncases (default 200):           number of pattern values for
    %                                       fitness estimation
    % ngen:                             number of generations
    % rep [3]:                          proportions of copy, crossover,
    %                                       and mutation operation used
    %                                       in breeding a new
population
    %   [ncopy, ncross, nmutate], where
    %   ncopy:                          proportion of copy operations
    %   ncross:                         proportion of crossover (  2)
    %   nmutate:                        proportion of mutations
    % seed {[<number of elements>, <number of individuals>],
    %       [<number of elements>, <number of individuals>]}
    %       (optional):
    %                                   specific individuals to include
    %                                       in initial population
    % Returns
    %    on [<number of elements>, npop]:
    %                                   "on" values (boolean) for last
    %                                       generation (entire
    %                                       population)
    %    ph [<number of elements>, npop]
    %                                   phase values for last
generation
    %                                       (in cycles)
    %    fitness [npop]:                fitness values for last
    %                                       generation
    % Supplying {on, ph} from one run as the "seed" argument for
another
    %   invocation permits a run to be continued, possibly with
    %   different values of some parameters.

    starttime = clock;
    th0 = 0; ph0 = 90;              % steering direction (azimuth,
elev)
    u0 = rect_sph([1; th0; ph0]);   % steering direction (rectangular)
    L = [7.5 7.5 0];                % determines size of neighborhood
                                    %   of steering direction excluded
                                    %   from sidelobe estimation
    if length(npop) > 1
        ncases = max(npop(2), 1);
    else
        ncases = 200;
    end
    npop = max(npop(1), 1);
    ngen = max(ngen, 0);
    rep = rep/sum(rep);
    ncross = round(rep(2)*npop);
    if mod(ncross, 2) == 1
        ncross = ncross - 1;
    end
    ncopy = round(sum(rep(1:2))*npop) - ncross;
    nmutate = npop - ncross - ncopy;
    for g = 0:ngen                  % for each generation:
        if g == 0                   % first time, create initial
                                    %   population
            if (nargin < 5)
                [on, ph] = randpop(size(xyz), npop);
            else
                on0 = seed{1};
                ph0 = seed{2};
                [on, ph] = randpop(size(xyz), npop - size(on0, 2));
                on = [on0==1, on==1];
                ph = [ph0, ph];
            end
        else                        % breed new population from old
            [dum, rk] = sort(fitness);  % compute ranks from fitnesses
            [dum, rk] = sort(rk);
            sel = roulette(rk, npop);   % select parents (rank-based)
            on = on(:, sel);
            ph = ph(:, sel);
            for i = ncopy+1 : 2 : ncopy+ncross-1
                [on(:, [i, i+1]), ph(:, [i, i+1])] = ...
                    cross(on(:, [i, i+1]), ph(:, [i, i+1]));
            end                         % do crossovers
            for i = ncopy+ncross+1 : npop
                [on(:, i), ph(:, i)] = mutate(on(:, i), ph(:, i));
            end                         % do mutations
        end
        fitness = zeros(1, npop);   % fitness computation
        for i = 1:npop                  % for each individual:
            xyzph = [xyz, ph(:, i)];        % append phases to
locations
```

```
            xyzph = xyzph(on(:, i), :);     % select "on" elements
            r = rand([1, ncases]);          % generate random set of
            th = rand([1, ncases])*(2*pi);  %   directions (points on
            w = sqrt(1 - r);                %   unit sphere) for
pattern
            r = sqrt(r);                    %   evaluation
            uvw = [r.*cos(th); r.*sin(th); w];
            uvw = [uvw, u0];                % include steering
direction
            pat = apattern(xyzph, uvw);     % evaluate pattern
            fitness(i) = rslav(pat, u0, uvw, L);
                                            % estimate ratio of average
                                            %   sidelobe level to main
                                            %   lobe
        end
        figure(3);                          % plot fitness distribution
        plot(10*log10(sort(fitness))); set(gca, 'Ylim', [-10 25]);
        [bestf, besti] = max(fitness);      % plot "best" pattern
        uvw = sqcirca(1/64);
        pat = makepattern(xyz, on, ph, besti, uvw);
        arect(pat, uvw);
        gen_bestf_altbest_numon = ...
            [g, bestf, rslav(pat, u0, uvw, L), sum(on(:, besti))]
                                            % print: generation number,
                                            %   "best" fitness,
                                            %   alternative estimate
for
                                            %   "best" pattern,
                                            %   number of "on" elements
        pause(0.3);

        if 0                                % placeholder for
                                            %   alternative stopping
                                            %   criterion
            break;
        end
    end
    start_end = [starttime; clock]          % print starting and
                                            %   stopping times

%=====================================================================
=
% "apattern" computes values of an array pattern

function pat = apattern(x, u)
    % x [<number of "on" elements>, 4]: locations and phases of "on"
    %                                   elements (phases in fourth
    %                                   column)
    % u [3, <number of directions>]:    points on unit sphere
(direction
    %                                   vectors) at which to sample
    %                                   pattern
    % Returns
    %   pat [1, <number of directions>]:
    %                                 array of pattern values
    % Alternatively, <number of directions> may be replaced with a list
    %   of two (or more) dimension, which then become the shape of the
    %   result.

    s = size(u); s = s(2:end);
    if length(s) > 1
        u = u(:,:);
    end
    if size(u, 1) < 3
        u(3,:) = sqrt(max(0, 1 - sum(u.^2, 1)));
    end
    if length(size(x)) > 2
        L = size(x, length(size(x)));
        x = reshape(x, prod(size(x))/L, L);
    end
    if size(x, 2) > 3
        u(4,:) = 1;
    end
    pat = sum(exp((2i*pi*x) * u), 1);
    pat = pat.*conj(pat)/size(x,1);
    if length(s) > 1
        pat = reshape(pat, s);
    end

%-----------------------------------------------------------------------
-
% "arect" displays a 3-D plot of a pattern.

function arect(pat, uvw, fig)
    % pat: [m, n]                        array of pattern values
    % uvw: [3, m, n]                     corresponding direction vectors
    %                                        where m and n are the
height
    %                                        and width of the array of
    %                                        pattern values.  Each
column
    %                                        is a triple of coordinates
    %                                        [u,v,w].
    % fig (optional):                    figure number

    if nargin < 3
        fig = 1;
    end
    s = size(uvw); s = s([2 3]);
    figure(fig);
    u = reshape(uvw(1,:), s);
    v = reshape(uvw(2,:), s);
    surfl(u, v, max(-10, 10*log10(max(pat,realmin))));
    zl = [-10, 5*ceil(2*log10(max(max(pat))))];
    set(gca, 'Zlim', zl);
    shading('interp');
    colormap(gray(64));

%-----------------------------------------------------------------------
-
% "cross" performs genetic crossover operation: breeds new individuals
%   from pairs of existing individuals (two children from each pair of
%   parents
```

```
function [on, ph] = cross(on, ph)
    % on [<number of elements>, 2]:     "on" values (boolean) for two
    %                                       parents
    % ph [<number of elements>, 2]:     phase values for two parents
    % Returns
    %   on [<number of elements>, 2]:   "on" values (boolean) for two
    %                                       children
    %   ph [<number of elements>, 2]:   phase values for two children

    lims = randint(1, 2, [1, size(on, 1)]);
    lim1 = min(lims);
    lim2 = max(lims);
    t = on(lim1:lim2, 1);
    on(lim1:lim2, 1) = on(lim1:lim2, 2);
    on(lim1:lim2, 2) = t;
    t = ph(lim1:lim2-1, 1);
    ph(lim1:lim2-1, 1) = ph(lim1:lim2-1, 2);
    ph(lim1:lim2-1, 2) = t;
    d = diff(ph([lim1, lim2], :), 1, 2);
    d = (mod(d+0.5, 1) - 0.5) .* (rand(2, 1) - 0.5);
    ph([lim1, lim2], :) = ph([lim1, lim2], :) + [d, -d];

%----------------------------------------------------------------------
-
% "makepattern" computes values of an array pattern; cover function for
%   apattern

function pat = makepattern(x, on, ph, i, uvw)
    % x [<number of elements>, 3]:      element locations (units of
    %                                       lambda)
    % on [<number of elements>, npop]:  "on" values (boolean) for
    %                                       population (return from
    %                                       athin)
    % ph [<number of elements>, npop]:  phase values for population
    %                                       (return from athin)
    % i:                                index of individual
    % uvw [3, <number of directions>]:  points on unit sphere
(direction
    %                                       vectors) at which to sample
    %                                       pattern
    % Returns
    %   pat [1, <number of directions>]:
    %                                   array of pattern values
    % Alternatively, <number of directions> may be replaced with a list
    %   of two (or more) dimension, which then become the shape of the
    %   result.

    pat = apattern([x(on(:,i),:), ph(on(:,i),i)], uvw);

%----------------------------------------------------------------------
-
% "mutate" performs genetic mutation operation: creates new individuals
%   by introducing random changes in existing individuals

function [on, ph] = mutate(on, ph)
    % on [<number of elements>, 1]:     "on" values (boolean) for one
    %                                       individual
    % ph [<number of elements>, 1]:     phase values for one individual
    % Returns
    %   on [<number of elements>, 1]:   "on" values (boolean) for
    %                                       mutated individual
    %   ph [<number of elements>, 1]:   phase values (boolean) for
    %                                       mutated individual

    lims = randint(1, 2, [1, size(on, 1)]);
    lim1 = min(lims);
    lim2 = max(lims);
    on(lim1:lim2) = rand(lim2-lim1+1, 1) >= 0.5;
    ph(lim1:lim2) = rand(lim2-lim1+1, 1);

%----------------------------------------------------------------------
-
% "randpop" creates a random population of individuals

function [on, ph] = randpop(size, npop)
    % size:                             size of element location (xyz)
    %                                       array (only the first
    %                                       dimension is used, i.e. the
    %                                       number of elements).
    % npop:                             population size
    % Returns
    %   on [size(1), npop]:             "on" values (boolean) for
    %                                       population
    %   ph [size(1), npop]:             phase values for population

    on = rand([size(1), npop]) >= 0.5;
    ph = rand([size(1), npop]);

%----------------------------------------------------------------------
-
% "rect_sph" computes rectangular coordinates (x, y, z) from spherical
%   coordinates (rho, theta, phi): radial distance, azimuth, and
%   elevation

function xyz = rect_sph(rtp)
    % rtp [3, ...]:                     array of spherical coordinates
    % Returns
    %   xyz [3, ...]:                   array of rectangular
coordinates

    rtp([2 3], :) = rtp([2 3], :)*(pi/180);
    r = rtp(1,:).*cos(rtp(3,:));
    xyz = [r.*cos(rtp(2,:)); ...
           r.*sin(rtp(2,:)); ...
           rtp(1,:).*sin(rtp(3,:))];
    xyz = reshape(xyz, size(rtp));

%----------------------------------------------------------------------
-
% "roulette" performs roulette-wheel selection

function sel = roulette(fitness, n)
    % fitness (row vector):             fitness values.  For rank-
    %                                       proportional selection,
    %                                       supply vector of ranks.
    % n:                                    number of individuals to
    %                                       select
    % Returns
    %   sel [n]:                            indices of selected individuals

    s = cumsum(fitness);
    r = sort(rand(1, n)) * s(length(s));
    [r, p] = sort([r, s]);
    sel = zeros(size(p)); sel(p) = [1:length(p)];
    sel = sel(1:n) - [0:n-1];
    sel = sel(randperm(n));

%----------------------------------------------------------------------
-
% "rslav" estimates the ratio of average sidelobe level of a pattern
%   to main-lobe level

function L = rslav(pat, u0, u, g)
    % pat [1, <number of directions>]:  pattern values
    % u0 [3, 1]:                        main-beam steering direction
    % u [3, <number of directions>]:    directions
    % g [1, 3]:                         criterion for excluding points
    %                                       "too close" to the main-
    %                                       beam direction from the
    %                                       average sidelobe estimate.
    %                                       Let u be a direction
vector.
    %                                       If the vector (u-u0).*g
    %                                       (coordinate-by-coordinate
    %                                       product) has magnitude 1 or
    %                                       less, the pattern value for
    %                                       direction u is excluded.
    %                                       E.g. g = [7.5 7.5 0]
    %                                       excludes a point u unless
    %                                       the distance of its
    %                                       projection into the xy-
plane
    %                                       from that of the steering
    %                                       direction is greater than
    %                                       1/7.5.
    % Alternatively, pat may be an array of 2 or more dimension (not a
    %   row vector) and the shape of u may be [3, size(pat)].
    % Returns
    %   L:                                  estimated ratio (compute
    %                                       10*log10(L) to convert
    %                                       to dB)

    u0 = reshape(u0, [length(u0), 1]);
    g = reshape(g, [length(g), 1]);
    s = size(u); s(1) = 1;
    d = sum(((u - repmat(u0, s)).*repmat(g, s)).^2);
    m = sum(pat(d > 1));
    n = max(pat(d <= 0.0025));
    if m == 0 | length(n) == 0
        L = 0;
    else
```

```
        L = n*sum(d(:) > 1)/m;
    end

%-----------------------------------------------------------------------
-
% "sqcirca" computes a grid of points for plotting purposes.
% It uses a mapping of the square -1 <= x <= 1, -1 <= y <= 1 onto
% the unit hemisphere u^2 + v^2 + w^2 = 1, w >= 1.
% The density of projections of grid points onto the unit circle in
% the uv-plane is approximately uniform.
%
function u = sqcirca(d)
    % d:                                  spacing of grid points in the
    %                                         square.  The number of grid
    %                                         points is n^2, where n is
    %                                         floor(1+1/d) or ceil(1/d),
    %                                         e.g. 129^2 = 16641 when
    %                                         d = 1/64.
    % Returns
    %   u [3, n, n] (n as above):       array of coordinate triples

    x = [-1:d:1];
    x = x + 0.5*(1 - x(end));
    [x, y] = meshgrid(x);
    p = (abs(x) > abs(y));
    q = ~p & (y ~= 0);
    u = x;  v = x; theta = x;
    theta(p) = ((pi/4)*y(p)) ./ x(p);
    theta(q) = ((pi/4)*x(q)) ./ y(q);
    u(p) = x(p) .* cos(theta(p));
    v(p) = x(p) .* sin(theta(p));
    u(q) = y(q) .* sin(theta(q));
    v(q) = y(q) .* cos(theta(q));
    w = sqrt(max(0, 1 - u.^2 - v.^2));
    u = cat(3, u, v, w);
    u = permute(u, [3 1 2]);

%=======================================================================
=
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C: LOCATION OF RANDOM ARRAY ELEMENTS

Element Number	X-Coordinate [inches]	Y-Coordinate [inches]	Element Number	X-Coordinate [inches]	Y-Coordinate [inches]
1	1.4182	-3.1934	13	-6.0295	-5.1197
2	1.9821	3.9394	14	3.2750	-0.6326
3	3.0061	2.1820	15	0.5112	-3.9564
4	5.1609	-6.3229	16	-1.8416	1.5413
5	1.7698	0.8005	17	5.4489	5.0811
6	-3.6236	-0.5907	18	-3.4405	5.2915
7	6.6931	5.2643	19	0.9727	3.3614
8	1.9670	-2.8319	20	-4.7703	-1.5505
9	-3.7821	-5.6546	21	6.7267	-0.2067
10	2.3215	6.2883	22	4.0856	3.9288
11	-5.1139	5.4905	23	4.6624	-3.8641
12	-6.6851	0.7956	24	-0.8178	3.0863

Table 3. Random Array Element Locations

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Altschuler, E. and Linden, D. (1996). ”Automating Wire Antenna Design Using Genetic Algorithms”. *Microwave Journal*. March 1996. p 74-86.

Billetter, D. (1989). *Multifunction Array Radar*. Norwood, Massachusetts: Artech House Publishing.

Boas, M. (1983). *Mathematical Methods in the Physical Sciences*. New York: John Wiley and Sons.

Cantrell, B., de Graaf, J., Leibowitz, L., Meurer, G., Parris, C., Stapleton, R. and Willwerth, F. (2002). “Develepment of Digital Array Radar”. *IEEE AESS Systems Magazine*. Volume 17. Number 3. March 2002. p. 22-27.

Chambers, L. (1995). *Practical Handbook of Genetic Algorithms: Applications Volume I*. New York: CRC Press.

Dorf, R. and Smith, R. (1992). *Circuits, Devices and Systems, 5th Edition*. New York: John Wiley and Sons.

Fogel, D. (2000). “What is Evolutionary Computation?”. *IEEE Spectrum*. February 2000. p. 26-32.

Giordano, N. (1997). *Computational Physics*. London: Prentice-Hall.

Griffiths, D. (1999). *Introduction to Electrodynamics, 3rd Edition*. London: Prentice-Hall.

Hecht, E. (1998). *Optics, 3rd Edition*. Reading, Massachusetts: Addison Wesley Longman Inc.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems*. Cambridge, Massachusetts: The MIT Press.

Icheln, C., Ollikainan, J. and Vainikainan, P. (2001). “Effects of RF Absorbers on Measurements of Small Antennas in Small Anechoic Chambers”. *IEEE AESS Systems Magazine*. Volume 16. Number 11. November 2001. p. 17-20.

Jacomb-Hood, A. and Purdy, D. (1999) "In Orbit Active Array Calibration for NASA's LightSAR". *IEEE Radar Conference*. Boston, Massachusetts: IEEE. p. 172-176.

Michielssen, E. and Weile, D. (1997). "Genetic Algorithm Optimization Applied to Electromagnetics: A Review". *IEEE Transactions on Antennas and Propagation*. Volume 45. Number 3. March 1997. p. 343-353.

Purdy, D. (2000). "Automated Process for Efficiently Measuring the Patterns of All Elements Located in a Phased-Array Antenna". *IEEE International Conference on Phased Array Systems and Technology*. p. 521-524.

Kasap, S. (2002). *Principles of Electronic Materials and Devices, 2nd Edition*. New York: McGraw-Hill.

Koza, J. (1992). *Genetic Programming*. Cambridge, Massachusetts: The MIT Press.

Reubens, P., ed. (2001). *Science and Technical Writing, 2nd Edition*. New York: Routledge.

Sanchez, E. and Tomassini, M. (1996). *Towards Evolvable Hardware: The Evolutionary Engineering Approach*. Berlin: Springer.

Skolnik, M. (2001). *Introduction to Radar Systems, 3rd Edition*. New York: McGraw-Hill.

Willis, N. (1995). *Bistatic Radar*. Silver Spring, Maryland: Technology Service Corporation.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. RADM David Altwegg, (ret.)
   Deputy Assistant Secretary of the Navy - Theater Combat Systems
   Washington, D.C.

4. CDR Billie S. Walden
   Program Executive Office – Theater Surface Combatants
   Washington Navy Yard
   Washington, D.C.

5. Dr. Thomas Kimbrell
   Naval Surface Warfare Center
   Dahlgren, Virginia

6. RADM Wayne Meyer, (ret.)
   W.E. Meyer Corporation
   Arlington, Virginia

7. CAPT Al Haggerty
   NAVSEA Code 426
   Washington, DC

8. Professor Mike Melich
   Institute for Joint Warfare Analysis
   Monterey, California

9. Professor Dave Jenn
   Department of Electrical and Computer Engineering
   Naval Postgraduate School
   Monterey, California

10. Professor Rod Johnson
    Institute for Joint Warfare Analysis
    Monterey, California

11. Professor Robert Harney
Naval Postgraduate School
Monterey, California

12. Professor William Maier
Naval Postgraduate School
Monterey, California

13. Dr. Dan Purdy
Office of Naval Research
Arlington, Virginia

14. Dr. Bobby Junker
Office of Naval Research
Arlington, Virginia

15. LCDR Lance Esswein
Naval Postgraduate School
Monterey, California

16. Nicholas Willis
Carmel, California

17. CAPT Michael Cassidy
Lockheed Martin Naval Electronics and Surveillance Systems
Moorestown, New Jersey

18. RADM Tim Hood, (ret.)
Lockheed Martin Government Electronic Systems
Moorestown, New Jersey

19. Peter Wilhelm
Naval Research Laboratory
Washington, D.C.

20. Robert Eisenhauer
Naval Research Laboratory
Washington, D.C.

21. Ed Senasack
Naval Research Laboratory
Washington, D.C.

22. Professor Herschel Loomis
Naval Postgraduate School
Monterey, California

23. Dr. Ray Bernstein
New Mexico State University
Las Cruces, New Mexico

24. Professor Paul Moose
Naval Postgraduate School
Monterey, California

25. Professor Curt Schleher
Naval Postgraduate School
Monterey, California

26. Lee Hammarstrom
Port Matilda, Pennsylvania

27. Lee Moyer
Defense Advanced Research Projects Agency
Arlington, Virginia

28. LT Jon Bartee
Naval Postgraduate School
Monterey, California