Shape-Changing Phased Arrays

Thesis by
David Elliott Williams

In Partial Fulfillment of the Requirements for the
Degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2022
Defended June 7th, 2022

David Elliott Williams
ORCID: 0000-0002-6213-4712

To the boy in Philly
who thought his life was over
and everyone else staring into The Abyss.

ACKNOWLEDGEMENTS

I decided to pursue a Ph.D. so that I could grow into an independent thinker, someone that could take an abstract idea, doggedly pursue it despite the uncertainty, and realize its full potential. When I first arrived at Caltech, I felt that I was limited to the things I had learned in a college classroom. Now, I know that with enough time and resources, I can pursue anything.

However, this kind of growth is challenging. It requires hard work, intense discomfort, and deep introspection. And most importantly, it requires the mentorship, guidance, support, and love of other people. Thus this thesis is not just the culmination of my personal academic work, but the culmination of the efforts of dozens of people that have supported me along the way. For all of this help, I am deeply grateful.

I want to begin by thanking my advisor, Professor Hajimiri. None of this growth would have been possible without his mentorship. Professor Hajimiri pushed me when I needed to be pushed, challenged me when I needed to be challenged, supported me when I needed help, fought for me when I needed an ally, and guided me when I felt lost. There are very few people that I have learned from as much as I have from Professor Hajimiri about scholarship, work, life, and myself. I know that I will be grateful for his hard work and dedication for the rest of my life.

Next I would like to thank the other members of my thesis committee, Professor Azita Emami, Dr. Sanders Weinreb, and Professor Kaushik Bhattacharya, for their guidance, feedback, and support.

I would like to thank my collaborators as part of the MURI initiative, especially Robert Lang, Yang Li, Charlie Dorn, and Professor Sergio Pellegrino, whose input has helped shape this thesis.

I would like to thank the Air Force Office of Scientific Research for supporting my work as part of MURI Grant FA9550-16-1-0566. I also would like to thank the Rogers Corporation for their donations of substrate materials over the years; never have free samples been more critical to scientific progress.

I want to thank Matan Gal for being a great friend and mentor early in my Ph.D. I do not know if anyone has ever enjoyed, or been as good at, getting me riled up more than him. I want to thank Austin Fikes for being a great colleague, editor, friend,

and source of support; I could not have asked for a better person to be in my cohort. I want to thank Craig Ives for his insights, his friendship, and the many deep, yet hilarious and distracting, philosophical conversations we have had over the years. I also want to thank all the other CHIC lab members whom I have had the fortune of learning from: Florian, Amir, Behrooz, Alex Pai, Brian, Costas, M. Reza, Noyan, Reza, Aroutin, Parham, Armina, Samir, Ailec, Oren, Debjit, Volkan, Alex Ayling, David, Alex White, Mohith, and Cole.

I want to thank David Hodge not just for his help fabricating the shape-changing phased array project, but also for his friendship, wisdom, and passion. I also would like to thank Michelle Chen both for single handedly keeping the lab operational and for the wonderful conversations we have had over the years.

I want to thank the rest of the administrators and staff that keep Moore running, but especially Tanya Owen, Anne Sullivan, and Christine Garske for their hard work and pleasant conversation. I want to thank Esthela Jalabert for being a constant source of positivity over the last six years. Her cheery smile when I greeted her always brightened my day, and her excited interest in what I was working on reminded me of why I wanted to be an engineer in the first place.

I want to thank Kathy Johnson, Dean Natalie Gilmore, and my proofreader; I literally could not have graduated without their help given how close my defense was to the deadline.

I have had many mentors outside of my lab over the last six years that have helped me to not just grow, but to achieve my goals and thrive. I want to begin by thanking Dr. Mark Glat for his support, wisdom, perception, and perspective. Next I want to thank Dr. Jenn Weaver for supporting me, helping me grow as an instructor, and starting me on the path to my next adventure. I want to thank Lucas Martinez for both teaching me to let-go, not think, and be creative, and for helping me stay sane over the last five years. I want to thank Grace Ho for teaching me how to function and for our many interactions that filled me with joy. As I have said for years, "If I ever get a Ph.D. it is singlehandedly [Grace's] fault." Finally, I want to thank Dr. Vanessa Tejada for being a great boss and source of support while I was an RA.

Speaking of being an RA, I want to thank all of my former students for a wild, fun, rewarding and *deeply* unique experience. I learnt more about leadership, politics, diplomacy, and myself in those two years than in the preceding 25. However, I could go without ever hearing the *Ride of the Valkyries* again.

I want to thank my friends in Pasadena and Los Angeles over the years, Riley, Morgan, Cecelia, Kathleen, Camilla, Nathan, Leo, Joži, Jade, Sarida, and Sonia. In particular, I want to thank Red for being an amazing friend at a truly critical period in my life. I want to thank my friends back east–Jeremy, David Mundschenk, Alejando, and Marly–all of whom I dearly miss and fill me with joy every time I talk to them.

I want to thank my editor Ethel for her continuous support, company, and insistence that my thesis needed more letters and commas. *Editor's Note:|}}}}}}}}}}}}* *\\\\\\\\\\\\\\\\\\\\\\\\\\\\]=p-[;'[l.p';='\=.*[1]

I want to thank my family–my Mother, my Father and Ali, Aunt Nancy and Paige, Perry, Sherlock, Nina, Gavin, and *especially* little Maddie–for their love, advice, and support. Sometimes it is critical to take a break from research and nothing is a nicer break then spending time with the people I love most. You all, more than anyone, have made me the person I am today. Thank you.

Finally, I want to thank my partner Rayden. They took a great risk moving to California to be with me and another by following me back east this summer. I do not know if I could have finished this dissertation without their love, support, kindness, and patience. They make me a better person.

A quick note: As anyone close to me knows, music is incredibly important to me. Each chapter in this thesis begins with a lyric from an album that helped me keep going during my time at Caltech. The quotes are in chronological order of when I first became obsessed with each album; the first chapter's quote reflects my first year at Caltech and the last chapter's reflects my last. Together they form a sort of musical journey though my time in graduate school. I hope you give the albums a listen.

[1] Ethel is my dog.

ABSTRACT

Historically, increasing the degrees of freedom in electromagnetic structures has revolutionized the capabilities of wireless systems and introduced new applications. While research on phased arrays has explored everything from antenna drive settings to the element placement, the array geometry is assumed to be a fixed parameter. This thesis summarizes the author's work developing shape-changing phased arrays. It demonstrates the fundamental trade-off between gain and steering range for a given geometry. Measurements of the first shape-changing phased array both verify this theory and demonstrate the ability to break this trade-off using geometric reconfiguration. In addition, the mathematical consequences of shape-change and their impact on the arrays electromagnetic properties are discussed. Programmable passive switching networks on flexible sheets embedded in the array are proposed to address these challenges. The ability of these structures to enhance array performance is demonstrated by *in-situ* optimization experiments on a demonstration array. The associated optimization problem is characterized with a statistical analysis on a simulated array. Finally, avenues for further research are proposed.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] D. E. Williams, C. Dorn, S. Pellegrino, and A. Hajimiri, "Breaking the tradeoff between maximum gain and steering range with an origami-inspired shape changing phased array," *To be submitted to IEEE Transactions on Antennas and Propagation*, 2022.

[2] D. E. Williams and A. Hajimiri, "Flexible meta-gaps to compensate for grating lobes in shape-changing phased arrays," *To be submitted to IEEE Transactions on Microwave Theory and Techniques*, 2022.

[3] ——, "In-situ optimization of meta-gaps to decrease side lobe levels and increase the field of view of a λ-spaced array," *To be submitted to IEEE Transactions on Microwave Theory and Techniques*, 2022.

[4] ——, "Meta-gaps for mechanically reconfigurable phased arrays," *Under review for 2022 52nd European Microwave Conference (EuMC)*, 2022,
Content used with permission by the copyright owner, the European Microwave Association (EuMA).

[5] D. E. Williams, C. Dorn, S. Pellegrino, and A. Hajimiri, "Origami-inspired shape-changing phased array," in *2020 50th European Microwave Conference (EuMC)*, 2021, pp. 344–347. DOI: 10.23919/EuMC48046.2021.9338189,
D. E. Williams participated in the conception of the project, developed the theory, designed the electromagnetic system (including tiles, integrated circuit, and antenna), wrote controlling code, performed the measurements, analyzed the data, and wrote the manuscript. Content used with permission by the copyright owner, the European Microwave Association (EuMA).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

NOMENCLATURE

APA. Aperture Projection Analysis.

AUT. Antenna/Array under Test.

CMOS. Complimentary Metal-Oxide Semiconductor.

CTAT. Complementary to Absolute Temperature.

DAC. Digital to Analog Converter.

EIRP. Effective Isotropic Radiated Power.

FoV. Field of View.

FSPL. Free Space Path Loss.

HPBW. Half Power Beam Width.

IC. Integrated Circuit.

LVCMOS. Low Voltage CMOS.

MBP. Main Beam Power.

MIMCAP. Metal-Insulator-Metal Capacitor.

MOSCAP. Metal-Oxide Semiconductor Capacitor.

PA. Power Amplifier.

PC. Personal Computer.

PCB. Printed Circuit Board.

PLL. Phase Locked Loop.

PS. Particle Swarm.

PTAT. Proportional to Absolute Temperature.

RF. Radio-Frequency.

RFIC. Radio-Frequency Integrated Circuit.

SA. Simulated Annealing.

SLL. Side Lobe Level.

SPI. Serial Peripheral Interface.

TSMC. Taiwan Semiconductor Manufacturing Company.

VCO. Voltage Controlled Oscillator.

VNA. Vector Network Analyzer.

VNS. Variable Neighborhood Search.

Chapter 1

INTRODUCTION

> And we came to see Time
> is taller than Space is wide.
>
> *Joanna Newsom*
> *Waltz of the 101st Lightborne*

1.1 Degrees of Freedom in Wireless Systems Revolutionize the World

The fundamental goal of a wireless system is to manipulate the local electromagnetic fields to create the desired far-field effect. The history of wireless technology has been the incorporation of new degrees of freedom to gain more sophisticated control over radiated and sensed waves. Each generation of advancement was predicated on the availability of new resources that expanded what was possible. And while enhancing the precision and flexibility of field manipulation appears to be a simple concept, each additional degree of freedom has had widespread and profound impacts on human society.

The First Generation: Enabling Radiation with Spark Gaps

Wireless technology was born in the late 19th century with the invention of the spark-gap transmitter by Heinrich Hertz. The transmitter consisted of two capacitors separated by long wires and a short air gap. The capacitors are charged to high voltages using a DC voltage source until the fields in the gap are strong enough to generate a spark. While sparking, charge would quickly accelerate between the capacitors and oscillate due to the inductance of the separating wire. These accelerating charges generate a damped electromagnetic wave that radiates away from the transmitter and can be detected remotely. Figure 1.1 shows an example of a commercial transmitter invented by Guglielmo Marconi.

While electrical telecommunications began with the invention of the electrical telegraph, the ability to communicate wirelessly with a spark-gap transmitter was revolutionary. It became possible to correspond with remote locations without a fixed cable in between, enabling ship-to-ship and ship-to-shore communication. Ships could quickly transmit distress calls when they began to sink, enabling rescue oper-

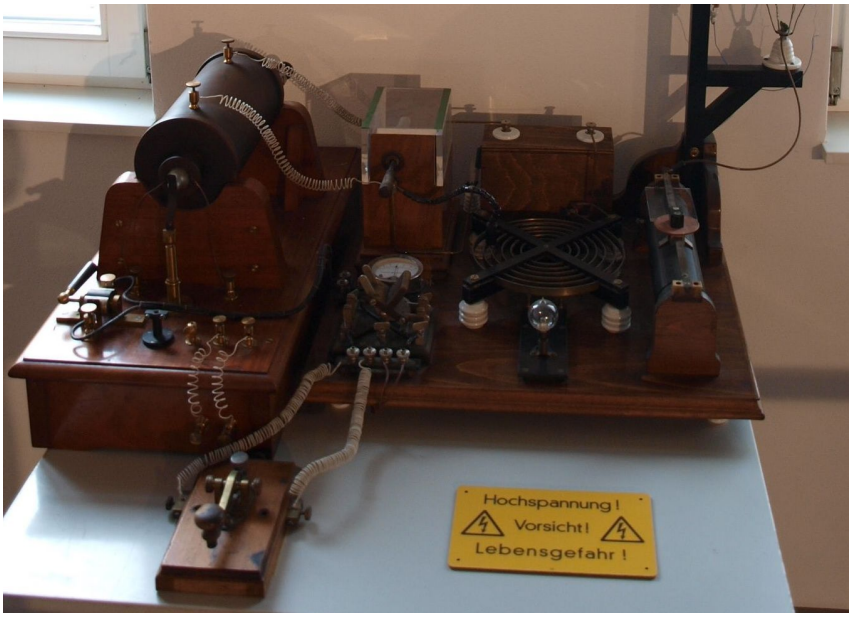


Figure 1.1: Spark gap transmitter in the Electric Museum in Frastanz, Austria.

ations that saved thousands of lives. Soon massive transmitters were built to enable transoceanic communication without having to lay underwater telegraph cables.

The Second Generation: Revolutionizing Society with Continuous Waves

Wireless systems really came into their own however with the invention of the vacuum tube and the development of continuous wave systems in the early 20th century. The precise frequency control of continuous wave systems provided isolation between different channels, greatly reducing the interference. Complex audio and video signals could be wireless transmitted via amplitude and phase control of constant frequency waves. A less obvious, but equally important, advancement was the impact of continuous waves on antenna design.

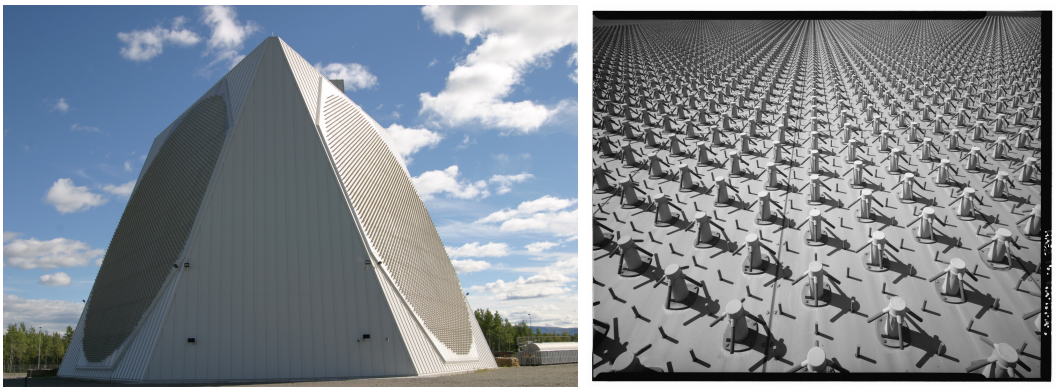
As spark-gap transmitters operated mostly in the kHz regime, only electrically short antennas were practical to construct. Thus the ability to shape the properties of the electromagnetic wave was limited. Continuous wave radios, however, could easily operate at hundreds of megahertz, thus enabling antennas with dimensions on the order of, or larger than, the wavelength. A wide range of antennas were developed for different applications, from highly directional antennas for long distance communication, to more isotropic ones for radio broadcasting. The ability to shape the radiation pattern enabled new wireless applications such as radio astronomy, radar, radio and video broadcasting, space communications, and GPS.

Figure 1.2: The Parkes radio telescope while receiving video from the moon of the Apollo 11 moon landing. © CSIRO via Wikicommons under CC BY 3.0

These new technologies had a momentous impact on human society. Radio broadcasting revolutionized popular music and brought the President of the United States into living rooms across the country in challenging times. Radar completely changed warfare, transportation, and weather broadcasting. Directional antennas enabled astronauts to communicate with the earth; Figure 1.2 is a picture of the Parkes radio telescope while it was receiving video of the moon landing, video that was broadcast around the world.

The Third Generation: Creating the Present with Phased Arrays

The next degrees of freedom to be introduced into wireless systems was precipitated by the development of solid-state electronics. Improvements in the f_{max} of silicon transistors enabled lightweight, low-cost power amplifiers at high frequencies, thus shrinking the size of radio technology. This size reduction further enhanced the impact of radio by making transceivers ubiquitous in phones, vehicles, and computers. The reduced size of transmitters also enabled a large number of them to be integrated into a single system, enabling phased arrays. While the ability of phased arrays to electronically alter their radiation patterns was well known, it was not until transmitters shrank that they could be small and cheap enough to be pragmatic.



(a) Upgraded SSPARS Array in Alaska. (b) Close up of Original Antenna Array.

Figure 1.3: The Precision Acquisition Vehicle Entry Phased Array Warning System (PAVES PAWS) was built in 1980 to detect incoming ballistic missiles.
Public Domain, US Federal Government–US Air Force.

Phased array's ability to electronically steer beams has revolutionized modern wireless systems. Radars can rapidly scan space, greatly increasing their speed and precision. Figure 1.3 shows the first solid-state phased array deployed, the Precision Acquisition Vehicle Entry Phased Array Warning System (PAVES PAWS) used by the U.S. airforce to detect incoming threats. Phased array radar has also greatly enhanced the capabilities of weather detection systems; according to the Director of the National Severe Storms Laboratory, "Phased array radar can extend [severe weather] warning lead times well beyond what was possible before" [1]. Phased array radar is also a key part of the development of autonomous vehicles.

Telecommunication systems have also been revolutionized by the use of multiple steerable beams in MIMO transmitters to increase bandwidth. Thanks to the relative ease of deploying wireless mobile networks, the number of people in the world with access to the internet has increased from 63% in 2015 to over 95% in 2021 [2], [3]. However, up to 30% of Africa's rural population still lacks mobile broadband coverage. In addition, the roll-out of broadband internet in the United States has been slow. In 2015, only 11% of Americans had access to more than one high-speed internet (100 Mbps) provider, four years later the percentage has only increased to 29% [4], [5]. Fortunately, phased arrays have also enabled the recent development of low-earth orbit satellite internet networks such as Starlink that will be able to supply broadband internet around the globe. This increase in access will spur both economic development in Africa and increased competition in the United States.

The Next Generation: Controlling the Future with Arbitrary Radiators

History has shown that increasing the degrees of freedom in wireless systems can revolutionize society. As in the past, the newly available resources in the present are a guide to what will be possible in the future. Over the last twenty years, computational power has vastly expanded, both in terms of its cost and availability, and in the advancement of new computational approaches such as machine learning and quantum annealing. And although the RF performance of silicon transistors has essentially plateaued, integration has advanced to the point that it is possible to fit an arbitrarily complex system into a small, cheap package. Research on origami-inspired structures, advancements in additive manufacturing, and the commercialization of flexible electronics have enabled the creation of arrays with almost arbitrary geometric structures. Thus, we can now construct and control systems with an extraordinary number of electrical and mechanical degrees of freedom. However, the difficulty in realizing the potential of this capability is how to properly leverage it to introduce *useful* degrees of freedom and create high-performing wireless systems. With this understanding, existing and emerging applications such as physically secure wireless links and wireless power transfer can be enhanced and entirely unforeseen applications can be enabled.

1.2 Contributions

It is clear that technological advancement has reached the point where near-arbitrary synthesis of radiation patterns is within our grasp. A radiator that can arbitrarily control both its geometry and the current distribution on its surface would be unconstrained in terms of what radiation patterns it could create. In order to explore the possibilities of such a "Universal Electromagnetic Surface," a Multidisciplinary University Research Initiative project was sponsored by the Air Force Office of Scientific Research. A interdisciplinary team of experts in Electromagnetics, Structures, Mechanics, Materials, Mathematics, and Origami was assembled to collaborate and study the various aspects of such a structure.

This thesis is a small part of that work and represents the first thrust into the world of shape-changing phased arrays. Over the last six years I have focused on addressing the electromagnetic questions prompted by shape change. What are the advantages of shape-change? What are the consequences? What kinds of system architectures are required to enable shape-change? Are there any unknown engineering and implementation challenges? If an array can change into arbitrary shapes, than what geometries are advantageous?

As with all research, the answers to these questions were sometimes surprising, sometimes relatively straightforward, and almost always introduced new questions. I developed the first shape-changing phased array to explore the associated implementation challenges [6]. The array also demonstrates the use of geometry as a design variable and that a single structure can exhibit both high maximum gain and a wide steering range, breaking the trade-off between the two. This array also demonstrated the key fact that in order to enable arbitrary shape change, the spacing of elements on the surface of the array must change, introducing grating lobes.

To address this challenge, I developed a passive, flexible, and programmable switching-networks that can unfold between tiles and change the near-field environment to compensate for the gaps. These *meta-gaps* proved to be a rich optimization problem that required careful application of measurement techniques and algorithms to explore. By experimenting with the meta-gaps, I characterized both the minimum enhancements they enable and aspects of the optimization problem they present.

1.3 Thesis Outline

The structure of the thesis is divided into two halves, with the first half providing the context and theory for the experiments that are performed in the second half. Chapter 2 provides background on antenna and phased array characterization that is used throughout the work. Chapter 3 details an in-depth theoretical justification for aperture projection analysis, which is used in Chapter 4 to perform an extensive analysis of different array geometries. Chapter 5 acts as a segue into the second half of the work by discussing the benefits of shape-changing phased arrays and the challenges associated with shape-change due to Gauss's *Theorema Egregium*.

The hardware designs of the radiating tiles that enable shape-change and used for experimentation are described in Chapter 6. Next, Chapter 7 presents measurements of the first shape-changing phased array that verify theory. The concept of flexible meta-gaps to address the increased element spacing required for shape-change is then introduced in Chapter 8. The optimization problem associated with meta-gaps is extensively explored in Chapter 9 using meta-hueristic algorithms and statistical analysis, while Chapter 10 demonstrates the enhancements enabled by meta-gaps through *in-situ* optimization of a meta-gap filled phased array. Finally, Chapter 11 summarizes the results and considers what is next. The appendices include extended mathematical derivations and source code for the optimization algorithms.

References

[1] T. Schoor, K. Pirtle, J. Murnan, and V. Farmer, "Nwrt: End of an era," National Severe Storms Laboratory, Sep. 2016. [Online]. Available: https://www.nssl.noaa.gov/about/history/nwrt-decommission/.

[2] "Measuring digital development: Facts and figures 2015," International Telecommunication Union, May 2015.

[3] "Measuring digital development: Facts and figures 2021," International Telecommunication Union, Dec. 2021.

[4] "Internet access services: Status as of june 30, 2019," United States Federal Communications Commision, Mar. 2022.

[5] "Internet access services: Status as of december 31, 2015," United States Federal Communications Commision, Nov. 2016.

[6] D. E. Williams, C. Dorn, S. Pellegrino, and A. Hajimiri, "Origami-inspired shape-changing phased array," in *2020 50th European Microwave Conference (EuMC)*, Jan. 2021, pp. 344–347, ISBN: 978-2-87487-059-0. DOI: 10.23919/EuMC48046.2021.9338189. [Online]. Available: https://ieeexplore.ieee.org/document/9338189/.

Chapter 2

PHASED ARRAY CHARACTERIZATION

> At any given moment, in the middle of a city
> there's a million epiphanies occurring
> and the blurring of the world beyond the curtain
> and the world within the person.
>
> *Kae Tempest*
> *Lionmouth Door Knocker*

In order to study phased arrays, it is important to understand their basic properties and behaviors, their associated figures of merit, and the measurement techniques used to characterize them. This chapter provides an important conceptual and experimental foundation on which later sections are built.

Section 2.1 begins by introducing the spherical coordinate system used throughout this work. This is followed by a background on various antenna and array properties in Section 2.2. Section 2.3 provides a general description on how phased arrays are able to perform electronic beam steering, and Section 2.4 illustrates how element spacing in an array alters the radiation pattern. Then, Section 2.5 describes the various figures of merit used in this work to characterize phased array performance. Section 2.6 details the measurement complexity of this characterization and the measurement approaches utilized in experiments. Finally, Section 2.7 delineates the optimization algorithm used by the arrays to steer beams.

2.1 Coordinate System

The coordinate system shown in Figure 2.1 is used throughout this work unless otherwise specified. The z-axis is defined to be *broadside* to the relevant antenna or array. Because all of the presented antennas are linearly polarized, the x-axis is defined to be in the polarization direction. Thus, the E-plane is equal to the xz-plane, the H-plane is equal to the yz-plane, and a planar array or antenna, like a patch, will lie in the xy-plane. References to the x- and y- planes are referring to the xz- and yz- planes and are thus equivalent to the E- and H- planes, respectively.

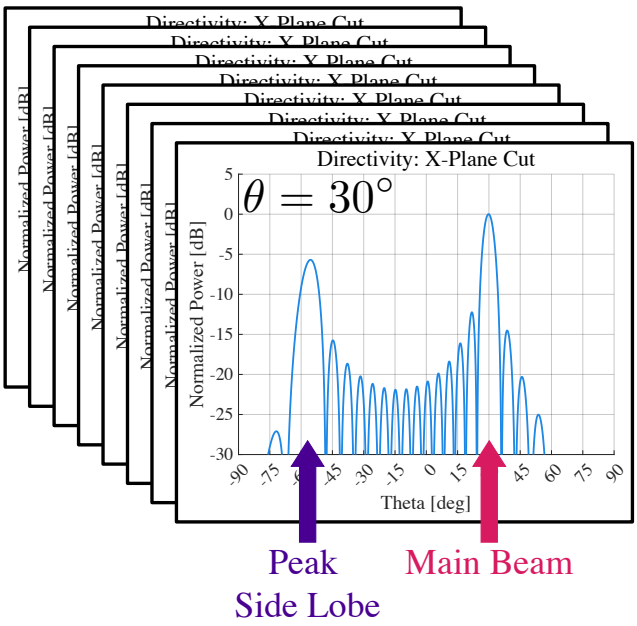


Figure 2.1: Coordinate System used throughout this work. $\hat{z}$ is normal to the array surface and $\hat{x}$ is aligned with the array polarization.

Spherical coordinates are defined by this Cartesian coordinate system, with θ being the angle with respect to the z-axis and ϕ being the angle in the xy-plane with respect to the x-axis. Thus, broadside is defined to be at $\theta = 0°$ and end-fire is defined to be at $\theta = 90°$. The E-plane is defined to be at $\phi = 0°$ and $180°$ and the H-plane is at $\phi = 90°$ and $270°$.

2.2 Antenna Properties

Antennas are transducers that convert voltages and currents at their input ports into electromagnetic fields that travel through free space and vice-versa. Excitations at their input ports induce currents along the surface of the antenna that in turn generate electromagnetic fields. A portion of these fields remain localized to the antenna like the fields of an inductor, capacitor, or transformer. The fields in this *near-field* region couple back into the port, altering its reactance. Another portion of the induced fields decouple from the antenna and radiate. These fields form traveling waves that carry power into the *far-field*, waves than can be used for communication, sensing, or power transfer. Antennas are the intermediary between electronics and free-space electromagnetics and thus their performance is critical for wireless communication, radar, wireless power transfer, and remote sensing.

There are numerous criteria that can be used to evaluate antenna performance, from

their port behavior, to their sensitivity to different polarizations, to their ability to steer radiation in particular directions. Because the research in this thesis is predominately concerned with latter, we introduce a non-exhaustive list of antenna properties and commonly used metrics that characterize the radiative properties of an antenna.

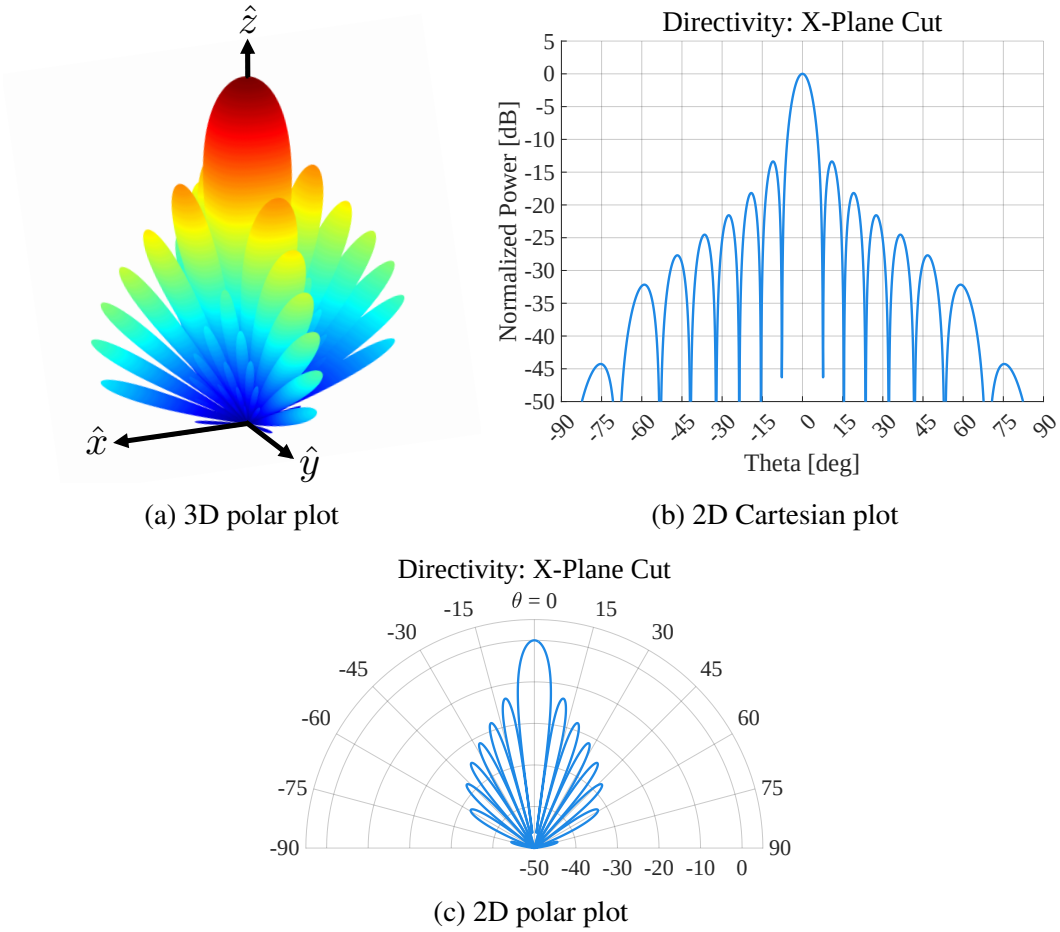
Reciprocity

A critical property of antennas is that they are *reciprocal*. Due to the linearity of Maxwell's equations and the reciprocity of both free space and commonly used materials, the behavior of an antenna does not depend on the direction that an electromagnetic wave is traveling. Therefore, the properties of an antenna are identical if it detecting incoming waves as a receiver, or shaping radiating waves as a transmitter. An antenna that effectively radiates in a particular direction will also receive effectively from that same direction. This fundamental property is useful for both conceptualization and measurement as the antenna can be treated as either a receiver or transmitter depending on which is more convenient.

Radiation Pattern

The *Radiation Pattern* of an antenna is a plot of the amount of power radiated in each direction in spherical coordinates, $P(\theta, \phi)$. It provides a comprehensive visualization of where power *is* and *is not* directed, demonstrating the array's ability to focus power in desired and undesired directions. The radiation pattern is the single most important property of an antenna as it defines its functionality. Some applications require broad patterns, such as a radio transmitter, while others require highly focused patterns, like a radio-telescope. Many of the metrics described below quantify various properties of this pattern.

There are multiple common ways of displaying radiation patterns, from a complete 3D spherical plot, to radial or rectangular plots along one axis, or *cut*. It is common to show cuts in the $E-$ and $H-$ planes, the two orthogonal planes that align with the $E-$ and $H-$ fields on the surface of a linearly polarized antenna. The radiated power can be presented in terms of raw or normalized power. Common normalization factors are the peak radiated power, the power injected into the port (see Gain below), or the average radiated power (see Directivity below). Figure 2.2 demonstrates different plots of the radiation pattern using the same data as an example.



(a) 3D polar plot

(b) 2D Cartesian plot

(c) 2D polar plot

Figure 2.2: Different visualizations of the same radiation pattern.

Lobes and Nulls

As can be seen in Figure 2.2, it is common for the radiation pattern of an antenna to be comprised of *lobes* and *nulls*, that is, regions of alternating high and low radiation due to the interference of fields radiated at different points on the antenna surface. Nulls are regions of destructive interference where little power is radiated and can clearly be seen at $\theta = \pm 8°, \pm 15°$, and at other angles in Figure 2.2b. These nulls separate lobes, regions of constructive interference where radiated power is concentrated. It is common for a radiation pattern to have a single *Main Lobe* that concentrates power in the desired direction, such as towards a target, and multiple *Side Lobes* that radiate power in undesired directions, such as away from the target. Because antennas are reciprocal, lobes indicate directions from which a receiving antenna will strongly pick up a signal while nulls indicate directions of greatly reduced sensitivity.

Free Space Path Loss

As the distance from an antenna increases, the radiated power is spread out over a larger area. Consider the power density a distance R from an isotropic antenna as an example. While the same total power is evenly distributed in each direction, the total power is divided over a larger spherical area, $4\pi R^2$. Thus the power density of radiated fields drops as a function of R^2. Intuitively, a receiving antenna at this distance will collect less power given the lower power density. This loss in power due to the propagation distance, d, between transmitting and receiving isotropic antennas is referred to as *free space path loss* and is given in Equation 2.1.

$$\text{FSPL} = \left(\frac{4\pi d}{\lambda}\right)^2 \tag{2.1}$$

Directivity

Given the effect of free space path loss, it it often desirable to *focus* the power radiated by an antenna in a given direction. Due to reciprocity, this focusing is equivalent to increasing both a receiving antenna's angular selectivity and the amount of power received from a given angle.

Directivity, D, is a measure of an antenna's ability to focus its radiated power. It is defined as the power radiated in a particular direction, $P(\theta, \phi)$, by the antenna relative to that of an isotropic antenna with the same total radiated power [1].

$$D(\theta, \phi) = \frac{P(\theta, \phi)}{P_{isotropic,\, total}(\theta, \phi)} \tag{2.2}$$

Because an isotropic radiator radiates equally in all directions, its power in any direction is the total radiated power averaged by the surface area of the unit sphere.

$$P_{isotropic,\, total}(\theta, \phi) = \frac{P_{radiated}}{4\pi} \tag{2.3}$$

Therefore, the directivity is proportional to the fraction of the total radiated power radiated in a given direction.

$$D(\theta, \phi) = 4\pi \frac{P(\theta, \phi)}{P_{radiated}} \tag{2.4}$$

Alternatively, the definition of directivity can be rewritten entirely in terms of the radiation pattern.

$$D(\theta, \phi) = \frac{4\pi P(\theta, \phi)}{\oiint P(\theta, \phi) d\theta d\phi} \tag{2.5}$$

Radiation Efficiency

Unfortunately, not all power injected into the ports of an antenna will radiate. The currents that generate the electromagnetic fields travel through conductors with non-zero resistance. In addition, the dielectrics used to separate conductors and guide fields have non-zero loss tangent. Therefore, some amount of power is dissipated as heat in both the conductors and the dielectric. This power loss detracts from the total amount of power that can be radiated, reducing the antenna gain. For some antennas, this loss can be quite high as fields are highly concentrated and large currents are induced.

The radiation efficiency, η, is the ratio of the total radiated power to the power injected into the port.

$$\eta = \frac{P_{radiateed}}{P_{port}} \tag{2.6}$$

Gain

The *Gain* of an antenna, $G(\theta, \phi)$, is a measure of an antennas ability to focus power in terms of *the power injected into its port*. Thus, Gain incorporates the both the directivity and radiation efficiency into a single metric. The complete of definition of Gain is the ratio of the radiated power in a given direction to that of an ideal isotropic antenna driven with the same power, P_{port} [1].

$$G(\theta, \phi) = \frac{P(\theta, \phi)}{P_{isotropic,\, port}(\theta, \phi)} \tag{2.7}$$

Because the isotropic antenna is lossless, all of the injected power is radiated and so the power radiated in any direction is the injected power averaged by the surface area of the unit sphere.

$$P_{isotropic,\, port}(\theta, \phi) = \frac{P_{port}}{4\pi} \tag{2.8}$$

Therefore, the gain is proportional to the fraction of *injected power* radiated in a given direction.

$$G(\theta, \phi) = 4\pi \frac{P(\theta, \phi)}{P_{port}} \tag{2.9}$$

Using Equations 2.4 and 2.6, the Gain can be simply expressed in terms of the antenna efficiency and directivity.

$$G(\theta, \phi) = \eta D(\theta, \phi) \tag{2.10}$$

Antenna Aperture

Antenna aperture, A_e, is defined as the power delivered by the antenna to a matched load, P_{port}, when irridated by an incident plane wave of uniform power density, S_{inc} [1].

$$A_e = \frac{P_{port}}{S_{inc}} \tag{2.11}$$

Note that while antenna aperture has units of square meters, it does not necessarily correspond to any *physical* area. Rather it is strictly defined by the electromagnetic properties of the antenna.

That said, a useful and accurate analogy is to think of the antenna aperture as a measure of how *electromagnetically large* the antenna is. For some antennas, such as a dipole or a Yagi-Uda, the aperture does not correspond to any physical dimension. For other antennas, like a horn or a parabolic dish, the aperture is closely related to the physical area of the opening or reflector. In light of this loose relation, A_e is commonly referred to as the *effective aperture* to distinguish it from any physical aperture.

Another important note is that the power used to define aperture in Equation 2.11 is dependent on the angle of incidence and polarization of the incident wave. Thus a more accurate definition of antenna aperture can be written as:

$$A_{e,\hat{p}}(\theta, \phi) = \frac{P_{port}}{S_{inc,\hat{p}}(\theta, \phi)}. \tag{2.12}$$

Whenever the angle of incidence or polarization is not specified, it is assumed that they are in the direction of maximum power absorption. In other words:

$$A_e = \max_{\{\hat{p},\theta,\phi\}} A_{e,\hat{p}}(\theta, \phi). \tag{2.13}$$

Relationship between Gain and Aperture

It turns out that the antenna gain and effective aperture are closely related. Intuitively as the aperture increases, a lossless antenna will receive more power from a given direction than a fixed isotropic antenna. Therefore, an increase in aperture should also increase both the antenna gain and directivity. The relationship between antenna gain and aperture is shown in Equation 2.14 [2].

$$G = 4\pi\eta\frac{A_e}{\lambda^2} \tag{2.14}$$

As expected, the gain and aperture are linearly related. The $\frac{4\pi}{\lambda^2}$ scaling factor accounts for the diffraction of the radiation related to the size of the aperture [3].

EIRP

When measuring an antenna, the absolute power measured is highly dependent on the gain of the measurement probe and the distance between the probe and the antenna under test due to the free space path loss. Thus the raw measured power is a relatively meaningless concept during characterization.

The Effective Isotropic Radiated Power, or EIRP, is a more useful measure of the radiated power. EIRP is the amount of total power that a theoretical isotropic antenna must radiate in order to achieve the same measured power density. It is not sensitive to measurement distance or probe gain as they are deembeded from the absolute power measurement.

$$\text{EIRP}_{dB} = P_{measured,dB} - G_{probe,dB} + \text{FSPL}_{dB} \tag{2.15}$$

Therefore, the EIRP is only dependent on the gain and the power injected into the antenna port.

$$\text{EIRP}_{dB} = G_{AUT,dB} + P_{port,dB} \tag{2.16}$$

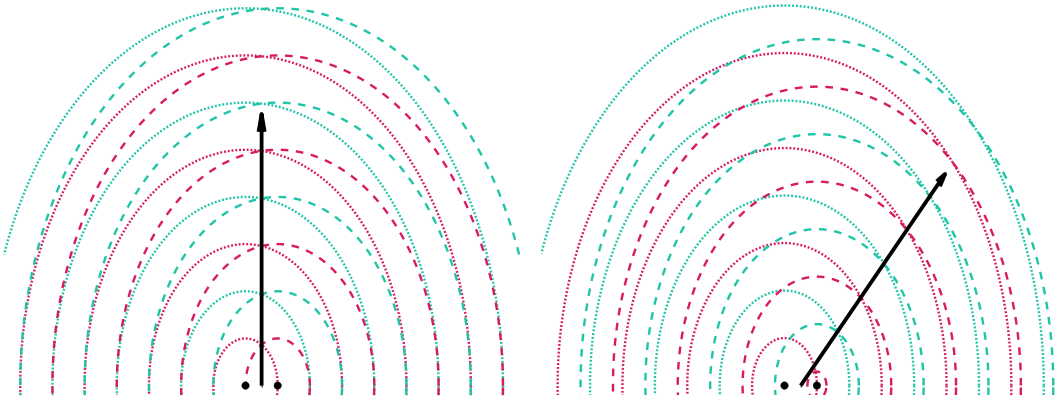
2.3 Overview of Phased Arrays

Phased arrays use multiple antennas in concert in order to improve performance. Generally speaking if an antenna is duplicated along a surface and the antenna output ports are tied together, the total power received by an incident wave increases and thus the antenna effective apertures combine to form a larger array aperture. From the perspective of a transmitting array, the fields radiated by each element interfere with each other and focus the radiated power along the center axis as seen in Figure 2.3a. This focusing increases the gain on the entire array.

However, the increase in aperture is not the most powerful benefit of using an array. By changing the phases of the fields radiated by each element in the array, the angular location at which power concentrates *changes* as shown in Figure 2.3b. As phase shifting can easily be accomplished electronically, a *phased array* can electronically alter the location of peak gain, a process commonly referred to as "steering a beam."

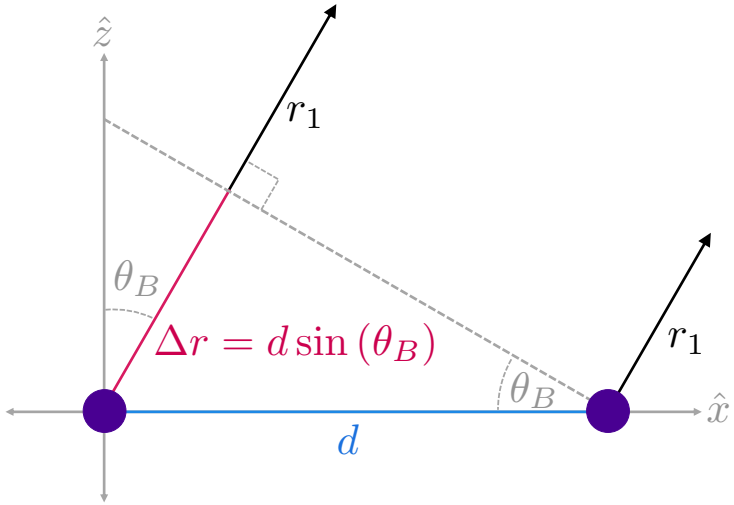
The locations of the constructive interference can be calculated analytically. Supposed that two elements spaced d meters apart are driven with a phase difference of $\Delta\phi$. For the waves radiated by each element to to interfere constructively, they must create a wavefront with equal phase. For a beam to form at angle θ_B, radiation from the further element must travel an additional

$$\Delta r = d\cos\left(\frac{\pi}{2} - \theta_B\right) \tag{2.17}$$



(a) Wavefronts of $\frac{\lambda}{2}$ spaced array. $\Delta\phi = 0°$.

(b) Wavefronts of $\frac{\lambda}{2}$ spaced array. $\Delta\phi = 127°$.



(c) The path length difference between antennas.

Figure 2.3: Demonstration of beam steering with a two element phased array. (a) and (b) show the constructive and destructive interference pattern of wave-fronts with 0° phase (red), and wave-fronts with 180° phase (green). (c) illustrates the geometric relationship between beam angle and path length difference.

meters as seen in Figure 2.3c. In terms of phase, this distance is equivalent to

$$\Delta r_{rad} = \frac{2\pi}{\lambda} d \sin(\theta_B) \tag{2.18}$$

radians. Thus in order to create constructive interference at θ_B, the phase of the closer element must lag that of the further element by this same amount and so the phase difference between the elements is:

$$\Delta\phi = \frac{2\pi}{\lambda} d \sin(\theta_B)\,. \tag{2.19}$$

Though derived for a two-element array, the same analysis holds for any N element linear array with constant spacing and phase shift between elements. Equation 2.19

can be written to solve for the location of the main beam given the element spacing, wavelength, and the phase difference between neighboring elements.

$$\theta_B = \arcsin\left(\frac{\lambda}{2\pi d}\Delta\phi\right) \tag{2.20}$$

2.4 Effects of Spacing on Radiation Pattern

Despite their benefits, there are limits to the capabilities of phased arrays. For the purposes of this dissertation, the most important factor limiting phased array performance is the *element spacing*, the separation between the centers of antennas within the array. Element spacing alters the radiation pattern of the array by introducing additional side lobes and the creation of *grating lobes*.

In an array, the fields radiated by every antenna constructively interfere with each other to concentrate the power in the main lobe. However for arrays with $N > 2$ elements, at some angles *most* of the fields interfere constructively while *some* interfere destructively. Power will still radiate in this direction, albeit less than in the main lobe direction. The locations of partial constructive interference thus introduce new side lobes into the radiation pattern. The properties of these side lobes are highly dependent on the locations of antennas in the array.

More critical than side lobes, however, is the introduction of *grating lobes*. If the elements in the array are spaced far enough apart, than there are *multiple* locations at which the fields from every antenna combine constructively. Thus power will be radiated equally[1] in both the main lobe and the grating lobe directions. The locations of these undesired peaks are equivalent to the diffraction pattern of radiation passing through a grating with the same spacing, hence the name "grating lobes."

Analytically grating lobes occur because waves will combine constructively if the phase difference is an integer multiple of 2π. Thus Equation 2.19 can be more accurately written as:

$$\frac{2\pi}{\lambda}d\sin(\theta_G) + 2\pi n \quad \forall n \in \mathbb{Z} \tag{2.21}$$

where θ_G is the angular location of a grating lobe[2]. Thus the locations of the grating lobes are given in Equation 2.22 [2].

$$\theta_G = \arcsin\left[\frac{\lambda}{2\pi d}\left(\Delta\phi + 2\pi n\right)\right] \quad \forall n \in \mathbb{Z} \tag{2.22}$$

[1] If the elements are isotropic.

[2] Or main-lobe for $n = 0$.

As can be seen from Equation 2.22, grating lobes only exist when the argument of the arcsin is between -1 and 1. Thus using 2.20 we can find identify the criteria for the existence of grating lobes when steering a beam to θ_B.

$$\left|\sin(\theta_B) + \frac{n\lambda}{d}\right| \leq 1 \quad \forall n \in \mathbb{Z} \tag{2.23}$$

The worst case scenario is when $\sin\theta_B = \pm 1$, or in other words, when $\theta_B = \pm 90°$.

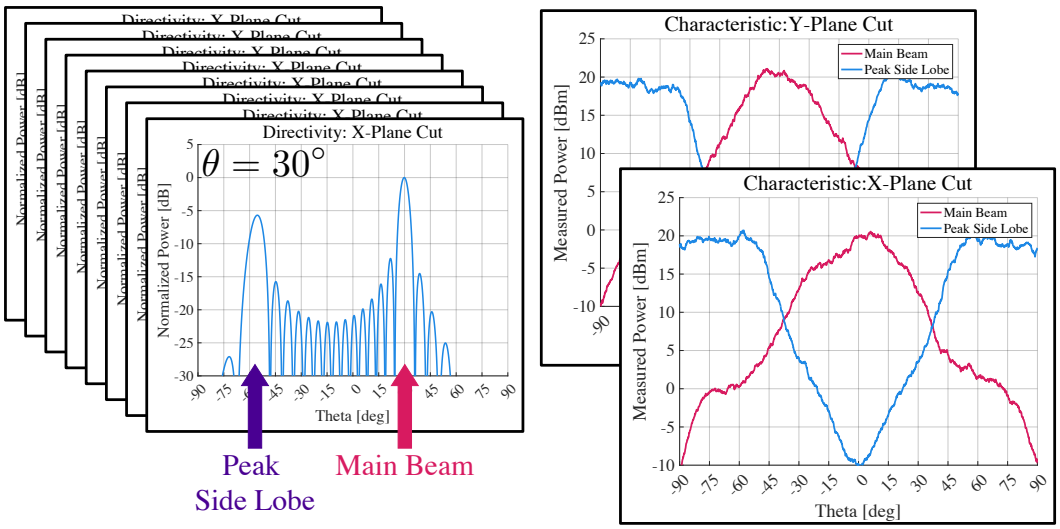
$$-1 + \frac{n\lambda}{d} \leq 1 \quad n \geq 1, n \in \mathbb{Z} \tag{2.24}$$

$$d \geq n\frac{\lambda}{2} \quad n \geq 1, n \in \mathbb{Z} \tag{2.25}$$

Equation 2.25 indicates the conditions under which grating lobes exist in addition to the number of grating lobes. Grating lobes are not possible if the spacing is less than $\frac{\lambda}{2}$. However, close examination of 2.23 indicates that in the case of $d = \frac{\lambda}{2}$, grating lobes only exist when the beam is steered to 90°, at which point the grating lobe is located at −90°. However many of the antennas commonly used in arrays, such as patch antennas, do not radiate at end-fire suppressing the −90° grating lobes. For this reason, arrays with half wavelength spacing are very common as they do not exhibit grating lobes.

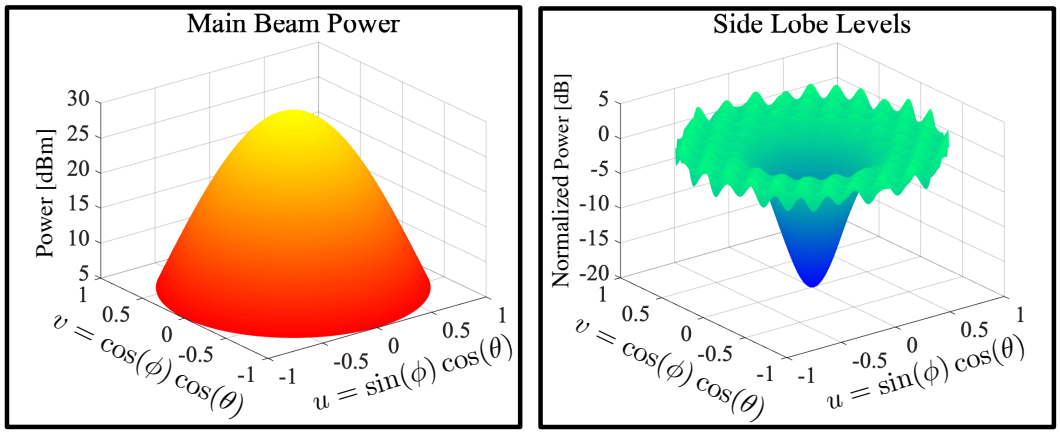
2.5 Phased Array Properties

The metrics given in Section 2.2 are measures of antenna properties. While these properties still hold for arrays, combining antennas in this way introduce new properties that need to be accounted for. Unlike antennas, phased arrays are capable of altering their radiation pattern electronically. However, the design of the array determines the resulting radiation patterns at different steering angles. Therefore, an ideal characterizations of phased array performance accounts for the behavior at *every* steering angle. Figure 2.4 illustrates how a phased array is completely characterized. Beams are steered and measured across each ϕ cut. A measure of interest, such as the main beam power or the peak side lobe power, are quantified for each beam angle. These measures can then be plotted across the steering range for each cut, as shown in Figure 2.4b, or as a 3D plot as shown in Figure 2.4c. Characterization values are only plotted along two axes in this work to reduce the number of measurements required. The following is a non-exhaustive list of measures that are used throughout this work.



(a) Beam radiation patterns.

(b) Array characteristics along different cuts.



(c) Array characteristic over full the steering range.

Figure 2.4: Complete characterization of array performance. (a) Radiation patterns are measured for each beam angle along a cut. Ideally, a full 3D pattern measurement is taken for each beam. (b) Data is extracted from beam patterns to calculate the characteristic along each cut. (c) Cut characteristics are combined to characterize array performance throughout steering range.

Main Beam Power

One measure of phased array performance is how much power is radiated by the beam throughout the steering range. This *Main Beam Power* (MBP) can be reported in terms of the maximum EIRP, Gain, or Directivity at each steering angle. Plots of the MBP thus indicate how the Gain of the array changes as beams are steered.

In general, the main beam power is limited by the radiation pattern of each individual element. For example, if each antenna in a planar array has a null at $\theta = 45°$, than the array itself cannot radiate at $\theta = 45°$. However, the main beam power is not

strictly equivalent to the element pattern as the coupling between elements in an array alters their respective patterns.

Main beam power is loosely connected to the side and grating lobes of the array as both represent radiated power that is not concentrated in the main beam. An interesting demonstration of this relationship is the case of an array with very large element spacing. As such a sparse array covers a wide area, it would *appear* to have a large effective aperture. Indeed, the main beam is highly concentrated due to the narrow interference pattern. However, the array also has multiple grating lobes due to the large spacing. These grating lobes redirect power away from the main lobe proportionate to the concentration of power within the main lobe. Thus the total gain, effective aperture, and main beam power do not increase. This phenomenon of trading beam width for grating lobes is known as "the sparse array curse."

–3 dB Steering Range

For most arrays, the main beam power changes throughout the steering range. The main beam power is maximum at some point, typically broadside, and begins to decrease as beams are steered away from the point. The -3 dB steering range is the portion of the steering range where the main beam power is no less than -3 dB below the maximum and thus the beam will radiate at least half as much power as the peak.

In general, the -3 dB steering range is a 2D region of (θ, ϕ) coordinates. However, it can also be specified by the difference in this region's boundaries through a given cut. For example, an elliptic -3 dB steering range with a major axis of 150° along the $x-$axis and a minor axis of 90° along the $y-$axis can be described as having a -3 dB steering range of 150° in the $x-$plane and 90° in the $y-$plane. While this cut based description is not complete, it is simplier to measure.

Side Lobe Level

An array's side and grating lobes have a large impact on antenna performance. Power radiated in these undesired lobes detracts from that in the main lobe and can interfere with other arrays. For a receiving array, these lobes increase the signal to noise ratio and enable blockers to jam the array output.

While grating and side lobes can have very complicated patterns, a simple measure of their effect is the *Side Lobe Level*, the relative strength of the peak side lobe with respect to the main beam power. It is common for the peak side lobe to be the largest

concern to antenna performance as the other side lobes have much lower magnitude. It is convenient to normalize the peak side lobe power by the main lobe power as the main concern of side lobes is the resulting degradation of gain and not the absolute power radiated in the side lobe. The absolute power radiated in the peak side lobe can be easily be calculated using the side lobe levels and the main beam EIRP.

In this work we take a broad definition of side lobe levels that includes grating lobes as they degrade array performance in the same manner as side lobes. In addition, in this work the main lobe and grating lobes are labeled by their intended function and not by their relative performance. It is thus possible for the side lobe levels to be *positive*, indicating that more power is radiated in the grating lobe than in the main lobe.

Field of View

When the power in the grating lobe becomes equal to that in the main beam, the labeling of the two directions becomes arbitrary. Due to spatial aliasing, the phase settings required to focus a beam in one direction is the same as that required to focus a beam in the other. Therefore, the main lobe can always be viewed as a grating lobe of a beam steered in a different direction.

If the main lobe is defined to be the lobe with more power, than it is impossible *by definition* to steer a beam beyond the point where the power in the main lobe and in the grating lobe are equal. Past that point, what was the grating lobe will have more power and so the labeling of the two beams will switch. There is thus a maximum range, referred to as the *Field of View*, over which beams can be steered.

As discussed above, lobes are labeled in this work by their intended function. It is far from *impossible* to direct beams outside the field of view in the sense that power can be concentrated at wide angles. As the presented research is often concerned with the behavior of beams steered at wide angles, it makes semantic sense to continue to refer to these beams as "the main lobe." We take responsibility for remembering that the performance of beams outside the field of view is generally atrocious.

The one important ramification of the labeling scheme is that the definition of field of view must be modified accordingly. In this case, the field of view is defined to be the angular region over which the side lobe level is negative. The resulting field of view is identical to that of the previous definition; the power in the main lobe is larger than that in the grating lobe.

2.6 Phased Array Characterization

Fully characterizing a phased array is an intensive measurement as the aggregate behavior of the array is determined by the properties of its comprising beams. As each beam direction is effectively a different antenna, a full two dimensional scan of the radiation pattern is required *for each beam* in order to measure how the side and grating lobes change as beams are steered. Therefore *completely* characterizing a phased array requires a large number of measurements.

Measurement Density

Supposed that the steering range is divided into $N_{b,\phi}$ cuts along the ϕ axis, each of which is sampled with $N_{b,\theta}$ beams spaced along the θ axis. Thus a total of $N_b = N_{b,\theta} \times N_{b,\phi}$ beams are measured to characterize the array. Each of these beams must itself be measured by sampling different points in the far-field[3]. Suppose the far-field is divided into $N_{m,\phi}$ cuts along the ϕ axis, each of which is measured at $N_{m,\theta}$ points along the θ axis. In total, a single beam requires $N_m = N_{m,\theta} \times N_{m,\phi}$ measurement points to characterize.

It is thus apparent that $N_{measurements} = N_b \times N_m$ measurements must be taken in order to characterize a phased array. To appreciate the scale of this task, consider characterizing a phased array in one hemisphere with 5° precision. In this case, $-90 \leq \theta \leq 90$ and $0 \leq \phi \leq 180$, each requiring 37 sample points. Therefore, $N_{b,\theta} = N_{m,\theta} = N_{b,\phi} = N_{m,\phi} = 37$ and 1,874,161 measurements must be taken.

For most phased array measurements, a single array is characterized. In the experiments in this dissertation however, multiple version of the array are measured and compared; different array geometries are compared in Chapter 7, while different meta-gap settings are characterized in Chapters 9 and 10. Due to the complexity of characterizing a *single* array, an efficient measurement approach is critical to characterizing *multiple* arrays in reasonable time. Thus the speed of two measurement methods are explored.

In typical radiation pattern measurements, either the antenna under test or the probe has to physically move between measurement points. The average time for such mechanical movement, T_{move}, is on the order of seconds. On the other hand, electronic operations like beam steering and field measurement can theoretically take less than a microsecond. Clearly the number of times each operation is performed will have a major impact on measurment speed. Therefore, in order to understand

[3]Near-field measurements can also be used, but these require the same number of sample points.

and optimize the speed of the measurement method the total time is analyzed in terms of the base operations.

```
def SequentialArrayCharacterization()
    for configuration in array_configurations:
        array.changeConfiguration(configuration)
        for beam_angle in scan_range:
            measurementRange.moveToAngle(beam_angle)
            array.optimizeBeam()
            for angle in measurement_space:
                measurementRange.moveToAngle(angle)
                measurement = measurementRange.measure()
```

Code Segment 2.1: Sequential Array Characterization

Sequential Measurement Approach

The most straightforward approach to measuring multiple phased arrays is to measure them sequentially. Code Segment 2.1 contains pseudo-code describing the measurement sequence; for each array the beam is scanned across the steering range and the entire measurement space is measured for each beam. Beams are steered using the beam optimization algorithm presented in Section 2.7.

Assume that the time to complete a measurement is T_{meas} and that the time to change array configurations is T_{change}. The time required to optimize each beam, T_{opt}, is fully discussed in Section 2.7 and given by Equation 2.38. Given these times and the measurement density, the time to measure each beam pattern is

$$T_{measure,\,pattern} = N_m \left(T_{move} + T_{meas}\right). \tag{2.26}$$

Thus the time to characterize a single array configuration is

$$T_{measure,\,array} = T_{change} + N_b \left[T_{move} + T_{opt} + T_{measure,\,pattern}\right]. \tag{2.27}$$

Combining these expressions with Equation 2.38 and rearranging gives an expression of the time required to measure M array configurations

$$T_{seq} = MT_{change} + MN_b \left[N_m + 1\right] T_{move} + MN_b \left[N_m + D\left(N - 1\right)\right] T_{meas} \tag{2.28}$$

where N is the number of tiles in the array and D is the optimization depth. Assuming $D\left(N-1\right)T_{meas} \ll T_{move}$, the time to characterize M phased array configurations sequentially is

$$T_{seq} \approx MT_{change} + MN_b(N_m + 1)T_{move}. \tag{2.29}$$

The rightmost term in Equation 2.29 is the time required to move between the measurement points for M configurations. For context, if T_{move} is one second, then it takes 90 minutes to characterize *a single configuration* along the E- and H-plane cuts with 5° precision. Clearly measuring one configuration after another is untenable for a large number of configurations.

```
def BatchedArrayCharacterization()
    # Optimization Phase
    for beam_angle in scan_range:
        measurementRange.moveToAngle(beam_angle)
        for configuration in array_configurations:
            array.changeConfiguration(configuration)
            phases = array.optimizeBeam()
            memory.saveBeamSettings(phases)

    # Measurment Phase
    for angle in measurement_space:
        measurementRange.moveToAngle(angle)
        for configuration in array_configurations:
            array.changeConfiguration(configuration)
            for beam_angle in scan_range:
                phases = memory.getBeamSettings(beam_angle)
                array.programBeam(phases)
                measurement = measurementRange.measure()
```

Code Segment 2.2: Batched Array Characterization

Batched Measurement Approach

The sequential approach is inefficient because it requires the range to repeatedly move between the same measurement points. However, if the configuration can be changed in the *midst* of the measurement than multiple configurations can be measured in parallel by iterating through them at each measurement point. Thus a large number of configurations can be characterized while only moving through measurement points once.

This intuition is the basis for the batched measurement method detailed by the psuedo-code in Code Segment 2.2. In a sense, the sequential measurement loop is "unwrapped" with the configurations changing for each measurement point instead of the measurement points changing for each configuration. In addition, beams are not measured immediately after optimization so that measurement points do not need to be revisited for multiple beams. Thus the measurement is broken into two phases, the optimization and the measurement phase.

During the first phase, the range moves between beam directions so that the phase settings required to steer the beam in that direction can be optimized. For each beam location, the array shifts between configurations and the beam is optimized. The resulting phase settings are stored for the measurement phase. In the second phase, the range moves between measurement points. At each point, the array shifts between configurations and beams using stored phase settings. The measurement point is thus measured for every combination of beam angle and array configuration.

Assume that the times to store and recall the phase settings and the time to program the beam are negligible when compared to the measurement time. In this case, the time to optimize the beam patterns for M configurations is

$$T_{phase,\,optimization} = N_b \left[T_{move} + M \left(T_{change} + T_{opt} \right) \right] \tag{2.30}$$

and the time to measure each of the beam patterns for M configurations is

$$T_{phase,\,measure} = N_m \left[T_{move} + M \left(T_{change} + N_b T_{meas} \right) \right]. \tag{2.31}$$

The sum of these expressions is the time required to characterize M array configurations. Thus using Equation 2.38 and rearranging gives Equation 2.32.

$$\begin{aligned} T_{batch} = M \left(N_b + N_m \right) T_{change} + \left(N_b + N_m \right) T_{move} \\ + M N_b \left[N_m + D \left(N - 1 \right) \right] T_{meas} \end{aligned} \tag{2.32}$$

It is clear that for a large enough number of configurations, the measurement time will be dominated by the field measurement time and the time to change configurations.

Comparison of Methods

Comparing Equation 2.32 to Equation 2.28 highlights the time trade-off made by unwrapping the loop. While the total time spent measuring fields (the third term in both expressions) is unchanged, the total movement time has decreased substantially and the total time spent changing between configurations has increased by a factor of $(N_b + N_m)$. Therefore, the relative speed of the two methods depends on the relative time to change between states.

For the the shape-changing phased array measurements in Chapter 7, the array geometry has to be manually reconfigured and reinforced with additional supports; a process that can take several minutes. Therefore, the sequential method is far superior despite the long movement time. In the meta-gap experiments in Chapters 9 and 10, however, the array configuration can be changed electronically in a few

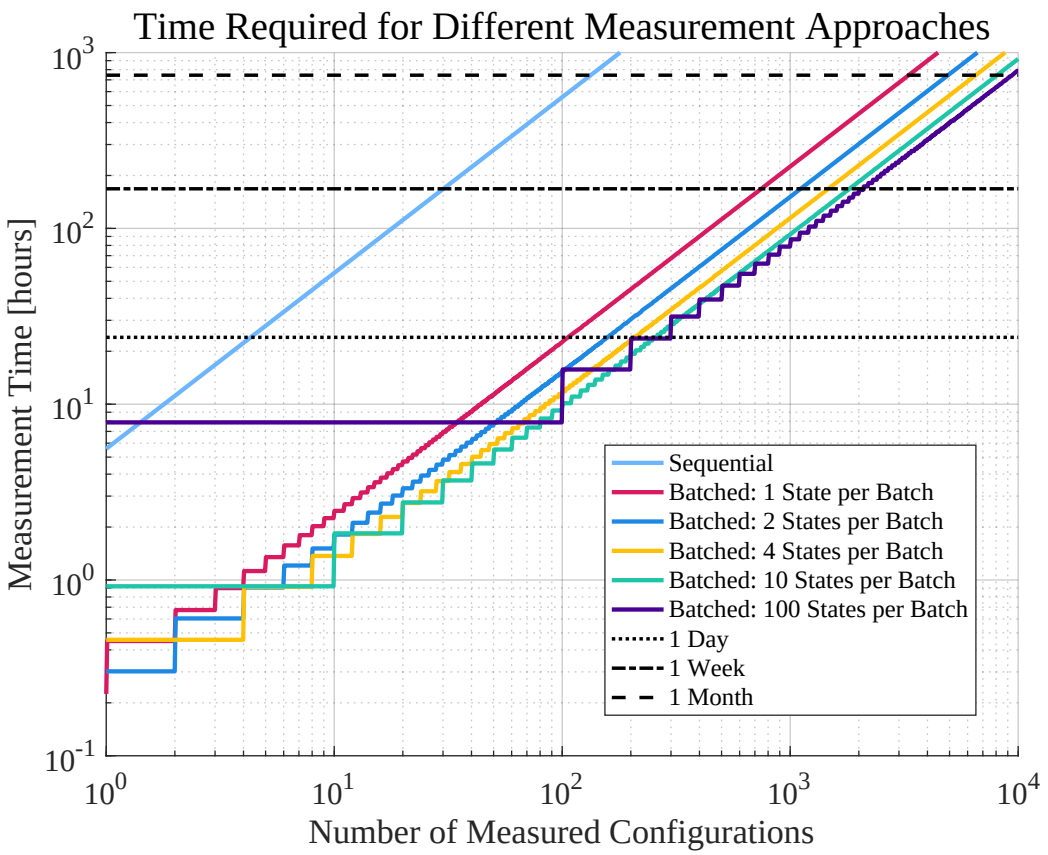


Figure 2.5: Time required to measure M array configurations using different characterization methods. Curves are based on execution times presented in Section 10.2 for meta-gap characterization. Larger batch sizes reduces measurement time until $\approx$ 100 states per batch, at which point the RF measurement time dominates.

milliseconds. Thus batched measurements provide a substantial speed improvement *critical* to the experiments.

One important downside to the batched method is that measurements for each configuration are only known at when the measurements for *all* of the configurations are complete. Therefore if the array measurements are to be used as part of an optimization, as they are in the meta-gap experiments, decisions about how to alter the configuration can only be made after the measurement is complete. Thus in an optimization context, it is advantageous to measure configurations in multiple *batches*[4] in order to get both the speed improvement of the batched method and the sequential method's ability to change the configurations to be measured on the fly. The measurement time of this method of sequential batches is simply Equation 2.32 times the number of batches.

[4]Hence the name of the method.

Figure 2.5 plots the time required by the two methods to measure M configurations with 5° precision from −90° to 90° in the $E-$ and $H-$ cuts. The total measurement time is calculated using the function execution times reported in Section 10.2. Multiple batch sizes are shown for the batch algorithm to show how the performance scales. As can be seen, increasing the batch size increases measurement speed. Around 100 states per batch however, the improvement ceases as the field measurement time dominates. It is clear that the total batch characterization time is enhanced by approximately 2 orders of magnitude when using the batched method. In fact, even one state per batch offers drastic improvement as multiple beams are measured at a single measurement location. That said measurements are time intensive even with the speed improvements; it takes 24 hours to measure 300 configurations.

2.7 Beam Optimization

In a phased array, each element radiates a signal on a fixed carrier frequency. Due to the linearity of Maxwell's equations in free space, the total field strength at any point is approximately the sum of sinusoids originating from each element. In the tile architectures used in this work, each sinusoid's phase is dependent on the propagation distance, the tile phase shift, and the path length between the tile and the shared reference source. The amplitude is dependent on the power radiated by each tile and the propagation distance. Thus the power radiated in each direction is proportional to a summation of time-shifted phasors with programmable phase and amplitude.

$$P(\theta, \phi) \propto \mathrm{Re}\left[\sum_{i=1}^{N} A_i e^{j(\Delta\Phi_i + \psi_i)}\right] \tag{2.33}$$

Equation 2.33 illustrates this phasor sum, with $\Delta\Phi_i$ being the unknown phase delay due to propagation and reference path length and A_o and ψ_i being the programmable amplitude and phase shift of each tile. Note that $\Delta\Phi_i$ is a function of array geometry and the angle of radiation (θ, ϕ).

To form a beam, the phases of the array must be aligned so that each element of the phasor sum adds constructively, i.e., has the same phase. While $\Delta\Phi_i$ is unknown, it is possible to identify the correct ψ_i by optimizing the measured power. To see this, consider a two element array with $\psi_0 = 0$ and $A_i = 1$. As the phase is relative to an arbitrary shared reference, we can assume that $\Delta\Phi_0 = 0$ without loss of generality.

$$P(\theta, \phi) \propto \mathrm{Re}\left[1 + e^{j(\Delta\Phi_1 + \psi_1)}\right] \tag{2.34}$$

$$P(\theta, \phi) \propto 1 + \cos\left(\Delta\Phi_1 + \psi_1\right) \tag{2.35}$$

As seen in Equation 2.35, the power is proportionate to the sum of a constant and a cosine. Thus $\Delta\Phi_1$ can be quickly identified by varying ψ_1 to optimize the measured power at (θ, ϕ), at which point $\phi_1 = -\Delta\Phi_1$.

Therefore, a two step phase approach can rapidly optimize the measured power. First, power is measured at $\psi_i = 0° = 360°$, $\psi_i = 120°$, and $\psi_i = 240°$. The optimal peak must exist between two of these points as they span the range of possible cosine arguments. In addition, these two points are at most 120° away from the peak and the third point is at least 120° away. As cosine is a monotonically decreasing function with respect to deviations less than 180° from its peak, the power of the third point will always be less than the first two. Therefore, the peak must exist between the two phases that radiate the most power.

Once these phases are identified, the peak can rapidly be found using repeated bisection. As cosine is convex between these initial endpoints, the radiated power will always be higher in the center than at the endpoints. In addition, as with the initial measurements the peak must be located between the midpoint and the endpoint with higher power. Thus by repeatedly bisecting the space and rejecting the lower power endpoint, the power can approach the peak with arbitrary precision. If the desired phase error between ϕ_1 and $-\Delta\Phi_1$ is x degrees, than the two element array can be optimized in

$$D = 3 + \log_2\left(\frac{120}{x}\right) \tag{2.36}$$

measurements. D is referred to as the *depth* of the optimization.

This optimization process can be used to optimize larger arrays by repeating it for each tile. Consider that when the two tile optimization process is complete, $\phi_1 = -\Delta\Phi_1$ and so the right hand side of Equation 2.35 is simply a constant. Thus if a third tile is now added, than the first two behave as a single reference tile with twice the radiated power. The phase of this third tile can be optimized using the same optimization approach, at which point the three tiles will behave as a single reference with three times the power. This process can be repeated for all of the tiles.

Thus to steer a beam of an N element array, one element is selected as the reference tile and the other tiles are "turned off" by setting $A_i = 0$ so that they do not radiate. Then each of the $N - 1$ remaining tiles are "turned on" one at a time and optimized using the two element optimization algorithm described above. In the end $\phi_i = -\Delta\Phi_i$ for all tiles and the maximum power is radiated in the direction of the measurement

probe (θ, ϕ). The number of measurements required to perform this optimization process for an N element array is:

$$n_{measurements} = (N-1)\,D = (N-1)\left[3 + \log_2\left(\frac{120}{x}\right)\right]. \tag{2.37}$$

Therefore, the total time required to optimize a beam is given by Equation 2.38.

$$T_{opt} = (N-1)\,DT_{meas} = (N-1)\left[3 + \log_2\left(\frac{120}{x}\right)\right]T_{meas} \tag{2.38}$$

The discussion above assumes that the measurement system is ideal. However, the optimization performance can degrade in the presence of noise. For example, if two of the endpoints or initial phases in the bisection process are close in magnitude, than measurement noise can result in the algorithm incorrectly selecting the wrong region to search. This has the effect of limiting the phase precision that the optimization can achieve. In addition, optimization measurements become more difficult with larger arrays as the change in power due to a single element becomes significantly smaller than the total power of the already optimized tiles. As measuring small differences in a high power signal is often difficult, the optimization performance will degrade. While there are other optimization approaches that are more robust to these challenges [4], the presented approach is sufficient for the array sizes used and optimization precision required in this work.

References

[1] C. Balanis, *Antenna Theory: Analysis and Design*, 4th ed. Wiley, 2016, ISBN: 1118642066.

[2] J. D. Kraus and R. J. Marhefka, *Antennas: For All Applications*, Third. McGraw-Hill, 2002, ISBN: 9780072321036.

[3] J. A. Shaw, "Radiometry and the friis transmission equation," *American Journal of Physics*, vol. 81, pp. 33–37, 1 Jan. 2013, ISSN: 0002-9505. DOI: 10.1119/1.4755780. [Online]. Available: http://aapt.scitation.org/doi/10.1119/1.4755780.

[4] A. Hajimiri, B. Abiri, F. Bohn, M. Gal-Katziri, and M. H. Manohara, "Dynamic focusing of large arrays for wireless power transfer and beyond," *IEEE Journal of Solid-State Circuits*, vol. 56, pp. 2077–2101, 7 Jul. 2021, ISSN: 0018-9200. DOI: 10.1109/JSSC.2020.3036895.

Chapter 3

APERTURE PROJECTION ANALYSIS

For me the difference
between love and hate,
weighs the same difference
between risotto and rice pudding.

Benjamin Clementine
Phantom of Aleppoville

In order to examine the effects of geometry on array performance, an analytic framework must be selected. A commonly used technique, *aperture projection analysis* (APA), is often assumed to hold without proof because of its intuitive nature. This chapter places this analysis technique on firm theoretical ground and reveals the underlying assumptions. Once established, aperture projection analysis is used in Chapter 4 to study the relationship between geometry and gain.

Section 3.1 begins by discussing a well established upper bound on directivity that assumes nothing about the radiating elements. Following this, Section 3.2 introduces the aperture projection analysis technique and Section 3.3 demonstrates how it produces a tighter bound on gain. Section 3.4 begins the theoretical analysis by clearly stating the assumptions on which APA is based. Section 3.5 then shows that the projected cross sectional area of the array is equal to its total effective aperture subject to those assumptions. Finally, Section 3.6 defends the most critical assumption, the cosine bound, via an intuitive conservation of energy argument and a thorough summary of the relevant literature.

3.1 Limits on Directivity

The maximum directivity of antennas has been the subject of extensive research since the 1940s [1]. Twenty years later, analysis of spherical radiating modes identified the relationship between the size of an antenna and its maximum directivity [2], [3]. While research on the exact relationship between quality factor, geometry, surface resistivity, and area continues to this day [4], there is a consensus on the general behavior of the bounds; the maximum directivity is a function of the radius of the

minimum enclosing sphere.

Any geometric object can be enclosed within a sphere of some radius, r_{min}. Regardless of the specific current distribution or locations of elements within the sphere, the minimum enclosing sphere only permits a finite number of radiating spherical modes, each of which is parameterized by a finite number of independent variables. Thus any radiation pattern created by the antenna has a finite number of degrees of freedom that can be exploited to increase the directivity. It turns out that the maximum directivity of the antenna is approximately half the number of degrees of freedom on the minimum enclosing sphere as there are always two orthogonal polarizations with the same directivity. These degrees of freedom, and thus the maximum directivity, can be summarized by the simple hueristic formula given in Equation 3.1.

$$D_{max} = (kr_{min})^2 + 3 \tag{3.1}$$

[5] verifies the accuracy of Equation 3.1 through an analysis counting the number of radiating modes as r_{min} increases and an in-depth paper study of measured antennas summarized by Figure 3.1. Electrically small antennas that are much smaller than a wavelength have at most 6 degrees of freedom and thus have a maximum directivity of 4.8 dBi, the gain of a Huygens source. For very large arrays the number of degrees of freedom grows proportionally to the surface area of the sphere and thus the directivity is approximately bounded by the cross-sectional area of the enclosing sphere. These two asymptotes are labeled in Figure 3.1 as the Huygens Source Bound and Geometric Optics Bound, respectively. Between these two extremes there is a mix of both prominent radiative and reactive modes that are not captured by either asymptote.

It should be noted that Figure 3.1 contains the measured directivities of three super-directive antennas that exceed the hueristic bound. These antennas have significant reactive fields that expand far beyond the minimum enclosing sphere. If the minimum enclosing sphere is expanded to include the maximum possible radius of these fields than the hueristic bound still applies. That being said, the heuristic limit is a practical one that can be exceeded in some scenarios, especially for electrically small antennas [5].

3.2 Aperture Projection Analysis

A common approach for estimating the power absorbed by a large array excited by an incident wave-front is to project the array aperture onto the wave-front plane and

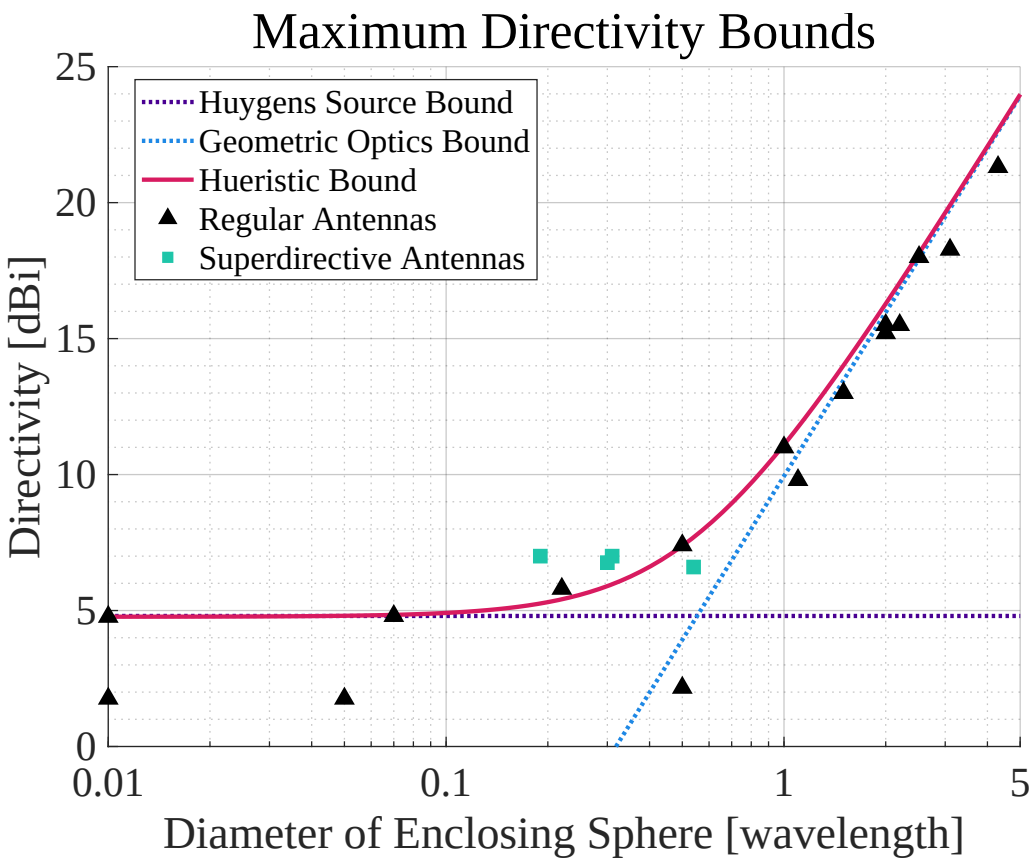


Figure 3.1: The maximum directivity of a radiator is related to the diameter of the smallest sphere that can enclose it. Measured directivities of antennas of all sizes, including superdirective, are compared to a hueristic upper bound. The bound is based on the number of degrees of freedom and known large and small radiator bounds. Bound makes no assumption about geometry or source locations. [5]

multiply by the wave's power density. By assuming that the array's aperture is closely related to its physical area, the projected cross-sectional area of the array provides a quick and intuitive estimate of the array's maximum gain in different directions as shown in Equation 3.2. This approach, which we shall refer to as *aperture projection analysis* or APA, is a powerful tool that can be used to characterize the behaviour of large arrays with different geometries.

$$G_{max}(\theta,\phi) = \frac{4\pi}{\lambda^2} A_{cross}(\theta,\phi) \tag{3.2}$$

3.3 A Tighter Bound on Gain

Aperture projection analysis is useful because it provides a tighter bound on the maximum gain than that given by Equation 3.1. To see this, picture a rectangular array whose side lengths (L and W) can change but its total area ($A = LW$) remains constant. Equation 3.3 predicts that the maximum gain of this array is:

$$G_{max} = \frac{4\pi}{\lambda^2} A. \tag{3.3}$$

The diameter of the minimum enclosing sphere however is equal to to the length of the rectangle's diagonal. Thus Equation 3.1 predicts that the maximum gain is:

$$r_{min} = \frac{\sqrt{L^2 + W^2}}{2} \tag{3.4}$$

$$G_{max} = \left(\frac{2\pi}{\lambda} \frac{\sqrt{L^2 + W^2}}{2} \right)^2 + 3 \tag{3.5}$$

$$G_{max} = \frac{\pi^2}{\lambda^2} \left(L^2 + W^2 \right) + 3 \tag{3.6}$$

$$G_{max} = \frac{4\pi}{\lambda^2} \left[\frac{\pi}{4} \left(\frac{A^2}{L^2} + L^2 \right) \right] + 3. \tag{3.7}$$

If L is very large, then:

$$G_{max} \approx \frac{4\pi}{\lambda^2} \left[\frac{\pi L^2}{4} \right]. \tag{3.8}$$

Comparing Equations 3.3 and 3.8, the maximum gain predicted by aperture projection analysis does not change with increasing L while that predicted by the minimum enclosing sphere grows without bound. Thus, if applicable, APA provides a more accurate and useful estimate. The reason that a lower bound can be established is that aperture projection analysis factors in the locations of elements within the enclosing sphere while the bound given by Equation 3.1 does not.

3.4 Aperture Projection Analysis Assumptions and Limitations

While aperture projection analysis is intuitive and relatively accurate in many scenarios, it is neither universally applicable nor strictly empirical. For example, it does not apply for many kinds of singular antennas such as an ideal dipole which has zero cross-sectional area but finite non-zero aperture. In addition, it is possible to

create an Huygens source with a directivity of 3 despite having a radius significantly smaller than a wavelength [5].

These limitations are due to the fact that Aperture Projection Analysis is an application of geometric optics and thus does not account for diffraction. In fact, APA is not derived directly from Maxwell's equations and instead relies upon several important assumptions listed below:

I Antennas are distributed uniformly along and oriented tangent to the array's surface.

II The array geometry can be approximated as a smooth continuous surface with acceptable error.

III The cosine bound on element gain applies, that is, $G(\theta) \leq \cos(\theta)G_{max}$.

a) The array curvature is smooth enough that it is appropriate to treat local coupling as similar to that in an infinite planar array.

b) The end-fire component of each element is negligible.

Assumption III is discussed extensively in Section 3.6. In order for this assumption to hold, both Assumptions III (a) and III(b) must also be valid. III(b) is true for many kinds of antennas, including the patch antennas used in this work.

3.5 Mathematical Foundation of Aperture Projection Analysis

Given the assumptions listed in Section 3.4, the calculation of an array's effective aperture is equivalent to the calculation of the array's cross-sectional area. In this section, we demonstrate this fact by first solving for the total array aperture without considering the specific geometry and then showing that the calculation of the cross-sectional area gives the same result. In the process we develop mathematical tools for calculating the cross-sectional area of an arbitrary *convex* geometry[1]. These tools will be used in Chapter 4 to compare the behavior of different arrays.

[1]Concave Geometries introduce the problem of self-occlusion which is not addressed in this work.

Total Array Aperture

As discussed in Section 2.2, antenna aperture is defined as the power absorbed by an antenna from an incident wave and delivered to a load.

$$A_e(\theta, \phi) = \frac{P_{delivered}}{S_{inc}(\theta, \phi)} \tag{3.9}$$

A phased array consists of multiple antennas and thus the total power delivered, $P_{delivered}$, to the load is equal to the sum of the powers delivered to each port, P_i.

$$P_{delivered} = \sum_i P_i \tag{3.10}$$

Thus the aperture of an array can be written as:

$$A_{array}(\theta, \phi) = \frac{\sum_i P_i}{S_{inc}(\theta, \phi)}. \tag{3.11}$$

Because each element is excited by the same incident wave with the same uniform power density, we can further modify the expression as follows:

$$A_{array}(\theta, \phi) = \sum_i \frac{P_i}{S_{inc}(\theta, \phi)} \tag{3.12}$$

$$A_{array}(\theta, \phi) = \sum_i A_i(\theta, \phi). \tag{3.13}$$

Thus the total array aperture is the sum of the apertures of its comprising elements *in the context of the array*. Note that this aperture is *not equivalent* to the aperture of the antenna in isolation because distortions in the near-field and mutual coupling will change the amount of power absorbed by the antenna[2].

Without making an assumption about the array's global geometry, we know from assumption I that each element will be located tangent to the array surface. Thus the local coordinate system that defines the element's pattern will be oriented with the $z-$ axis parallel to the surface normal vector $\vec{N}_i$, as shown in Figure 3.2. Therefore, an incident wave in the global direction $\vec{S}(\theta, \phi)$, maps to an angular position of $\vec{S}_i(\psi_i, \gamma_i)$ in this local coordinate system.

$$\psi_i = \arccos\left(\frac{\vec{N}_i \cdot \vec{S}}{\left|\vec{N}_i\right|\left|\vec{S}\right|}\right) \tag{3.14}$$

[2]A common error is the assumption that placing high gain antennas in an array will greatly enhance the gain. This is not the case as the coupling between the antennas reduces their apertures when placed into the array [6].

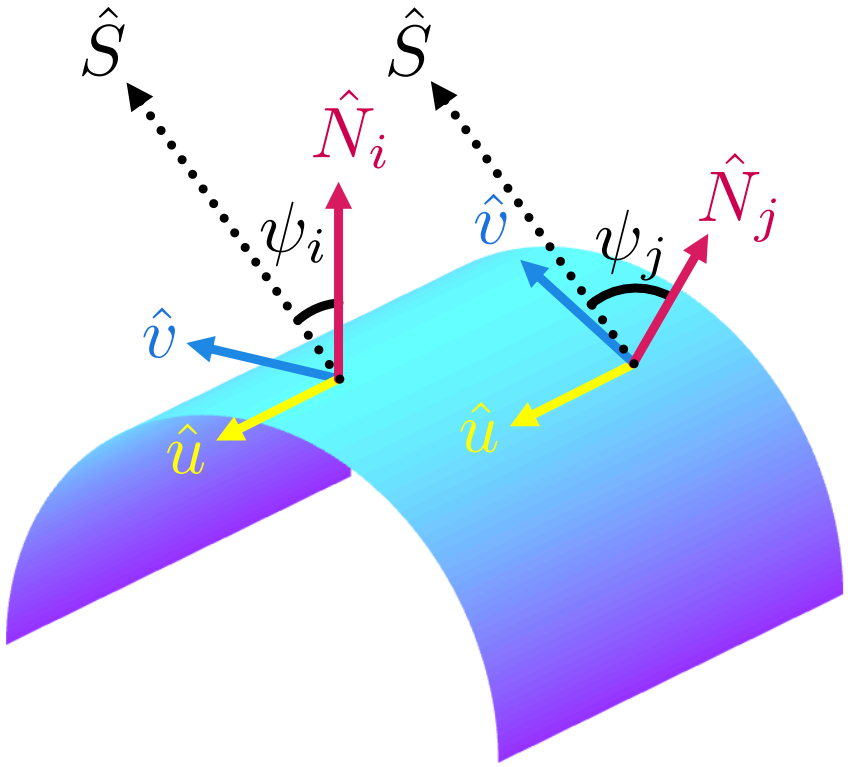


Figure 3.2: Local coordinate system for elements at different locations in the array. $\hat{u}$ and $\hat{v}$ are tangent to the surface while $\hat{N}_i$ is normal to it. Element patterns are thus oriented with their $\hat{z}_i$ axis parallel to $\hat{N}_i$. An incident wave or observer in direction $\hat{S}$ is located at $\theta = \psi_i$ in the local coordinate system of each element.

Assumption I also states that the elements are distributed uniformly along the array's surface. Thus we can define two orthogonal directions along the surface, u and v, and indicate the separation of element i from its neighbors as Δu_i and Δv_i [7]. With this definition, the maximum total aperture of element i is $\Delta u_i \Delta v_i$. Combined with assumption III, we can conclude that the maximum aperture of element i is:

$$A_i(\theta, \phi) \leq \Delta u_i \Delta v_i \cos(\psi_i). \tag{3.15}$$

The maximum aperture of the array is:

$$A_{array}(\theta, \phi) \leq \sum_i \cos(\psi_i) \Delta u_i \Delta v_i. \tag{3.16}$$

Under assumption II, we can approximate the power absorbed by each discrete element as the sum of powers absorbed by infinitesimal elements along a smooth curve with the same bounds as shown in Figure 3.3. Thus we can rewrite Equation 3.15 as:

$$A_i(\theta, \phi) \leq \int_{u_i}^{u_{i+1}} \int_{v_i}^{v_{i+1}} \cos(\psi_i) d\Omega. \tag{3.17}$$

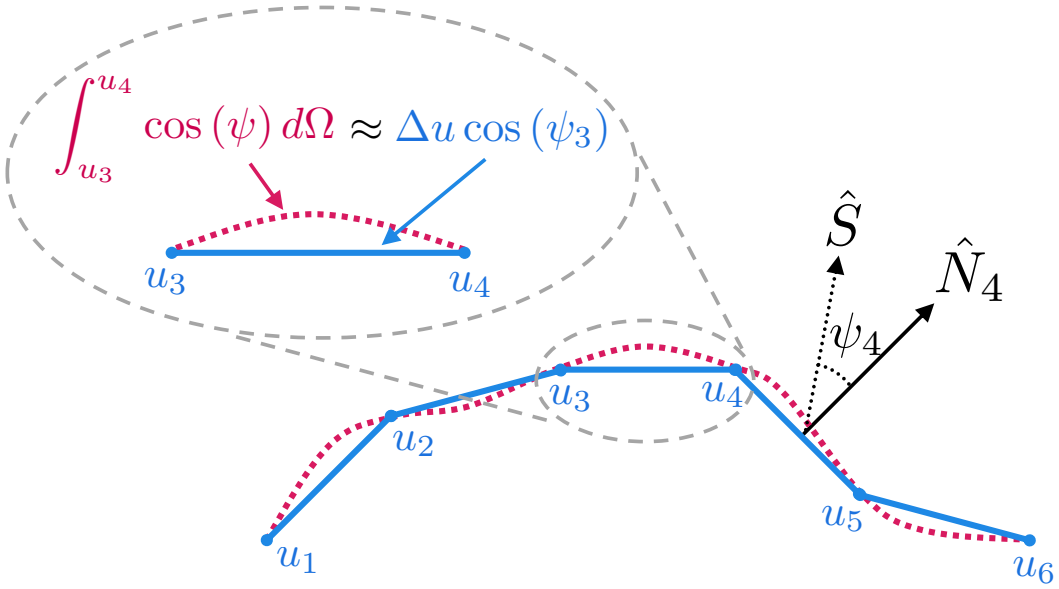


Figure 3.3: An illustration of the meaning and consequence of Assumption II. If the array geometry can be approximated as a smooth continuous surface, than each discrete element can be approximated by a set of infinitesimal elements along the surface with the same endpoints. The total power absorbed by a single discrete element, $\cos(\psi)\Delta u$, is approximately equal to the sum of the powers absorbed by these infinitesimal elements. Therefore, the discrete sum is approximately equal to a continuous integration along the surface.

The bounds of the integrals in Equation 3.17 cover the area of each element, therefore summing the integrals of each element is the same as integrating over the entire surface area of the array. Thus the maximum aperture of the array can be written as a continuous integral:

$$A_{array}(\theta, \phi) \leq \oiint \cos(\psi) d\Omega. \tag{3.18}$$

As will be shown below, Equation 3.18 is equal to the cross-sectional area of the array geometry.

Projected cross-sectional area

Any conformal array can be represented as a two dimensional surface in $\mathbb{R}^3$ parameterized by the surface patch $(x, y, z) = \sigma(u, v)$. Now suppose there is an observer at $\vec{O}(\theta_o, \phi_o)$ in the far field. To this observer, the array is projected onto an observation plane, (x', y'), and will appear to be a 2D object. The area of this 2D object in the observation plane is equal to the cross-sectional area of the array in the $\vec{O}(\theta_o, \phi_o)$ direction.

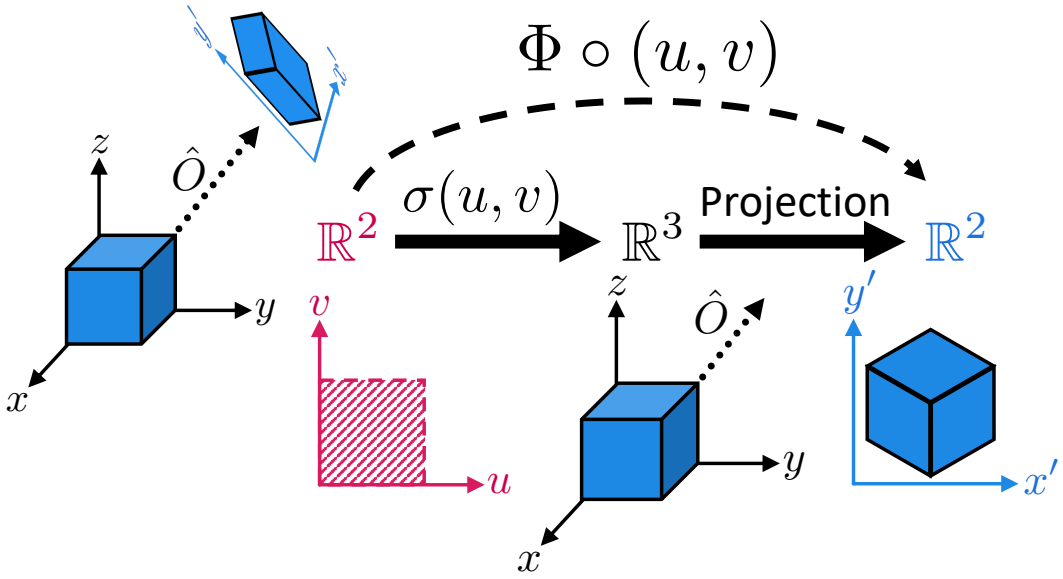


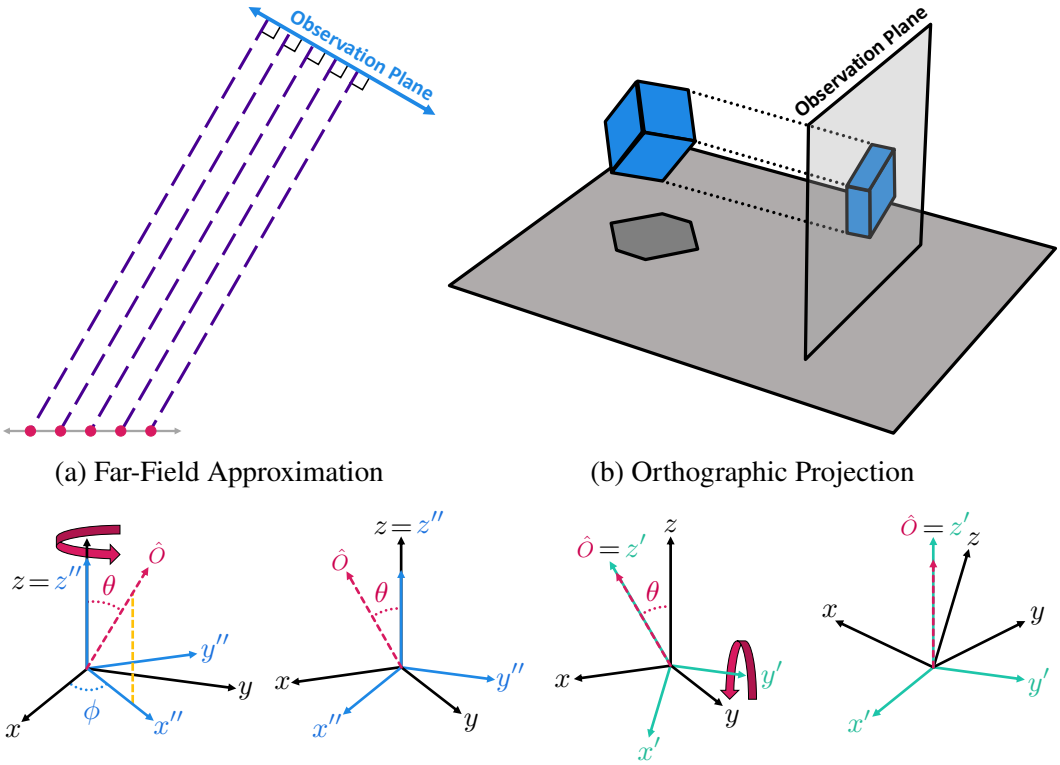
Figure 3.4: The projection of a surface patch onto an observation plane located at $\vec{O}(\theta_o, \phi_o)$ can be understood as a mapping of $(x', y') = \Phi \circ (u, v)$ through $\mathbb{R}^3$. The three dimensional surface patch is defined by the function $\sigma(u, v)$ over the two dimensional parameterization plane. This three dimensional patch is then projected onto the two-dimensional observation plane. These two operations can be combined to form a direct mapping, Φ.

Note that the projection is a mapping $\mathbb{R}^3 \Rightarrow \mathbb{R}^2$ and that the surface patch itself is a mapping $\mathbb{R}^2 \Rightarrow \mathbb{R}^3$. Taken together, projection of a surface patch can be viewed as mapping $(x', y') = \Phi \circ (u, v)$, $\mathbb{R}^2 \Rightarrow \mathbb{R}^3 \Rightarrow \mathbb{R}^2$, from the parameterizaion plane to the observation plane. Thus the cross-sectional area of an array can be calculated by an integral over the parameterization plane using the appropriate Jacobian.

$$A_{cross} = \iint_R |J(\Phi)| \, du dv \tag{3.19}$$

The nature of the projection is critical to calculating the cross-sectional area of the array. The far-field approximation makes two major assumptions that explicitly define this projection. The first assumption is that the fields observed in the far field are plane waves strictly traveling in the $\vec{\rho}$ direction. The second is that the observer is infinitely far away and so the rays from each radiating component to the observer are parallel. Combined, these two assumptions assert that a line perpendicular to the observation plane can be drawn from each point on the surface to the observation plane. This is the definition of an orthographic projection [8].

Suppose now that we define a new coordinate system (x', y', z') such that the observer is located on the $z'-$ axis. In this coordinate system, the orthograpic projection is



(a) Far-Field Approximation (b) Orthographic Projection

(c) (x', y', z') (d) Rotate z' by $-\phi$ (e) (x'', y'', z'') (f) Rotate x'' by $-\theta$

Figure 3.5: The far field approximation (a) and the orthographic projection (b) are equivalent as they both assume that the observer is located infinitely far away and thus the rays from each element to the observer are parallel. (c)-(f) illustrate the steps required to calculate the orthogonal projection. (c) an intermediary coordinate system (x'', y'', z'') is defined by $z'' = z$ and x'' is equal to the projection of $\hat{O}$ onto the $xy-$ plane. (d) the coordinate system is rotated about z' by $-\phi$. (e) (x', y', z') is defined by $y' = y''$ and $z' = \hat{O}$. (f) the coordinate system is rotated about y' by $-\theta$.

simply a mapping of the $x'y'-$ plane at $z' = 0$ to a parallel plane $z' = \infty$. Thus the cross-sectional area of the object in the observation plane is the cross-sectional area of the object in the $x'y'-$ plane.

As shown in Figure 3.5, the (x, y, z) coordinate system can then be mapped to the (x', y', z') coordinate system via two rotations. This can be accomplished by rotating each point around the $z-$ axis by $-\phi$ to (x'', y'', z'') such that observer is located in the $x''z''-$ plane and then rotating it around the new y'' axis by $-\theta$ to (x', y', z') such that the observer is located along the z' axis. While in general three rotations are necessary to perform an arbitrary rotational transformation, the final rotation about the $z'-$ axis can be ignored because it does not effect the projected area. In matrix

form these rotations are as follows:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \tag{3.20}$$

Using this transformation and the orthnographic projection, the mapping of a point at (x, y, z) onto the observation plane is thus:

$$x' = x\cos(\phi)\cos(\theta) + y\sin(\phi)\cos(\theta) - z\sin(\theta) \tag{3.21}$$

$$y' = -x\sin(\phi) + y\cos(\phi). \tag{3.22}$$

The mapping of a point on the parameterization plane onto (x, y, z) is simply the definition of the surface patch, $(x, y, z) = \sigma(u, v)$. Equations 3.21 and 3.22 can be conveniently written in terms of the surface patch using vector notation.

$$x' = \vec{A} \cdot \vec{\sigma} \tag{3.23}$$

$$y' = \vec{B} \cdot \vec{\sigma} \tag{3.24}$$

where

$$\langle x, y, z\rangle = \vec{\sigma}(u, v) \tag{3.25}$$

$$\vec{A} \equiv \langle\cos(\phi)\cos(\theta), \sin(\phi)\cos(\theta), -\sin(\theta)\rangle \tag{3.26}$$

$$\vec{B} \equiv \langle -\sin(\phi), \cos(\phi), 0\rangle\,. \tag{3.27}$$

Now the Jacobian of the mapping Φ, $(u, v) \Rightarrow (x', y')$, can be written explicitly as

$$|J(\Phi)| = \left\| \begin{bmatrix} \frac{\partial}{\partial u}\vec{A} \cdot \vec{\sigma} & \frac{\partial}{\partial v}\vec{A} \cdot \vec{\sigma} \\ \frac{\partial}{\partial u}\vec{B} \cdot \vec{\sigma} & \frac{\partial}{\partial v}\vec{B} \cdot \vec{\sigma} \end{bmatrix} \right\|, \tag{3.28}$$

noting that $\vec{A}$ and $\vec{B}$ do not depend on u or v:

$$|J(\Phi)| = \left\| \begin{bmatrix} \vec{A} \cdot \vec{\sigma}_u & \vec{A} \cdot \vec{\sigma}_v \\ \vec{B} \cdot \vec{\sigma}_u & \vec{B} \cdot \vec{\sigma}_v \end{bmatrix} \right\|. \tag{3.29}$$

Plugging in $\vec{A}$ and $\vec{B}$ and solving for the determinant gives:[3]

$$|J(\Phi)| = (\vec{\sigma}_u \times \vec{\sigma}_v) \cdot \langle\cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta)\rangle. \tag{3.30}$$

[3]The algebraic steps showing this equivalence can be found in Appendix A.

This is precisely the dot product between the standard normal vector of the patch, $\vec{N}_\sigma \equiv \vec{\sigma}_u \times \vec{\sigma}_v$ [7], and a unit vector in the direction of the observer $\vec{O} \equiv \langle \cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta) \rangle$. Thus

$$|J(\Phi)| = \vec{N}_\sigma \cdot \vec{O} \tag{3.31}$$

which is equivalent to

$$|J(\Phi)| = \left\|\vec{N}_\sigma\right\| \left\|\vec{O}\right\| \cos(\psi) \tag{3.32}$$

$$= \left\|\vec{\sigma}_u \times \vec{\sigma}_v\right\| \cos(\psi) \tag{3.33}$$

where ψ is the angle between the normal vector on the surface and the observation vector.

Plugging Equation 3.33 into Equation 3.19 gives an expression relating the cross-sectional area of an array to its parameterization.

$$A_{cross} = \iint_R \cos(\psi) \left\|\vec{\sigma}_u \times \vec{\sigma}_v\right\| du dv \tag{3.34}$$

From differential geometry, we know that $\left\|\vec{\sigma}_u \times \vec{\sigma}_v\right\|$ is the Jacobian of the mapping from the parameterization space to the area on the surface [7]. In other words, if $d\Omega$ is a infinitesimal segment of area on the surface, then

$$d\Omega = \left\|\vec{\sigma}_u \times \vec{\sigma}_v\right\| du dv \tag{3.35}$$

and

$$A_{surface} = \iint_R \left\|\vec{\sigma}_u \times \vec{\sigma}_v\right\| du dv = \oiint d\Omega. \tag{3.36}$$

Thus the cross-sectional area of an array can also be expressed as

$$A_{cross} = \oiint \cos(\psi) d\Omega. \tag{3.37}$$

Equivalence of Total Aperture and Cross-Sectional Area

Equation 3.37, the cross-sectional area of the array, is equal to the right hand side of Equation 3.18, the upper bound on the array aperture. Therefore, under the assumptions listed in Section 3.4, the cross-sectional area of an array provides an upper bound on its aperture.

$$A_{array}(\theta, \phi) \leq A_{cross} \tag{3.38}$$

This provides a mathematical basis for what is known intuitively, the cross-sectional area of an array provides an accurate bound on the its aperture. In fact, Equation 3.37 provides a very effective mathematical tool for calculating the maximum aperture of an array, especially when it is written in the form of Equation 3.39.

$$A_{cross} = \iint_R \vec{N}_\sigma \cdot \vec{O} du dv \tag{3.39}$$

It should be noted that the integrand is signed and only accounts for an infitesimal segment of the array. Therefore segments that face away from the observer will decrease the value of the integral. In addition, segments that face the observer but are occluded by another segment of the array, will still contribute to the integral. Neither of these contributions to the cross-sectional area are valid and thus should be avoided. The backwards orientation problem can easily be avoided by careful choice of integration bounds in an analytic computation, or nulling negative integrands in a numeric computation.

The occlusion problem is more difficult to account for because it necessitates comparing each segment in the array to check whether they occlude each other. Fortunately there are known computational techniques from computer graphics research, such as Z-buffering, to address this problem [9]. Analytically the problem is more challenging, so in this work we restrict our analysis to convex geometries that cannot occlude themselves.

3.6 Cosine Bound on Element Gain

Assumption III in Section 3.4 is the most critical for establishing the validity of Aperture Projection Analysis. It also is the least obvious as to why it should be true. It certainly does not hold for individual antennas, like a dipole, and it is not immediately clear why the maximum gain of an element must decrease as a cosine when placed in an array. Although it is commonly cited in literature, [10]–[12], it is critical to establish the validity of the cosine bound assumption in order to support the use of Aperture Projection Analysis. This section examines multiple arguments in support of the bound. It is shown that the cosine bound is generally applicable except in end-fire, i.e., when $\theta \approx \pm 90°$.

Antennas in close proximity to other antennas behave differently than when they are isolated. The existence of other radiators changes the near-field environment, both through mutual coupling and by altering the boundary field conditions. Consider an element in isolation; as there are no other sources the field behavior is completely

determined by the antenna. Assuming there is no metal nearby, the fields are free to take on any orientation and magnitude. Now suppose that there two identical antennas in close proximity driven completely out of phase. Due to the symmetry, the field contributions from both antennas should be identical, except with opposite phase, along the plane halfway between them. Thus there must be no fields halfway between the antennas. In an array, each interior antenna is surrounded by other antennas, further altering its properties. This new boundary condition changes the near-field environment and thus alters both the radiation pattern of the individual element and its impedance. As the boundary conditions change with element excitation, it is clear that the element impedance and radiation patterns must also change with the excitation. This effect is commonly modeled in electromagnetic solvers using primary-secondary, or master-slave, boundary conditions.

The effect of coupling on element radiation pattern is shown in [13]. An extensive analysis of self-and mutual coupling in an array of element spacing $0.5\lambda \leq d \leq \lambda$ shows both the strong effect of steering angle on element impedance and the reduction of each element's broadside gain to

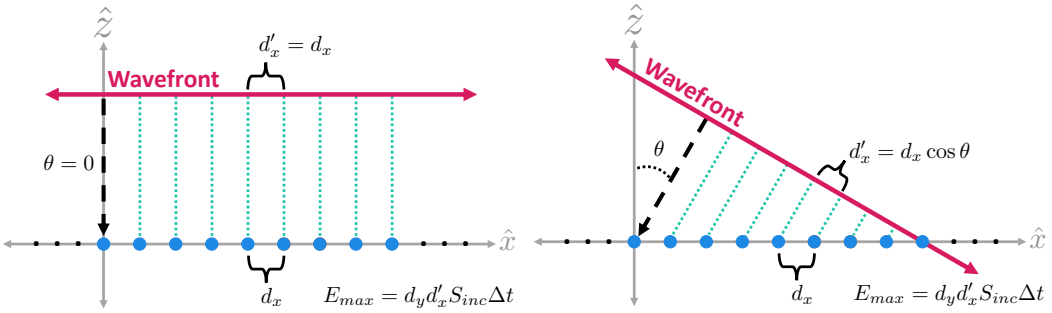
$$G = 4\pi \left(\frac{d}{\lambda}\right)^2 \tag{3.40}$$

regardless of its gains when isolated. Equation 3.40 is precisely the gain predicted by Equation 2.14 assuming that the aperture is equal to the average area per element in the array.

The cosine upper bound on element aperture, and thus gain, is well studied. [14] and [15] demonstrate how mutual coupling drastically alters the antenna impedance as a beam is scanned and [16] shows that a cosine element radiation pattern maximizes the gain by reducing this effect. [17] and [18] use the active reflection coefficient[4], and an analysis of the grating lobes in a planar array, to calculate the distortion of the element pattern due to array excitation, reaching the conclusion that the mutual coupling of a 0.5λ spaced array forces the element pattern to be bounded by a cosine.

[19] models the effect of the boundary field conditions between elements in an array with waveguides whose dimensions change as the beam is steered. The radiation resistance of these waveguides is then calculated and used to quantify the amount of power, relative to that of an isotropic antenna, required to create a given field strength. The result of this analysis supports the cosine bound on element aperture.

[4] A function of the mutual coupling and element excitation.



(a) Broadside incidence. $d' = d$ (b) Incidence angle θ. $d' = d\cos(\theta)$

Figure 3.6: Conservation of energy argument for the $\cos\theta$ bound. An infinite pulsed plane wave pulse of power density S_{inc} and duration Δt is incident on a two dimensional infinite array with element spacing d_x and d_y. The maximum energy available for each element to absorb is $E_{max} = d_y d'_x S_{inc} \Delta t$, where d'_x is the effective spacing between elements along the wavefront in the $xz-$ plane.

[20] and [21] find that the cosine bound is applicable in cases where the end-fire effect is negligible through an analysis of the array factor of a linearly polarized square aperture and an array of half wavelength spaced isotropic elements. [22] demonstrates the cosine bound on the array directivity, except when radiating end-fire, through an analysis of the radiation due to current modes in a planar array. Finally, [23] utilizes a Floquet analysis of modes radiated by an infinite rectangular array to prove the cosine bound.

Note that all of the analyses in the cited literature are based on planar arrays and that the mutual coupling between elements is critical to the creation of the cosine bound. Therefore, the array's curvature has to be smooth enough that it is appropriate to treat the coupling between elements as similar to that in an infinite planar array. As elements in an array mostly couple with their immediate neighbors, this is not a particularly restrictive condition.

Conservation of Energy Argument

A simple thought experiment provides intuitive reasoning for why the aperture of individual elements must be bounded by a cosine. Consider an infinite planar array with element spacing d oriented in the $xy-$plane. Now supposed an incident impulse plane wave, infinite in space but finite in duration, passes through the array as shown in Figure 3.6a. The wavefront is parallel to the array plane and has a finite uniform power density, S_{inc}, so there is a finite amount of power that can be absorbed by each element. On average, the surface area per unit element of the array is d^2. Therefore if the array completely absorbs the incident wave, the average power that can be

absorbed by each element is at most

$$P_{absorbed} = d^2 S_{inc}. \tag{3.41}$$

Now consider a plane wave that is not parallel to the array plane but at an incident angle θ that is completely absorbed by the array. As shown in Figure 3.6b, the same power density is spread out over multiple elements as the effective spacing of elements along the wavefront is reduced by a factor of $\cos\theta$. Thus a reduction of element aperture must occur due to the conservation of energy. The average power that can be absorbed by each element is at most

$$P_{absorbed} = d^2 S_{inc} \cos(\theta). \tag{3.42}$$

One way to understand why the power absorbed by each element is reduced by cosine on average is to break the incident wave into components traveling perpendicular and parallel to the array. The perpendicular component of the incident wave behaves as in Figure 3.6a and is completely absorbed. However the parallel component of the incident wave must travel along the array, and thus will be absorbed by an another element. Thus for an element in the *center* of the array, the parallel component has already been absorbed, leaving only the perpendicular component, $S_{inc}\cos(\theta)$. Thus on average the power absorbed by each element is bounded by $\cos\theta$.

This component analysis raises an important question that leads to a caveat in the cosine bound; if the parallel component is absorbed by another element in the array, than how can *that* element be bound by a cosine? The answer is that it cannot, and that cosine bound only applies to elements *in the middle of an array*. The first few elements of the array absorb the lateral component of the power and behave differently than the rest of the array. Therefore, although the cosine bound implies that arrays cannot receive end-fire power, a finite array can as the edge elements are not bound by a cosine. However, for very large arrays the power absorbed by the elements *on average* drops to zero as most of the array is subject to the cosine bound.

References

[1] C. J. Bouwkamp and N. G. de Bruijn, "The problem of optimum antenna current distribution," *Philips Research Reports*, vol. 1, pp. 135–158, 2 Jun. 1945.

[2] R. Harrington, "On the gain and beamwidth of directional antennas," *IRE Transactions on Antennas and Propagation*, vol. 6, pp. 219–225, 3 Jul. 1958, ISSN: 0096-1973. DOI: 10.1109/TAP.1958.1144605.

[3] R. F. Harrington, "Effect of antenna size on gain, bandwidth, and efficiency," *Journal of Research of the National Bureau of Standards, Section D: Radio Propagation*, vol. 64D, p. 1, 1 Jan. 1960, ISSN: 1060-1783. DOI: 10.6028/jres.064D.003.

[4] M. Gustafsson and M. Capek, "Maximum gain, effective area, and directivity," *IEEE Transactions on Antennas and Propagation*, vol. 67, pp. 5282–5293, 8 Aug. 2019, ISSN: 0018-926X. DOI: 10.1109/TAP.2019.2916760. [Online]. Available: https://ieeexplore.ieee.org/document/8718501/.

[5] P.-S. Kildal, E. Martini, and S. Maci, "Degrees of freedom and maximum directivity of antennas: A bound on maximum directivity of nonsuperreactive antennas.," *IEEE Antennas and Propagation Magazine*, vol. 59, pp. 16–25, 4 Aug. 2017, ISSN: 1045-9243. DOI: 10.1109/MAP.2017.2706659. [Online]. Available: http://ieeexplore.ieee.org/document/7954002/.

[6] A. Hajimiri, B. Abiri, F. Bohn, M. Gal-Katziri, and M. H. Manohara, "Dynamic focusing of large arrays for wireless power transfer and beyond," *IEEE Journal of Solid-State Circuits*, vol. 56, pp. 2077–2101, 7 Jul. 2021, ISSN: 0018-9200. DOI: 10.1109/JSSC.2020.3036895. [Online]. Available: https://ieeexplore.ieee.org/document/9270598/.

[7] A. Pressley, *Elementary Differential Geometry*. Springer London, 2010, ISBN: 978-1-84882-890-2. DOI: 10.1007/978-1-84882-891-9. [Online]. Available: http://link.springer.com/10.1007/978-1-84882-891-9.

[8] "Orthographic projection," *Merriam-Webster Dictionary*, [Online]. Available: https://www.merriam-webster.com/dictionary/orthographic%5C%20projection.

[9] W. Straßer, "Schnelle kurven- und flächendarstellung auf grafischen sichtgeräten," Technischen Universität Berlin, Apr. 1974.

[10] C. Balanis, *Antenna Theory: Analysis and Design*, 4th ed. Wiley, 2016, ISBN: 1118642066.

[11] D. Pozar, "The active element pattern," *IEEE Transactions on Antennas and Propagation*, vol. 42, pp. 1176–1178, 8 1994, ISSN: 0018926X. DOI: 10.1109/8.310010. [Online]. Available: http://ieeexplore.ieee.org/document/310010/.

[12] L. Josefsson and P. Persson, *Conformal Array Antenna Theory and Design*. John Wiley & Sons, Inc., 2006.

[13] J. Allen, "Gain and impedance variation in scanned dipole arrays," *IRE Transactions on Antennas and Propagation*, vol. 10, pp. 566–572, 5 Sep. 1962, ISSN: 0096-1973. DOI: 10.1109/TAP.1962.1137903. [Online]. Available: http://ieeexplore.ieee.org/document/1137903/.

[14] S. Edelberg and A. Oliner, "Mutual coupling effects in large antenna arrays: Part 1–slot arrays," *IRE Transactions on Antennas and Propagation*, vol. 8, pp. 286–297, 3 May 1960, ISSN: 0096-1973. DOI: 10.1109/TAP.1960.1144837. [Online]. Available: http://ieeexplore.ieee.org/document/1144837/.

[15] P. Carter, "Mutual impedance effects in large beam scanning arrays," *IRE Transactions on Antennas and Propagation*, vol. 8, pp. 276–285, 3 May 1960, ISSN: 0096-1973. DOI: 10.1109/TAP.1960.1144839. [Online]. Available: http://ieeexplore.ieee.org/document/1144839/.

[16] P. Hannan, "The element-gain paradox for a phased-array antenna," *IEEE Transactions on Antennas and Propagation*, vol. 12, pp. 423–433, 4 Jul. 1964, ISSN: 0096-1973. DOI: 10.1109/TAP.1964.1138237. [Online]. Available: http://ieeexplore.ieee.org/document/1138237/.

[17] W. Wasylkiwskyj and W. Kahn, "Element patterns and active reflection coefficient in uniform phased arrays," *IEEE Transactions on Antennas and Propagation*, vol. 22, pp. 207–212, 2 Mar. 1974, ISSN: 0096-1973. DOI: 10.1109/TAP.1974.1140756. [Online]. Available: http://ieeexplore.ieee.org/document/1140756/.

[18] ——, "Element pattern bounds in uniform phased arrays," *IEEE Transactions on Antennas and Propagation*, vol. 25, pp. 597–604, 5 Sep. 1977, ISSN: 0096-1973. DOI: 10.1109/TAP.1977.1141654. [Online]. Available: http://ieeexplore.ieee.org/document/1141654/.

[19] H. Wheeler, "The radiation resistance of an antenna in an infinite array or waveguide," *Proceedings of the IRE*, vol. 36, pp. 478–487, 4 Apr. 1948, ISSN: 0096-8390. DOI: 10.1109/JRPROC.1948.229650. [Online]. Available: http://ieeexplore.ieee.org/document/1697673/.

[20] R. Bickmore, "A note on the aperture of electrically scanned arrays," *IRE Transactions on Antennas and Propagation*, vol. 6, pp. 194–196, 2 Apr. 1958, ISSN: 0096-1973. DOI: 10.1109/TAP.1958.1144576. [Online]. Available: http://ieeexplore.ieee.org/document/1144576/.

[21] M. King and R. Thomas, "Gain of large scanned arrays," *IRE Transactions on Antennas and Propagation*, vol. 8, pp. 635–636, 6 Nov. 1960, ISSN: 0096-1973. DOI: 10.1109/TAP.1960.1144910. [Online]. Available: http://ieeexplore.ieee.org/document/1144910/.

[22] R. Elliott, "Beamwidth and directivity of large scanning arrays: Last of two parts," *Microwave Journal*, vol. 1964, pp. 74–82, Jan. 1964.

[23] S. P. Skobelev, "On the ideal gain of a radiating element in a planar array," in *2008 12th International Conference on Mathematical Methods in Electromagnetic Theory*, Jun. 2008, pp. 305–307, ISBN: 978-1-4244-2284-5. DOI: 10.1109/MMET.2008.4580976. [Online]. Available: https://ieeexplore.ieee.org/document/4580976/.

Chapter 4

EFFECT OF ARRAY GEOMETRY

> I cower at the thought of other people's expectations
> And yet still, hand over mine to them.
>
> ---
>
> *Parquet Courts*
> *Tenderness*

The geometric shape of a conformal array has a strong impact on the array behavior. While the details of tile placement and spacing on the surface impacts the patterns of *individual* beams, the entire geometry bounds the maximum gain and 3 dB steering range of the array. This chapter explores the effects of array geometry using aperture projection analysis, showing the existence of a trade-off between gain and steering range.

Section 4.1 begins with a brief overview of the literature on conformal arrays. The methodology used to compare geometries is established in Section 4.2 while Section 4.3 introduces the parameterizations of the basic geometries to be evaluated. Analytic formulas of the shapes' maximum broadside gain are presented in Section 4.4 while a numeric analysis of their steering ranges is performed in Section 4.5. The chapter concludes with the demonstration of the trade-offs between gain and steering range inherent in the different array geometries in Section 4.6.

4.1 Arrays of Different Shapes

Phased arrays conformal to different geometries, such as planar, cylindrical, and spherical, have been extensively explored. Studies of planar and conformal arrays[1]–[3] have shown that curved arrays have a larger scan range, but lower broadside gain, than planar arrays. However, these improvements are not guaranteed [4] as the implementation of element spacing and feed systems have a large impact on beam patterns and polarization matching. Pattern synthesis is particularly difficult as, unlike planar arrays, the element pattern cannot be factored out from the array factor [5]. The properties of even more exotic geometries, such as ellipsoids, hyperboloids, paraboloids, and ogives, have also been studied [6].

Moving beyond planar arrays opens up the design space to new applications. The most common conformal array application is the integration of antennas and arrays into geometries designed for other applications, such as the wing of an airplane or the nosecone of a rocket. This enables aircraft to employ multiple antenna systems without increasing drag; commercial aircraft can have over 20 different antennas [7] while some military aircraft have up to 70 [8]. Due to their aerodynamic properties, conic arrays and conic-cylindrical hybrid arrays (CYLCON) have been closely studied [9]–[12].

However, non-planar array geometries have useful radiative properties for other applications. Faceted phased arrays that emulate curved surfaces with planar segments are commonly used to increase array scan range. Due to their azimuthal symmetry, cylindrical arrays can be used in applications that require minimal beam distortion and large cross-polerization isolation such as high accurate polarimetric phased array radar [13], [14]. The spherical arrays have been explored [15], [16] as their wide scanning range is useful for tracking satellites. In addition, the precise beam control at wide steering angles of spherical antennas is useful for military GPS applications as they can null ground based jamming signals without eliminating the signals from low-elevation GPS satellites [17].

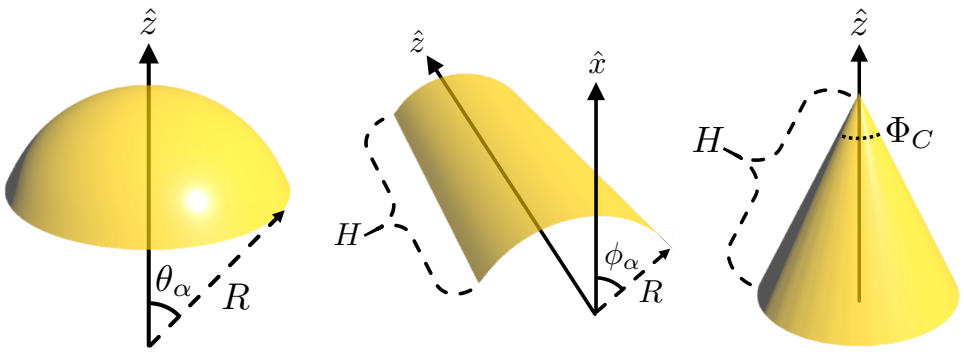
4.2 Aperture Projection Analysis of Array Geometries

As discussed in Chapter 3, the maximum gain of an array is determined by its cross-sectional area. Depending on the array geometry, this cross-sectional area *changes* as an observer rotates around the array. Thus at some angle the cross-section is maximized, resulting in the maximum gain. For most geometries, the maximum gain decreases as an observer deviates from this angle, eventually dropping -3 dB below the maximum, limiting its steering range.

The effect of different geometries on the maximum gain and steering range can thus be quantified by analyzing the change in cross-sectional area from different observation angles. However in order to compare the merits of different geometries, they must be normalized in some manner. Otherwise, the ideal geometry would simply be an infinitely large sphere because it has infinite cross-sectional area no matter the observation angle[1].

Normalization of the array surface area provides the most useful general comparison. In addition to the geometry, the most important factors that influence array

[1] Unfortunately, constructing such an array would be impractical.



(a) Spherical Cap (b) Cylinder (c) Cone

Figure 4.1: Parameterizations of Simple Array Geometries.

performance are the number of elements and their relative spacing. While other factors such as the element positions, the maximum aperture, and the array fill factor are also important, they are closely related to these three factors. Therefore to isolate the effect of geometry, the spacing and number of elements should be kept constant. Unfortunately, as detailed in Section 5.2, this is not strictly possible due to changes in Gaussian curvature. However, the spacing and number of elements can be kept approximately constant by ensuring that the array surface areas are the same.

To perform the analysis we start by defining parameterizations of different simple geometries–Planar, Spherical, Cylindrical, and Conic–and analyzing their cross-sectional areas at different orientations. Then by normalizing the surface areas, their relative maximum broadside gains are calculated. Finally analytic and numerical calculations examine the trade-off between gain and steering range in the different geometries. This chapter presents and analyzes the results as given; the full analysis of each geometry's parameterization, surface area, and maximum gain are included in Appendix B.

4.3 Parameterization of Simple Array Geometries

Figure 4.1 shows the basic geometries analyzed in this chapter. This section provides descriptions of their parameterizations and their surface areas.

Planar Array

A planar array is the simpliest geometry, consisting of a rectangle in the $xy-$ plane. It can be simply parameterized by the location along each axis, $0 \leq x \leq W$ and $0 \leq y \leq L$. While the aspect ration of $\frac{W}{L}$ is very important for the specific beam patterns, it actually makes no different when considering the total aperture of the

array. The planar array surface area $A_{plane} = WL$ is the basis for the normalizations of the other geometries.

Spherical Array

A spherical array is parameterized by its radius, R, and its location in spherical coordinates (θ, ϕ) with $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. Of interest more than a strict sphere, however, is a spherical cap that covers a portion of the sphere from $\theta = 0$ to some critical angle, θ_α. Such a cap can describe the behavior of spherical domes with identical surface area, but different curvatures including nearly planar, a hemisphere, and a complete sphere. The surface area of such an array is given in Equation 4.1.

$$A_{surface,sphere} = 2\pi R^2 \left(1 - \cos(\theta_\alpha)\right) \tag{4.1}$$

Cylindrical Array

A cylindrical array is parameterized by its radius, R, and its location in cylindrical coordinates (ϕ, z) with $-\pi \leq \phi \leq \pi$ and $0 \leq z \leq H$. As with the spherical array, the radial extent of the cylinder can be constrained in order to examine incomplete cylinders with different curvatures. In this case, θ is constrained to $\pm\phi_\alpha$. The surface area of such an incomplete cylindrical array is given in Equation 4.2.

$$A_{surface} = 2\phi_\alpha RH \tag{4.2}$$

Conic Array

While there are many different conic shapes, this analysis focuses on a complete right circular cone with an apex angle of Φ_C. Such cone can be parameterized by the location along its slant line and its angle with respect to the $x-$ axis, (θ, h) with $0 \leq \theta \leq 2\pi$ and $0 \leq h \leq H$. The surface area of this conic array is given by Equation 4.3.

$$A_{surface} = \pi H^2 \sin\left(\frac{\Phi_C}{2}\right) \tag{4.3}$$

4.4 Relative Broadside Gain of Different Geometric Shapes

Given the parameterization and surface areas of these basic geometric shapes, their maximum broadside gain compared to that of a planar array with the same surface area can be calculated using Aperture Projection Analysis. For this analysis, broadside is defined as the angle with maximum cross-sectional area, with the exception of the conic array where it is defined to be along the cone's axis of revolution[2].

[2] As will be seen in Section 4.5, the angle of maximum projected area changes with Φ_c.

Planar Array

When viewed from broadside, a the full area of the planar array is in view. Therefore the cross-sectional area $A_{cross,plane} = A_{plane}$ and the maximum possible gain is

$$G_{max,plane} = 4\pi \frac{A_{plane}}{\lambda^2}. \tag{4.4}$$

Spherical Array

For an area smaller than a hemisphere[3], the entire spherical cap is visible at broadside. In this case, the relative maximum gain can calculated from the surface area of the array A_{plane} and the radius of the sphere, R, as shown in Equation 4.5.

$$G_{max,sphere} = G_{max,plane} - \left[\frac{A_{plane}}{\lambda R}\right]^2 \tag{4.5}$$

For areas larger than a hemisphere, part of the array becomes occluded and does not contribute to the maximum gain. In this case, the maximum gain can be expressed in terms of χ the ratio of the surface area of array to that of a sphere with radius R. It should be noted that these expressions are equivalent for $\chi \leq 0.5$.

$$\chi = \frac{A_{plane}}{4\pi R^2} \tag{4.6}$$

$$G_{max,sphere} = \begin{cases} (1-\chi)\, G_{max,plane} & 0 \leq \chi \leq 0.5 \\ \dfrac{1}{4\chi} G_{max,plane} & 0.5 \leq \chi \leq 1 \end{cases}. \tag{4.7}$$

Considering Equation 4.5, it is clear that curvature of the spherical array decreases the broadside gain. For very large R, the array is essentially planar and so the gain is unaffected. However, the gain is reduced by half when the array covers an entire hemisphere. The relative gain is reduced by another factor of two in the case of a complete sphere as half of the array is occluded. Therefore, the maximum gain of a spherical array is -6 dB less that a planar array of the same surface area.

Cylindrical Array

For $\phi_\alpha \geq \frac{\pi}{2}$, part of the cylinder surface area is occluded. Thus like the spherical array, the relationship between the cylindrical and planar gains is a piece-wise continuous function as shown in Equation 4.8.

$$G_{max,cylinder} = \begin{cases} \mathrm{sinc}(\phi_\alpha) G_{max,plane} & 0 \leq |\phi_\alpha| \leq \frac{\pi}{2} \\ \dfrac{1}{\phi_\alpha} G_{max,plane} & \frac{\pi}{2} \leq |\phi_\alpha| \leq \pi \end{cases}. \tag{4.8}$$

[3] $A_{plane} \leq 2\pi R^2$

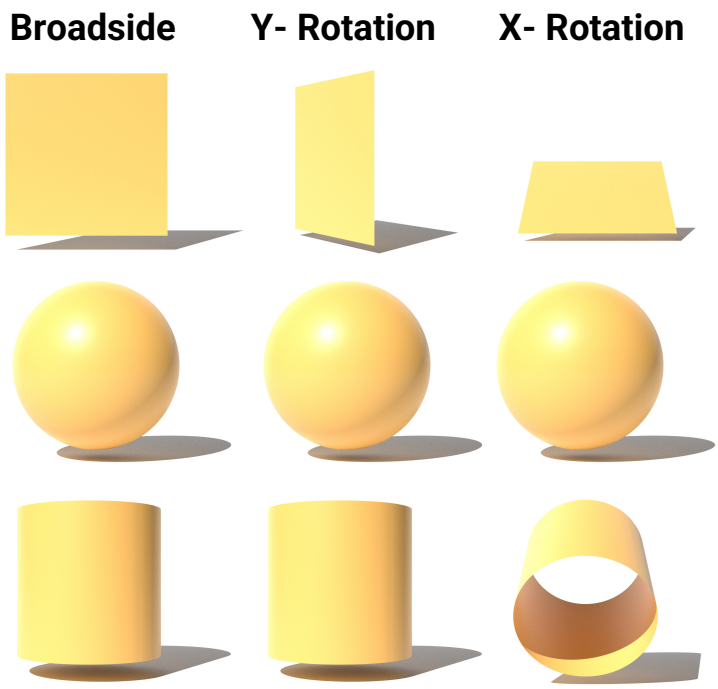


Figure 4.2: Change in cross-sectional area of array geometries under rotation.

Again it is clear that the curvature of a cylindrical array reduces its maximum gain. For arrays with small ϕ_α the gain is unaffected. However, the gain is reduced by a factor of $\frac{2}{\pi}$ as the curvature increases and the array covers half of the cylinder. For a complete cylinder the gain is $\frac{1}{\pi}$ smaller, or approximately -5 dB less, than that of a planar array with equivalent surface area.

Conic Array

The gain behavior of a conic array is more complicated than the other geometries examined because the angle of maximum cross-sectional area changes with the apex angle. Imagine the case where $\Phi_C = \pi$; the conic array is simply a planar circle as the cone has no height. Thus the maximum gain is along the axis of revolution. As Φ_C decreases however, the cross-sectional area along the axis of revolution decreases as the radius of the cone's base decreases and the height increases. Eventually at $\Phi_C = 0$, the cone collapses into a one dimensional line with no cross-sectional area along the axis of revolution.

However, consider an observer located perpendicular to the axis of revolution. This observer can see at most half of the cone's surface area as the rest is occluded. In the degenerate example of $\Phi_C = 0$ however, the entire "surface area" of the linear

array is oriented in the plane perpendicular to the axis of revolution. Thus this perpendicular observer is located at the the angle of maximum cross-sectional area. Therefore it is clear that the angle of maximum gain must change with Φ_C.

The angle of maximum gain is difficult to calculate analytically in all cases due to the complicated occlusion behavior of the cone. As discussed in Section 3.5, the integration bounds of Equation 3.39 must be carefully chosen to ensure that the area of occluded portions of the array are not included. Unlike the other shapes, the cone's parameterization requires complicated functional bounds in order to exclude occluded areas, rendering the analytic results cumbersome.

To simplify the analytic analysis, the maximum gain of the conic array along the axis of revolution is analyzed instead. As shown in Equation 4.9, the gain is proportional to the sign of half the apex angle.

$$G_{max,cone} = \sin\left(\frac{\Phi_C}{2}\right) G_{max,plane} \tag{4.9}$$

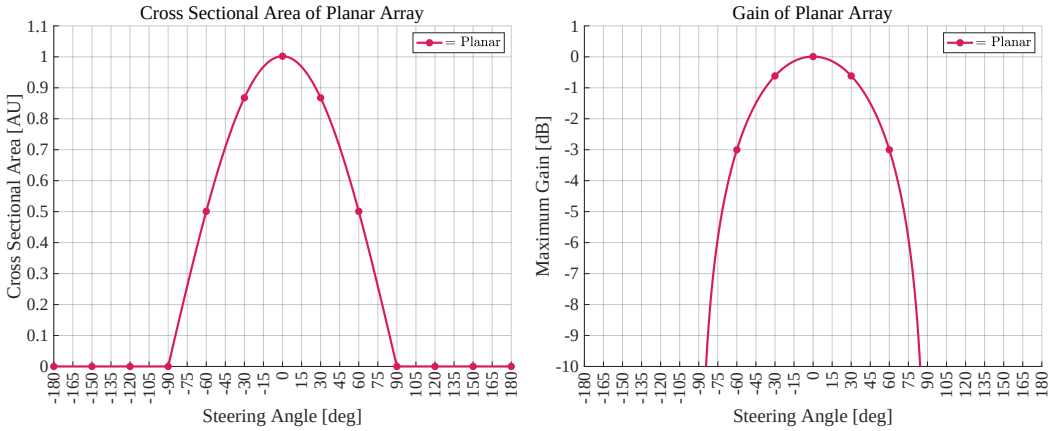
4.5 Change of Gain with Steering Angle

As can be seen in Figure 4.2, the array geometry also determines how the cross-sectional area, and thus gain, changes under rotation. This rotation is equivalent to steering the beam to different directions. The cross-sectional area of any convex geometry from any observation angle can be easily calculated using numeric integration of Equation 3.39. Numeric integration automatically accounts for occlusion by ignoring portions of the surface that face away from the observer, or equivalently, have a negative integrand.

This technique is used to calculate the change in cross-sectional area, and thus the maximum gain, across the steering range in orthogonal planes from $-180°$ to $180°$. Results are plotted for different θ_α, ϕ_α, and Φ_C. In all cases, the array surface area and maximum gain is normalized to that of the planar array.

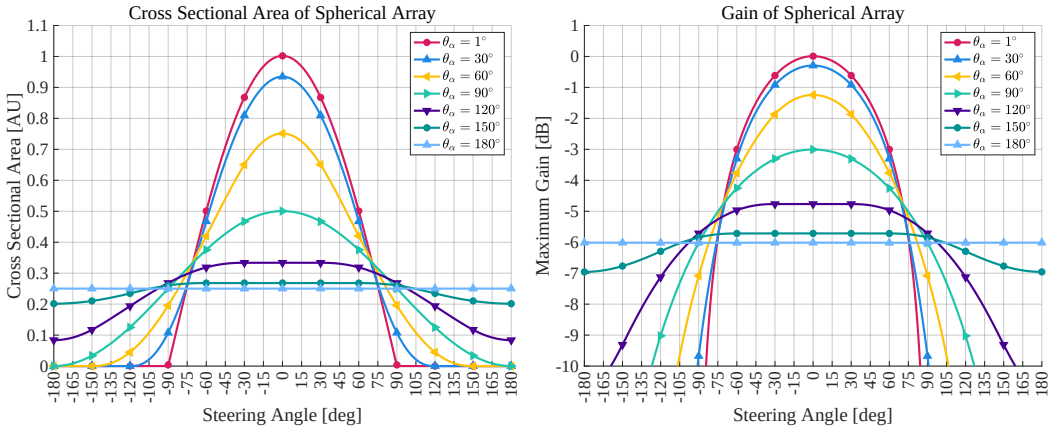
Planar Array

Figure 4.3 shows the change in cross-sectional area and maximum achievable gain with steering angle for a planar array. Due to rotational symmetry, the behavior is identical in the $xz-$ and $yz-$ plane cuts. The cosine relationship between the cross-sectional area and steering area is clearly visible. The maximum broadside gain is 0 dB by definition. The -3 dB steering range is $120°$ with the gain dropping rapidly afterwords.



(a) Cross-sectional area (b) Maximum gain

Figure 4.3: Maximum gain and cross-sectional area of a planar array.

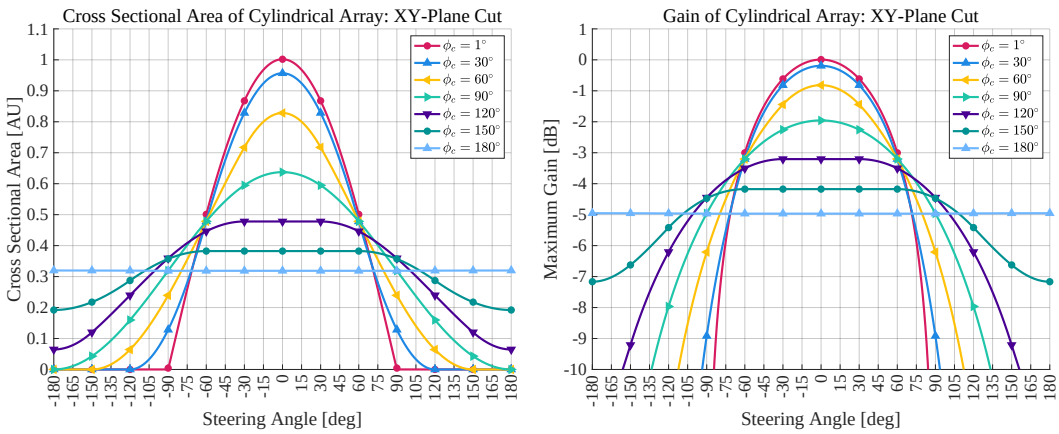


(a) Cross-sectional area (b) Maximum gain

Figure 4.4: Maximum gain and cross-sectional area of spherical cap arrays with different θ_α.

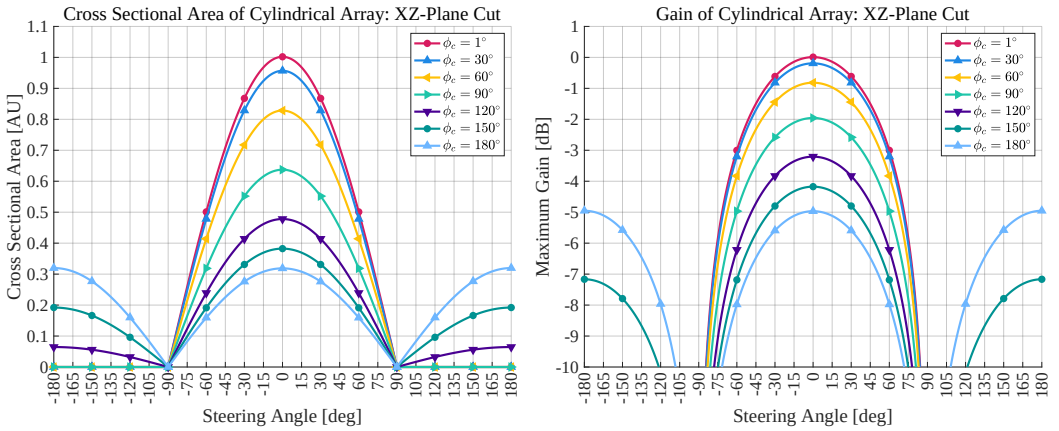
Spherical Array

The maximum gain and cross-sectional area over the steering range for various spherical caps is depicted in Figure 4.4. As with the planar array, the $xz-$ and $yz-$ plane cut performance is identical. At one extreme, $\theta_\alpha = 1°$, the cap is essentially a plane and behaves accordingly. At the other extreme, $\theta_\alpha = 180°$, the array is a complete sphere and thus the cross-sectional area does not change with steering angle and the gain is a constant -6 dB. For a hemisphere, $\theta_\alpha = 90°$, the gain changes with steering angle less drastically than planar, exhibiting a -3 dB steering range of $180°$ with a broadside gain of -3 dB. By comparing the array results to that of the planar case, it is clear that all of the spherical caps have a higher maximum gain than planar array at large steering ranges. The steering angle at which the hemisphere



(a) Cross-sectional area: $xy-$plane.

(b) Maximum gain: $xy-$plane.



(c) Cross-sectional area: $xz-$plane.

(d) Maximum gain: $xz-$plane.

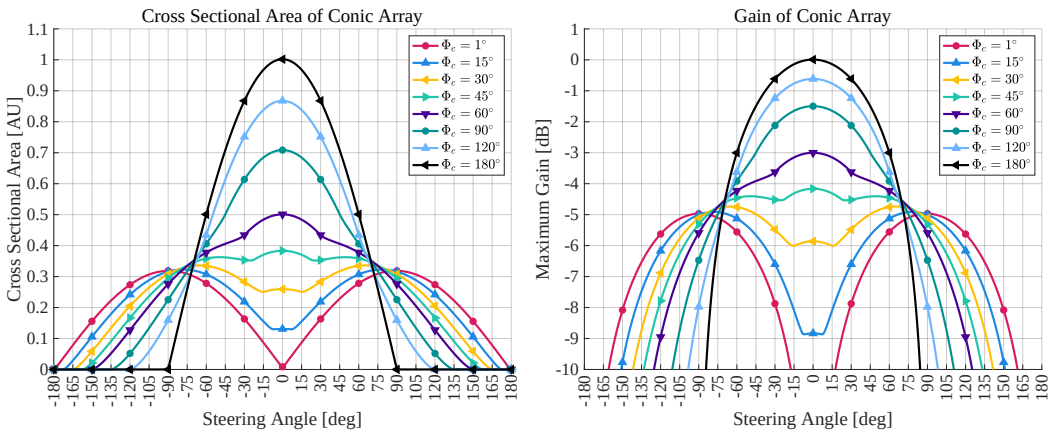
Figure 4.5: Maximum gain and cross-sectional area of cylindrical arrays with different ϕ_α.

has higher maximum gain is 70.5° while the sphere outperforms at angles above 75.5°.

Cylindrical Array

Unlike the spherical, conic, and planar arrays, the cylindrical array is asymmetric resulting in different gain behaviors in the $xz-$ and $xy-$ plane cuts. Thus Figure 4.5 includes area and gain plots for both cuts. The cylinder is oriented around the $z-$ axis and thus is curved in the $xy-$ plane cut. Again cylinders with different ϕ_α are shown, ranging from the essentially planar $\phi_\alpha = 1°$ to the complete cylinder $\phi_\alpha = 180°$.

As in the spherical case, the increased curvature drops the maximum gain to approximately -2 dB for the half-cylinder and -5 dB for the complete cylinder. However,



(a) Cross-sectional area

(b) Maximum gain

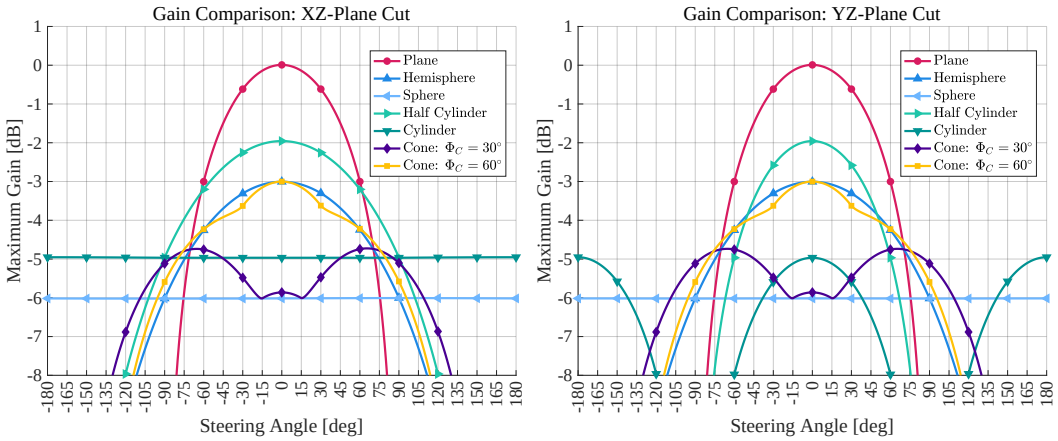
Figure 4.6: Maximum gain and cross-sectional area of conic arrays with different Φ_C.

this curvature decreases the sensitivity to steering angle in the $xy-$ plane cut, increasing the -3 dB steering range to 180° for the half-cylinder. The complete cylinder's gain is completely insensitive to the steering angle in the $xy-$ plane. The half and full cylindrical arrays have higher gain than the planar array starting at approximately 62° and 71.4°, respectively. Thus the cylindrical arrays strictly outperform the spherical arrays in the $xy-$ plane.

The cost of this increased performance is evident in the $xz-$ plane gain characteristic shown in Figure 4.5d. Due to the lack of curvature in this direction, the gain has the same angular sensitivity as a planar array. However as the curvature increases in the $xy-$ plane, the broadside gain in both axes decreases accordingly. Thus the cylindrical array is strictly worse than the planar array in the $xz-$ plane. It has the same -3 dB steering range of 120° and does not outperform the planar array at any angle. The only exception is for cylinders with $\phi_\alpha \geq 90°$, at which point *some* of the area is located on the far side of the array. This backside area creates an angular region in the $xz-$ plane where the gain is non-zero. However, a null must exist that isolates the two regions at $\theta = 90°$.

Conic Array

The maximum gain and cross-sectional area over the steering range of different right circular cones is shown in Figure 4.6. Once again the array is rationally symmetric so the behaviour is the same in both $xz-$ and $yz-$ plane cuts. When $\Phi_C = 180°$ the array is a plane, while at $\Phi_C = 1°$ the array is essentially a one dimensional line.



(a) Maximum Gain in $xz-$plane. Cylinders have curvature in this plane.

(b) Maximum Gain in $yz-$plane. Cylinders are straight in this plane.

Figure 4.7: Comparison of the maximum gain over steering range of different array geometries.

As can be seen, as Φ_C decreases, the maximum gain decreases as the gain becomes less sensitive to steering angle. However the maximum gain moves away from the axis of revolution, $\theta = 0$, once Φ_C is less than approximately 45°. As discussed in Section 4.4, this is the point where the the side of the array has larger cross sectional area despite the occlusion. As expected at $\Phi_C \approx 0$ the maximum gain is bifurcated at $\theta = \pm 90°$.

Interestingly, between the planar and bifurcated extremes the conic arrays exhibits a mix of their behavior. This decreases the sensitivity of the gain to steering angle, increasing the -3 dB to approximately 191° for $\Phi_C = 60°$ and 226° for $\Phi_C = 30°$. In addition, the two conic arrays perform better than the planar array at steering angles greater than 69.4° and 70.4°, respectively. Remarkable, the maximum gain of these arrays is only 3 dB and 4.7 dB less than planar; the arrays exhibit a better maximum gain vs. steering range trade-off than the spherical arrays.

Comparison of Specific Geometries

Figure 4.7 directly compares the gains of a key geometries, a plane, a hemisphere, a sphere, a half cylinder, a cylinder, and two cones. Each shape has been normalized to have the same surface area. In this analysis, the cylinder and half cylinder are both orientated along the y-axis and normal to the $z-$axis. The plane has the largest broadside gain and highest sensitivity to steering angle. The sphere has the lowest broadside gain but is insensitive to steering angle. The cylinder is insensitive to steering angle in one direction and has a larger broadside gain than the sphere.

Both the hemisphere and half cylinder have larger broadside gains than their full counterparts because their entire surface area is observable. It is thus clear that more curvature in an axis increases the steering range at the cost of decreasing the maximum gain.

Finally, the cones behave quite different to the other geometries due to their non-convex sensitivity to steering range. The $\Phi_C = 60°$ cone behaves similarly to, but slightly outperforms, the hemisphere. The $\Phi_C = 30°$ cone has the largest steering range in both directions with the exception of the sphere which has lower maximum gain.

4.6 Trade-off Between Maximum Gain and Steering Range

The results in Section 4.5 hint that there is a fundamental trade-off between the maximum gain and steering range. Curvature, or kinks in the case of a cone, extends the angular region where parts of the surface area are visible. Flatness on the other hand, concentrates the orientation of the surface area in a particular direction. Thus in a sense curvature spreads out the gain over a larger angular region, reducing the maximum gain but decreasing its angular sensitivity.

However, the trade-off between gain and steering range is not the same for every geometric shape. Plots of the change in the maximum gain and the -3 dB steering range versus the critical parameters, θ_α, ϕ_α, and Φ_C, are used to explore this trade-off. As the steering range of the cylindrical array is different in $xz-$ and $xy-$ plane cuts, the *average* steering range between cuts is used. In all plots, the dashed line corresponds with the maximum gain compared to the planar array while the solid line corresponds with the average steering range between cuts. The critical parameters are oriented so that the array is approximately planar on the left-hand side.

Spherical Array

Figure 4.8 shows the change in maximum gain and steering range of spherical caps as θ_α increases. As can be seen, the steering range does not appreciably increase until $\theta_\alpha \approx 45°$, at which point the maximum gain has decreased by over 0.5 dB. After this point the -3 dB steering range rapidly increases to 360° at $\theta_\alpha \approx 135°$. At this point, the geometry is not a complete sphere, but the gain never decreases -3 dB below its maximum. It is only at $\theta_\alpha = 180°$ that the array is completely spherical, and there is no variation in gain with steering angle.

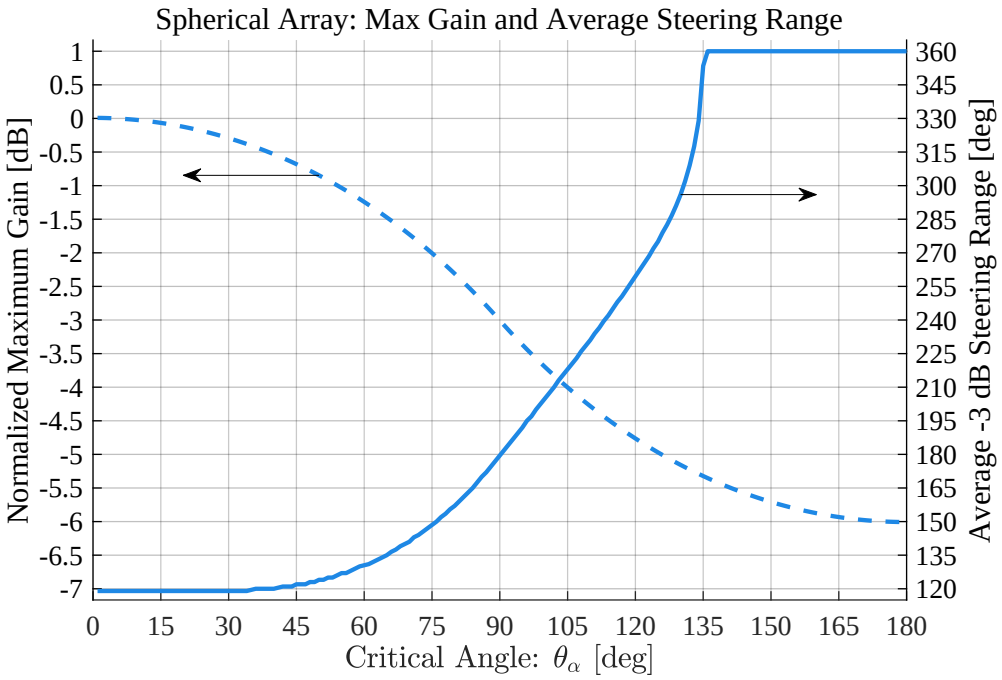


Figure 4.8: Trade-off between the maximum gain and the average -3 dB steering range for spherical arrays as θ_α changes. The $\theta_\alpha = 0°$ array is effectively planar.

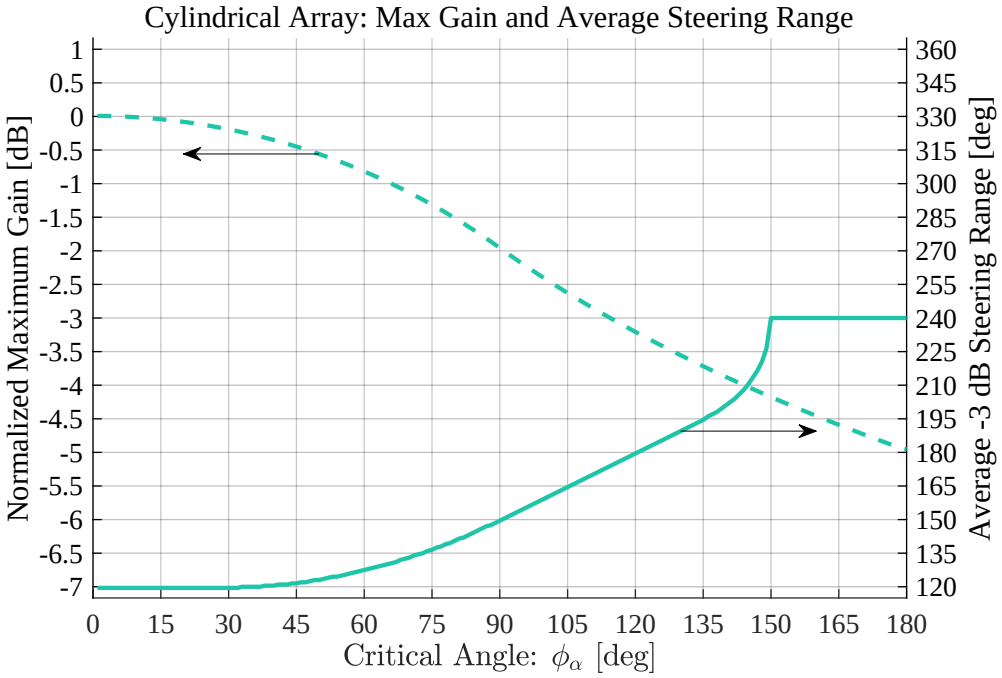


Figure 4.9: Trade-off between the maximum gain and the average -3 dB steering range for spherical cap arrays as ϕ_α changes. The $\phi_\alpha = 0°$ array is effectively planar. Note that the steering range increase is entirely in one plane.

Cylindrical Array

The change in maximum gain and steering range of cylinders with different ϕ_α is shown in Figure 4.9. The are similar to that of the spherical array, however, the changes in both gain and steering range are more gradual. An important note is that while the average steering range is reported for fair comparison, the increase in steering range is entirely concentrated in one direction while the other remains at 120°. Thus the steering range in the one axis can easily be derived from the chart by doubling the difference between the curve value and 120°. The average steering range of the array reaches 240° at $\phi_\alpha = 150$, at which point the difference in front and back side gain is less than 3 dB. It is only at $\phi_\alpha = 180$ that the cylinder is complete and there is no variation in gain along the curved axis.

Conic Array

Figure 4.10 shows the change in maximum gain and steering range of circular cones as the apex angle, Φ_C, decreases. Initially planar, the cone steering range only marginally increases as the gain drops off quickly. Then the gain and steering range trade roughly equally in the range of $105° \leq \Phi_C \leq 45°$. As mentioned when discussing Figure 4.6, the maximum gain moves away from the axis of revolution at $\Phi_C = 45°$ and the trade-off changes dramatically with the steering range increasing much faster than the maximum gain decreases. At around $\Phi_C = 20°$ however, the peaks in maximum gain bifurcate enough that the gain along the axis of revolution drops 3 dB below the maximum gain and the steering range splits into two distinct regions with a steering range approximately equal to 120°.

All Geometries

Figure 4.11 directly compares the trade-off between gain and average steering range for the different geometries. Each curve plots the maximum gain versus the steering range as its critical parameter changes. Each geometry starts in the top left corner as they are effectively planes with a normalized maximum gain of 0 dB by definition and a steering range of 120° However as curvature is introduced, the steering range increases while the gain decreased.

The optimal geometry for a desired steering range can thus be inferred by selecting the curve with the most gain at the appropriate location on the $x-$ axis. For example, conic arrays provide the most gain, approximately -2.75 dB, for a average steering range of 180°. As can be seen, the relative trade-off behavior is fairly complex as the optimum geometry alternates between cylindrical, conic, and spherical arrays.

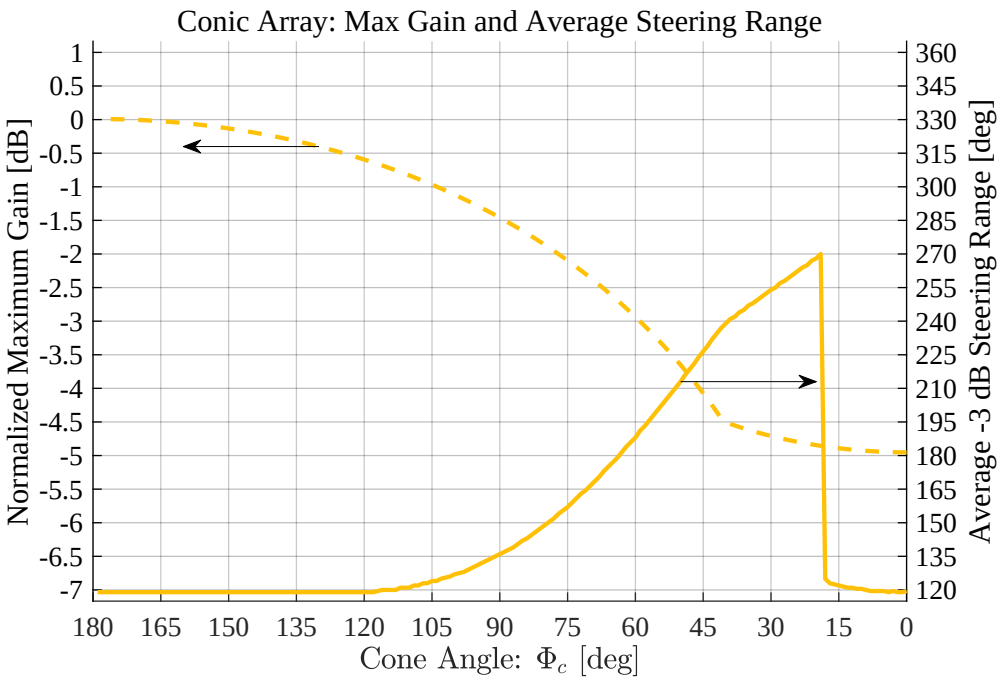


Figure 4.10: Trade-off between the maximum gain and the average −3 dB steering range for conic arrays as Φ_C changes. The $\Phi_C = 180°$ array is effectively planar.

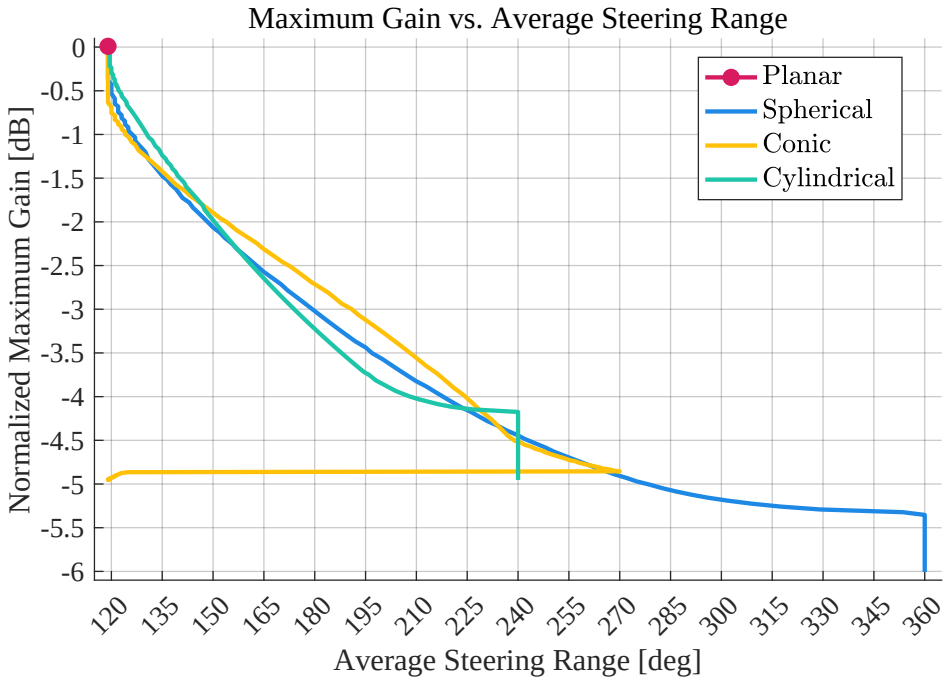


Figure 4.11: Direct comparison of the trade-off between the maximum gain and the average −3 dB steering range for different array geometries. Note that the cylindrical array's improvements in steering range are entirely in one plane.

Table 4.1: Summary of Array Performance for Different Geometries.

Geometry	Max Gain [dB]	3dB Steering Range [deg]	Surpasses Planar [deg]
Planar	0	120	-
Hemisphere	-3	180	70.5
Sphere	-6	360	75.5
Half-Cylinder	-1.96	180 [xy], 120 [z]	62.1 [xy], Never [z]
Cylinder	-4.97	360 [xy], 120 [z]	71.4 [xy], Never [z]
Cone ($\Phi_C = 30$)	-4.7	225.8	70.4
Cone ($\Phi_C = 60$)	-3	191.2	69.4

However, it is clear that only spherical caps can have an average steering range above 270°.

Again, the improved steering range of the cylindrical array is entirely in one axis yet the average is reported. Interestingly, the cylindrical arrays provide the most performance at low average steering ranges. The single axis steering range trade-off with gain can be derived from Figure 4.11 by doubling the x coordinate of each point in the cylindrical curve. By doing this it is immediately apparent that the cylindrical array always provides the most gain for a given steering range in one axis.

Table 4.1 provides a summary of the performance of the different basic geometries evaluated throughout the chapter. As can be clearly seen, cylinders are optimal if a large steering range is only necessary in one axis. The spherical array provides the largest steering range of any array while the planar array provides the largest gain. For intermediate steering ranges, conic arrays provide more gain than the spherical caps. Finally, all curved geometries provide higher gain than planar for beams steered somewhere between 62° and 76°.

References

[1] R. Guy, "Spherical coverage from planar, conformal and volumetric arrays," in *IEE National Conference on Antennas and Propagation*, Mar. 1999, pp. 287–290, ISBN: 0 85296 713 6. DOI: 10.1049/cp:19990072. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/cp_19990072.

[2] L. Marantis, P. Brennan, and A. Kanatas, "Spherical harmonic theory investigations for spherical antenna arrays," in *2020 14th European Conference on Antennas and Propagation (EuCAP)*, Mar. 2020, pp. 1–5. DOI: 10.23919/EuCAP48036.2020.9135326.

[3] A. Hizal, "Wide angle scanning conformal phased array on a spherical surface," in *2013 IEEE International Symposium on Phased Array Systems and Technology*, Oct. 2013, pp. 259–266, ISBN: 978-1-4673-1127-4. DOI: 10.1109/ARRAY.2013.6731838.

[4] P. Bevelacqua and C. Balanis, "The influence of array geometry on system performance," in *2006 IEEE Antennas and Propagation Society International Symposium*, 2006, pp. 3327–3330, ISBN: 1-4244-0123-2. DOI: 10.1109/APS.2006.1711325.

[5] W. Kummer, "Basic array theory," *Proceedings of the IEEE*, vol. 80, pp. 127–140, 1 1992, ISSN: 00189219. DOI: 10.1109/5.119572.

[6] L. Josefsson and P. Persson, *Conformal Array Antenna Theory and Design*. John Wiley & Sons, Inc., 2006.

[7] M. Hopkins, J. Tuss, A. Lockyer, *et al.*, "Smart skin conformal load-bearing antenna and other smart structures developments," in *38th Structures, Structural Dynamics, and Materials Conference*, Apr. 1997. DOI: 10.2514/6.1997-1163.

[8] S. Schneider, C. Bozada, R. Dettmer, and J. Tenbarge, "Enabling technologies for future structurally integrated conformal apertures," in *IEEE Antennas and Propagation Society International Symposium. 2001 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.01CH37229)*, 2001, pp. 330–333, ISBN: 0-7803-7070-8. DOI: 10.1109/APS.2001.959730. [Online]. Available: http://ieeexplore.ieee.org/document/959730/.

[9] V. Jaeck, L. Bernard, K. Mahdjoubi, *et al.*, "Conformal phased array in a small conical shape for communications at 5.2 ghz," in *2016 IEEE International Symposium on Phased Array Systems and Technology (PAST)*, Oct. 2016, pp. 1–6, ISBN: 978-1-5090-1447-7. DOI: 10.1109/ARRAY.2016.7832633.

[10] Y. Dong, F. Cheng, and L. Cao, "Performance analysis of conformal conical array for airborne radar," in *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, Dec. 2019, pp. 1–4, ISBN: 978-1-7281-2345-5. DOI: 10.1109/ICSIDP47821.2019.9173024.

[11] Y. F. Wu and Y. J. Cheng, "Conical conformal shaped-beam substrate-integrated waveguide slot array antenna with conical-to-cylindrical transition," *IEEE Transactions on Antennas and Propagation*, vol. 65, pp. 4048–4056, 8 Aug. 2017, ISSN: 0018-926X. DOI: 10.1109/TAP.2017.2716404.

[12] Y. Xia, B. Muneer, and Q. Zhu, "Design of a full solid angle scanning cylindrical-and-conical phased array antennas," *IEEE Transactions on Antennas and Propagation*, vol. 65, pp. 4645–4655, 9 Sep. 2017, ISSN: 0018-926X. DOI: 10.1109/TAP.2017.2730241.

[13] C. Fulton, J. L. Salazar, Y. Zhang, *et al.*, "Cylindrical polarimetric phased array radar: Beamforming and calibration for weather applications," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, pp. 2827–2841, 5 May 2017, ISSN: 0196-2892. DOI: 10.1109/TGRS.2017.2655023.

[14] G. Zhang, S. Karimkashi, L. Lei, *et al.*, "A cylindrical polarimetric phased array radar concept - a path to multi-mission capability," in *2013 IEEE International Symposium on Phased Array Systems and Technology*, Oct. 2013, pp. 481–484, ISBN: 978-1-4673-1127-4. DOI: 10.1109/ARRAY.2013.6731875. [Online]. Available: http://ieeexplore.ieee.org/document/6731875/.

[15] H. Steyskal, A. Hessel, and J. Shmoys, "On the gain-versus-scan trade-offs and the phase gradient synthesis for a cylindrical dome antenna," *IEEE Transactions on Antennas and Propagation*, vol. 27, pp. 825–831, 6 Nov. 1979, ISSN: 0096-1973. DOI: 10.1109/TAP.1979.1142192.

[16] P. Knott, "Design and experimental results of a spherical antenna array for a conformal array demonstrator," in *2007 2nd International ITG Conference on Antennas*, Mar. 2007, pp. 120–123, ISBN: 978-3-00-021644-2. DOI: 10.1109/INICA.2007.4353945.

[17] I. J. Gupta, T.-H. Lee, K. A. Griffith, *et al.*, "Non-planar adaptive antenna arrays for gps receivers," *IEEE Antennas and Propagation Magazine*, vol. 52, pp. 35–51, 5 Oct. 2010, ISSN: 1045-9243. DOI: 10.1109/MAP.2010.5687504.

Chapter 5

BENEFITS AND DRAWBACKS OF SHAPE CHANGING PHASED ARRAYS

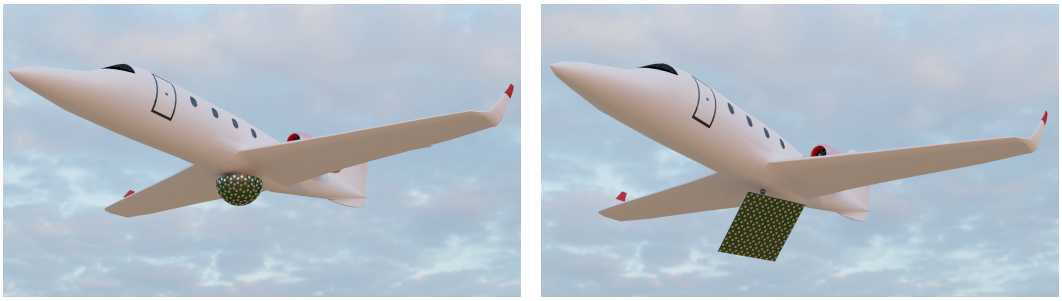
Out of the gloom, a figure approaches
wrapped in a garment of northern light.
Heaves me free. Gives me a potato.
Escorts me to the edge of town,
then turns on the heel.
Failing to hide, a tail striped,
following always the cusp of night.

Richard Dawson
Shapeshifter

The results of Chapter 4 suggests that an array could break the trade-off between gain and steering range by reconfiguring its geometry. This chapter explores the theoretic advantages and challenges of such shape-changing phased arrays that are experimentally explored in the subsequent chapters. It begins with a discussion of the benefits of shape-changing phased arrays and their possible applications in Section 5.1. Then, Section 5.2 presents the mathematical restrictions of shape change imposed by Gauss's *Theorema Egregium*. Section 5.3 follows with a brief overview of the art of origami as it provides insight into addressing these restrictions. Prior work on origami-inspired electromagnetic structures is reviewed in Section 5.4 before Section 5.5 concludes the chapter discussing the effects of increased spacing.

5.1 Shape-Changing Phased Array Benefits and Applications

As discussed in Chapter 4, the array geometry presents a fundamental trade-off between steering range and maximum gain. Curvature is a given plane reduces sensitivity of gain to steering angle in that plane, thus extending the steering range. However, this curvature directly reduces the broad-side cross sectional area and thus the maximum achievable gain. More broadly, the array geometry constrains the radiation patterns that can be synthesized by restricting the locations and orientations of potential surface currents.



(a) Hemispherical array to rapidly search a large region for a target.

(b) Planar array to boost signal fidelity once target is located.

Figure 5.1: Shape-changing phased arrays enable multi-phase applications such as search and rescue for a lost hiker.

Much of the exploration of phased arrays has assumed that the geometry is fixed, limiting the space of achievable radiation patterns. There is thus an opportunity to enhance the capabilities of phased arrays by introducing dynamic geometric reconfigurability into the design space. Phased arrays capable of *in situ* geometric reconfiguration can break the trade-off between steering range and gain and increases the diversity of radiation patterns that can be synthesized by the array.

Such shape-changing phased arrays enable new applications by dynamically conforming their shapes to the geometry best suited for a given task. Thus a shape-changing array can function as an array version of the Swiss Army knife, performing multiple functions as part of different applications. However, shape-changing phased arrays also enable a single array to serve multiple functions as part of the same operation.

Figure 5.1 illustrates an example of such a multi-phase operation, search and rescue. Imagine an airplane is flying over a dense forest trying to find a lost hiker. The plane is equiped with a shape-changing phased array functioning either as a telecommunications array or as a radar. Initially, the array is in a hemisphere configuration in order to rapidly scan the full forest below as it searches for the hiker. Once a signal is detected and the location of the hiker is identified, the array changes into a planar configuration pointed in the correct direction. This planar array maximizes the signal strength to enable communication with the hiker. From a radar perspective, the planar array decreases the array's beam-angle, thus providing more angular resolution and a more detailed radar image.

Another advantage provided by shape-changing phased arrays is that they can tune

Figure 5.2: Rapidly deployable nodes with shape-changing phased arrays can create a self-configuring communications network for disaster relief. Nodes located near users take on hemispherical shapes to increase coverage. Other nodes, such as those on the top of buildings, form planar arrays to boost SNR for high throughput back-haul links.

the trade-off between gain and steering range by increasing the curvature as needed. Therefore, shape-changing phased arrays are beneficial for applications where the array needs are not known in advance. For example, consider a region that has just been struck by a disaster like an earthquake or a hurricane. Such disasters often take out telecommunications systems precisely when they are needed most to coordinate disaster relief, communication about the disaster to the effected public, and help victims connect with friends and family. Despite their importance, it can take a long period of time to restore functionality to these networks. It took a week for cell service to be restored to parts of Florida after Hurricane Michael made landfall, and parts of Puerto Rico were without service for months after Hurricane Maria [1]. The challenge with telecommunications in a disaster zone is the variability in the environment. Network designers put a great deal of care into the selection and construction of mobile sites to ensure optimal performance. In a disaster-relief zone there is no such luxury; there might not be an ideal location in a neighborhood filled with floodwaters and destroyed houses.

Shape-changing phased arrays could have a major impact on disaster response by enabling a network of portable nodes that can be easily carried into a recovery area by disaster response personnel as shown in Figure 5.2. These nodes would form a

self-configuring network, with the array geometry at each node adapting to suit the needs of the network. End nodes would take on spherical or cylindrical geometries in order to maximize areal coverage and communicate with end users. Other nodes would form the network backbone, taking on planar shapes in order to maximize bandwidth to support back-haul. As it is not initially clear what the situation on the ground is, and thus what the network needs are, it is advantages to have an array that can adapt to any need. In addition, shape-change allows nodes to adapt if the network needs change due to more nodes being added or nodes being disabled by the potentially hostile environment.

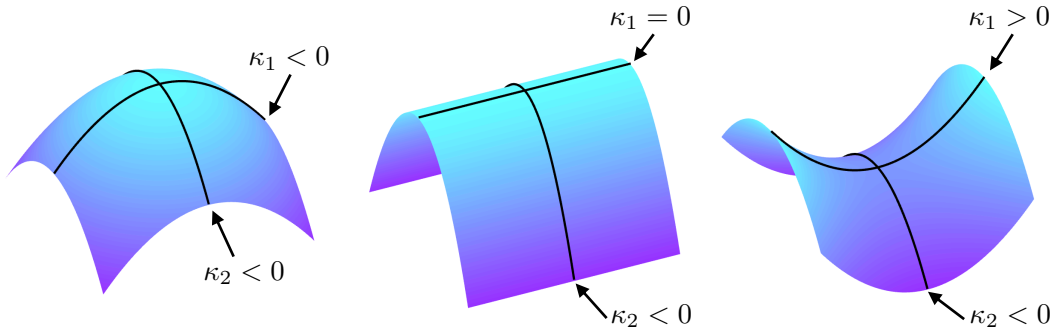
The ability to tune array performance is useful for other applications as well. Shape-change could help with the regular roll-out of telecommunications networks by allowing one antenna to serve multiple functions and adjust to the precise needs of the deployment site. Space applications benefit from in-situ tuning to adjust performance post-launch. With rapid shape-change, minor adjustments in array geometry in real time could be used to enhance beam performance and isolation in MIMO systems depending on the locations of users.

Finally, a more abstract but academically interesting application is the development of arbitrary radiators. Shape-change removes the restrictions imposed on surface currents imposed by the geometry, greatly increasing the radiation patterns that can be synthesized. Truly arbitrary RF synthesis may be possible by combining shape change with recent advances in re-configurable RF systems [2], [3]. Such arbitrary control would enable very precise beam synthesis and compensate for limitations in the element pattern.

5.2 Implications of Gauss's *Theorema Egregium*

In addition to introducing new opportunities, shape-changing phased arrays also introduce new challenges that must be explored. In addition to the evident mechanical and electrical implementation issues, there are mathematical consequences to the abstract notion of shape-change due to Gauss's *Theorema Egregium*[1]. To understand this theorem and its ramifications, we must establish the concept of Gaussian curvature.

[1] Latin for "Remarkable Theorem."



(a) $K = \kappa_1\kappa_2 > 0$ (b) $K = \kappa_1\kappa_2 = 0$ (c) $K = \kappa_1\kappa_2 < 0$

Figure 5.3: Examples of geometries with different Gaussian curvature, K.

Gaussian Curvature

The *curvature*, κ, of a curve in $\mathbb{R}^3$ is a measure of how much it deviates from a straight line, with the sign indicating the direction of the deviation. Thus any curve along a 2-dimensional surface in $\mathbb{R}^3$ will have some curvature. While there are an infinite number of such curve that go through a given point P on the surface, there is always at least one with a maximum curvature, κ_1, and at least one with a minimum curvature κ_2. These maximum and minimum curvatures are the *principle curvatures* of the surface at the point P. The *Gaussian curvature*, K, of a surface at a given point is the product of the principle curvatures as shown in Equation 5.1.

$$K = \kappa_1\kappa_2 \tag{5.1}$$

The sign and value of the Gaussian curvature dictate the shape of a surface at a given point. As shown in figure 5.3, a positive value indicates that any curve through the point will deviate in the same direction, forming a dome. A negative value however indicates that the deviation of deviation is opposite, forming a saddle point. Zero Gaussian curvature indicates that a straight line in $\mathbb{R}^3$ can be drawn through the point on the surface. Gaussian curvature is a local measure defined that can vary at different points along the surface. While most shapes will have regions with positive K and regions with negative K, some geometries have constant Gaussian curvature along their surfaces. For example, the Gaussian curvature of a sphere of radius R is R^{-2}. Also, a plane has constant zero Gaussian curvature because it does not curve anywhere. Other shapes that have constant zero Gaussian curvature are refered to as *developable surfaces* or *flat surfaces* [4].

Gauss's *Theorema Egregium*

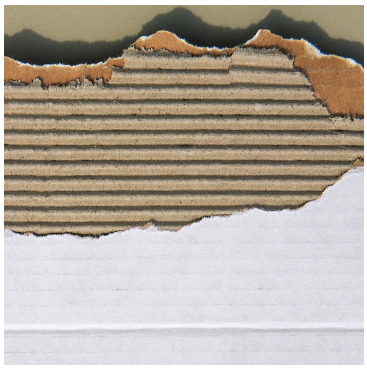
Gauss's *Theorema Egregium* states that "the Gaussian curvature of a surface is invariant under local isometry" [4]. This means that if a surface is transformed in such a way that the *distance* between each pair of points on the surface is preserved, than the Gaussian curvature of the new surface is the same as the old. Of more immediate interest, however, is the contrapositive; if the Gaussian curvature of the new surface is *not* the same, than the distance between each pair of points on the surface is *not* preserved. Thus in order to transform a surface of one Gaussian curvature, like a plane, into another with a different curvature, like a sphere, than the distance between at least one pair of points on the surface *must* change.

The *Theorema Egregium* applies physical structures, and therefore any deformation that changes the Gaussian curvature must also change the distance between composite elements. For stretchable materials, the atomic lattice spacing changes or polymer chains fold/unfold to allow the material length to change. For example as a latex balloon inflates into larger spheres, the latex stretches to account for the change in Gaussian curvature. A rigid structure, however, cannot stretch and thus must split into segments with gaps in between. For example, chain mail is made of interlocking metal rings that are rigid individually, but can conform to different shapes as the gaps within the rings allow the distance between them to adjust.

Necessity of Increased Element Spacing and Gaps

The *Theorema Egregium* has numerous applications, and profound consequences, in a variety of fields. Planar rigid materials like paper or a sheet of steel can be corrugated in order to increase their strength in one direction. Because planes have zero Gaussian curvature, if one direction is curved than the orthogonal direction must not be curve or it will tear (introducing gaps). Thus corregated materials leverage the shear strength of the material to counteract forces that would introduce additional curvature. This is precisely why folding a slice of pizza provides additional rigidity. Gauss's *Theorema Egregium* also impacts cartography as maps of the globe must be distorted when drawn on a planar sheet. The theorem also explains why it is impossible to neatly wrap a basketball as a gift; the wrapping paper must crumple or tear to sit flat on the surface of the ball.

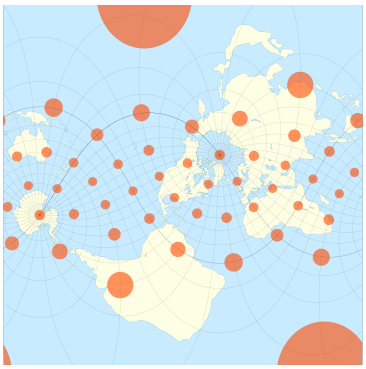
While stretchable electronics is a subject of ongoing research and development, it is still an immature technology [5], [6]. Instead, this work focuses on rigid shape-changing arrays made out of traditional materials in order to explore the their



(a) Cardboard is corrugated to increase its strength.
Public Domain.

(b) Folding a slice of pizza prevents sagging. Cropped Photo by Louise Ma / WNYC.

(c) Trying to wrap a ball as a gift creates unsightly wrinkles and folds.



(d) A planar map of the earth must be distorted.
© Justin Kunimune / Wikimedia Commons / CC-BY-SA-4.0

Figure 5.4: Applications and effects of Gauss's *Theorema Egregium*.

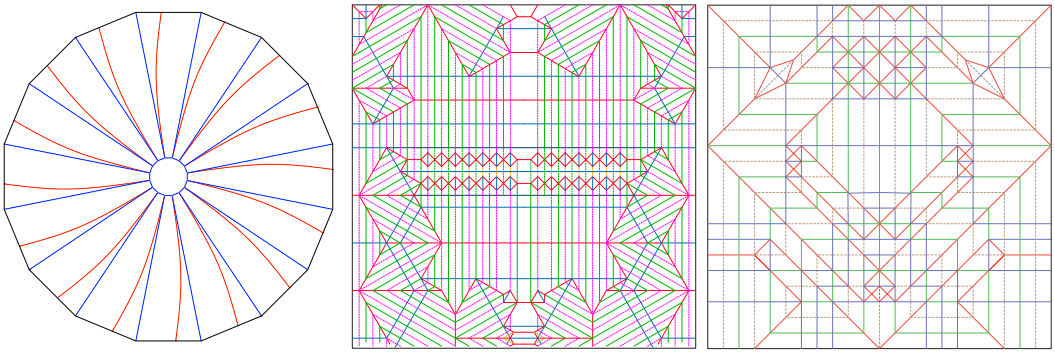
fundamental challenges and solutions. Modern printed circuit boards and antennas are commonly comprised of rigid metal and dielectrics that cannot stretch, flex, or compress. While phased arrays built on flexible substrates have recently been demonstrated [7]–[10], such substrates are flexible in a sense similar to a sheet of paper; they can bend but they cannot compress or stretch. Thus even flexible arrays are considered "rigid" by the *Theorema Egregium*.

Questions arise from the rigidity of electronic materials and the *Theorema Egregium*. Is it mathematically possible for an array of rigid elements to perform *arbitrary* shape change? If so, how can it be accomplished and what are the limitations? If gaps must be introduced between antennas, how does this effect the electromagnetic properties of the array? Can this effect be leveraged or compensated for? The rest of this work seeks to address these questions through theory, simulations, and experiment.

(a) "Fatpot16" [11]

(b) "Cyclomatus Metallifer" [12]

(c) "Dragon Drop" [13]



(d) "Fatpot16" Crease Pattern

(e) "Cyclomatus Metallifer" Crease Pattern

(f) "Dragon Drop" Crease Pattern

Figure 5.5: Examples of origami artwork and crease patterns. All works by Robert J. Lang. Used with permission.

5.3 Background on Origami

The mathematical restraints on shape-change enforced by Gauss's *Theorema Egregium* can be addressed by looking to origami for inspiration. Origami is the practice of folding planar sheets of paper into beautiful works of art. While origami traditionally discourages cutting the paper, the related art kirigami uses both folds and cuts. Both practices rely on intricate folding patterns to create potentially complex geometries as shown in Figure 5.5.

Because paper ideally does not stretch, it cannot change its Gaussian curvature as distances along its surface cannot be altered. An ideal fold does not alter the Gaussian curvature because the curvature of the fold line is always zero. Every possible piece of origami art has the same Gaussian curvature as a sheet of paper, zero[2]. Yet, origami art can take forms that seem to have non-zero Gaussian curvature.

[2]Strictly speaking, physical paper can stretch to some degree and realized folds are not ideal. In addition, techniques like wet-folding are sometimes used to improve the paper flexibility [14]. Therefore, some amount of change in Gaussian curvature is possible in origami. However, these factors do not affect the theoretical validity of this section.

Figure 5.6: "Rattlesnake" by Robert J. Lang. Used with permission [15].

While it appears that both arts violate Gauss's *Theorema Egregium*, they actually circumvent it by *emulating* arbitrary shapes without changing the Gaussian curvature of the paper itself. They do not violate the theorem because distances along the surface *of the paper* do not change. The geometry created by the folding pattern is only illusory; the full surface area of the object is concealed by its outermost layers.

The *Theorema Egregium* is apparent when considering just the external surface area of the origami object. For example, an origami sphere has a different gaussian curvature than a sheet of paper and thus much change its surface area. Indeed the external surface area shrinks during folding as portions of the planar surface area are hidden inside the sphere. Thus, two points on the piece of paper are located closer to each other on the emulated spherical surface. The necessity of gaps is evident when considering the opposite transformation. The distance between two points on the surface of the sphere increases as the paper unfolds; these potential "gaps" are filled by the hidden paper.

The use of cuts in kirigami is another method by which the *Theorema Egregium* can be circumvented. Cuts introduce gaps that allow the distance between points on either side of the cut to increase. This change in distance allows the Gaussian curvature of the emulated surface to change and its surface area to grow. However, the emulated surface will not be completely filled by the original surface area as the

gaps created by cuts are empty.

It is also important to note that the non-folding portions of the origami shape, the *facets*, are limited to shapes with zero Gaussian curvature because the *Theorema Egregium* applies locally. Therefore, it is impossible to create a perfect sphere with origami as the shape will always be comprised of facets that only curve in one direction. This mathematical restriction is apparent when considering the sphere-like origami pot in Figure 5.5a. However, this restriction can be nearly unnoticeable with sufficiently small facets and clever folding, as clearly illustrated in Figure 5.6.

Origami and Kirigami demonstrate important consequences of rigid shape-change.

I Folds enable rigid materials to emulate arbitrary shapes.

II Portions of the surface area must be hidden when the surface area shrinks.

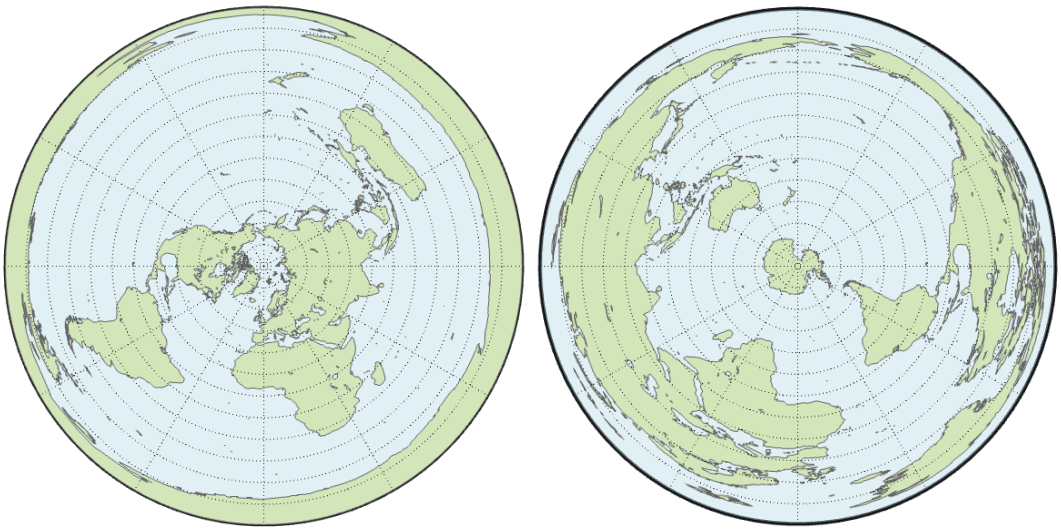
III Surface gaps must be introduced when the surface area grows.

IV Non-folding portions of the shape cannot change their local Gaussian curvature.

5.4 Prior Art of Shape-Changing Electromagnetics

The idea of combining origami folding techniques and antennas is not new. There has been much work on origami antennas, that is, single element radiating structures that use origami to achieve a particular goal such as deployability [16], [17] or for adjustable frequency [18]–[20], polarization [21], and radiation pattern [22], [23]. In general, these structures are single port devices whose radiation properties are not electronically controllable. While some origami antennas have been embedded into an array, in these cases the array itself is fixed; the shape change occurs within an element and not across the array.

Exploration of mechanical reconfigurability across an array has been much more limited. Origami has been used to design frequency-selective surfaces [24], [25]; mechanically reconfigurable arrays of passive elements that can change shape in order to alter their interactions with incident electromagnetic fields. They are essentially reconfigurable RF metamaterials and show promise for novel antenna designs and applications. However, the lack of ports on the elements precludes these surfaces from being used as antennas. In addition, reconfigurable patch arrays with at most four elements have been explored [26], [27]. However, these arrays are driven with a single feed and are not capable of beam steering like a phased array,



(a) Centered at the North Pole. (b) Centered at the South Pole.

Figure 5.7: Maps of the Earth using the azimuthal equidistant projection.

and thus behave like a single antenna element. The full opportunity presented by geometric reconfigurable arrays has not been explored.

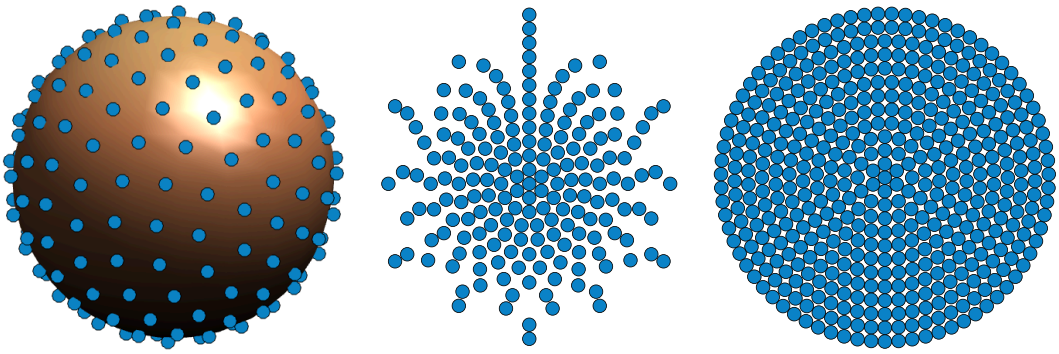
5.5 Consequences of Increased Spacing and Gaps

As discussed in Section 5.2, the spacing between elements must change in order to enable shape-change. As discussed in Section 2.4, the element spacing of an array determines the locations and strengths of side lobes, the locations and strengths of grating lobes, and the field of view. Thus, Gauss's *Theorema Egregium* has profound consequences on array performance.

Assume that there is a shape-changing array, rigid or stretchable, that can shift between a planar and a spherical configuration. As discussed, the distances between its elements must change. In general, a spacing below $\frac{\lambda}{2}$ is undesirable due to the associated increase in coupling that can degrade the element matching, efficiency, and radiation pattern. Therefore, it is reasonable to design the array so that the spacing is $\frac{\lambda}{2}$ in one configuration and is larger in the other. In this analysis we explore an array with minimum spacing in the planar state[3].

In order to demonstrate how shape-change effects side lobes, a transformation between points on the two geometries must be selected. Unfortunately there are an unlimited number of such transformations and the transformation of a realized array would be determined by the specifics of its mechanical structure. Thus there is not

[3] A similar analysis can be performed with the opposite assumption.



(a) Densely packed spherical array.

(b) Unwrapped spherical array.

(c) Densely packed circular array.

Figure 5.8: Demonstration of the effect of Gauss's Theorema Egregium on element spacing. (a) A densely packed spherical array with elements separated by approximately 0.5λ. (b) The location of elements in the same array after being mapped to a planar array using the azimuthal equidistant projection. (c) A densely packed circular array with the same aperture but elements separated by approximately 0.5λ.

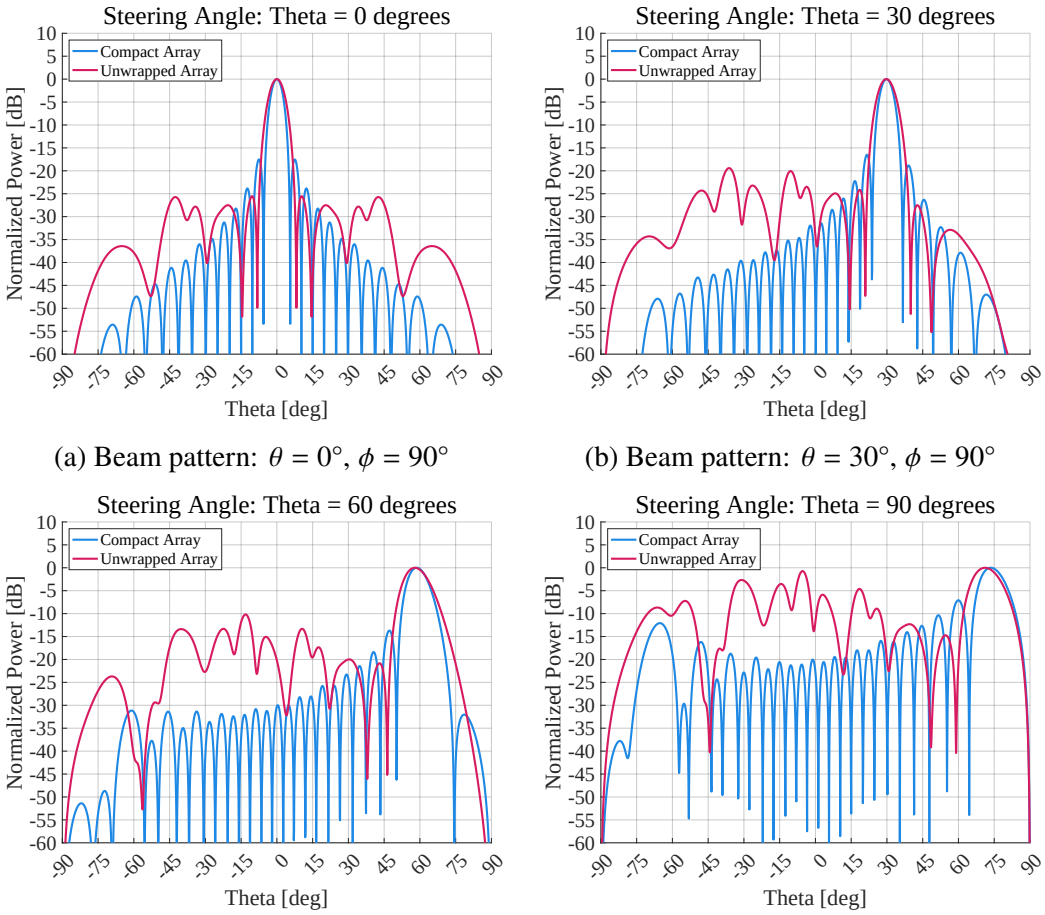
a set effect on side lobes and so any transformation selected in abstract is arbitrary. That said, a reasonable transformation choice is the *azimuthal equidistant projection* between a spherical array and a circular planar array [28]. The azimuth equidistant projection is formed by selecting a point on the sphere and drawing lines of constant azimuth centered on that point. The selected point then becomes the center of the planar array and the azimuth lines are "unrolled" to lay flat in the plane like a star. The mapping between spherical coordinates to polar coordinates is given in Equations 5.2 and 5.3. This projection thus preserves the radial spacing of elements and their azimuth coordinates. Thus the distortion is entirely in the azimuth direction and is concentrated at edge of the circular array. Figure 5.7 illustrates this distortion using azimuthal equidistant projections of the Earth centered on the north and south poles.

$$r_{polar} = R_{sphere}\theta_{sphere} \tag{5.2}$$

$$\theta_{polar} = \phi_{sphere} \tag{5.3}$$

The azimuthal equidistant projection gives good insight into the effect of shape change on side lobes because it maintains both the radial spacing and the azimuthal symmetry at each radius. Therefore, the distortion is fairly evenly distributed along all azimuth cuts of the array[4].

[4] As a comparison, the Mercator projection exhibits different spacial distortion in the x- and y-plane cuts.



(a) Beam pattern: $\theta = 0°$, $\phi = 90°$

(b) Beam pattern: $\theta = 30°$, $\phi = 90°$

(c) Beam pattern: $\theta = 60°$, $\phi = 90°$

(d) Beam pattern: $\theta = 90°$, $\phi = 90°$

Figure 5.9: Comparison of beam patterns for the unwrapped and circular arrays in Figure 5.8. Beams are steered every 30°. As can be seen, the arrays have similar beam widths as they have approximately the same aperture. However, the decreased element density in the unwrapped array greatly increases the side lobes, especially at wide steering angles.

Figure 5.8a depicts a spherical array with densely packed elements; the distance between elements on the surface is approximately, and no less than, $\frac{\lambda}{2}$. Using the azimuthal equidistant projection, the antenna elements are mapped to the circular planar array shown in Figure 5.8b. The minimum spacing between elements remains $\frac{\lambda}{2}$, however the spacing increases greatly at the edge of the array reducing the fill factor and increasing the grating lobes. We compare this "unwrapped array" to a densely packed circular array with the same aperture as shown in Figure 5.8c. As with the spherical array, the distance between elements in this "dense array" is approximately, and no less than, $\frac{\lambda}{2}$. Notwithstanding the improvements that can be made by optimizing element placement or tapering the element drive, Figures

5.8a and 5.8c are representative of the ideal spherical and planar arrays that do not undergo shape change and thus illustrate their expected grating lobe performance. Thus the general degredation in phased lobes due to shape change can be explored by comparing the patterns of the arrays in Figures 5.8b and 5.8c.

Figure 5.9 shows the normalized beam patterns for the dense and unwrapped arrays when steered to $\theta = 0°$, $30°$, $60°$, and $90°$ in the $\phi = 0°$ direction. The element pattern is modeled as $\cos\theta$. The unwrapped array exhibits large side lobes at every steering angle, often 20 dB higher than those in the dense array pattern at the same angles. The dense array is able to suppress the grating lobes a $\theta = 90°$ by over 10 dB, while the unrwapped array contains numerous side lobes within a few decibels of the main lobe. One important note is that the pseudo-random distribution of elements at the fringe of the unwrapped array does distribute the power radiated in side lobes across the pattern and unlike the dense array. The peak side lobes of the dense array, which are located immediately next to the main lobe, are higher than the peak side lobes of the unwrapped array for the $\theta = 0°$ and $\theta = 30°$ patterns. However, the total power is radiated in the side lobes is less in dense array due to its higher fill factor.

References

[1] T. Frank, *Cell phone service must be restored quicker after hurricanes*, Oct. 2019. [Online]. Available: https://www.scientificamerican.com/article/cell-phone-service-must-be-restored-quicker-after-hurricanes/.

[2] C. R. Chappidi, X. Lu, X. Wu, and K. Sengupta, "Antenna preprocessing and element-pattern shaping for multi-band mmwave arrays: Multi-port transmitters and antennas," *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2020, ISSN: 0018-9200. DOI: 10.1109/JSSC.2020.2967545.

[3] X. Lu, C. R. Chappidi, X. Wu, and K. Sengupta, "Antenna preprocessing and element-pattern shaping for multi-band mmwave arrays: Multi-port receivers and antennas," *IEEE Journal of Solid-State Circuits*, pp. 1–1, 2020, ISSN: 0018-9200. DOI: 10.1109/JSSC.2020.2967544.

[4] A. Pressley, *Elementary Differential Geometry*. Springer London, 2010, ISBN: 978-1-84882-890-2. DOI: 10.1007/978-1-84882-891-9. [Online]. Available: http://link.springer.com/10.1007/978-1-84882-891-9.

[5] Z. Huang, Y. Hao, Y. Li, *et al.*, "Three-dimensional integrated stretchable electronics," *Nature Electronics*, vol. 1, pp. 473–480, 8 Aug. 2018, ISSN: 2520-1131. DOI: 10.1038/s41928-018-0116-y.

[6] M. J. Yun, Y. H. Sim, D. Y. Lee, and S. I. Cha, "Highly stretchable large area woven, knitted and robust braided textile based interconnection for stretchable

electronics," *Scientific Reports*, vol. 11, p. 4038, 1 Dec. 2021, ISSN: 2045-2322. DOI: 10.1038/s41598-021-83480-x.

[7] M. D. Kelzenberg, P. Espinet-Gonzalez, N. Vaidya, *et al.*, "Ultralight energy converter tile for the space solar power initiative," in *2018 IEEE 7th World Conference on Photovoltaic Energy Conversion (WCPEC) (A Joint Conference of 45th IEEE PVSC, 28th PVSEC & 34th EU PVSEC)*, Jun. 2018, pp. 3357–3359, ISBN: 978-1-5386-8529-7. DOI: 10.1109/PVSC.2018.8547403.

[8] A. C. Fikes, A. Safaripour, F. Bohn, B. Abiri, and A. Hajimiri, "Flexible, conformal phased arrays with dynamic array shape self-calibration," in *2019 IEEE MTT-S International Microwave Symposium (IMS)*, Jun. 2019, pp. 1458–1461, ISBN: 978-1-7281-1309-8. DOI: 10.1109/MWSYM.2019.8701107.

[9] M. Gal-Katziri, A. Fikes, F. Bohn, B. Abiri, M. R. Hashemi, and A. Hajimiri, "Scalable, deployable, flexible phased array sheets," in *2020 IEEE/MTT-S International Microwave Symposium (IMS)*, Aug. 2020, pp. 1085–1088, ISBN: 978-1-7281-6815-9. DOI: 10.1109/IMS30576.2020.9224066.

[10] A. Fikes, O. S. Mizrahi, A. Truong, F. Wiesemuller, S. Pellegrino, and A. Hajimiri, "Fully collapsible lightweight dipole antennas," in *2021 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting (APS/URSI)*, Dec. 2021, pp. 545–546, ISBN: 978-1-7281-4670-6. DOI: 10.1109/APS/URSI47566.2021.9704302.

[11] R. J. Lang, *Fatpot16*, 2008. [Online]. Available: https://langorigami.com/artwork/fatpot16/.

[12] ——, *Cyclomatus metallifer, opus 562*, 2010. [Online]. Available: https://langorigami.com/artwork/cyclomatus-metallifer-opus-562/.

[13] ——, *Dragon drop, opus 823*, 2022. [Online]. Available: https://langorigami.com/artwork/dragon-drop/.

[14] M. Rao, *'wet' origami artist turns damp paper into gorgeous sculptures*, Aug. 2015.

[15] R. J. Lang, *Rattlesnake, opus 539*, 2008. [Online]. Available: https://langorigami.com/artwork/rattlesnake-opus-539/.

[16] J. Costantine, Y. Tawk, I. Maqueda, *et al.*, "Uhf deployable helical antennas for cubesats," *IEEE Transactions on Antennas and Propagation*, vol. 64, pp. 3752–3759, 9 Sep. 2016, ISSN: 0018-926X. DOI: 10.1109/TAP.2016.2583058. [Online]. Available: http://ieeexplore.ieee.org/document/7496961/.

[17] S. I. H. Shah, M. M. Tentzeris, and S. Lim, "A deployable quasi-yagi monopole antenna using three origami magic spiral cubes," *IEEE Antennas and Wireless Propagation Letters*, vol. 18, pp. 147–151, 1 Jan. 2019, ISSN: 1536-1225. DOI: 10.1109/LAWP.2018.2883380. [Online]. Available: https://ieeexplore.ieee.org/document/8543872/.

[18] X. Liu, S. V. Georgakopoulos, and S. Rao, "A design of an origami reconfigurable qha with a foldable reflector [antenna applications corner]," *IEEE Antennas and Propagation Magazine*, vol. 59, pp. 78–105, 4 Aug. 2017, ISSN: 1045-9243. DOI: 10.1109/MAP.2017.2706649. [Online]. Available: http://ieeexplore.ieee.org/document/8002721/.

[19] X. Liu, S. Yao, N. Russo, and S. Georgakopoulos, "Tri-band reconfigurable origami helical array," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Jul. 2018, pp. 1231–1232, ISBN: 978-1-5386-7102-3. DOI: 10.1109/APUSNCURSINRSM.2018.8608197. [Online]. Available: https://ieeexplore.ieee.org/document/8608197/.

[20] Y. Wu, A. Vallecchi, Y. Yang, *et al.*, "3d printed modular origami inspired dielectrics for frequency tunable antennas," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, Jul. 2019, pp. 903–904, ISBN: 978-1-7281-0692-2. DOI: 10.1109/APUSNCURSINRSM.2019.8889209.

[21] S. Yao and S. V. Georgakopoulos, "Origami segmented helical antenna with switchable sense of polarization," *IEEE Access*, vol. 6, pp. 4528–4536, Dec. 2018, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2787724. [Online]. Available: https://ieeexplore.ieee.org/document/8240591/.

[22] X. Liu, S. Yao, and S. V. Georgakopoulos, "Mode reconfigurable bistable spiral antenna based on kresling origami," in *2017 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Jul. 2017, pp. 413–414, ISBN: 978-1-5386-3284-0. DOI: 10.1109/APUSNCURSINRSM.2017.8072249. [Online]. Available: http://ieeexplore.ieee.org/document/8072249/.

[23] N. E. Russo, C. L. Zekios, and S. V. Georgakopoulos, "Origami multimode ring antenna based on characteristic mode analysis," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, Jul. 2019, pp. 2037–2038, ISBN: 978-1-7281-0692-2. DOI: 10.1109/APUSNCURSINRSM.2019.8888860. [Online]. Available: https://ieeexplore.ieee.org/document/8888860/.

[24] K. Fuchi, J. Tang, B. Crowgey, A. R. Diaz, E. J. Rothwell, and R. O. Ouedraogo, "Origami tunable frequency selective surfaces," *IEEE Antennas and Wireless Propagation Letters*, vol. 11, pp. 473–475, 2012, ISSN: 1536-1225. DOI: 10.1109/LAWP.2012.2196489. [Online]. Available: http://ieeexplore.ieee.org/document/6189735/.

[25] S. M. A. M. H. Abadi, J. H. Booske, and N. Behdad, "Exploiting mechanical flexure as a means of tuning the responses of large-scale periodic structures," *IEEE Transactions on Antennas and Propagation*, vol. 64, pp. 933–943, 3 Mar. 2016, ISSN: 0018-926X. DOI: 10.1109/TAP.2015.2513418. [Online]. Available: http://ieeexplore.ieee.org/document/7368872/.

[26] S. R. Seiler, G. Bazzan, K. Fuchi, *et al.*, "An origami inspired circularly-polarized folding patch antenna array," in *2018 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Jul. 2018, pp. 181–182, ISBN: 978-1-5386-7102-3. DOI: 10.1109/APUSNCURSINRSM.2018.8608608. [Online]. Available: https://ieeexplore.ieee.org/document/8608608/.

[27] D. Rohde, S. Noghanian, Y.-h. Chang, and S. K. Sharma, "Two element series fed origami antenna," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, Jul. 2019, pp. 1111–1112, ISBN: 978-1-7281-0692-2. DOI: 10.1109/APUSNCURSINRSM.2019.8889219. [Online]. Available: https://ieeexplore.ieee.org/document/8889219/.

[28] J. P. Snyder, *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, Dec. 1997.

Chapter 6

EXPERIMENTAL PLATFORM

> You think you can't, you wish you could.
> I know you can, I wish you would.
>
> *Screamadelica*
> *Slip Inside This House*

This chapter presents the hardware design of a tile based array architecture capable of undergoing shape change. Two generations of the architecture are presented. These designs serve as the experimental platforms used to explore the properties of shape-changing phased arrays and meta-gaps in Chapters 7 and 10.

Section 6.1 begins by describing the flexible array architecture capable of dynamically adjusting the location and orientation of its elements. The specifics of the first generation of this architecture are detailed in Section 6.2. Section 6.3 presents the design of the custom RFIC at the heart of this first generation[1]. Finally, Section 6.4 examines the second generation hardware used to power the meta-gap experiments in Chapter 10.

6.1 Array Architecture

To accomplish the goal of dynamically reorientable elements, the array consists of independent tiles that are held in place by a mechanical backbone. Each tile is a self-contained RF system capable of synthesizing and radiating an RF signal with arbitrary phase and amplitude. Tiles are squares with a side length approximately equal to $\frac{\lambda}{2}$ so that they can be joined together into a planar rectangular array with $\frac{\lambda}{2}$ spacing, minimizing grating lobes.

The "front side" of the tile contains the electronics necessary to generate such a signal, while the "back side" contains a ground-plane backed antenna that radiates the signal. The radiator ground plane isolates the front side electronics from the back side antenna to minimize unwanted coupling of the radiated wave into the electronics and unwanted perturbations of the antenna near-field by the front side conductors.

[1]The high density MIM-MOS capacitor in this section were designed by Matan Gal, while the shift registers and DAC were designed by Behrooz Abiri.

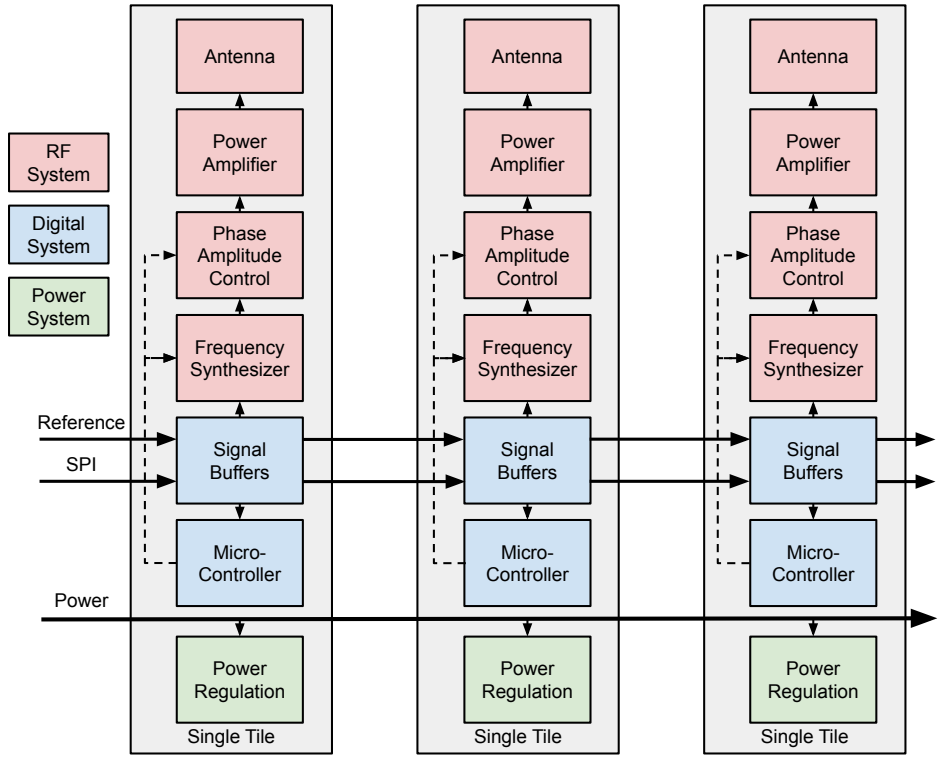
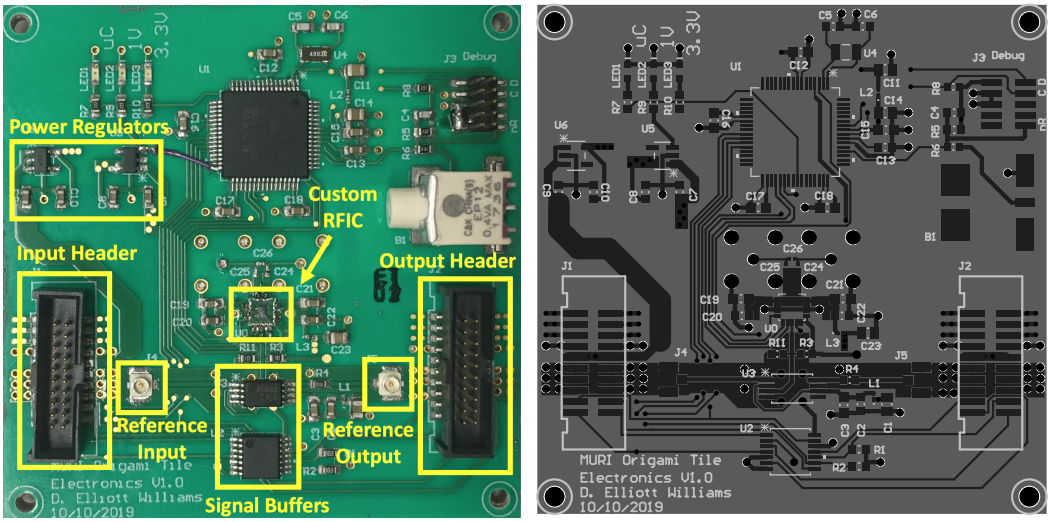


Figure 6.1: Diagram of flexible array architecture. The array consists of independent identical tiles that act as RF sources. Each tile can radiate signals locked to a reference with programmable amplitude and phase. Programming, power, and reference signals are transmitted from one tile to the next.

The antenna is driven by a via passing through an opening in the ground plane. Figure 6.1 shows a high level diagram of the flexible array architecture used in this work. In abstract, the RF system consists of a frequency synthesizer, a programmable phase amplitude modulator, and a power amplifier (PA). The synthesizer locks to a low frequency reference signal, generating a signal at the desired frequency with a deterministic phase offset from the reference source. The modulator adjusts the phase and amplitude so that the RF output will have a desired phase offset from the reference source and output power. Finally, the PA drives the antenna in order to radiate the signal.

Thus, the tiles are synchronized to the reference with a programmable phase offset and amplitude, allowing arbitrary control of the relative phase shift and strength of the waves radiated by each tile. Arbitrary phase control allows beams to be steered no matter the orientation and location of the tiles, while arbitrary amplitude control enables amplitude tapering along the array surface to adjust the beam patterns.

Each tile requires programming and reference signals and power in order to function.



(a) Annotated photo of the tile components. (b) Layer map of routed traces.

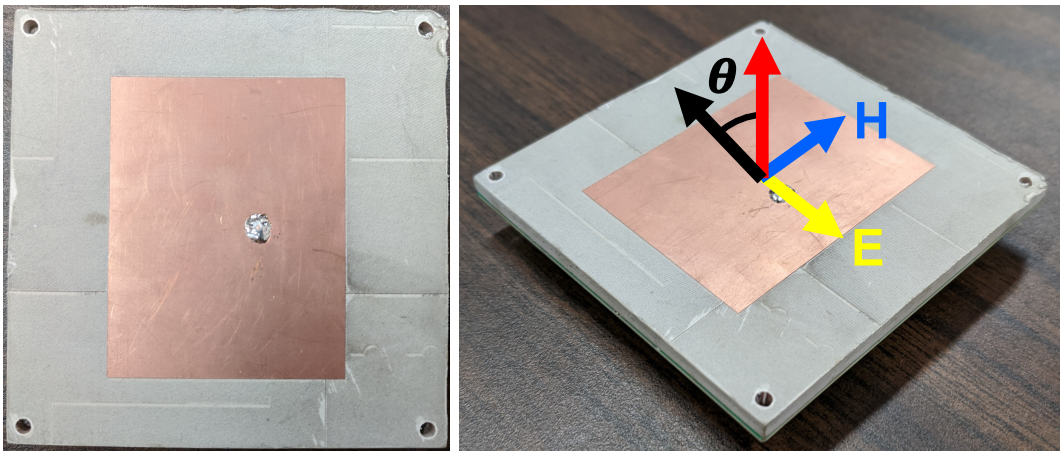
Figure 6.2: First Generation Tile Electronics Layout.

Signals and power are fed from one tile to the next in a daisy-chain configuration using flexible wires. Each tile acts as a buffer, amplifying and re-transmitting the reference and programming signals to prevent signal degradation. A separate motherboard drives the first tile and interfaces the array with the controller PC, power supply, and reference generator. A low frequency (≈100MHz) reference signal is used to reduce the power required to buffer and distribute the reference. Programming is accomplished using a SPI bus in broadcast mode, with each tile acting as a secondary assigned with a unique address and the motherboard acting as the main. Finally, power is distributed at a higher voltage and regulated locally on each tile in order to compensate for the conductive losses.

This modular daisy-chain architecture allows the number of tiles in the array to be easily adjusted to suit the needs of the experiment. The flexible wires allow dynamic reconfiguration of the tile locations and orientations. Tiles are held in place by a mechanical backbone via nylon standoffs that connect to the corners of the tile, minimizing the perturbation of the antenna field distribution.

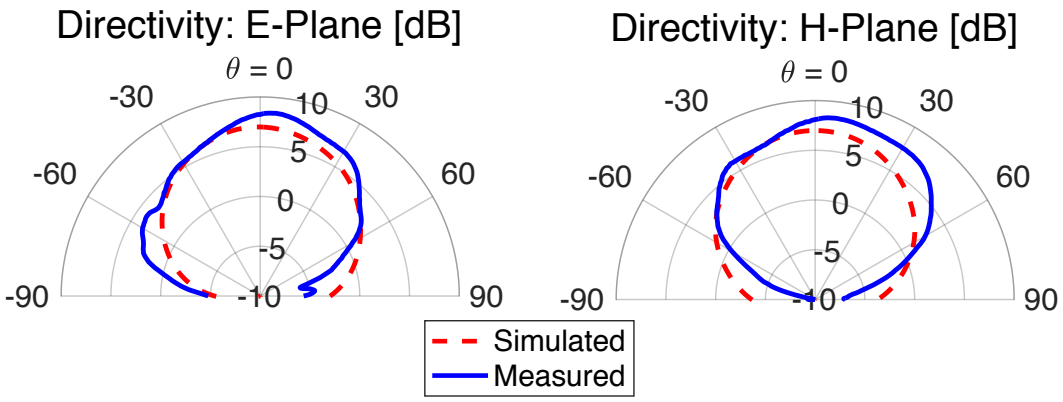
6.2 First Generation Flexible Array Tile

The first generation tile is used for the shape change experiments described in Chapter 7. The 6cm square tile consists of two PCBs, an electronics board and an antenna board, that are soldered together to form a single PCB stack up. The two board approach was used to reduce manufacturing costs and to enable independent characterization.



(a) Antenna top view

(b) Measurement coordinate system



(c) Directivity measurements in E- and H- planes assuming no backside radiation

Figure 6.3: First Generation Tile Antenna Design and Measured Patterns. Reprinted with permission from the copyright holder, EuMA.

Electronics Design

Figure 6.2a shows the electronics side of the first generation tile. The board has three copper layers separated by Rogers 4350B dielectric. The top two layers are signal layers that are also used to route supply traces. The bottom layer is a ground plane that is directly soldered to the ground plane of the antenna board. The board layout is shown in Figure 6.2b.

The heart of the tile is a custom 2.6 GHz RFIC containing the entire RF system discussed in Section 6.3. The RFIC is bonded to the board with an interposer PCB. The reference is fed to the RFIC by a two output clock buffer (CDCLVC1102PWR) that is also used to buffer the reference signal for the next tile. To minimize reflections, the reference is carried from tile to tile using 50Ω U.FL coaxial cables and distributed across the tile using matched coplanar waveguides.

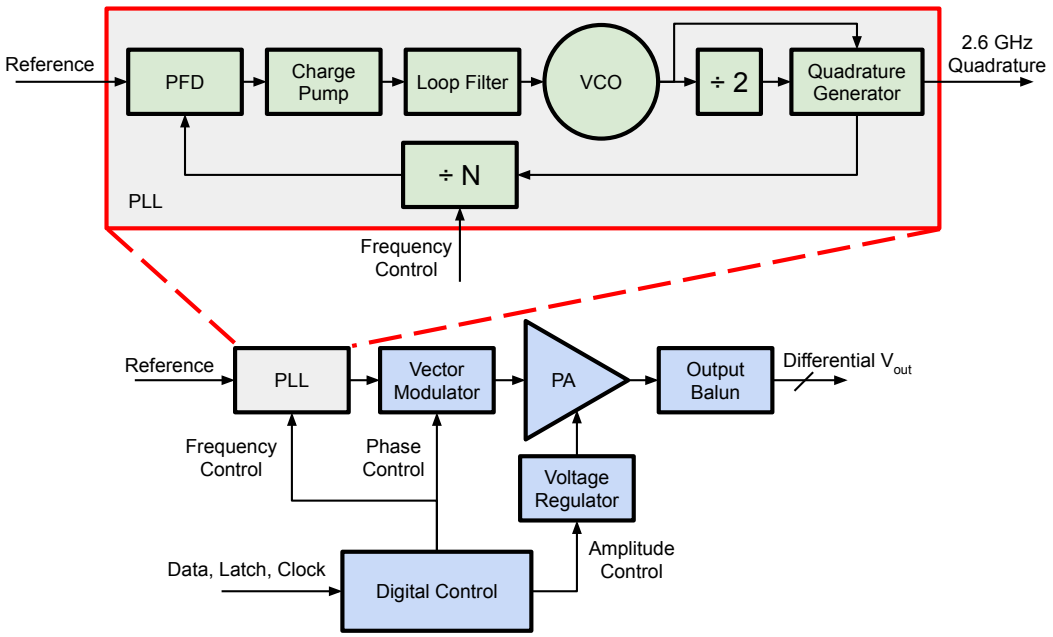


Figure 6.4: Antenna driver RFIC block diagram. Reprinted with permission from the copyright holder, EuMA.

The daisy-chain nature of the architecture necessitates a long conductor, and thus relatively high resistance, between the power supply and the last tile in the array. This resistance causes a drop in the supply voltage on the last time when the tiles draw high current. To combat this, the supply is distributed at 5V and two linear regulators (MIC5504-3.3YM5 and TCR3DF10) on each tile are used to convert it into local 3.3V and 1V supplies.

Antenna Design

Figure 6.3a shows the antenna side of the first generation tile. The antenna is a via-fed linearly polarized patch fabricated on a 120 mil Rogers 6002 substrate. The patch length is 32.27 mm and its width is 41.08 mm. An L matching network on the electronics board tune out the feed inductance and matches the antenna to 50Ω at 2.6 GHz. Figure 6.3c shows the simulated and measured antenna patterns. The measured directivity and HPBW of the antenna is 8.9 dBi and 60.9°, respectively. The simulated efficiency is 98.1%.

6.3 2.6 GHz Driver Integrated Circuit

The first generation tile is powered by a custom RFIC fabricated in TSMC's 65nm CMOS process. The chip is a 2.6 GHz driver with programmable amplitude and phase. The chip dimensions are only 0.8mm by 1.25mm, allowing it to be easily

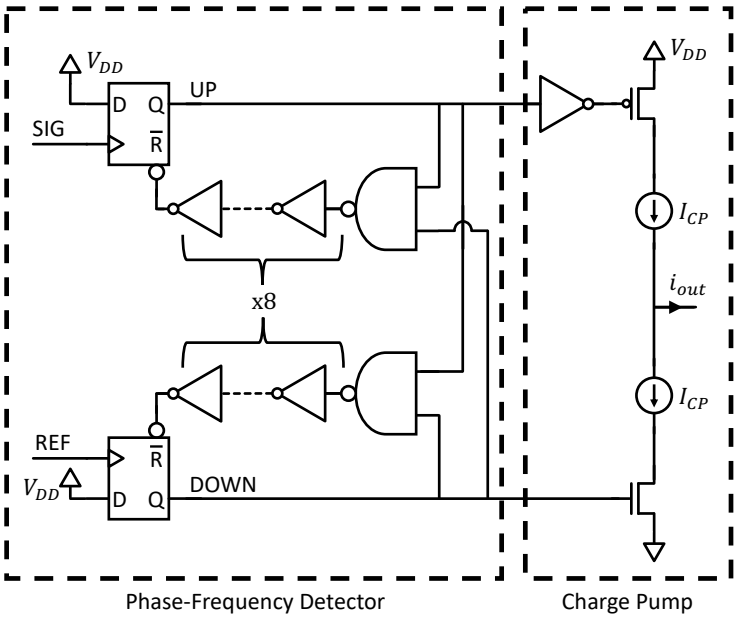
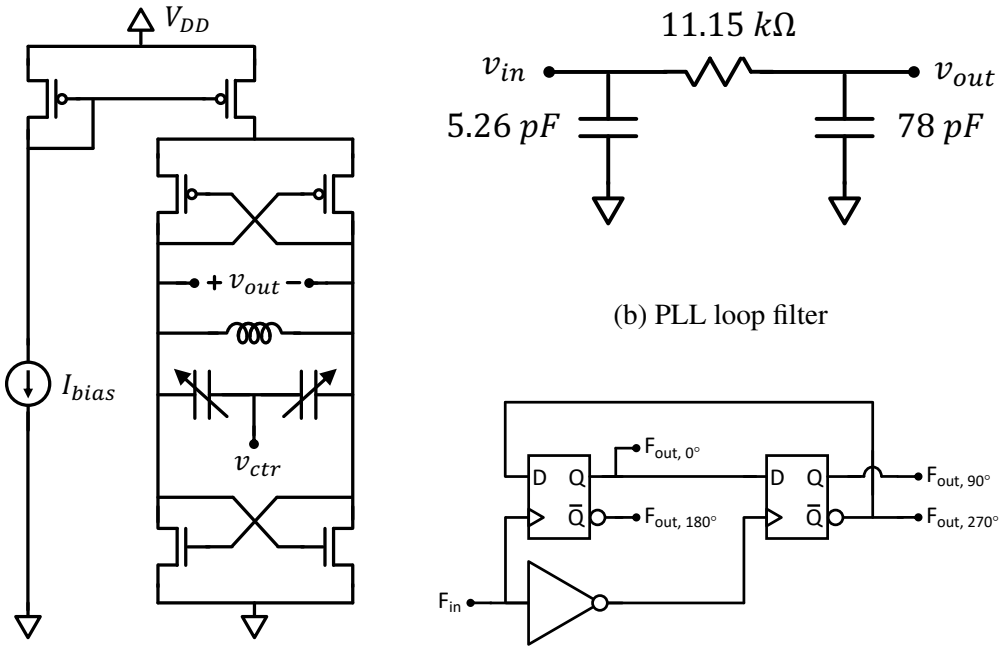


Figure 6.5: Schematic of the phase frequency detector and charge pump.

integrated into systems with tight area constraints, such as area constrained tiles or multi-port antennas.

Figure 6.4 shows a system level block diagram of the circuit. A type-II Phased Locked Loop (PLL) with a programmable divider generates quadrature signals at the output frequency $f_{out} = Nf_{ref}$. The quadrature signals drive a vector modulator that serves as a phase shifter and a PA driver. The regulated supply of the output PA is programmable, enabling amplitude control of the output. Finally, the PA drives a on-chip transformer acting as a balun to enable differential and single-ended outputs.

The type-II PLL consists of a phase frequency detector (PFD), charge pump, loop-filter, voltage controlled oscillator (VCO), and divider chain. The output frequency is determined by a 5-bit programmable divider originally presented in [1]. The VCO operates at twice the desired output frequency and divided by two in order to generate quadrature outputs. Division and quadrature generation is achieved using digital flip-flops with inverted clocks in a feedback loop. The VCO is a 5 GHz P/NMOS stacked cross-coupled oscillator. The oscillator tank consists of a two-turn inductor and MOSCAP varactors. The VCO consumes 2.63 mW and has a measured gain of 544 MHz V^{-1}. The PFD provides the charge pump with UP and DOWN signals so the latter can add and remove charge from the loop filter using 100 μA current pulses. The PLL is stabilized by a 2nd order loop filter, consisting of a 5.46 pF MIM capacitor, a 11.15 kΩ poly-silicon resistor, and an area-compact



(a) VCO

(b) PLL loop filter

(c) Divider and quadrature generator

Figure 6.6: Schematics of PLL frequency generation blocks.

78pF combination MIM and MOS capacitor. The PLL has a tuning range from 2.46 to 2.73 GHz when programmed with a division ratio of 32 and -25 dB reference spurs.

The vector modulator is a tuned amplifier that combines differential I and Q currents with programmable weights at the resonant output node. The current weights are controlled using 64 identical cascoded transistors in parallel with each cascode transistor turned on or off using a digital control bit. Because the transistors are the same size, the total current is determined by a 64 bit thermometer code. Two differential swap multiplexers are used to invert the signs of the vector modulator drive in order to obtain phase shifts in all four quadrants. The output summing node is an LC tank formed by the capacitance of the power amplifier input transistors, fixed MIM capacitors, and a five-turn inductor.

The PA drive is determined by Equation 6.2

$$V_{PA} = I\cos(\omega t) + Q\sin(\omega t) \tag{6.1}$$

$$V_{PA} = \sqrt{I^2 + Q^2}\cos\left(\omega t + \arctan\left[\frac{Q}{I}\right]\right) \tag{6.2}$$

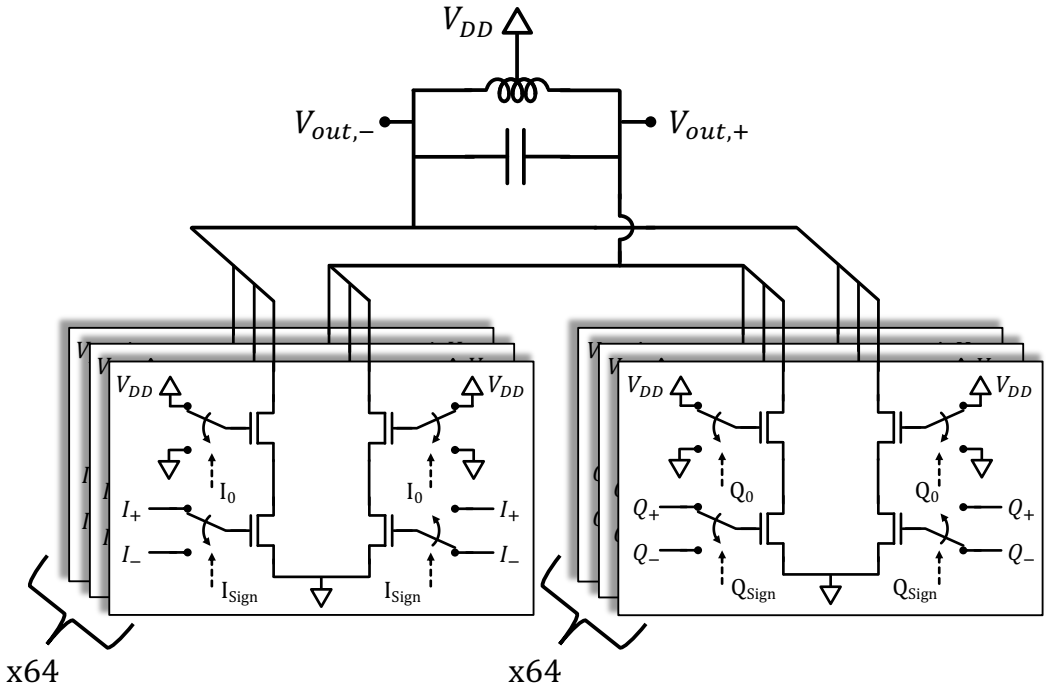


Figure 6.7: Schematic of the four quadrant vector modulator.

where I and Q are the strengths of the in-phase and quadrature signals. Thus, the phase and amplitude of the power amplifier drive are determined by the thermometer codes for the I and Q amplifiers and the sign control bits of the input multiplexers. The phase of the PA drive is the phase of the output signal, while the amplitude determines how hard the PA is driven and thus the PA's class of operation, linearity, and efficiency.

The output stage is a cascoded class E/F^{-1} differential switching PA. The DC bias of both the base and cascode transistors are determined by programmable DACs to enable different modes of operation. The differential transistors drive a custom on-chip transformer that is used as part of the waveform shaping filter, as an impedance transformer, and as a balun to enable the chip to drive both differential and single-ended outputs. The 2:1 transformer is 80.9% efficient and converts a 50 Ω load into 16.5 Ω. The realized Power Amplifier has a maximum measured drain efficiency of 15.9% and can output up to 22.2 mW.

The PA power supply is controlled by a programmable linear regulator capable of supplying up to 400 mA. By reducing the supply voltage, the output voltage swing, and thus output power, of the PA is reduced. The linear regulator consists of a large PMOS transistor driven by an operational amplifier. Negative feedback is used to force the output voltage to match an input reference voltage supplied by am 8-bit

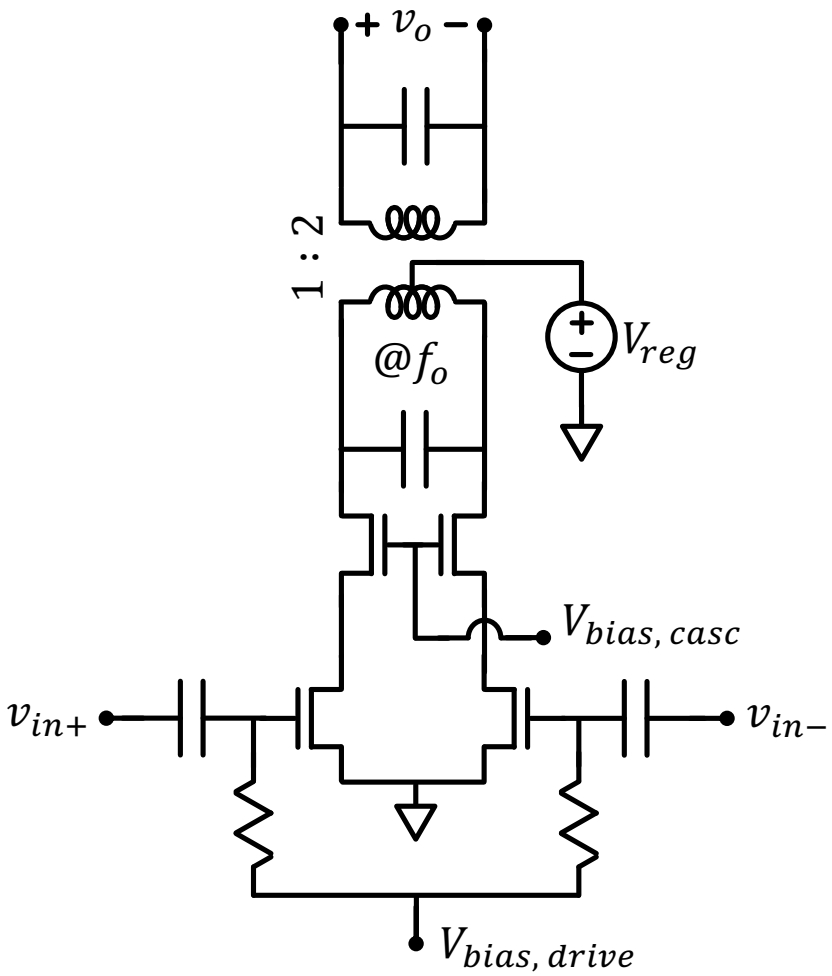
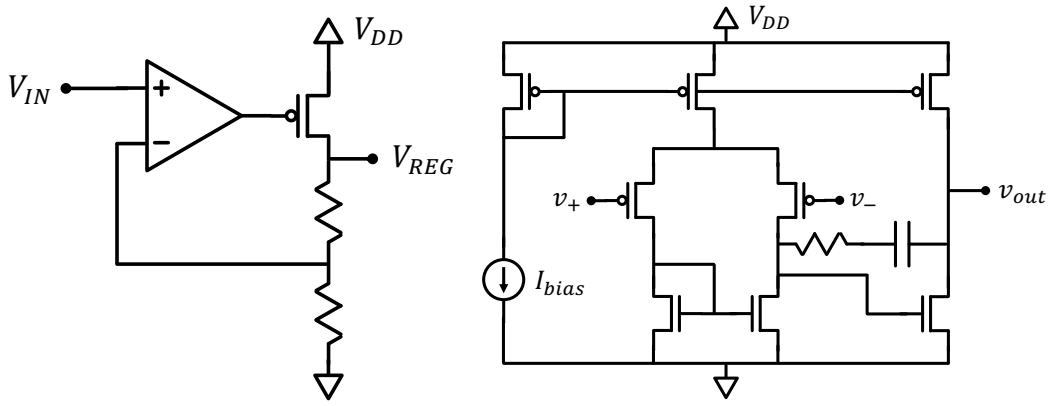


Figure 6.8: Schematic of the power amplifier.

DAC. The two stage operational amplifier consists of a differential PMOS amplifier input stage followed by a single ended NMOS common source amplifier output stage. An output buffer stage is unnecessary because the amplifier drives the gate of a PMOS transistor with high input impedance.

Biasing in the chip is based on a bandgap reference current source. The core of the bandgap is two differentially sized pmos diodes that are used to generate PTAT and CTAT currents. The total temperature independent current is 100μA.This currents is fed into current mirrors that route the bias throughout the chip.

Digital control is accomplished using a set of five 8-bit, and one 16-bit, shift registers. DATA and CLK signals shift input data into the registers. When a latch enable (LE) signal goes high, the final 7-bits are compared with a hardwired address to determine whether the shifted data will be stored and driven to the rest of the chip. Three of



(a) Voltage controlled voltage regulator. (b) Two stage op-amp at the core of the regulator.

Figure 6.9: Schematic of the PA regulator.

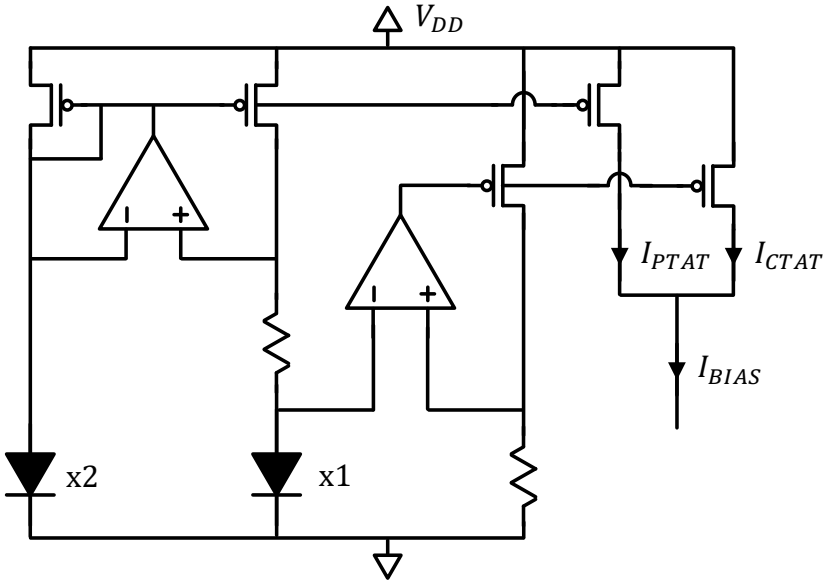


Figure 6.10: Schematic for the bandgap reference generator.

the 8-bit shift registers drive 8-bit R2R DACs and control the regulator reference voltage and the PA bias voltages. The remaining two 8-bit registers control PLL settings and debug features. The 16-bit register controls the vector modulator, with 2-bits controlling the input multiplexers and the remaining 14 bits being converted into the two 64-bit thermometer codes by a Binary-to-Thermometer controller.

The total IC consumes $\approx$ 175mW in typical operation. A die photo of the chip is shown in Figure 6.11 and the total area of each block and sub-module is listen in Table 6.1.

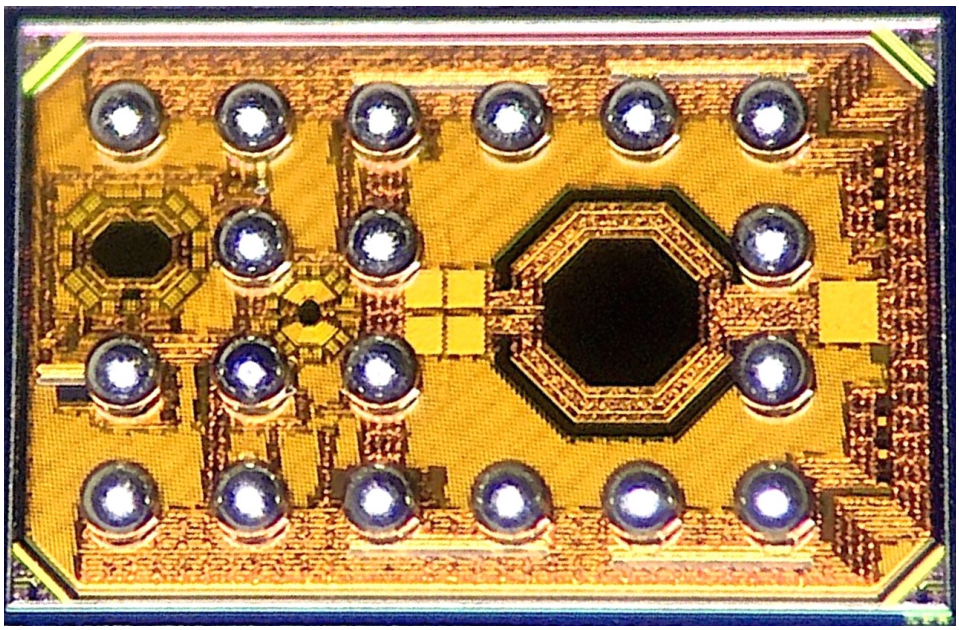


Figure 6.11: Die photo of 2.6 GHz driver integrated circuit.

6.4 Second Generation Flexible Array Tile

The second generation tile addresses shortcomings with the first generation and includes the capacity to support meta-gaps or other peripherals. Meta-gaps require much higher power consumption that would greatly reduce the efficiency of the first generation's linear-regulators. In addition, rapid programming speeds are critical for efficient meta-gap characterization as described in Chapter 9. Thus the second generation replaces the OR Gate buffers in the first generation that limited the programming bandwidth. The second generation also makes use of the two soldered PCB stack-up but includes assembly modifications to enhance the reliability of the RF connection between boards. The second generation tile is used for the meta-gap experiments described in Chapter 9.

Electronics Design

Figure 6.12a shows the electronics side of the second generation tile. The 6cm square PCB consists of two signal layers, a supply plane, and a ground plane. Layers are separated by FR4 instead of Rogers to minimize cost with minimal performance degradation due to the size of the board and frequency of operation. As in the first generation, the bottom ground plane is soldered to the ground plane of the antenna. The board layout is shown in Figure 6.12b.

The RF system in the second generation is comprised with off the shelf components

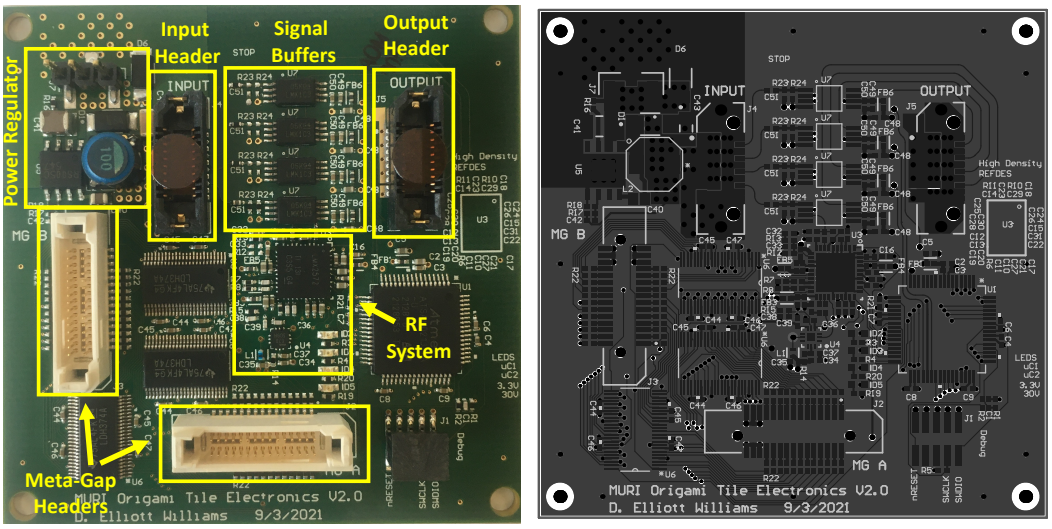
Table 6.1: 2.6 GHz Driver RFIC Component Block Areas

Circuit Block	Length [um]	Width [um]	Area [10^3 um^2]
Phase Frequency Detector	12	5	0.060
Charge Pump	18	15.8	0.284
Loop Filter	193	94	18.1
Voltage Controlled Oscillator	223	197	43.9
Quadrature Generator	5	3.7	0.018
Programmable Divider	38	5.6	0.213
Phased Locked Loop Total	330	197	65.0
Power Amplifier Core	288	191	55.0
Output Transformer	398.6	295.6	117.8
Power Amplifier Total	604.6	295.6	178.7
Regulator Op-Amp	96.7	48.4	4.6
Regulator Total	133	100	13.3
Digital to Analog Converter	78	62.7	4.9
Vector Modulator Controller	16.4	16.3	0.267
Programming Shift Registers	98.5	32.5	3.2
Digital Total	-	-	18.1
Bandgap Reference	20	16	0.320
Biasing Circuit Total	60.7	46.6	2.8
Vector Modulator	197	158	31.1
Total Chip Area	1250	800	1000

to minimize cost. The design consists of a PLL (LMX2572RHAT) and a PA (SST12LP07A-QXBE) operating at 2.5 GHz. The PLL contains an integrated VCO and provides programmable output phase and amplitude control. A 78.125 MHz reference signal is fed to the PLL by a two output LVCMOS clock buffer (LMK1C1102PWR) that also buffers the reference for the next tile.

The tile also serves as a controller for two neighboring meta-gap sheets. Two headers on each tile are used to control switches on two neighboring meta-gap sheets. Each header consists of 24 programmable outputs that drive the connected PIN diode switches with 10 mA. The state and drive of the 48 output pins are managed by six 8-bit flip-flops (distributed between three SN74LVCH16374ADGVR chips) configured as two 24-bit flip flops. These flip-flips share a 24 bit data bus and are selectively programmed using separate CLK lines.

SPI programming signals are buffered by the same LVCMOS clock buffer IC as the reference in order to enable high speed programming. Due to the change in buffers, the second generation tile does not have read-back capability and so the array oper-



(a) Annotated photo of the tile components. (b) Layer map of routed traces.

Figure 6.12: Second Generation Tile Electronics Layout.

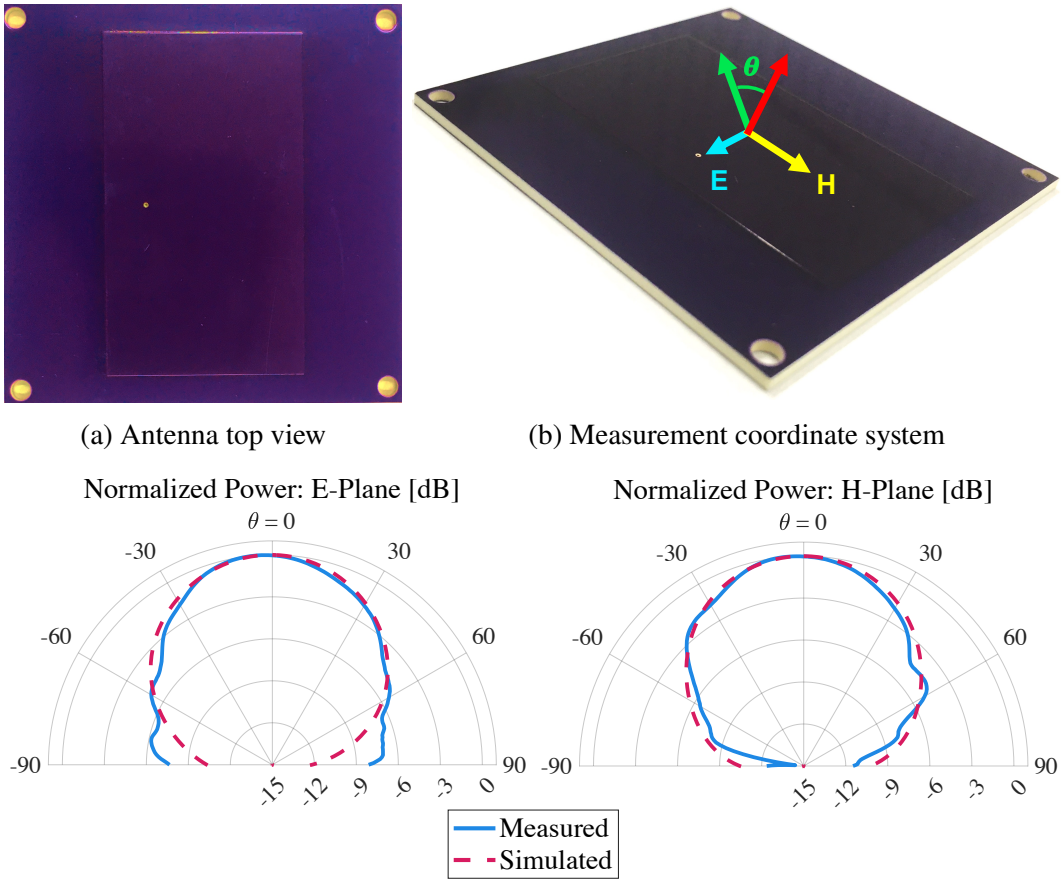
ates exclusively in broadcast mode. An ARM microcontroller (ATSAMD21J17A) decodes incoming SPI packets and programs the PLL, PA, and meta-gap flip-flops as necessary. Each tile can be addressed with either a unique tile address for selective programming, or a global broadcast address for shared instructions.

Power, ground, programming, and reference signals are distributed through the array using a single 20 pin cable in order to reduce array wiring complexity. The cable, a Samtec Edge Rate® Coax Cable Assembly, is an array of micro-coax cables matched to 50Ω. The cables are selected to minimize reflections for the reference and programming signals. The remaining pins are capable of supporting up to 4A of supply current. Reference and programming signals are distributed across the tile using matched microstrip lines to reduce reflections.

In order to reduce the total current carried by the cables and improve efficiency, power is distributed at 30V and down-converted to 3.3V by a local DC to DC converter (R1245S003C) on each tile. The resulting architecture can support up to 120W of power. Depending on the meta-gap settings, the power consumption of the array in operation ranges between 36W and 64W, but is around 44W during typical operation.

Antenna Design

Figure 6.13 shows the antenna side of the second generation tile. The antenna is a via-fed linearly polarized patch fabricated on a 60 mil Rogers 4350B substrate. The



(a) Antenna top view

(b) Measurement coordinate system

(c) Normalized Power measurements in E- and H- planes

Figure 6.13: Second Generation Tile Antenna Design and Measured Patterns.

patch length is 29.7 mm and its width is 51 mm. The antenna is covered with solder mask to aid with board-to-board soldering; simulations showed that the solder mask introduced negligible loss as the operation frequency. The antenna, including the via feed, is designed to be matched to 50Ω at 2.5 GHz with a RF choke inductor that also supplies power to the PA. Manufacturing variation detuned the antenna by 35 MHz. Figure 6.13c shows the simulated and measured antenna patterns. The measured directivity and HPBW of the antenna is 7.4 dBi and 79.8°, respectively. The simulated efficiency is 84.0%.

References

[1] C. Vaucher, I. Ferencic, M. Locher, S. Sedvallson, U. Voegeli, and Z. Wang, "A family of low-power truly modular programmable dividers in standard 0.35-μm cmos technology," *IEEE Journal of Solid-State Circuits*, vol. 35, pp. 1039–1045, 7 Jul. 2000, ISSN: 0018-9200. DOI: 10.1109/4.848214. [Online]. Available: http://ieeexplore.ieee.org/document/848214/.

Chapter 7

ORIGAMI-INSPIRED SHAPE-CHANGING PHASED ARRAY

> Because running will be the way
> your life and mine will be described.
> As in "the long run"
> Or, as in having given someone "A run for his money"
> Or, as in "running out of time."
>
> *Gil Scott-Heron & Makaya McCraven*
> *Running*

This chapter presents measurements of the first shape-changing phased array, verifying the theory presented in Chapter 4 and demonstrating a single structure that breaks the trade-off between gain and steering range. The first generation tiles detailed in Chapter 6 are attached to a flexible metal backbone that can switch between planar, cylindrical, and spherical shapes.

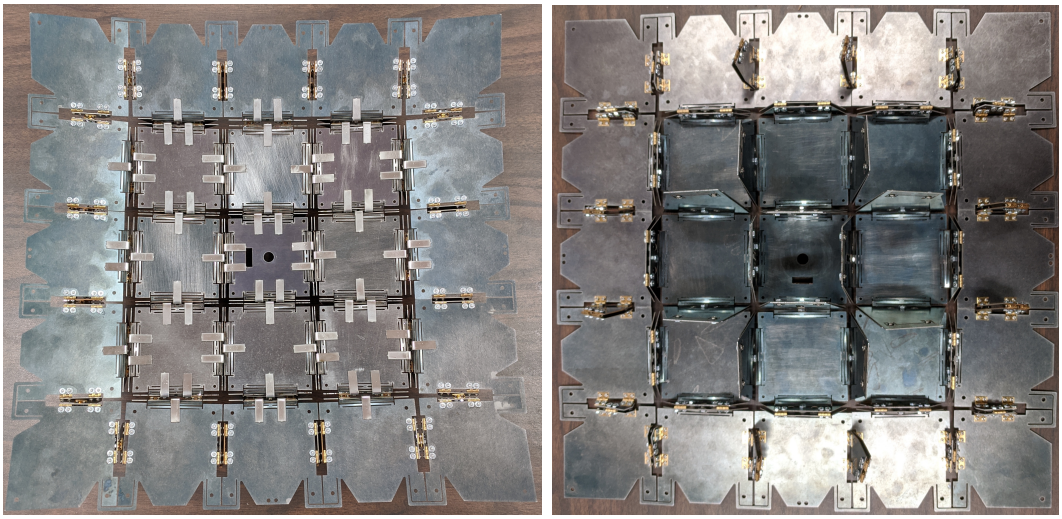
Section 7.1 presents the design of the array and the metal backbone[1]. Next, Section 7.2 describes the measurement setup used to characterize the array. Section 7.3 then presents measurement results that are analyzed in Section 7.4. Finally, conclusions are drawn in Section 7.5.

7.1 Design

The array consists of 15 identical 6 cm square radiating tiles that are held in a particular shape by an origami-inspired metal backbone. The 15 tiles are arranged in a 3-by-5 rectangle[2]. The backbone allows the shape of the array to be easily altered by shifting between configurations that hold tiles in the correct locations and orientations. The tiles provide independent phase and amplitude of radiated 2.6 GHz signals, allowing beams to be steered using digital programming. The full design details of the first generation tile used in this array is presented in Section 6.2.

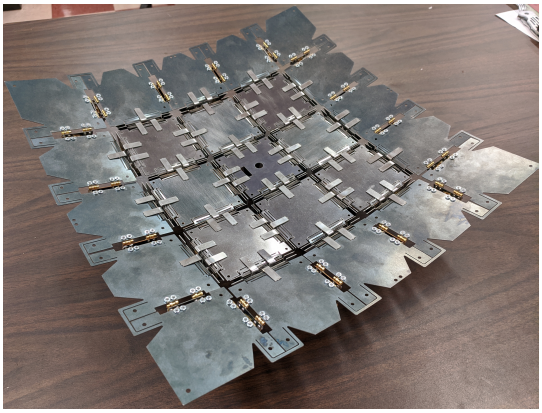
[1] The mechanical design, analysis, and fabrication work in this Section was performed by my collaborator Charles Dorn, and the backbone "folding pattern" was designed by my collaborator Robert Lang.

[2] Unfortunately, a full 5-by-5 array could not be characterized due to damage to some of the tiles during testing.

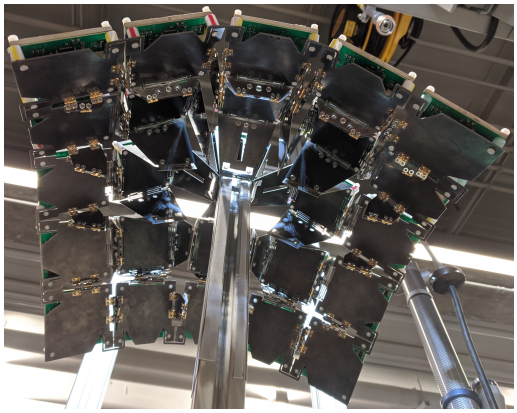


(a) Top view

(b) Back view



(c) Oblique view



(d) Underneath with attached tiles

Figure 7.1: The 5-by-5 element steel backbone that enables shape-change.

The backbone structure shown in Figure 7.1 allows for reconfiguration of the antenna tiles between spherical, planar, and cylindrical configurations. The geometry consists of a 5-by-5 grid of squares placed on the surface of a sphere of radius 19.6 cm. Adjacent squares are connected by pairs of trapezoids joined by compliant hinges. In the spherical configuration, the trapezoids open into deep valley folds. From the spherical configuration, the valley folds can be closed and the trapezoids brought into contact to achieve planar and cylindrical configurations. In the planar configuration, the antennas are arranged in a 6.75 cm grid, or 0.585λ at 2.6 GHz.

The backbones "folding pattern" was designed by first identifying tile locations on the sphere that minimized the the RMS sum of the surface spacing between tiles subject to a no-shear condition and the requirement that the edge elements touch and form a continuous rim. Than, joints were placed between adjacent tiles so that

the tiles could rotate relative to each other and lie flat. The no-shear conditions simplifies the hinge design as the tiles do not move laterally from their neighbors when unfolding. The joined rim condition reduces the degrees of freedom to improve rigidity.

The resulting geometry has 24 kinematic degrees of freedom, so the structure is designed to be bi-stable in the spherical and planar configurations to simplify actuation and facilitate reconfiguration. To achieve bi-stability, torsional springs are embedded on each fold and fold angles are restricted using stops. Specific rest and stop angles of the hinges are selected such that both the spherical and planar configurations are at local strain energy minima. The mountain folds are given a rest angle of zero (fully open) while the valley folds are given a rest angle of π radians (fully folded). The strain energy of the structure is

$$E = \sum_{i \in I_m} \frac{k_m}{2} \theta_i^2 + \sum_{j \in I_v} \frac{k_v}{2} \left(\theta_j - \pi\right)^2 \tag{7.1}$$

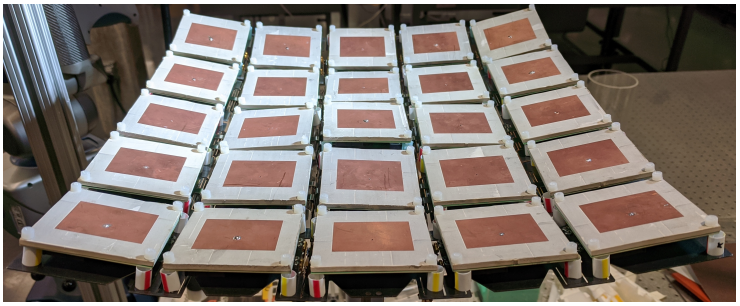
where I_m and I_v index mountain and valley folds and k is the torsional stiffness.

The chosen design guarantees that both planar and spherical configurations are stable, ensuring that the structure can maintain both shapes and is easily actuated between the two. Figure 7.2 shows the combined backbone and tile structure in the different configurations. As can be see, the planar and spherical shapes are self-stable and do not need external supports. The cylindrical configurations is accessed by tightening a cable between opposite sides of the structure as shown in Figure 7.2e.

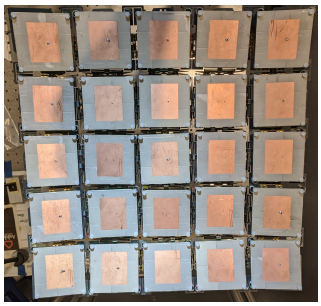
To build the structure, compliant hinges have been designed as lamina emergent torsional connections [1] cut by water jet from 0.025 inch thick spring steel. Since the plates and stops are not perfectly rigid and the compliant hinges are not perfect folds, the prototype slightly deviates from ideal planar, spherical, and cylindrical shapes. In addition, the geometries are distorted under gravity due to the unexpectedly high weight of the tiles. However, the geometries approximate the desired shapes closely enough with the aid of external supports to demonstrate the advantages of a reconfigurable array.

7.2 Measurement Setup

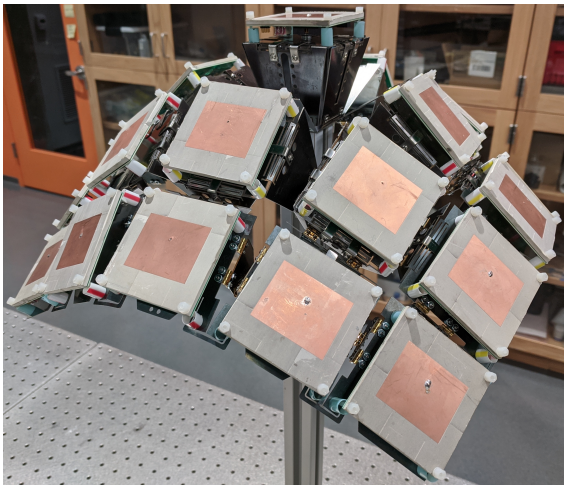
To demonstrate the benefits of shape change, the broadside EIRP is measured across the steering range for the array in planar, cylindrical, and spherical configurations. Figure 7.3 shows a diagram of the measurement setup. A vector network analyzer



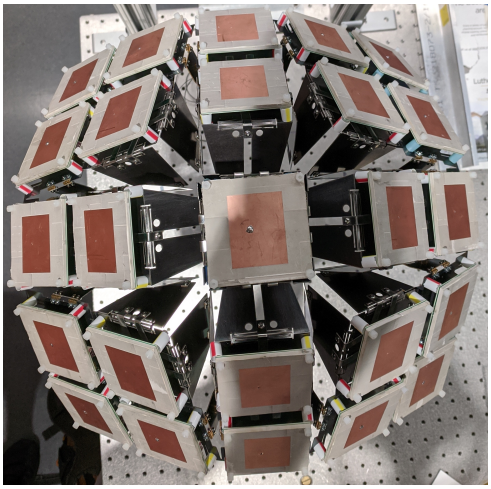
(a) Planar side view



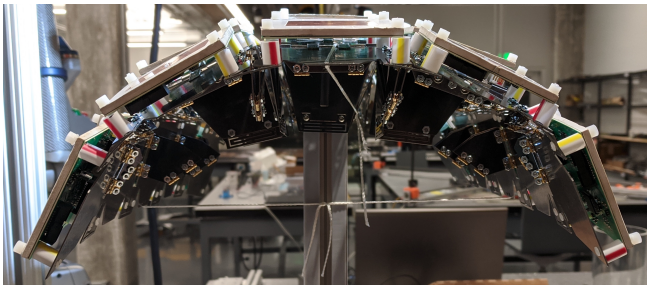
(b) Planar top view



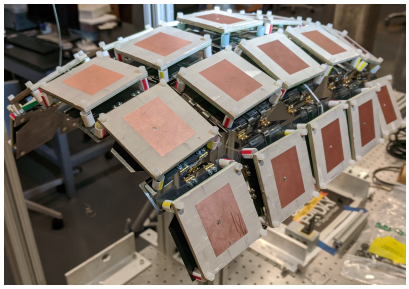
(c) Spherical oblique view



(d) Spherical top view



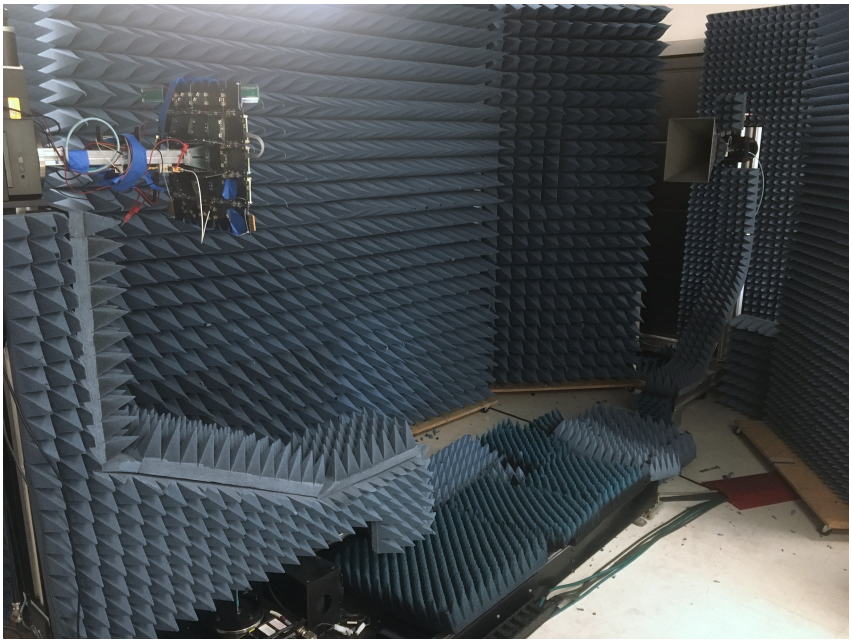
(e) Cylindrical side view



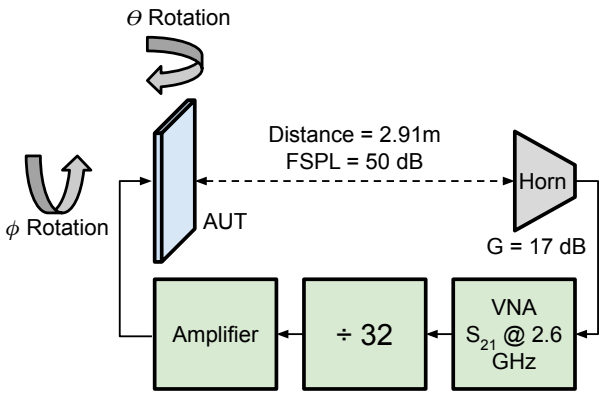
(f) Cylindrical oblique view

Figure 7.2: The shape-changing phased array populated with 25 tiles. No external supports are used to restrain the geometry, with the exception of a single string in the cylindrical configuration.

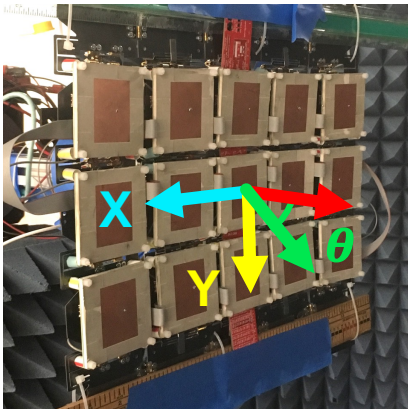
(Agilent Technologies N5230A) is set to measure S_{21} at 2.6 GHz. A pair of frequency dividers (total division ratio of 32) convert the 2.6 GHz output of the vector network analyzer (VNA) to a 81.25 MHz reference signal. A microwave amplifier buffers the reference signal and supplies it to the antenna under test (AUT). Each tile within the array up-converts the reference back to 2.6 GHz, adjusts the phase, and radiates a fixed power. The combined radiated field is measured by a horn antenna located



(a) Measurement Range



(b) Diagram of measurement setup

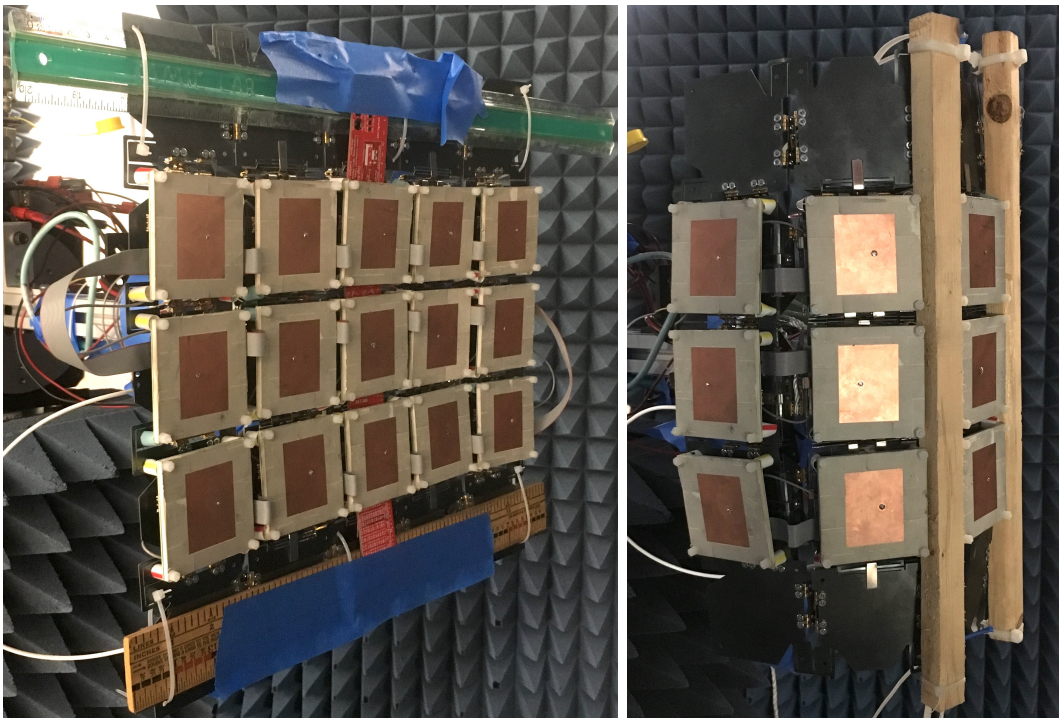


(c) Measurement coordinate system

Figure 7.3: Shape-changing phased array measurement setup.

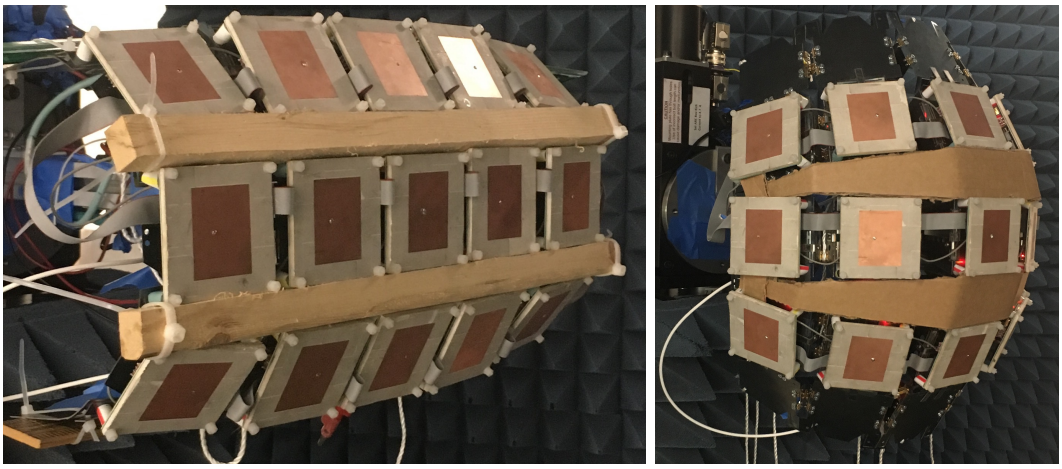
2.91 meters from the AUT and connected to the VNA. This distance is larger than the Fraunhofer distance of every measured array configuration. The measured S_{21} is converted to EIRP using a fixed calibration factor. This calibration factor is calculated by measuring a fixed radiation pattern with both the VNA and a power meter and comparing the results.

To measure the radiation pattern, the AUT is mounted horizontally on a far-field scanner and rotated by an angle θ around the vertical as shown in Figure 7.3a. The array is rotated about its center axis in order to measure different ϕ cuts. To steer a beam to a given angle, the AUT is rotated in the opposite direction and the power



(a) Planar configuration

(b) y-cylinder configuration



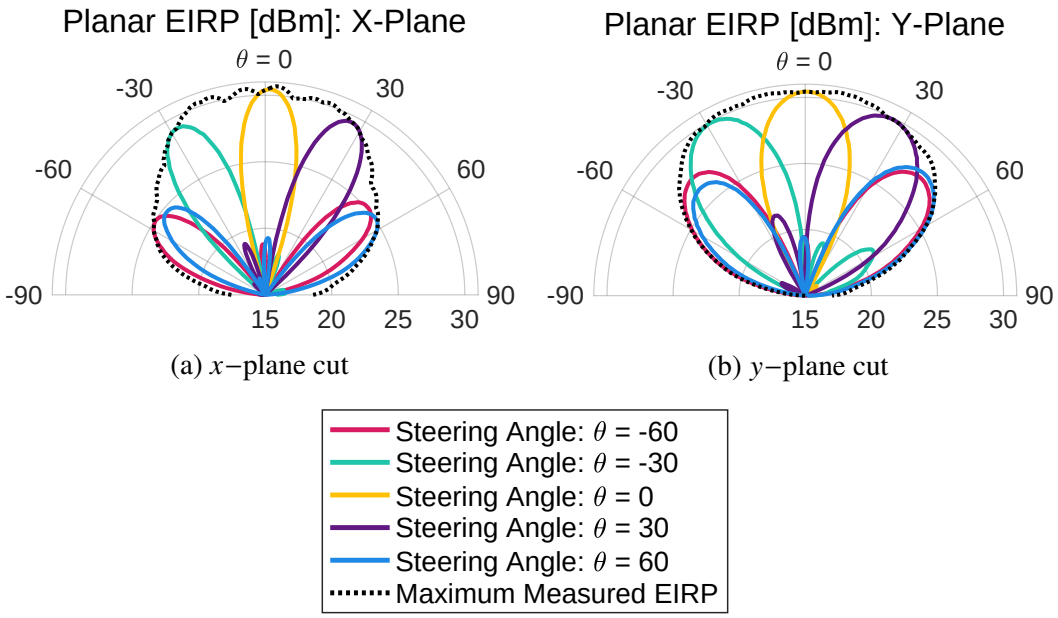
(c) x-cylinder configuration

(d) Spherical configuration

Figure 7.4: The 3-by-5 tile origami-inspired shape-changing phased array as measured. Additional restraints are required to hold the geometry in the proper configuration given the weight of the tiles. Reprinted with permission from the copyright holder, EuMA.

received by the horn antenna is maximized by using the optimization algorithm discussed in Section 2.7.

EIRP is used as a proxy for the antenna gain in order to reduce measurement complexity. True characterization of the gain requires a compete 2D measurement



(a) $x-$plane cut (b) $y-$plane cut

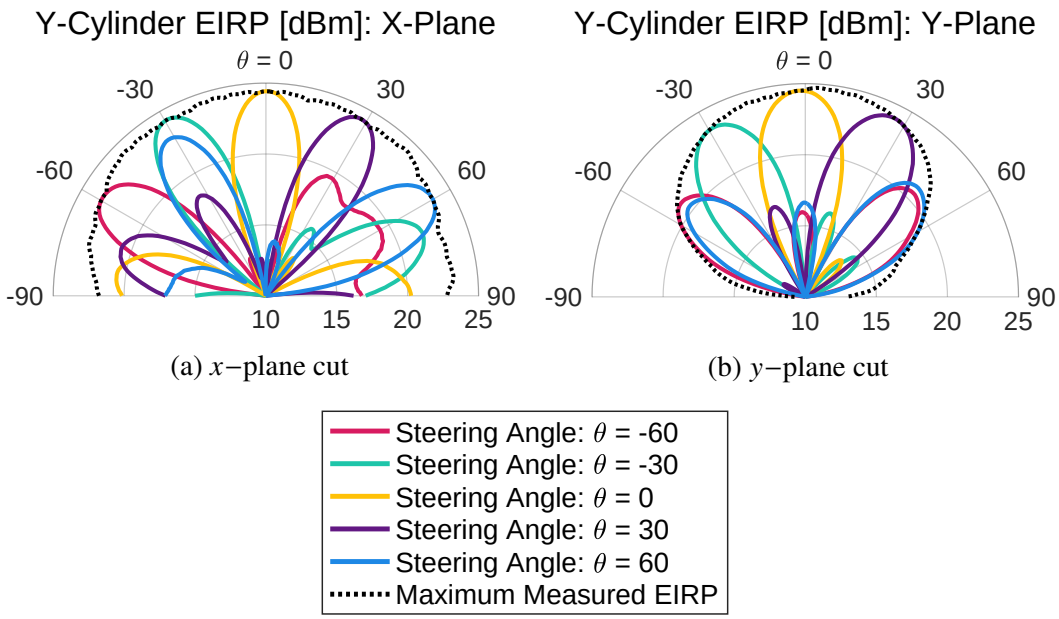
Figure 7.5: Collection of measured beam patterns for the planar configuration. The maximum measured beam power at each angle is indicated by the black dashed line.

of the radiation pattern. The EIRP however, only requires measurements of the peak beam power and is proportional to the gain assuming that the *total* power radiated by the array is held constant. In order to quantify the change in gain over the steering range, beams are steered in 5° increments from −90° to 90° in the x-plane and in the y-plane. An $x-$ or $y-$plane cut of the beam is then measured, depending on which plane the beam is steered in. Because each of these individual patterns contains the maximum achievable EIRP in the steering direction, the maximum of the measured patterns in a plane approximates the maximum achievable EIRP over the steering range in that plane. This array characteristic is measured for each configuration shown in Figure 7.4: a plane, a sphere, and cylinders oriented in the $x-$ and $y-$ directions.

7.3 Measurement Results

Figures 7.5 through 7.8 show a subset of the EIRP beam patterns measured in the $x-$ and $y-$ planes for the planar, spherical, and cylindrical configurations. While beams are measured every 5°, only the beams located every 30° are plotted to facilitate comprehension. Thus these patterns represent only a subset of the 296 total measured beam patterns. Each plot also includes the maximum measured EIRP across the steering range.

Figures 7.5 shows the beam patterns for the planar array shown in Figure 7.4a.



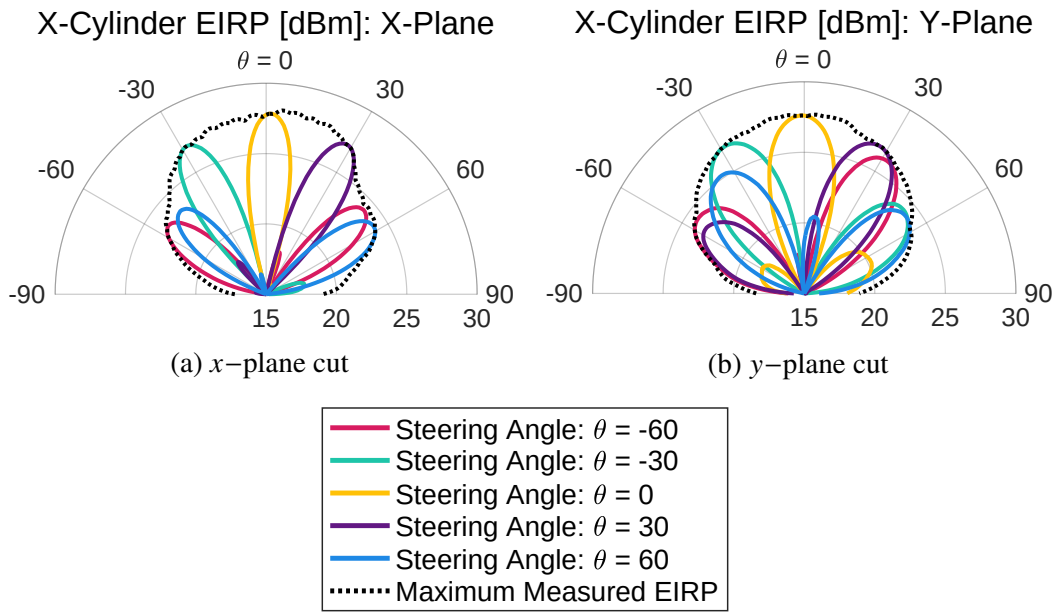
(a) $x-$plane cut (b) $y-$plane cut

Figure 7.6: Collection of measured beam patterns for the $y-$cylinder configuration. The maximum measured beam power at each angle is indicated by the black dashed line.

The beam patterns resemble those one would expect for planar array with 0.585λ spacing. It has a high broadside EIRP that rapidly drops off with steering angle. The sidelobes are generally low until wide steering angles when they become comparable to the main lobe. Due to the rectangular aperture of the 3-by-5 array, the beams are more concentrated in the $x-$plane than they are in the $y-$plane due to the increased aperture in that plane. The beams are so concentrated in the $x-$plane that there is a notable ripple in the maximum EIRP measurement as there is a sharp drop in power between beam angles.

Figures 7.6 shows the beam patterns for the $y-$cylinder array shown in Figure 7.4b. Note that the $y-$cylinder is curved along the $x-$plane and its curvature is quite aggressive. As can clearly be seen, the maximum EIRP curve is more even along the $x-$plane than it is in the $y-$plane. However, the broadside EIRP is much lower than the planar array. The increased spacing between tiles due to the aggressive curvature results in large side lobes at all steering angles. Finally, the $y-$plane patterns resemble those of the planar array.

Figures 7.7 shows the beam patterns for the $x-$cylinder array shown in Figure 7.4c. This cylinder is curved along the $y-$plane and has a more gradual curvature than the $y-$cylinder. Thus the resulting beam patterns have lower side lobes in the $y-$plane



(a) $x-$plane cut (b) $y-$plane cut

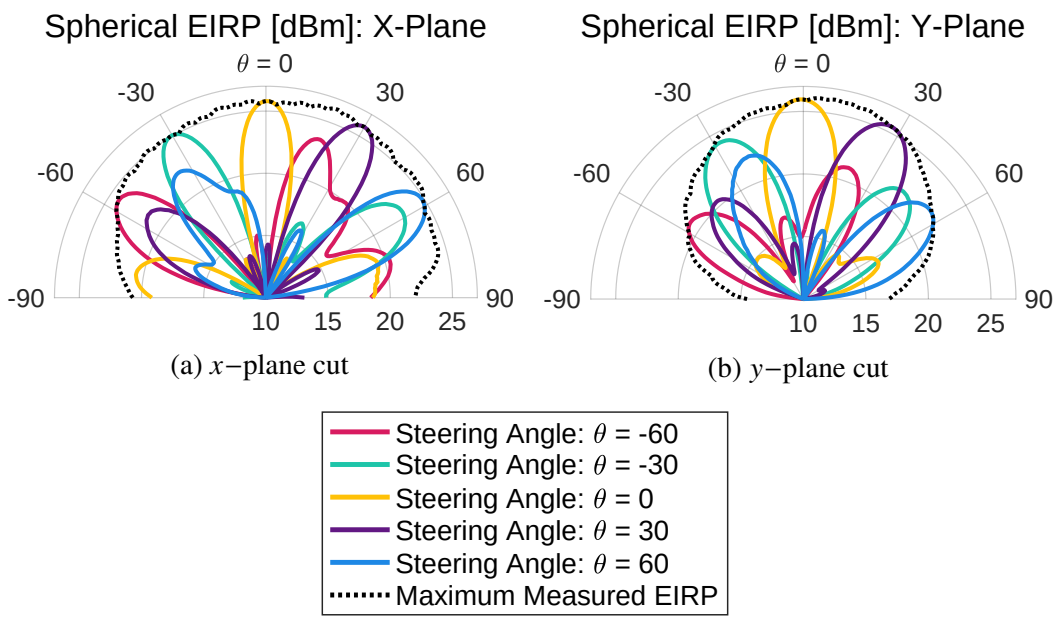
Figure 7.7: Collection of measured beam patterns for the $x-$cylinder configuration. The maximum measured beam power at each angle is indicated by the black dashed line.

due to the smaller spacing. However, the angular coverage of the array is smaller due to the short surface length in the $y-$ direction and the large radius of curvature. Therefore, the array exhibits only a minor increase in maximum EIRP at wide angles despite the curvature. The broadside EIRP of the Array is between that of the planar and $y-$cylinder arrays.

Figures 7.8 shows the beam patterns for the spherical array shown in Figure 7.4d. Note that the spacing in tiles is much larger in this configuration than in the planar due to the change in Gaussian curvature. As a result, the array exhibits high side lobes throughout its steering range. In addition, the total aperture of the array increases due to the larger spacing. Despite the lower fill factor, this larger aperture results in a broadside EIRP higher than that of the $y-$cylinder array. Like the $x-$cylinder, the modest angular coverage due to the array's short vertical length reduces the increase in maximum EIRP at wide steering angles in the $y-$plane. However, the $x-$plane exhibits much better EIRP at wide steering angles.

7.4 Analysis of Geometric Behavior

Figure 7.9 shows the maximum measured EIRP of beams steered along the x- and y- planes for each configuration. The measured curves follow the trends expected given the theory presented in Chapter 4. In particular, shapes with curvature in a



(a) $x-$plane cut (b) $y-$plane cut

Figure 7.8: Collection of measured beam patterns for the spherical configuration. The maximum measured beam power at each angle is indicated by the black dashed line.

given plane exhibit lower maximum EIRP at broadside but less angular sensitivity in that plane than their straight counterparts.

The spherical and y-cylinder configurations are curved along the x-plane and thus achieve a higher EIRP at wide steering angles than the planar or x-cylinder configurations which are not curved. Similarly in the y-plane, the EIRP of the curved spherical and x-cylinder configurations drops off more gradually at wide angles than the straight planar and y-cylinder configurations. Finally, the maximum broadside EIRP is largest for the planar array. These EIRP results indicate that the array's gain behaves similarly.

Figure 7.9 shows some non-idealities that are important to address. Due to the asymmetry inherent in a rectangular 3-by-5 array, each configuration is longer in the x-plane than in the y-plane. Therefore, there is a larger reduction in broadside cross-sectional area and a larger increase in angular coverage due to curvature in the x-plane than in the y-plane. This results in the x-cylinder having a larger maximum broadside EIRP than the y-cylinder. In addition, the x-cylinder has a higher angular sensitivity in the y-plane than the y-cylinder has in the x-plane. It is expected that the two geometries would exhibit identical maximum broadside EIRP and complimentary angular sensitivities in the case of a square array.

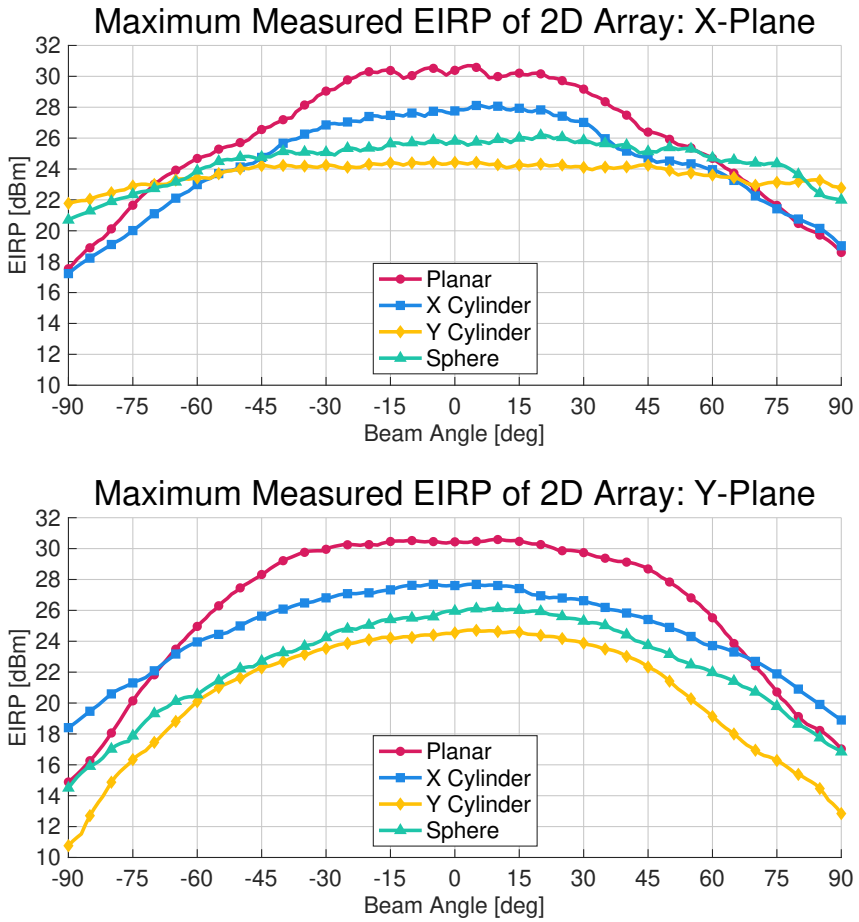


Figure 7.9: Maximum measured EIRP as a beam is steered in the x- and y- planes for the 3-by-5 tile array in different geometric configurations. Reprinted with permission from the copyright holder, EuMA.

As can be seen in Figure 7.4, the element spacing, and thus the total surface area, is larger in the spherical configuration due to the change in Gaussian curvature. This increase in total area partially compensates for the loss in broadside cross-sectional area due to curvature, making the spherical maximum broadside EIRP higher than the y-cylinder but lower than the planar or x-cylinder configurations. This partial area compensation comes at the cost of increased grating lobes. Finally, the sphere exhibits higher angular sensitivity in the y-plane than in the x-plane due to the rectangular asymmetry.

The results in Figure 7.9 also provide a measure of the -3 dB steering range of the different geometries. The planar array has an average steering range of 88° while the spherical array has an average steering range of 118°. Meanwhile, the $x-$ and $y-$cylinders have average steering ranges of 92.5° and 138°, respectively. The measured maximum gain and -3 dB steering ranges in each plane for the four

Table 7.1: Measured Performance of Shape-Changing Phased Array Geometries

Geometry	Max EIRP [dBm]	-3 dB Steering Range: X-Plane [deg]	-3 dB Steering Range: Y-Plane [deg]
Planar	30.7	76	100
X-Cylinder	28.1	82	103
Y-Cylinder	24.7	180	96
Spherical	26.2	145	91

geometries are listed in Table 7.1. While the predicted trade-off between steering range and max gain predicted by aperture projection analysis are clear in the EIRP and $x-$plane steering range results, the $y-$plane steering range results do not follow the expected pattern. It is thus clear that the short side length of the array in the $y-$plane was insufficient to effect the steering range.

7.5 Conclusions

The general consistency between the geometric theory preseented in Chapter 4 and the measurement results indicates that aperture projection analysis is a valid analytic model[3] for estimating the gain properties of different geometries. In addition, the array demonstrated that by using shape change it is possible to break this trade-off. By shifting between the Y-Cylinder and Planar geometries alone, the array can increase its steering range by 104° or increase its broadside gain by 6 dB. The reconfigurable nature of the array allows it adapt itself to the geometry best suited for a given application. This reconfiguration is easy to control due to the bi-stable design of the array backbone. This origami-inspired shape-changing phased array is an important first step towards enabling the applications discussed in Section 5.1.

References

[1] J. O. Jacobsen, G. Chen, L. L. Howell, and S. P. Magleby, "Lamina emergent torsional (let) joint," *Mechanism and Machine Theory*, vol. 44, pp. 2098–2109, 11 Nov. 2009, ISSN: 0094114X. DOI: 10.1016/j.mechmachtheory.2009.05.015. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0094114X09001116.

[3] At least for arrays of patch antennas.

Chapter 8

META-GAPS

> Yo, I treat this like my thesis
> well-written topic, broken down into pieces.
> I introduce then produce, words so profuse.
> It's abuse how I juice up this beat, like I'm deuce.
>
> ---
>
> *Lauryn Hill*
> *Final Hour*

The increased element spacing required for shape change introduces grating lobes that are detrimental to array performance. This chapter introduces the concept of a meta-gap, flexible sheets of patterned metal located in the gaps between tiles, to compensate for the increased spacing. The concept evolves naturally from the design and optimization problem posed by using passive structures to manipulate fields under various excitations.

Section 8.1 begins the chapter by motivating the use of passive flexible metal structures in the gaps. Next Section Section 8.2 argues for the use of optimization for design. Section 8.3 discusses the challenge of electromagnetic design via optimization and the benefits of using the Lavaei-Babakhani method. The idea of a switchable meta-gap structure based on this approach is presented in Section 8.4 and the implemented design is described in Section 8.5. Finally, Section 8.6 concludes the chapter by reviewing the literature of similar concepts, the use of switches and meta-materials to alter antenna and array performance.

8.1 Motivation

As discussed in Section 5.5, the distances between elements in an shape-changing phased array must increase, thus increasing the side lobe levels of the array and altering the antenna coupling. In addition, an array of rigid elements must introduce gaps in the array to achieve arbitrary shape change. Thus, the fill factor of the array decreases reducing the fraction of power that can be concentrated in the main beam. From this perspective, it appears that a shape-changing phased array will have worse performance that a fixed array with the same geometry due to the introduction of gaps.

However, these gaps also present an opportunity in the form of extra unused area on the radiation surface. For example, additional radiators could be introduced to fill the gaps, increasing the fill factor and reducing grating lobes. However, there are often regions where the small gap size or irregular geometry precludes additional antennas. In addition, when folded, these active elements would be completely obscured, an inefficient use of resources.

Instead, meta-material inspired passive metal networks on flexible sheets can be placed in the unused area between antennas in sparse arrays. These *meta-gaps* can manipulate the near-field environment in order to compensate for the increased spacing. The metal structure could behave like a reflector, directing incident power towards a neighboring antenna, increasing its effective aperture and thus the array fill factor. Mutual-coupling between the antenna and the meta-gap metal could alter the port impedance to improve matching. Meta-gaps could also work in concert to suppress surface modes across the array associated with grating lobes. These flexible meta-gaps could improve performance of both deployable arrays and rigid shape-changing phased arrays.

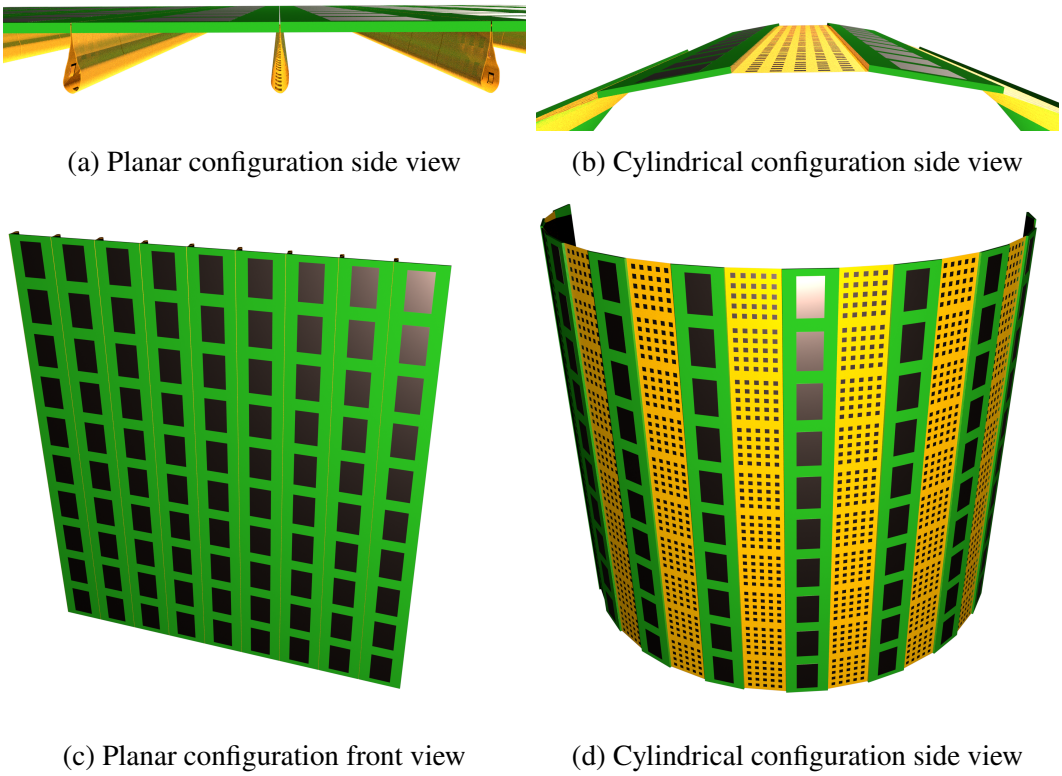
Figure 8.1 illustrates the advantages of *flexible* meta-gaps. In a planar shape there are no gaps, so the meta-gaps fold behind ground plane backed antennas where they will minimally interact with the radiating surface. When the array changes into a cylinder[1], the meta-gaps are deployed between the antennas and are configured to mitigate the effects of increased antenna spacing or otherwise enhance the array performance.

8.2 Motivation for Optimization Approach

Any metal located in the near-field of an antenna will affect its performance through mutual coupling. The fields generated by the antenna induce secondary currents on the metal surface which in turn generate secondary fields. These fields in turn induce currents on the antenna surface, perturbing the current distribution. Thus the far-field radiation pattern is altered by both the perturbation of the antenna currents and the re-radiation by the secondary currents. The port impedance will also change due to the modification of the antenna currents. Finally, the induced currents on the metal dissipate power, reducing the antenna efficiency.

However, these modifications are not strictly a bad thing. The distortion of the far-

[1]Technically, cylinders are developable surfaces and thus the element spacing does not need to increase. However, a cylinder with folds provides a clearer visualization than a sphere and still effectively illustrates the concept.



(a) Planar configuration side view

(b) Cylindrical configuration side view

(c) Planar configuration front view

(d) Cylindrical configuration side view

Figure 8.1: Illustration of how meta-gaps can be deployed to fill the gaps in a shape-changing array. (a) In planar configuration the sheets fold behind the tiles. (b) In cylindrical configuration they expand to fill the gaps. Reprinted with permission from the copyright holder, EuMA.

field pattern can be desirable; Yagi-Uda antennas use passive resonant conductors to increase the directivity of a dipole. The distance between an antenna and a ground plane is commonly adjusted in order to alter the port impedance. In a sense, the patch in an aperture fed patch antenna is a "parasitic" metal that couples power from the aperture into the desired radiating mode. Thus metal within the near-field environment of an antenna can either benefit or harm its performance. The same can be argued for metal in the near-field of an array.

The challenge is identifying what metal patterns benefit the antenna performance. In fact, the entire process of antenna design is arguably finding solutions to this problem. While there are many antenna design methods, the most common is leveraging intuition about the behavior of known structures to propose new structures or modifications to old ones that are then verified with analytic and numeric calculations.

Unfortunately while electromagnetic fields in a linear media are linear with respect to the excitation, they are not linear with respect to the boundary conditions[2]. Adding conductors alters the boundary conditions of the system and therefore can drastically change its fundamental behavior. Resonances can be introduced or removed. Surface currents and coupling can be drastically altered. So, while intuition based on existing structures is an excellent guide for designing new ones, new designs are not guaranteed to function as expected; this is why designs must be checked with numeric calculations. A more critical concern is that this method of design can miss designs with excellent performance despite having little resemblance to any known structure.

The difficulty of this design problem is exacerbated in the scenario where the system is subject to different field excitations, as is the case when the structure is embedded within a phased array. As the beam is steered, the near-field environment of the array is altered, changing the currents induced on the surface of the conductor and thus its effect on the far-field pattern and the mutual coupling. This effect is observed in phased arrays by the element impedance and patterns changing with excitation as discussed in Section 3.6. Thus the behavior of the structure must be considered under *every desired excitation*.

While the above concerns about excitations and non-linearity always apply in electromagnetic design, how relevant they are depends on the specific problem. Unfortunately, both are relevant in the design of meta-gaps. While it is highly likely that there *exists* a particular pattern of metals that can be placed in the gaps of an array to boost performance, it in not intuitively clear *what* that pattern is. It might be desirable to introduce parallel metal lines that resonate to reduce coupling between antennas like a psuedo-bragg reflector. It could also be beneficial to have a set of rings that strongly couple to the magnetic field and store the energy like an inductor, altering the port impedance in a way that compensates for the change due to beam steering. Or perhaps the ideal design would intersperse rings and lines. While there are any number of intuitive arguments for patterns that *might* enhance performance when placed in the gaps of an array, there is not a set of patterns that are *known* to do so. Thus a different approach should be taken in order to identify and demonstrate well-performing patterns that can then be used as a basis for intuitive reasoning.

[2]For example, a wave of any frequency can propagate along the surface of a conductor. However if four conductors are used to form a waveguide, only waves above the cutoff frequency can propagate between the conductors. This exclusionary behavior cannot be described by the sum of waves traveling along the individual conductors.

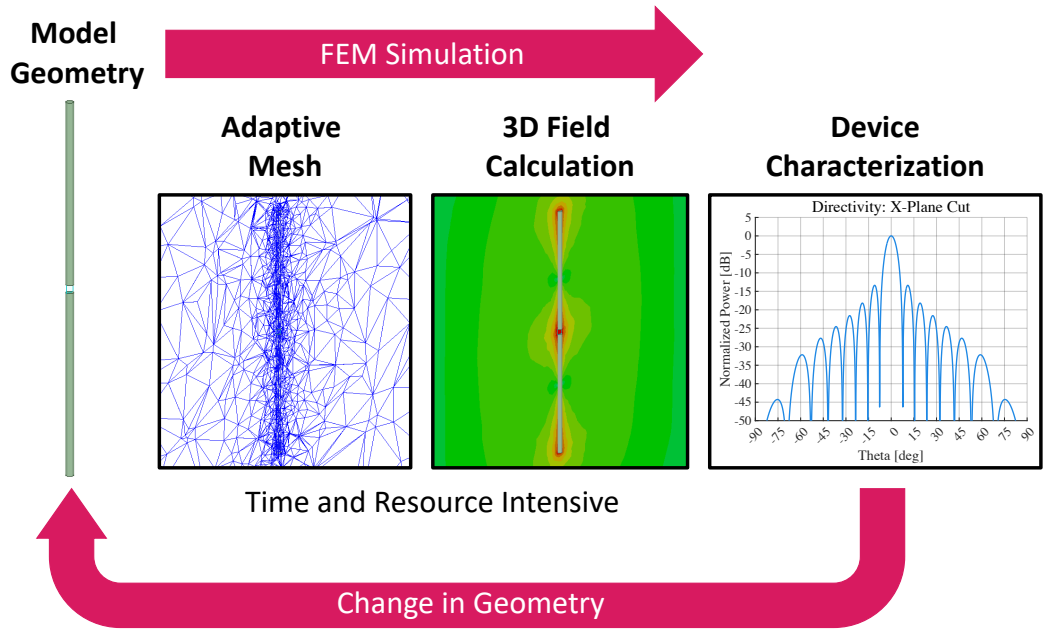


Figure 8.2: Typical optimization loop for electromagnetic design using Finite Element Analysis (FEM). FEM is time and resource intensive, greatly limiting the number of iterations.

Optimization methods enable the exploration of a large number of candidate patterns in order to identify good performing solutions. In addition, the relative performance of different optimization methods can be used to infer the nature of the design space and properties of good performing solutions.

8.3 Electromagnetic Simulation and Optimization

The challenge with optimization-based electromagnetic design is the computational complexity of solving Maxwell's equations for a given set of boundary conditions. The three main approaches to solving this problem, Method of Moments (MoM), Finite Element Method (FEM), and Finite-Difference Time-Domain (FDTD) each discretize either a simulation volume or surface area into discrete segments in order to locally solve discrete versions of Maxwell's equations. To be accurate, the discritization must be small enough to accurately capture the field behavior in that local region. Thus areas with fine geometric features require a denser mesh, increasing the computational resources required to solve the problem. Solving Maxwell's equations is computationally intensive for systems with complicated geometries, large simulation volumes, and/or high operating frequencies.

Because Maxwell's equations are linear with respect to excitation, once the elec-

tromagnetic system has been solved its behaviour under different excitations can be rapidly calculated from the solution. Changes to the geometry of the system, however, invalidate the mesh and the solution derived from it, thus requiring the entire computational problem to be repeated.

Figure 8.2 shows a typical optimization loop for geometric design based on FEM simulation. A discrete mesh is generated around the model geometry, which is then used to solve the electromagnetic fields. These fields are then used to characterize the performance of the geometry, such as the radiation pattern or impedance matching. Based on these results, the geometry is altered and the process repeats. Each iteration requires another FEM simulation, greatly increasing the computational resources required. It is not uncommon for a single FEM simulation to take anywhere from 15 minutes to over 24 hours. Therefore, the use of FEM simulation in the loop greatly restricts the capability of the optimizer.

Ideally, the field solver in the optimization loop would not require repeatedly meshing the simulation space and solving Maxwell's equations. The Lavaei-Babakhani method instead uses network embedding to characterize different design decisions using a single FEM simulation.

Network Embedding

Any linear electromagnetic system can be described by an M port network, S'. This network can be divided into two sub-networks connected together by N internal ports, with one containing all M external ports. Figure 8.3 shows the resulting decomposition consisting of an $M + N$ port *embedding* network S and an N port *embedded* network S_e using a circuit as an example. Using this decomposition, the effect of altering the embedded network on total network can be calculated [1].

By labeling the port order of S such that the first M ports are external and the remaining N ports are internal, it can be broken into the sub-matrices shown in Equation 8.1

$$S = \left[\begin{array}{c|c} S_{mm} & S_{mn} \\ \hline S_{nm} & S_{nn} \end{array}\right] \tag{8.1}$$

where S_{mm} is the externally reflected waves given an external excitation, S_{mn} is the externally reflected waves given an internal excitation, S_{nm} is the internal reflected waves given an external excitation, and S_{nn} is the internal reflected waves given an internal excitation. By labeling the internal and externally incident waves as V_n^+ and V_m^+, and the internal and externally reflected waves as V_n^- and V_m^-, we can analyze

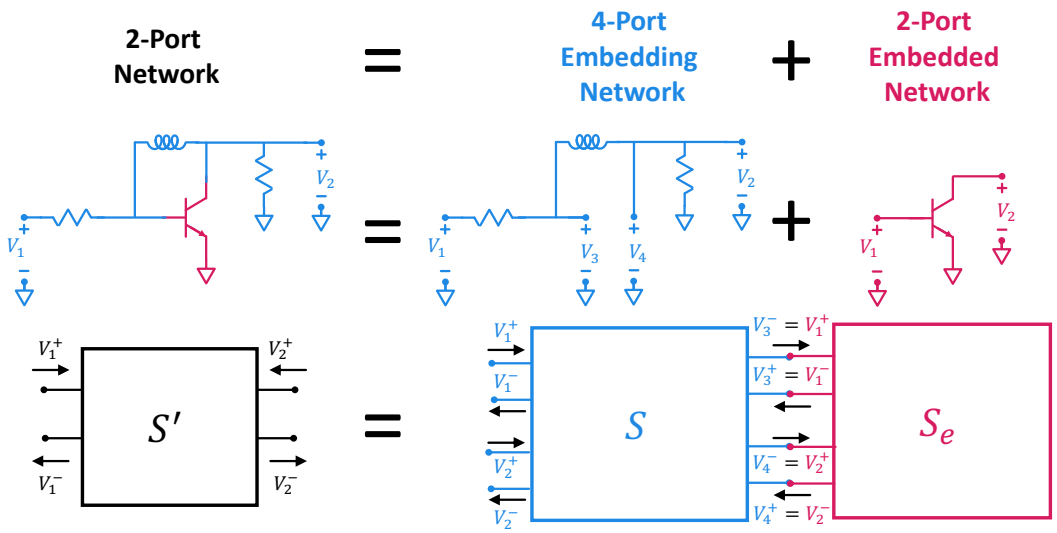


Figure 8.3: A 2-port network can be decomposed into a 4-port embedding network, with 2 internal and 2 external ports, and a 2-port embedded network with 2 internal ports.

their interactions with the embedded network.

$$V^- = SV^+ \tag{8.2}$$

$$\left[\begin{array}{c} V_m^- \\ \hline V_n^- \end{array}\right] = \left[\begin{array}{c|c} S_{mm} & S_{mn} \\ \hline S_{nm} & S_{nn} \end{array}\right] \left[\begin{array}{c} V_m^+ \\ \hline V_n^+ \end{array}\right] \tag{8.3}$$

Equation 8.3 can be rewritten as two separate equations.

$$V_m^- = S_{mm}V_m^+ + S_{mn}V_n^+ \tag{8.4}$$

$$V_n^- = S_{nm}V_m^+ + S_{nn}V_n^+ \tag{8.5}$$

The relationship between the internal incident, V_n^+, and reflected,V_n^-, waves is determined by the N port embedded network, S_e. Note that waves reflected by the *embedding* network, V_n^-, are incident to the *embedded* network and that waves incident to the *embedding* network, V_n^+, are those reflected by the *embedded* network.

$$V_n^+ = S_e V_n^- \tag{8.6}$$

We can use Equations 8.4, 8.5, and 8.6 to solve for the total network behavior. First we plug Equation 8.6 into Equation 8.5 and solve for V_n^-

$$V_n^- = S_{nm}V_m^+ + S_{nn}S_e V_n^- \tag{8.7}$$

$$\left[I - S_{nn}S_e\right] V_n^- = S_{nm}V_m^+ \tag{8.8}$$

$$V_n^- = \left[I - S_{nn}S_e\right]^{-1} S_{nm}V_m^+ \tag{8.9}$$

with I symbolizing the n by n identity matrix. We then use Equation 8.6 again to solve for V_n^+.

$$V_n^+ = S_e \left[I - S_{nn}S_e\right]^{-1} S_{nm}V_m^+ \tag{8.10}$$

We can now solve for V_m^- exclusively in terms of V_m^+ using Equation 8.4.

$$V_m^- = \left(S_{mm} + S_{mn}S_e \left[I - S_{nn}S_e\right]^{-1} S_{nm}\right) V_m^+ \tag{8.11}$$

Note that the M external ports determine the behavior of the total system $S^{'}$.

$$V_m^- = S^{'}V_m^+ \tag{8.12}$$

Thus we can can solve for the total system behavior, $S^{'}$, given the embedding network, S, and the embedded network, S_e, using Equation 8.13.

$$S^{'} = S_{mm} + S_{mn}S_e \left[I - S_{nn}S_e\right]^{-1} S_{nm} \tag{8.13}$$

As will be shown, Equation 8.13 is a powerful tool that can greatly reduce the computation time required to characterize the effects of boundary conditions on electromagnetic performance.

The Lavaei-Babakhani Method

The Lavaei-Babakhani method uses network embedding to remove boundary conditions from the electromagnetic simulation and instead enforce them as part of a network optimization problem [2]. The ports of an electromagnetic system are divided into three groups, the *control ports*, the *excitation ports*, and the *sensor ports*. The excitation ports are the typical ports found in an electromagnetic simulation that relate the behavior of the system to an externally connected network. The sensor ports are connected to miniature probe conductors embedded in the simulation volume to measure the strength of the field at a desired location and polarization. Finally, the control ports are used to enforce boundary conditions that alter the relationship between the excitation and sensor ports.

Control ports are placed in the gaps of a grid of small conductors squares within the simulation volume. From an EM simulation perspective, these ports enforce a specific field boundary condition between the two conductors. The orientation of the electric and magnetic fields are fixed, but their relative amplitude and phase are determined by the port excitation. Because Maxwell's equations are linear with respect to the excitation, the simulator can calculate the Green's function for the

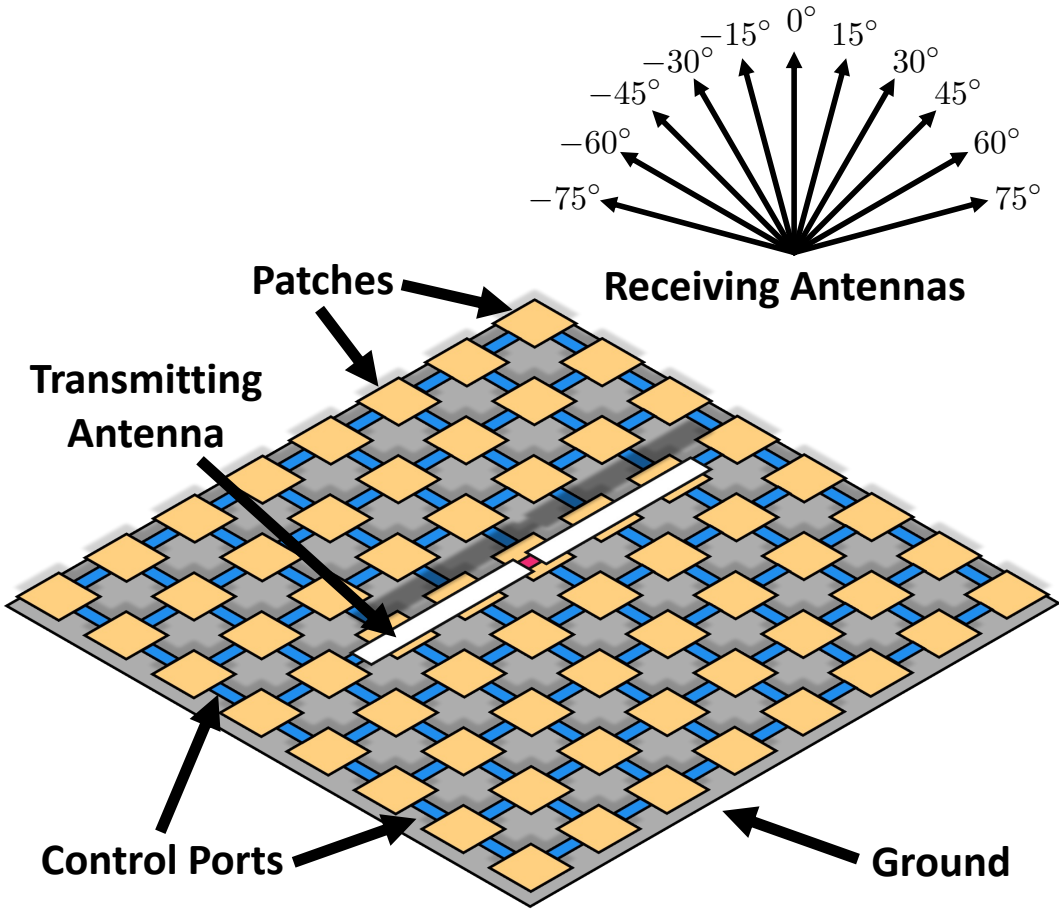


Figure 8.4: An example of the Lavaei-Babakhani Method. A mesh of conductors sits between a dipole and a ground plane. The control ports (blue) determine the boundary conditions between patches in the mesh. The excitation port (red) drives the dipole. Sensor ports (not shown) are attached to receving antennas every 15° in the far-field. The resulting EM simulation will create an $s-$parameter matrix that relates the sensed fields to the excitation given the boundary conditions imposed by the control ports [2].

space without knowledge of the excitation. In turn this Green's function is used to calculate the relationship between ports in the simulation volume, usually in the form of an s-parameter matrix. Thus by changing the external network connected to the control ports, specific field boundary conditions can be enforced.

Figure 8.4 shows an example of an antenna design problem to be solved [2]. The goal is to find a conductor pattern that improves performance when placed in between a ground plane and a dipole. There is $M_e = 1$ excitation port at the dipole input, N control ports in the conductor mesh, and a set of M_s sensor ports distributed along a sphere in the far-field to measure the radiation pattern. A single electromagnetic

simulation of the structure will generate an $(M_e + M_s + N) \times (M_e + M_s + N)$ s-parameter matrix that can be represented as an $M_e + M_s + N$ port system with the M_s ports capturing the radiation pattern of the dipole when excited by the M_e excitation ports given the boundary conditions imposed by the N control ports. Since the goal of the optimization problem is to find a conductor pattern, the control ports enforce whether their is a metal connection (a short) or not (an open). These two boundary conditions can be represented by an s-parameter of $\Gamma = -1$ and $\Gamma = 1$, respectively. Thus the boundary conditions can be enforced by connecting the N ports of the system model to an N-port *controller matrix* of the form shown in Equation 8.14 where $\Gamma_i = \pm 1$.

$$S_{controller,switch} == \begin{bmatrix} \Gamma_1 & 0 & \dots & 0 \\ 0 & \Gamma_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \Gamma_n \end{bmatrix} \tag{8.14}$$

The power of the Lavaei-Babakhani method comes from the realization that, given this setup, calculating the effect of the boundary conditions on the behavior of the system is simply a network embedding problem. The single EM simulation provides an $M + N$ embedding network with $M = M_e + M_s$ external ports and N internal ports. Changes to the system performance with a change in boundary conditions can be quickly solved by altering $S_{controller}$ and using Equation 8.13. In this light, Equation 8.13 highlights the non-linearity of Maxwell's equations with respect to the boundary conditions; the total network behavior is related to the boundary conditions imposed through a matrix inverse.

The Lavaei-Babakhani method allows for rapid optimization as different design decisions can be evaluated orders of magnitude faster than with an electromagnetic simulation. However, there are drawbacks to this method. The biggest challenge is that optimizing the switch controller given by Equation 8.14 is an NP-Hard problem [2]. Another challenge is the complexity of the electromagnetic simulation. Because each design choice, measurement point, and excitation requires a separate port, the total number of ports can be very large. This is especially true when optimizing metal structures embedded within a 2D array. A single simulation can easily have hundreds of ports, if not thousands. The increased number of ports drastically increases the computational resources needed to perform the analysis and process the resulting s-parameter model. In addition, in antenna optimization problems, sensor ports must be placed in the far-field to detect the radiation pattern. For array

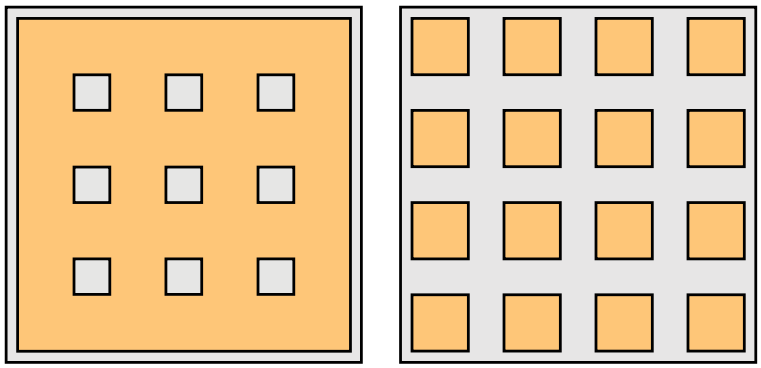
problems this requires a very large simulation volume, requiring volume based simulation methods like FTDT and FEM to use a large amount of memory. The simulation problem might be so immense that it cannot even be solved without a supercomputer. For large array problems, performing several simulations with more efficient techniques such as Domain Decomposition (DDM) might consume fewer computational resources. Finally, a challenge that hampers any computational optimization technique is that the performance of the final system critically depends on the assumptions and accuracy of the electromagnetic model. A metal pattern that is optimal for the model might not be optimal for the physical array with manufacturing variation and without any simplifying assumptions.

8.4 Switched Meta-Gap Operating Principle

Inspired by the Lavaei-Babakhani method and its limitations, we present a switch-controlled meta-gap structure capable of dynamic reconfiguration. The structure consists of a grid of metal squares connected to each other by RF switches. These switches can be turned on and off to create conductive pathways that control how and where currents are excited. As the density of the grid increases, the size of each metal square decreases and the number of switches increases. Thus individual squares minimally interact with the near-field environment as they cannot support substantial surface currents. However, as switches are enabled, the conductive path length increases and surface currents can be supported and steered in a particular shape. With a fine enough grid, the switches can essentially program the shape of conductors on the surface into an arbitrary configuration. In a sense, the concept is similar to the pixel-based automated design of waveguide couplers presented in [3].

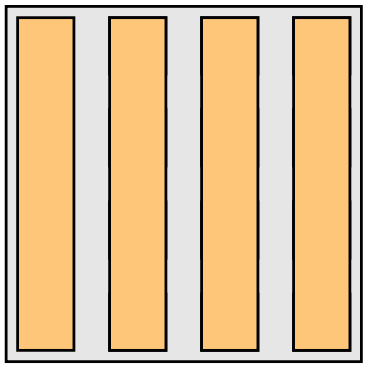
In this ideal scenario, the switched meta-gap can emulate any desired metal structure. Figure 8.5 illustrates this capability. If all of the switches are turned on, than the grid forms a single metal sheet and the meta-gaps behave like a solid ground plane between antennas. If the switches are all off, than no conductive paths exist and the meta-gaps become nearly transparent. These two extremes represent complete reflection and transmission of an incident wave, greatly altering the fields in the gaps. In addition, parallel lines or resonant rings can be formed, potentially introducing resonances in the near-field.

This switched-passive network concept enables the exploration of the possibilities of meta-gaps along multiple fronts. In a sense, this structure is a physical implementation of the Lavaei-Babakhani method and thus is well adapted to optimization

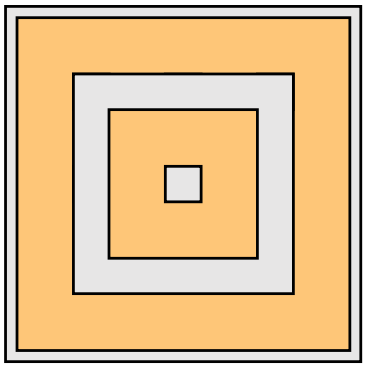


(a) All switches are on.

(b) All switches are off.



(c) Vertical parallel lines.



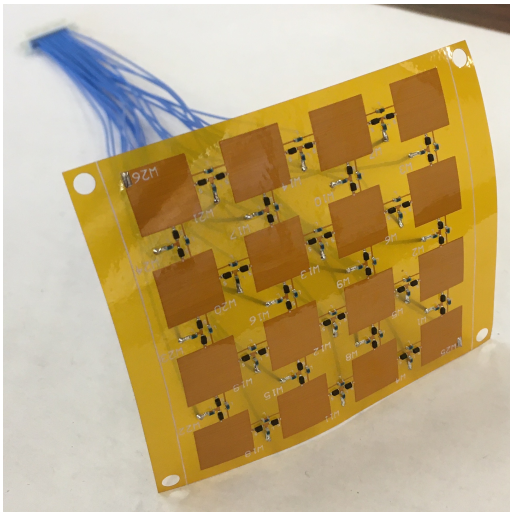
(d) Concentric rings.

Figure 8.5: An ideal meta-gap can emulate any metal pattern by turning a particular set of switches on and off.

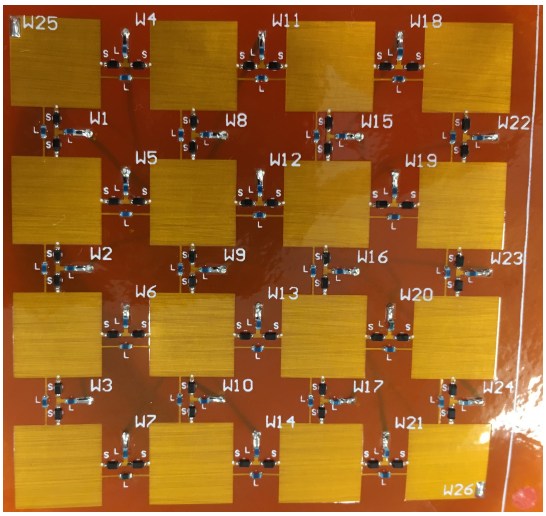
in simulation. In addition, by using programmable RF switches the conductive pattern on the sheets can be optimized on a physical array. This *in-situ* optimization avoids the accuracy issues of simulation because the optimal solution *is* the optimal solution for the physical array. In addition, concerns about simulation complexity are irrelevant *in-situ* as the array behavior is "calculated" instantaneously[3]. Ideally, every possible meta-gap pattern can be achieved and measured using the switched passive network, making it an excellent platform to explore meta-gap performance.

Of course, this idealized design is impossible for several reasons. The first is that the switches themselves have a minimum length, limiting the density of the grid. More importantly, the switches are not-ideal and thus do not create perfect isolation nor conduction between neighboring squares. Because power from the near-field is dissipated through the switches, the array efficiency decreases are more switches are added. Finally, the best performing RF switches in terms of isolation and loss are

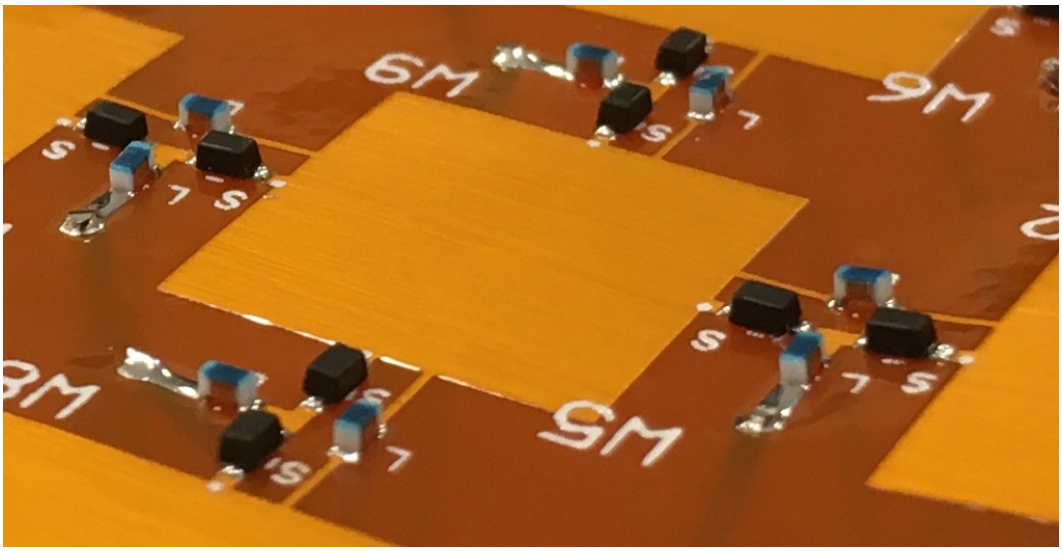
[3] Although not *measured* instantaneously as discussed in Chapter 10.



(a) Full view of sheet, header, and wires.



(b) Top view of meta-gap sheet.



(c) Closeup of switch layout.

Figure 8.6: Manufactured meta-gap sheet. Each array contains a 4-by-4 grid of conductors connected with 24 RF switches. Switches are formed by back to back PIN diodes and are driven with a 32 AWG wire connected to the backside.

PIN diodes which require active biasing, actively consuming power. Thus a dense grid can consume a significant amount of power. Finally, the hardware complexity of programming a large number of switches precludes very dense grids.

8.5 Switched Meta-gap Design

The switched meta-gap sheet shown in Figure 8.6 was designed to explore the possibilities presented by meta-gaps via *in-situ* optimization. The sheet is comprised of a 0.5-oz copper layer between two 1-mil layers of flexible polyimide. This thin

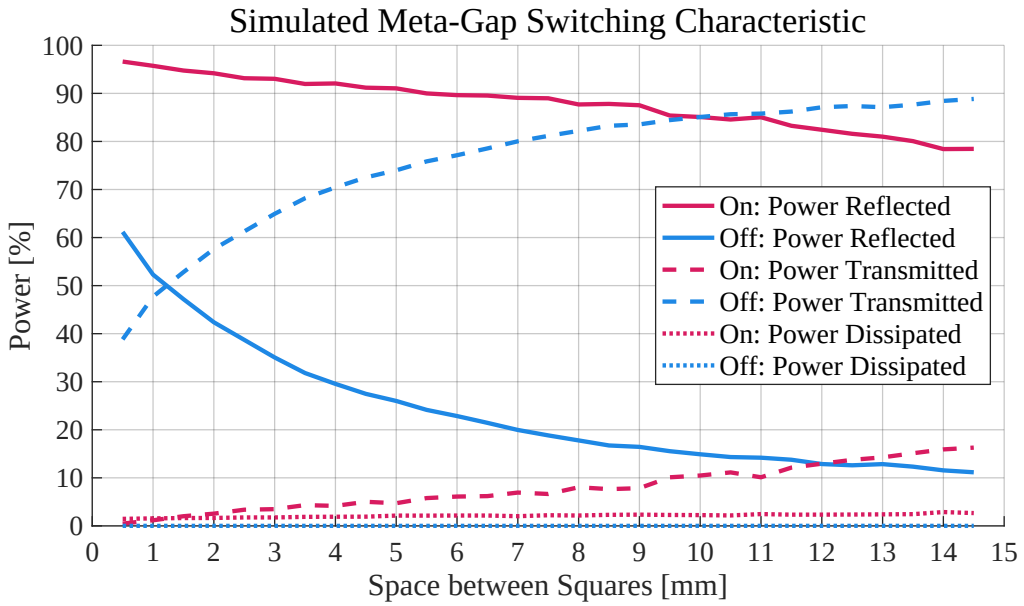


Figure 8.7: Simulated behavior of the meta-gap sheet as the gap size between conductors increases. Percentage of power reflected, transmitted, and dissipated is shown for cases when the switches are all on and all off. The on resistance of switches is based on measurement. Reprinted with permission from the copyright holder, EuMA.

substrate was selected to maximize flexibility and minimize its effect on the near-field environment. The sheet is held in place by four corner mount holes that connect to radiators. The exposed surface is a 6cm square, half of a wavelength at 2.5 GHz. The metal forms a 4-by-4 grid of 9.7mm squares separated by 5.3mm gaps. Neighboring square in the grid are connected by programmable RF switches as shown in 8.6c.

The metal grid was designed to achieve maximum variation in reflectivity and transparency when all switches are simultaneously turned on and off. When all of the switches are on, the surface would ideally behave like a ground plane, maximizing the reflection of incident waves. When all of the switches are off, the surface should be largely transparent to incident waves. Figure 8.7 shows the results of a FEM analysis of the structure to characterize how the metal spacing alters the surface's ability to reflect incident plane waves. A space size of 5.3mm was selected to minimize the power reflected in the off state while reflecting 90% of the power in the on state.

Figure 8.8 shows the schematic for the RF switches. The core of the switch is a pair of back to back PIN diodes (MACOM MADP-007433-12790T). Like all diodes,

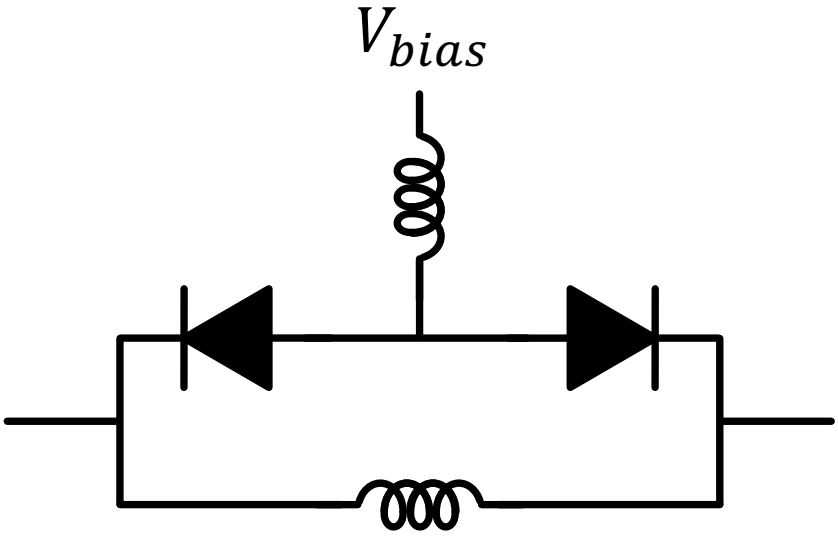


Figure 8.8: Schematic of RF switch.

PIN diodes can behave as conductors or insulators depending on the DC bias voltage across their terminals. PIN diodes make particularly good RF switches due to their long intrinsic regions. When on, this region is flooded with carriers that are unable to quickly exit the region when a large voltage is applied. Thus at high frequencies, carriers remain in the intrinsic region even if the applied voltage drops below the threshold voltage. Thus the PIN diode behaves as a fairly linear resistor at high frequencies, at the cost of reduced switching speed. When off, the intrinsic region increases the length of the parasitic capacitance and thus increases isolation between the terminals at high frequencies. Their low linear on-resistance in addition to their high isolation make PIN diodes excellent RF switches.

Back to back diodes are used in order to establish an independent DC bias for each switch. The center node of the switch is biased with a 30 nH RF choke inductor (COILCRAFT 0402DC-30NXJRW), while a common DC ground is established on adjacent squares by a second 30 nH inductor in parallel with the switch. The inductors prevent leakage of the RF signal. Each switch is controlled via a header (SAMTEC SESDT-15-32-G-07.0-L) containing 24 control wires and 2 ground wires. The lightweight 32 AWG wires are connected to the sheet from the back side in order to maximize sheet flexibility and minimize unwanted near-field interactions.

The measured switching behavior of the structure is shown in Figure 8.9. At 2.5GHz, the diode insertion loss is -0.67 dB when on, and the isolation is 15.3 dB when off. The measured relationship between bias current and insertion loss at 2.5GHz is

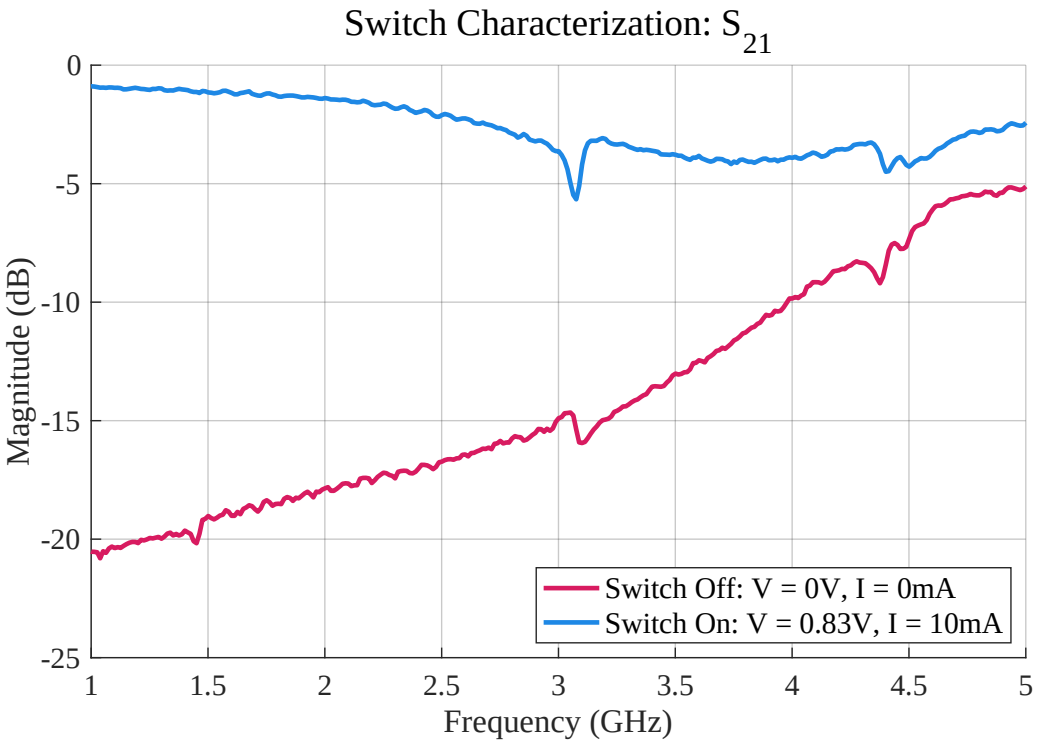


Figure 8.9: Measured insertion loss and isolation of switch test structure. Reprinted with permission from the copyright holder, EuMA.

shown in Figure 8.10. A bias current of 10 mA was selected in order to balance insertion loss with power consumption. While higher bias currents reduce insertion loss further, they do so with diminishing returns. This bias is programmed and supply by the second generation tiles described in Section 6.4.

8.6 Related Work

The use of switches and meta-materials to alter antenna and array performance is well explored. PIN diodes have been used to alter the operation frequency [4], polarization [5], and radiation patterns of both antennas and arrays [6], [7]. Techniques include using switches to reconfigure the conductor geometry [8], [9], alter resonator modes [10], change the array feed [11], or just simply select different elements [12]–[14]. Switches have also been used to modify parasitic conductors and adjust the near-field environment, such as enabling and disabling directors and reflectors in a Yagi–Uda antenna [15], [16].

While PIN diodes are commonly used, other RF switches have been explored to exploit different performance trade-offs. Other RF switches include vacators [17], MEMS switches [18]–[20], optically controlled photoconductive switches [21], and

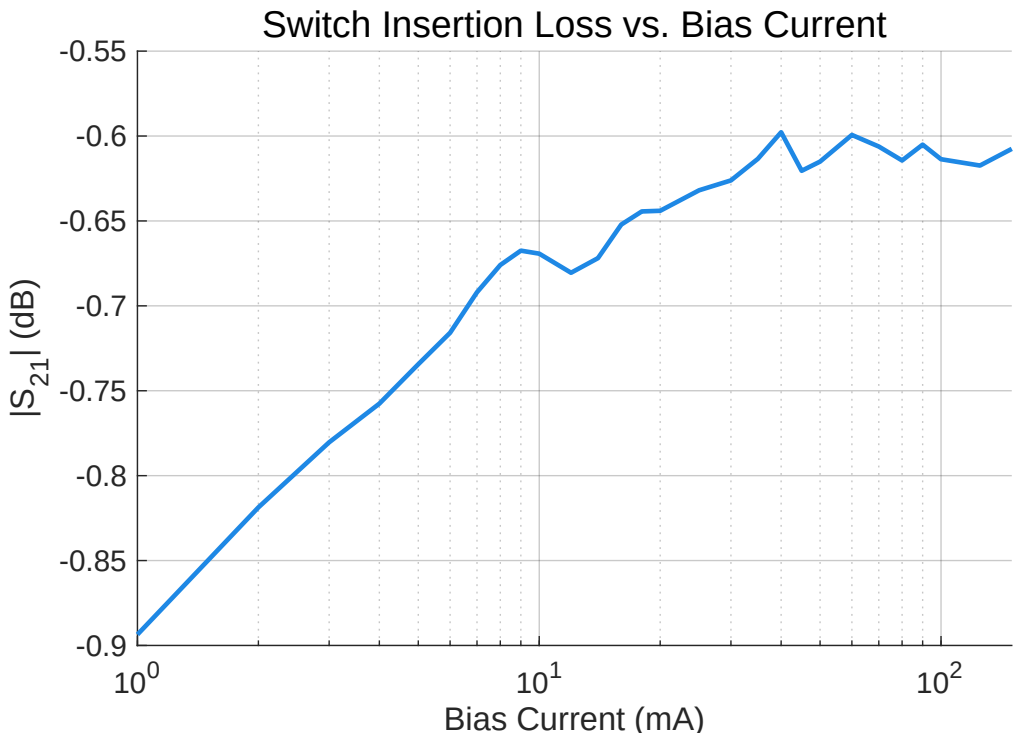


Figure 8.10: Measured insertion loss of switch test structure versus bias current.

thermally controlled materials [22].

Embedding meta-materials in radiating structures is another well-traveled research avenue. Research has shown that the coupling between antennas in an array can be reduced by up to 20 dB by placing meta-materials between the antennas [23]–[25]. This increased isolation enabled the size of a five element linear array to be reduced to 1.18λ [26]. In another work, a meta-material is used as a reactive impedance substrate to reduce the size of a patch antenna to 0.1λ [27]. [28] found that the bandwith of a patch antenna can be enhanced by over 200% by placing a meta-material sheet between the patch and the ground. Finally, a meta-material filter can be used to dynamically alter an antenna's polarization [29].

There has also been interesting work that combined meta-material concepts with switches. [30] demonstrated that switches can be used to change the properties of a frequency-selective surface to selectively reflect certain frequencies. Thus the radiation pattern of a dipole can be modified surrounding it with programmable sheets that can be either transparent or reflective [31].

Switched RF structures and meta-materials are useful for a broad range of applications. Switched parasitic structures can be used to modulate the radiation pattern

to enable physically secure wireless communication [32]. Pattern reconfiguration allow enables blind optimization of transceiver patterns to mitigate interference and multi-path effects using a constant modulus algorithm [33]. Reconfigurable antennas are also useful for in-orbit satellite adaptation, enhanced MIMO systems, and cognitive radio techniques [34]. Surfaces of programmable meta-materials can control the properties of a reflected wave, enabling both programmable reflector arrays [35], [36] and passive relays [37]. These surfaces have been developed for holography at THz frequencies [38]. Meta-materials can also be engineered to create anisotropic surfaces that bends RF energy around an object, low loss waveguides, and THz modulation [39].

References

[1] R. Spence, *Linear Active Networks*. John Wiley & Sons, Jan. 1970, ISBN: 047181525X.

[2] J. Lavaei, A. Babakhani, A. Hajimiri, and J. C. Doyle, "Solving large-scale hybrid circuit-antenna problems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, pp. 374–387, 2 Feb. 2011, ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2072010. [Online]. Available: http://ieeexplore.ieee.org/document/5635369/.

[3] C. Sideris, A. Hajimiri, C. Yang, *et al.*, "Automated design of a 3d printed waveguide surface coupler," in *2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Jul. 2015, pp. 318–319, ISBN: 978-1-4799-7815-1. DOI: 10.1109/APS.2015.7304545.

[4] M. Shirazi, J. Huang, T. Li, and X. Gong, "A switchable-frequency slot-ring antenna element for designing a reconfigurable array," *IEEE Antennas and Wireless Propagation Letters*, vol. 17, pp. 229–233, 2 Feb. 2018, ISSN: 1536-1225. DOI: 10.1109/LAWP.2017.2781463.

[5] F. Yang and Y. Rahmat-Samii, "A reconfigurable patch antenna using switchable slots for circular polarization diversity," *IEEE Microwave and Wireless Components Letters*, vol. 12, pp. 96–98, 3 Mar. 2002, ISSN: 1531-1309. DOI: 10.1109/7260.989863.

[6] P. Ngamjanyaporn, C. Phongcharoenpanich, and M. Krairiksh, "A wideband phase-shifterless switched-beam array antenna," in *2018 IEEE-APS Topical Conference on Antennas and Propagation in Wireless Communications (APWC)*, Sep. 2018, pp. 1–2, ISBN: 978-1-5386-6765-1. DOI: 10.1109/APWC.2018.8503739.

[7] V. Semkin, F. Ferrero, A. Bisognin, *et al.*, "Beam switching conformal antenna array for mm-wave communications," *IEEE Antennas and Wireless*

Propagation Letters, pp. 1–1, 2015, ISSN: 1536-1225. DOI: 10.1109/LAWP.2015.2426510.

[8] A. G. Besoli and F. D. Flaviis, "A multifunctional reconfigurable pixeled antenna using mems technology on printed circuit board," *IEEE Transactions on Antennas and Propagation*, vol. 59, pp. 4413–4424, 12 Dec. 2011, ISSN: 0018-926X. DOI: 10.1109/TAP.2011.2165470.

[9] Y. Sung, T. Jang, and Y.-S. Kim, "A reconfigurable microstrip antenna for switchable polarization," *IEEE Microwave and Wireless Components Letters*, vol. 14, pp. 534–536, 11 Nov. 2004, ISSN: 1531-1309. DOI: 10.1109/LMWC.2004.837061.

[10] S. Nikolaou, R. Bairavasubramanian, C. Lugo, *et al.*, "Pattern and frequency reconfigurable annular slot antenna using pin diodes," *IEEE Transactions on Antennas and Propagation*, vol. 54, pp. 439–448, 2 Feb. 2006, ISSN: 0018-926X. DOI: 10.1109/TAP.2005.863398.

[11] J. Ouyang, "A circularly polarized switched-beam antenna array," *IEEE Antennas and Wireless Propagation Letters*, vol. 10, pp. 1325–1328, 2011, ISSN: 1536-1225. DOI: 10.1109/LAWP.2011.2176308.

[12] Y. Chen, L. Zhang, Y. He, W. Li, and S.-W. Wong, "A pattern reconfigurable siw horn antenna realized by pin diode switches," in *2021 Computing, Communications and IoT Applications (ComComAp)*, Nov. 2021, pp. 112–115, ISBN: 978-1-6654-2798-2. DOI: 10.1109/ComComAp53641.2021.9653080.

[13] V. A. Nguyen and P. S. Ook, "Compact switched and reconfigurable 4-ports beam antenna array for mimo applications," in *2011 IEEE MTT-S International Microwave Workshop Series on Intelligent Radio for Future Personal Terminals*, Aug. 2011, pp. 1–3, ISBN: 978-1-4577-0961-6. DOI: 10.1109/IMWS2.2011.6027210.

[14] M. R. Kamarudin, P. S. Hall, F. Colombel, and M. Himdi, "Cpw-fed disc-loaded monopole array antenna with integrated pin diode switches," in *2007 International workshop on Antenna Technology: Small and Smart Antennas Metamaterials and Applications*, Mar. 2007, pp. 396–399, ISBN: 1-4244-1088-6. DOI: 10.1109/IWAT.2007.370158.

[15] S. Zhang, G. Huff, J. Feng, and J. Bernhard, "A pattern reconfigurable microstrip parasitic array," *IEEE Transactions on Antennas and Propagation*, vol. 52, pp. 2773–2776, 10 Oct. 2004, ISSN: 0018-926X. DOI: 10.1109/TAP.2004.834372.

[16] W. H. Chen, J. W. Sun, X. Wang, *et al.*, "A novel planar switched parasitic array antenna with steered conical pattern," *IEEE Transactions on Antennas and Propagation*, vol. 55, pp. 1883–1887, 6 Jun. 2007, ISSN: 0018-926X. DOI: 10.1109/TAP.2007.898643.

[17] J. Costantine, Y. Tawk, S. E. Barbin, and C. G. Christodoulou, "Reconfigurable antennas: Design and applications," *Proceedings of the IEEE*, vol. 103, pp. 424–437, 3 Mar. 2015, ISSN: 0018-9219. DOI: 10.1109/JPROC.2015.2396000.

[18] C. Jung, M. Lee, G. Li, and F. DeFlaviis, "Reconfigurable scan-beam single-arm spiral antenna integrated with rf-mems switches," *IEEE Transactions on Antennas and Propagation*, vol. 54, pp. 455–463, 2 Feb. 2006, ISSN: 0018-926X. DOI: 10.1109/TAP.2005.863407.

[19] G. Rebeiz, K. Entesari, I. Reines, *et al.*, "Tuning in to rf mems," *IEEE Microwave Magazine*, vol. 10, pp. 55–72, 6 Oct. 2009, ISSN: 1527-3342. DOI: 10.1109/MMM.2009.933592.

[20] E. Brown, "Rf-mems switches for reconfigurable integrated circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 46, pp. 1868–1880, 11 1998, ISSN: 00189480. DOI: 10.1109/22.734501.

[21] C. Panagamuwa, A. Chauraya, and J. Vardaxoglou, "Frequency and beam reconfigurable antenna using photoconducting switches," *IEEE Transactions on Antennas and Propagation*, vol. 54, pp. 449–454, 2 Feb. 2006, ISSN: 0018-926X. DOI: 10.1109/TAP.2005.863393.

[22] Z. Su, M. Vaseem, W. Li, S. Yang, and A. Shamim, "Additively manufactured frequency/radiation pattern reconfigurable antenna based on monolithically printed vo2 switch," in *2019 13th European Conference on Antennas and Propagation (EuCAP)*, Mar. 2019.

[23] S. Luo, Y. Li, T. Jiang, and B. Li, "Fss and meta-material based low mutual coupling mimo antenna array," in *2019 IEEE International Symposium on Antennas and Propagation and USNC-URSI Radio Science Meeting*, Jul. 2019, pp. 725–726, ISBN: 978-1-7281-0692-2. DOI: 10.1109/APUSNCURSINRSM.2019.8888621.

[24] Y. Li and K. Yu, "High isolation two-element mimo antenna array based on embedded meta-material cells," in *2017 International Symposium on Antennas and Propagation (ISAP)*, Oct. 2017, pp. 1–2, ISBN: 978-1-5386-0465-6. DOI: 10.1109/ISANP.2017.8229005.

[25] W. Li, Y. Liu, and Y. Li, "Meta-material based mutual coupling reduction of circularly polarized array," in *2016 IEEE International Symposium on Antennas and Propagation (APSURSI)*, Jun. 2016, pp. 511–512, ISBN: 978-1-5090-2886-3. DOI: 10.1109/APS.2016.7695964.

[26] K. Buell, H. Mosallaei, and K. Sarabandi, "Metamaterial insulator enabled superdirective array," *IEEE Transactions on Antennas and Propagation*, vol. 55, pp. 1074–1085, 4 Apr. 2007, ISSN: 0018-926X. DOI: 10.1109/TAP.2007.893373.

[27] H. Mosallaei and K. Sarabandi, "Antenna miniaturization and bandwidth enhancement using a reactive impedance substrate," *IEEE Transactions on Antennas and Propagation*, vol. 52, pp. 2403–2414, 9 Sep. 2004, ISSN: 0018-926X. DOI: 10.1109/TAP.2004.834135.

[28] L. Bernard and V. Jaeck, "Investigations on bandwidth enhancement of low cost printed phased array with reactive impedance substrates," in *2013 IEEE International Symposium on Phased Array Systems and Technology*, Oct. 2013, pp. 279–284, ISBN: 978-1-4673-1127-4. DOI: 10.1109/ARRAY.2013.6731842.

[29] H. L. Zhu, S. W. Cheung, X. H. Liu, and T. I. Yuk, "Design of polarization reconfigurable antenna using metasurface," *IEEE Transactions on Antennas and Propagation*, vol. 62, pp. 2891–2898, 6 Jun. 2014, ISSN: 0018-926X. DOI: 10.1109/TAP.2014.2310209.

[30] M. Bouslama, M. Traii, A. Gharsallah, and T. A. Denidni, "Reconfigurable dual-band 3d frequency selective surface unit-cell," in *2015 IEEE International Symposium on Antennas and Propagation & USNC/URSI National Radio Science Meeting*, Jul. 2015, pp. 1264–1265, ISBN: 978-1-4799-7815-1. DOI: 10.1109/APS.2015.7305021.

[31] M. Bouslama, M. Traii, T. A. Denidni, and A. Gharsallah, "Beam-switching antenna with a new reconfigurable frequency selective surface," *IEEE Antennas and Wireless Propagation Letters*, vol. 15, pp. 1159–1162, 2016, ISSN: 1536-1225. DOI: 10.1109/LAWP.2015.2497357.

[32] A. Babakhani, D. B. Rutledge, and A. Hajimiri, "Transmitter architectures based on near-field direct antenna modulation," *IEEE Journal of Solid-State Circuits*, vol. 43, pp. 2674–2692, 12 Dec. 2008, ISSN: 0018-9200. DOI: 10.1109/JSSC.2008.2004864.

[33] M. Krairiksh, "Development of a handset adaptive antenna using phased-array of switched-beam elements," in *2008 International Workshop on Antenna Technology: Small Antennas and Novel Metamaterials*, Mar. 2008, pp. 87–90, ISBN: 978-1-4244-1522-9. DOI: 10.1109/IWAT.2008.4511297.

[34] C. G. Christodoulou, Y. Tawk, S. A. Lane, and S. R. Erwin, "Reconfigurable antennas for wireless and space applications," *Proceedings of the IEEE*, vol. 100, pp. 2250–2261, 7 Jul. 2012, ISSN: 0018-9219. DOI: 10.1109/JPROC.2012.2188249.

[35] D. Sievenpiper, J. Schaffner, H. Song, R. Loo, and G. Tangonan, "Two-dimensional beam steering using an electrically tunable impedance surface," *IEEE Transactions on Antennas and Propagation*, vol. 51, pp. 2713–2722, 10 Oct. 2003, ISSN: 0018-926X. DOI: 10.1109/TAP.2003.817558.

[36] S. V. Hum and J. Perruisseau-Carrier, "Reconfigurable reflectarrays and array lenses for dynamic antenna beam control: A review," *IEEE Transactions on*

Antennas and Propagation, vol. 62, pp. 183–198, 1 Jan. 2014, ISSN: 0018-926X. DOI: 10.1109/TAP.2013.2287296.

[37] C. Huang, A. Zappone, G. C. Alexandropoulos, M. Debbah, and C. Yuen, "Reconfigurable intelligent surfaces for energy efficiency in wireless communication," *IEEE Transactions on Wireless Communications*, vol. 18, pp. 4157–4170, 8 Aug. 2019, ISSN: 1536-1276. DOI: 10.1109/TWC.2019.2922609.

[38] S. Venkatesh, X. Lu, H. Saeidi, and K. Sengupta, "Reconfigurable multifunctional terahertz holographic metasurface using cmos chip tiling," in *2021 IEEE 19th International Symposium on Antenna Technology and Applied Electromagnetics (ANTEM)*, Aug. 2021, pp. 1–3, ISBN: 978-1-6654-0335-1. DOI: 10.1109/ANTEM51107.2021.9518737.

[39] C. L. Holloway, E. F. Kuester, J. A. Gordon, J. O'Hara, J. Booth, and D. R. Smith, "An overview of the theory and applications of metasurfaces: The two-dimensional equivalents of metamaterials," *IEEE Antennas and Propagation Magazine*, vol. 54, pp. 10–35, 2 Apr. 2012, ISSN: 1045-9243. DOI: 10.1109/MAP.2012.6230714.

Chapter 9

META-GAP OPTIMIZATION

Once upon a time called Now
Somebody say, "Is there funk after death?"
I say, "Is seven up?"

Parliament
P. Funk (Wants to Get Funked Up)

Unfortunately, the optimization of passive switching networks is an NP-Hard problem in general. The number of possible configurations grows exponentially with the number of switches and there is not a known effective heuristic. However, the electromagnetic specifics of using switchable meta-gaps between tiles could permit a such heuristic. Therefore, a set of metahueristic optimization algorithms are used to explore the properties of the unknown optimization problem and demonstrate the ability for meta-gaps to increase array performance.

Section 9.1 opens the chapter by showing how the different meta-gap switch settings can be encoded in binary and mapped to integers. This is followed by a discussion of the optimization complexity in Section 9.2. Section 9.3 presents the criteria over which the meta-gap states states will be optimized. The employed optimization framework and stochastic optimization algorithms are detailed in Section 9.4, providing insight into the kinds of properties each algorithm exploits. Next a set of simulated statistical experiments are performed in Section 9.5 to examine the statistical properties of the algorithms when applied to this optimization problem. Section 9.6 analyzes the experimental results and draws conclusions.

9.1 Representation of Meta-gap State

By turning on and off different switches on the meta-gap sheets, the behavior of the entire array changes. A particular set of switch settings is thus referred to as the *state* of the meta-gaps. The different states will effect the array performance differently. Thus the different states must be characterized in order to find the state with the best performance: the *optimal state*.

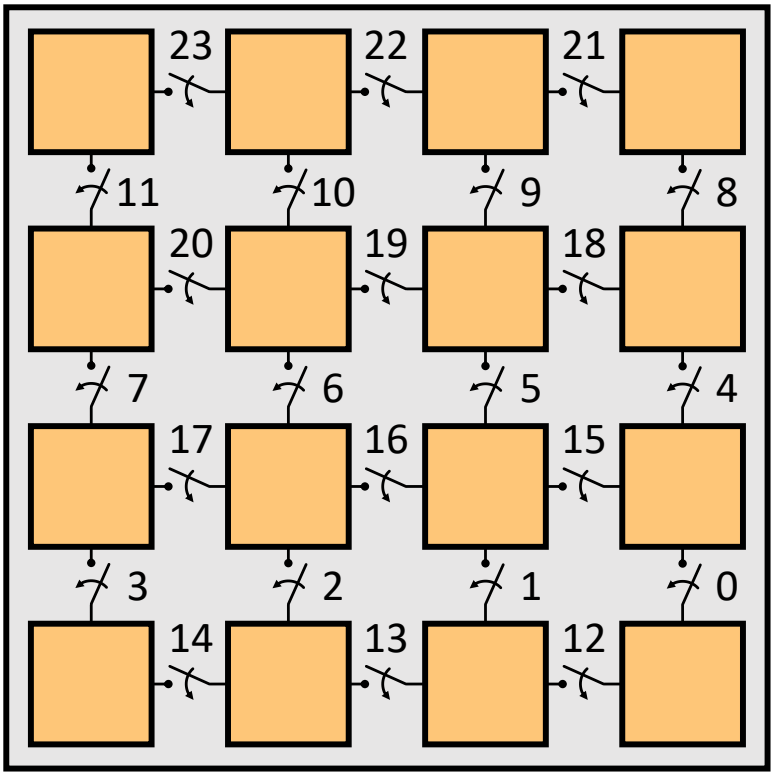


Figure 9.1: The labeling scheme used for switches on each meta-gap sheet. A sheet can thus be represented by a 24-bit number. For example in state 1, only the bottom right vertical switch is on, while only the top left horizontal switch is on in state 2^{23}.

To facilitate array communication, algorithm programming, and conceptual reasoning, it is helpful to represent each state with a unique integer. Each independent[1] switch in the array can either be "on" or "off," which is represented as a binary "1" or "0," respectively. Thus a collection of N independent switches can thus be represented with an N-bit binary number where a bit at a particular index, S_k, represents the status of particular switch.

For example, Figure 9.1 shows the labeling scheme used for each meta-gap sheet. The 24 switches on the sheet can be represented with 24 bits, with the first 12 bits representing the vertical switches and the second 12 representing the horizontal ones. Switches are indexed within each orientation by their row and column location, (i, j). This N digit binary number can also be interpreted as an unsigned integer, S,

[1] As described in Section 9.2, some switches are deliberately *not* independent in order to restrict the search space.

using Equation 9.1.

$$S = \sum_{k=0}^{N-1} S_k 2^k \tag{9.1}$$

Using this scheme, the state of the meta-gaps can be uniquely referred to by an integer between 0 and $2^N - 1$. It is immediately apparent that the number of possible states scales exponentially with the number of switches. Each meta-gap sheet contains 24 switches and thus has $2^{24} = 16{,}777{,}216$ different configurations. Because the experimental array measured in Chapter 10 contains 960 switches, it can be configured into over 10^{289} different states, over 200 orders of magnitude more than the estimated number of atoms in the observable universe[2].

This state definition also highlights the connectivity of the state space. Flipping the status of one switch changes one bit in the state. Thus the state space can be represented with a graph where the nodes are states and edges are single bit flips. This means that each state is directly connected to N neighboring states and that the graph can be traversed from a given state to any other state within N bit flips. With this in mind we can define the *k-neighborhood* of a state as the set of states within k bit flips of the given state. Note that this is equivalent to the set of states with a Hamming distance of k from the given state. The size of the k-neighborhood grows exponentially, with the 1-neighborhood containing N states and the N-neighborhood containing every state.

Finally, states 0 and $2^N - 1$ are special as they refer to the states where all switches are "off" and "on," respectively. As discussed in Section 8.4 these states would ideally correspond to the meta-gaps sheets acting as empty gaps and as a ground plane. The "all-off" and "all-on" states are used as helpful baselines throughout measurements.

9.2 Problem Scaling

It is clear from the Section 9.1 that meta-gaps present a high-dimensional space of configurations with a far smaller subspace of useful solutions. Thus, optimization algorithms must be used to search the space for the optimal configurations. However, due to the non-linearity of Maxwell's equations with respect to boundary conditions, the distribution of performance among different states is potentially highly non-linear and disjoint. Turning on a single switch could complete a loop or tune a conductor to just the right length and introduce resonance that greatly alters array performance.

[2]The number of atoms can be inferred to within a few orders of magnitude of 10^{80} using the cosmological parameters measured in [1].

Thus the optimization problem is more akin to a combinitorial problem than a non-linear continuous optimization. Indeed, optimizing this kind of switching network has been proven to be NP-Hard, in general and there is not a known effective convex heuristic [2].

Due to the computational complexity of the meta-gap optimzation problem, an exponential amount of time is required in order to identify the optimal configuration. Instead, we employ stochastic metaheuristic search algorithms in order to identify *preferred*, if not optimal, configurations. These partially randomized algorithms are capable of quickly searching a large state-space and locating increasingly optimal local maxima. Metaheuristics are an efficient method for characterizing the optimization problem and indentifying a lower bound on meta-gap performance.

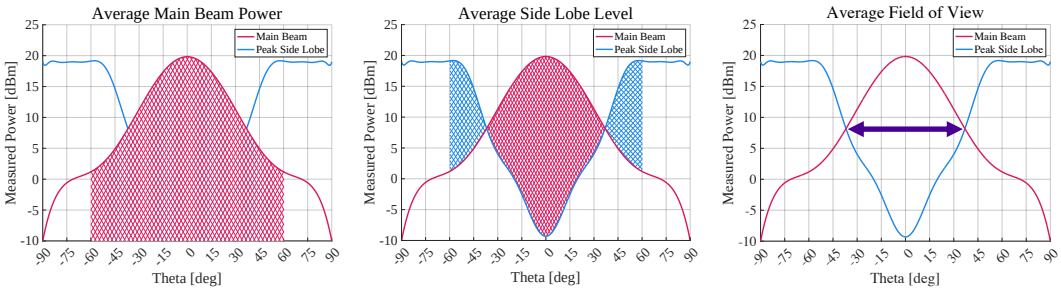
Another approach to managing the problem complexity is to limit the degrees of freedom by mapping multiple switches to the same bit. For example, potentially desirable symmetries in the array and/or sheet can be enforced by linking switches on opposite sides of the line of symmetry together. This technique reduces the search space by making it more course, increasing the effect each degree of freedom has on the array performance. Because the restricted search space is a subset of the unrestricted space, the optimal performance in the unrestricted space is at least as good as that in the restricted space.

Initially, the meta-gap sheets in the array are initially forced to have the same switch settings in order to restrict the search space to 24 degrees of freedom in order to give rough insight into meta-gap behavior. Less constrained symmetries are introduced in Section 10.5 and explored in Sections 10.6 and 10.7.

9.3 Optimization Criteria

In order for meta-gaps to mitigate the effects of increased spacing, a measure of the array's side and grating lobes must be used as an optimization criteria. As discussed in Section 2.6, full array characterization requires a large number of measurements. In light of this measurement density and the complexity of the optimization problem, it is critical to select an efficient optimization criteria and measurement approach in order to maximize the number of states that can be explored in a finite amount of time.

Ideally the gain, side lobe levels, and field of view would be measured over the full scan range of the array. However, such a scan requires a large number of beams, each of which must be measured at a large number of points. Instead, the number



(a) Main Beam Power (b) Side Lobe Levels (c) Field of View

Figure 9.2: Visualization of Optimization Criteria. The main beam power is integrated within the theoretical field of view, ex. ±60°. The difference between the main beam power and peak side lobe power is integrated within the theoretical field of view. The field of view optimization maximizes the angular difference between the first crossings of the main beam power and the peak side lobe power.

of measurements can be reduced by measuring the main beam power (MBP), the side lobe level (SLL), and the field of view (FoV) in the E- and H-plane cuts. These orthogonal cuts provide a decent measure of array performance without requiring a full 2D scan due to the rectangular arrangements of the arrays being optimized.

Maximizing the main beam power indirectly suppresses the side lobes by reducing the power lost in them, although it could also increase the total radiated power by improving the input impedance of the antennas. Minimizing the side lobe levels suppresses the side lobe power without altering the main lobe power[3]. Finally, optimizing the field of view explores whether meta-gaps are able to alter the effective spacing between elements to shift the locations of grating lobes.

In order to characterize the performance of the array with a single number, the *average* MBP and SLL over the field of view, and the *average* FoV over the different ϕ cuts, are used as the optimization criteria as shown in Figure 9.2. The MBP and SLL averages are calculated linearly in units of Watts instead of in decibels. Using an average ensures that the array performance is over its entire steering range and not just in one direction. MBP and SLL are only optimized over the FoV because, as discussed in Section 2.5, a main beam located outside of the field of view is a grating lobe of another beam within the field of view. Thus, increasing the main beam power outside the field of view simply increases side lobe levels inside it. This restricted scan range has the added benefit of greatly reducing the number of points that need to be measured; the field of view of an ideal λ- spaced array is only ±30°.

[3]Note that the easiest way to minimize the side lobe power *in general* is to minimize all radiated power, which is clearly not a useful solution.

Finally, as explained in Section 10.2, a more reliable measure of meta-gap sheet performance is its relative performance to a fixed baseline. Therefore, the pattern measurements of a given state are normalized by those of a baseline state where each switch is turned off. Thus the optimization criteria used can best be expressed as the average main beam power within the field of view versus the baseline state, the average side lobe level within the field of view versus the baseline state, and the average field of view. The MBP and SLL are presented in units of decibels compared to baseline while the FoV is listed in units of degrees.

9.4 Optimization Algorithms

In general, optimization algorithms that account for the specifics of a problem will outperform abstract metahueristic algorithms [3]. However, it is not obvious how a given set of switches will alter the array behavior, due to the strongly non-linear behavior of switched networks and the complex interactions of parasitic elements to different electromagnetic excitations. Therefore, four different metahueristic algorithms are employed to both explore meta-gap performance and to provide insight into the general structure of the optimization space and solutions.

The selected algorithms–Genetic, Particle Swarm, Variable Neighborhood Search, and Simulated Annealing–take advantage of different properties of the optimization problem [4]. In addition, a simple random search is used both as a baseline comparison and to provide insight into the statistical distribution of state performance. Thus the properties of the problem structure can be illuminated by comparing the relative performance of these methods.

The same framework, shown in Code Segment 9.1, is utilized to execute each algorithm, referred to as an *optimizer*. The optimizer repeatedly supplies a batch of states to be measured, evaluates the performance of the batch, and determines whether the algorithm should terminate. Once terminated, the optimizer returns the optimal state explored.

In order to reduce the measurement time and allow more states to be explored, the batched measurement method described in Section 2.6 is used to characterize the states. Because multiple states are explored simultaneously, the algorithm cannot evaluate any state in the batch until the entire batch is complete. Therefore, the utilized algorithms are designed to operate on batches of states instead of making decisions after each state is evaluated sequentially.

Each round of measurement is broken up into two phases. In the optimization phase,

```
while not optimizer.isFinished():
    batch_states = optimizer.getNextStates()
    for beam_angle in scan_range:
        measurementRange.moveToAngle(beam_angle)
        for state in batch_states:
            array.programState(state)
            optimumPhaseSettings[state][beam_angle] =
            ↪  array.optimizePhase()
    for measurement_angle in measurement_space:
        measurementRange.moveToAngle(measurement_angle)
        for state in batch_states:
            array.programState(state)
            for beam_angle in scan_range:
                phase_settings =
                ↪  optimumPhaseSettings[state][beam_angle]
                array.programPhases(phase_settings)
                measurement = measurementRange.measure()
    optimizer.evaluateMeasurements(batch_states)
optimizer.getOptimalState()
```

Code Segment 9.1: Optimization Framework Pseudo-Code.

the measurement range moves to each desired beam direction and optimizes a beam for each state in the batch, storing the beam phase settings. The beam steering algorithm employed is described in Section 2.7. In the measurement phase, the measurement range moves between measurement angles and iterates over the states and beam angles at each.

The framework interfaces with the optimizer through four functions called every batch: `isFinished()`, `getNextStates()`, `evaluateMeasurements()`, and `getOptimalState()`. The implementation of these functions are mostly shared between algorithms. The function `isFinished()` simply evaluates whether the desired number of batches has been evaluated. The Main beam power, side lobe level, or field of view of the measured patterns is characterized and stored by `evaluateMeasurements()`. Finally, `getOptimalState()` returns the best state stored by `evaluateMeasurements()`. In this work, only `getNextStates()` differs between the algorithms and thus it completely determines their behavior. The implementation of this function, the philosophy behind it, and the resulting optimization behavior is discussed for each algorithm in the following sections. To demonstrate the different behaviors, each section includes a diagram of the algorithm maximizing the non-linear function shown in Figure 9.3. The full source code of the implemented algorithms is is contained in Appendix D.

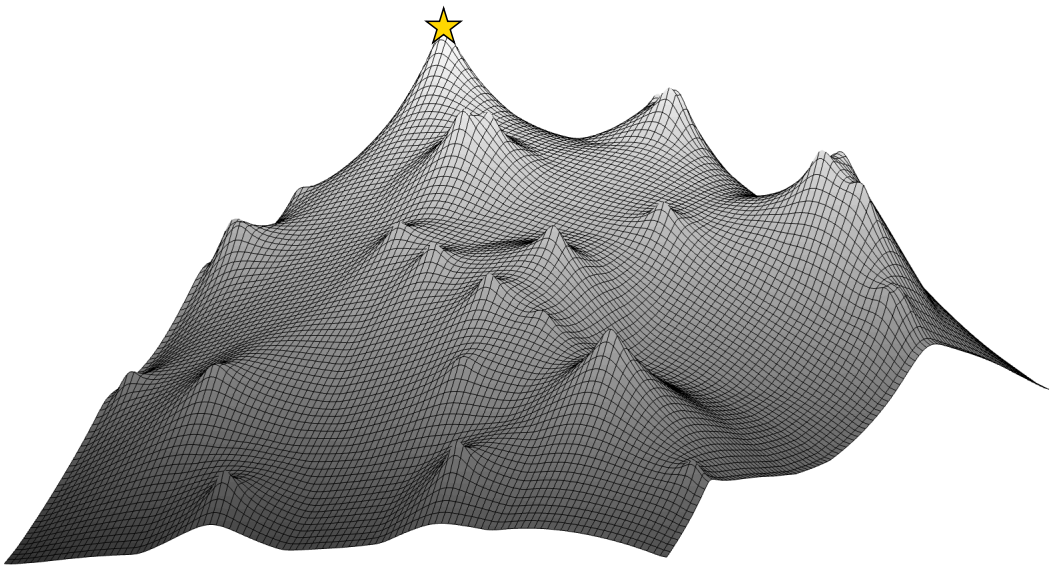


Figure 9.3: An example non-linear function to be maximized by the metahueristic algorithms in order to compare their behavior. The global maximum is marked by a gold star.

```
def getNextStates():                                                    1
    next_states = []                                                    2
    for i in range(states_per_batch):                                   3
        random_state = getRandomInteger(0,2^number_of_bits-1)           4
        next_states += [random_state]                                   5
    return next_states                                                  6
```

Code Segment 9.2: Random Search Pseudo-Code: getNextStates().

Random Search

Random search is the simplest possible algorithm that can be used to explore a large search volume. It selects the states in the next batch by randomly selecting an integer between 0 and $2^N - 1$. Because each state is equally likely to be selected, the algorithm uniformly explores the state space. Because random search is a simple approach that does not use measurement results to improve the search, it provides a useful benchmark for assessing how well other algorithms utilize this information.

The behaviour of random search can be understood probabilistically. Suppose that each state in the state space is ordered by its performance in a histogram. This histogram can be thought of as the probability distribution of performance when randomly selecting a state and can be represented by the cumulative distribution

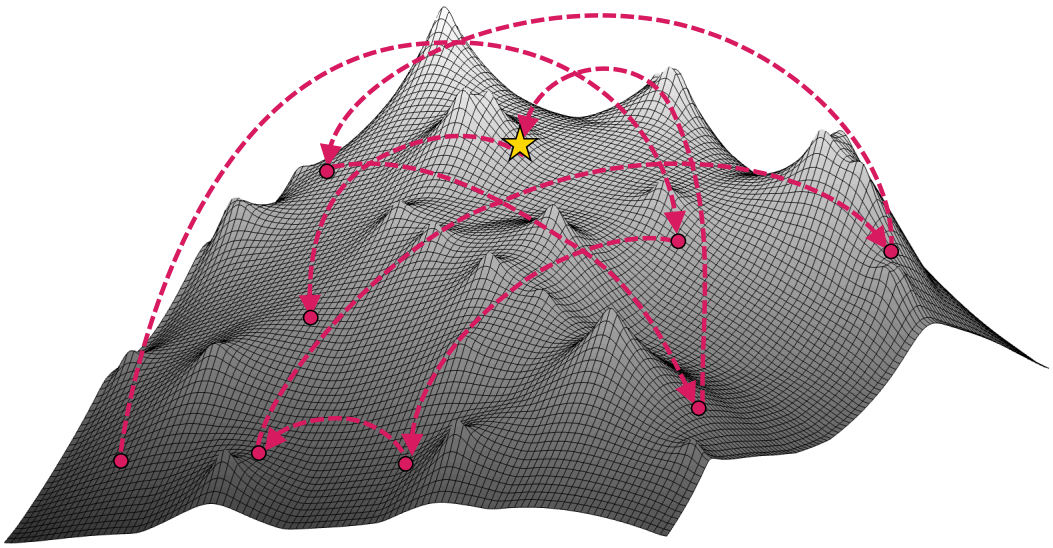


Figure 9.4: The random search moves through the optimization space at random, storing the best value it encounters. While clearly sub-optimal, it does cover the entire search space uniformly.

function $F_X(x)$. The random search repeatedly samples this distribution and returns the best performing state selected. The longer the algorithm runs, the more likely it is that a better state will be selected. As shown in Appendix C, we can write the expected value returned by the random search after examining k states in terms of $F_X(x)$.

$$E[X_k] = \int_0^\infty \left[1 - F_X(x)^k\right] dx - \int_{-\infty}^0 F_X(x)^k dx \tag{9.2}$$

The random search algorithm also provides useful information on the distribution of the state space because the collection of states it explores can be analyzed statistically. This collection is a random sample of a much larger population and so its distribution approaches that of the entire state space. The *sample mean* and *unbiased sample variance* shown in Equations 9.3 and 9.4 are unbiased estimators of the population mean and variance. Thus the average of the explored state characterization values describes the average effect of the meta-gap sheets, while the variance gives insight into how much different meta-gaps settings alter array performance.

$$\bar{X} = \frac{1}{n} \sum_{i=1}^{k} X_i \tag{9.3}$$

Table 9.1: Genetic Optimization Analogy

Terminology	⟷	Description
Organism	⟷	A abstract entity used to generate states to be measured
Population	⟷	A set of organisms that generates states in each batch
Generation	⟷	Batch number
Genome	⟷	State number
Genes	⟷	Values of a particular set of bits
Gene	⟷	Value of a particular bit
Fitness	⟷	Optimization characteristic for a measured state

$$s^2 = \frac{n}{n-1}\sigma_X^2 = \frac{n}{n-1}\left[\frac{1}{n}\sum_{i=1}^{k}\left(X_i - \bar{X}\right)^2\right] \tag{9.4}$$

Finally, random search is useful for verifying the accuracy of measurement results. Because each batch of states explored is a random sample, the sample average characterization value for each batch should not drastically vary. Any trend in the average indicates that *something other* than the meta-gap states is changing over the course of the measurement and distorting the measured performance of each state. This distortion will cause states to have better (or worse) performance than they should and thus invalidate decisions made by the optimizer. This factor could be anything from temperature variation over the long measurement period to power fluctuations to programming errors.

Genetic Optimization

Genetic optimization is inspired by evolution and how selective pressure, genetics, and reproduction optimizes biological organisms for niche environments [5]. Because genetic optimization is an analogy at its core, it is helpful to describe its behavior using biological terms. A state is viewed as an *organism* whose behavior is determined by its *genome*, the particular switch settings that are represented as a binary number. Thus each bit in a state is analogous to a single gene. Each batch of states is viewed as a new *generation* of a *population*. When states are measured, the *fitness* of the organism is determined by the optimization characteristic. The genetic algorithm uses selective pressure based on this fitness to direct the optimization. A summary of the analogy terminology is shown in Table 9.1.

As shown in Code Segment 9.3, the general behavior of genetic optimization is divided into three steps, selection, recombination, and mutation. In evolution, better adapted organisms are more likely to reproduce. Thus in the selection step, pairs of organisms are randomly selected to be parents of the next generation based

```
def getNextStates():
    if batch_number == 0:
        next_states = getInitialPopulation()
    else:
        parents = selection()
        children = recombination(parents)
        next_states = mutation(children)
    return next_states
```

Code Segment 9.3: Genetic Optimization Pseudo-Code: getNextStates().

on their fitness. As in sexual reproduction, a child organism is created by selecting genes from the two parents in the recombination phase. However, reproduction is imperfect and mutations are introduced into the child's genome. Thus a selection of the child's genes are randomly altered in the mutation phase.

These three steps each serve an important purpose in the optimization process. The selection step removes poorer performing organism from the population. If selection is repeated over and over, the population would consist of the fittest organism, optimizing the search criteria. Recombination introduces new organisms based on the genes of existing organism, thus allowing the optimization to explore more of the search space in search of even better performing states. However, there is a risk of stagnation because recombination only passes genes that already exist in one generation to the next; no new states can be explored if the population only consists of one organism. Thus mutation introduces novel genes to the population to prevent stagnation.

Because genes are propagated from one generation to the next, the initial population has an impact on the final performance. Because genes that are not present in the initial population can only be introduced through mutation, it is important that desired genes are in the initial population so they can propagate. While it is possible to initialize the population with known genes, without prior-knowledge of what genes perform well it is best to generate the initial population by randomly selecting states as in random search.

Code Segment 9.4 shows pseudo-code for the selection process used in this work. The list of states in a generation is sorted by their performance. A weight is assigned to each state based on its order in this sorted list. These weights are then used to randomly select states, with replacement, to be parents for the next generation. The best performing state is more likely to be picked than the worst performing state, so

```
def selection():
    orderedStates = sortByPerformance(previous_states)
    stateWeights = orderInList(orderedStates)
    parents = []
    for i in range(states_per_batch):
        parents +=
        ↪  randomlyPickTwoStates(orderedStates,stateWeights)
    return parents
```

Code Segment 9.4: Genetic Optimization Pseudo-Code: selection().

```
def recombination(parents):
    children = []
    for parent_pair in parents:
        child = '0b0'
        for bit_index in range(number_of_bits):
            bit = randomlySelectBitFromParents(
            ↪  parent_pair,bit_idx)
            child[bit_index] = bit
        children += [child]
```

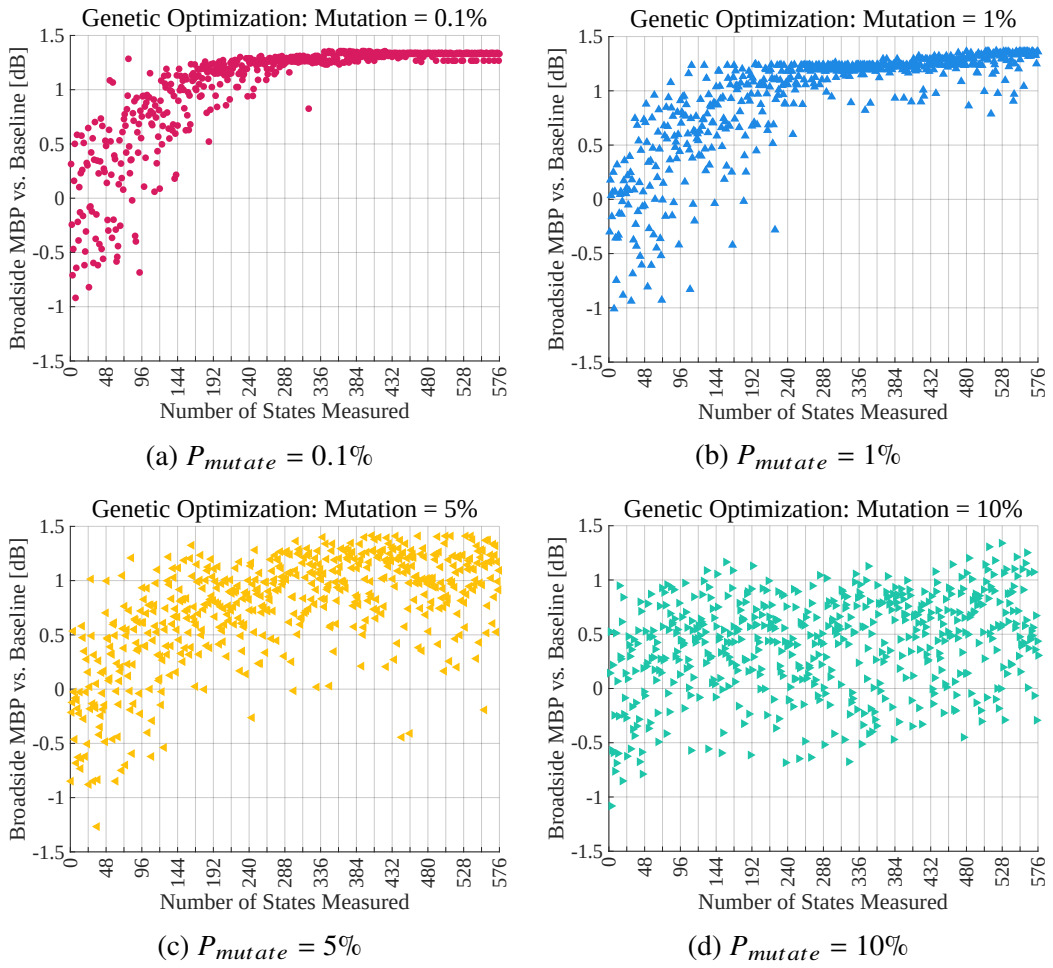
Code Segment 9.5: Genetic Optimization Pseudo-Code: recombination().

```
def mutation(children):
    next_states = []
    for child in children:
        for bit_index in range(number_of_bits):
            coin_flip = getRandomFloat(0,1.0)
            if coin_flip < p_mutate:
                child[bit_index] = flipBit(child,bit_index)
        next_states += [child]
    return next_states
```

Code Segment 9.6: Genetic Optimization Pseudo-Code: mutation().

its genes will be more prevalent in the next generation. While it is possible to assign weights based on the *performance* of each state, it is expected that the difference in performance between meta-gap states to be small and thus a fitness based weight would not apply as much selected pressure.

Code Segment 9.5 shows pseudo-code for the recombination process utilized. For each child, the value of each bit in the state is randomly selected from one of the two parents. A common alternative approach is to randomly pick an index and select bits before it from one parent and after it from the other. This method is



(a) $P_{mutate} = 0.1\%$ (b) $P_{mutate} = 1\%$

(c) $P_{mutate} = 5\%$ (d) $P_{mutate} = 10\%$

Figure 9.5: Effect of mutation probability on the genetic optimization during *in-situ* optimization of broadside main beam power enforcing identical meta-gap settings on each sheet.

useful when neighboring bits in the state are expected to have a correlated effect on the performance. However, adjacent bits do not necessarily correspond to adjacent switches in our implementation, and thus they are not expected to be correlated.

Code Segment 9.6 shows pseudo-code for the implemented mutation scheme. The value of each bit in each child's state is flipped with a fixed probability, `p_mutate`. The mutation probability has a large impact on the convergence and performance of the optimizer. Figure 9.5 shows the performance of the genetic algorithm optimizing the broadside power of the array presented in Section 10.1 with different mutation probabilities. A low mutation probability emphasizes the selection and recombination process but does not introduce new genes into the population. Thus, the algorithm will quickly converge but can get stuck in a local optimum as genes that aren't in the initial population are not introduced. On the other hand, a high

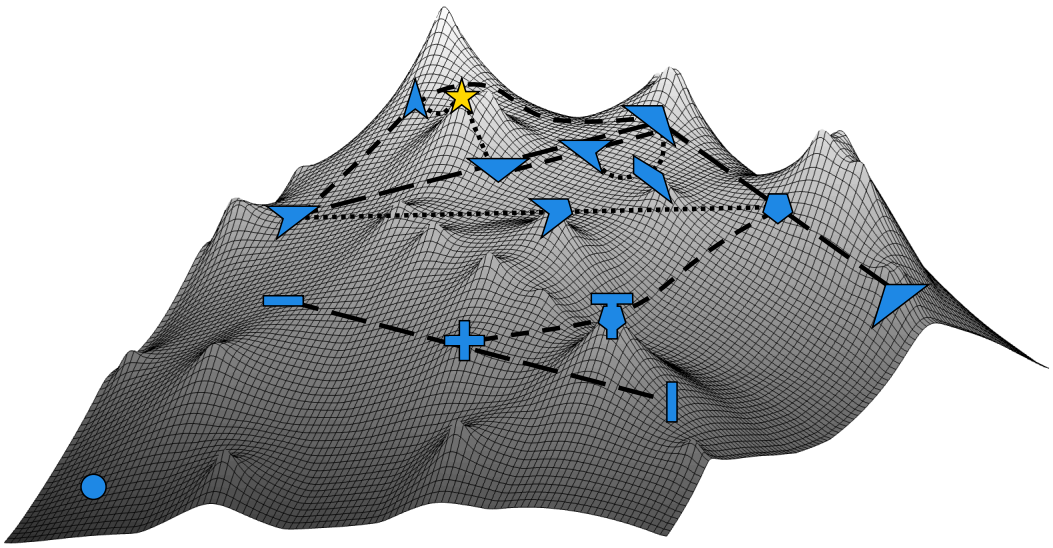


Figure 9.6: Genetic optimization selects high performing states, recombines their genes, and produces children with the occasional mutation. The shape of each state explored represents its genes. Children are placed in the center of dashed lines that connect their parent states. The length of the dashes delineates the generation of the child, with longer dashes indicating earlier generations. Mutations are indicated by curved lines and by new features being introduced into the shape. As can be seen, recombination allows wide regions of space to be quickly covered and selection concentrates children in a higher performing subspace. However, areas outside of the existing population cannot be explored without mutation.

mutation probability can explore more of the search space but converges slower as the selective pressure is reduced. A 50% mutation probability results in the genetic algorithm becoming random search as each bit has a 50% of being 1 or 0 regardless of its value pre-mutation. The presented experiments use a 1% mutation probability because it has good convergence behavior given the numbers of states explored.

Genetic optimization performs well if the solution space is comprised of *features*, that is, a set of distinct patterns that have an abstract meaning. Part of the reason that genetic optimization is successful in biology is that, while there are a massive number of possible DNA combinations, the DNA is mapped first to amino acids and then to proteins. A random string of DNA does not necessarily correspond to a sequence of amino acids that form a stable protein. Thus, it is favorable to keep fragments of DNA that encode specific proteins intact and swap them instead of the base pairs. The code for proteins are an example of features in the biological solution space that form building blocks for different organisms.

```
def getNextStates():
    if batch_number == 0:
        particles = initializeParticles()
    else:
        swarm_optimum = getSwarmOptimum()
        for particle in particles:
            position = particle.getPosition()
            position_value = particle.getValue()
            if position_value > particle.individualOptimum():
                particle.updateIndividualOptimum(position,
                ↪  position_value)
            if position_value > swarm_optimum:
                updateSwarmOptimum(position, position_value)
    next_states = []
    for particle in particles:
        particle.moveParticle()
        next_states += [particle.getPosition()]
    return next_states
```

Code Segment 9.7: Particle Swarm Pseudo-Code: getNextStates().

Particle Swarm

Particle swarm is another biology inspired optimization algorithm, in this case, the social behavior of animals like a flock of birds or a school of fish [6]. The algorithm is based on a simplified model of animal swarming. In a swarm, the behaviour of individual organisms is determined both by the individual's senses and by information obtained from the group as a whole. For example, if a fish in a school senses a predator and adjusts course, neighboring fish will also adjust course even if they have not yet detected the threat. In a sense, the group exhibits an awareness greater than its composite parts. It is this "swarm intelligence" that particle swarm optimization seeks to exploit.

As with genetic optimization, it is helpful to use an analogy and its terminology to understand the behavior of particle swarm. Originally designed for continuous optimization problems, the classic analogy is to treat each variable as a separate axis in an $N-$ dimensional state space. Each explored state is a particular value for each of the $N-$ variables and thus corresponds with a *position* in the state space. Particle swarm treats the states in each batch as the positions of a *swarm* of *particles*. With each new batch, the positions of these particles change according to each particle's *velocity* and new states are explored. Thus batches correspond to *time* passing as the particle moves through the state space.

To emulate swarm behaviour, each particle determines its trajectory using a combi-

Table 9.2: Particle Swarm Analogy

Terminology	⟷	Description
Particle	⟷	A abstract entity that selects states to be measured
Swarm	⟷	A set of particles that determine the next batch of states
Position	⟷	State number
Velocity	⟷	Change in state number between batches
Time	⟷	Batch number
Value	⟷	Optimization characteristic for an explored state
Individual Optimum	⟷	Most valuable state explored by a specific particle
Swarm Optimum	⟷	Most valuable state explored
Sensing	⟷	Measuring the optimization characteristic

nation of information "sensed" by itself and by the group. Each position is associated with a *value* determined by the optimization criteria. As each particle traverses the space it remembers the most valuable position it has encountered thus far, its *individual optimum*, and adjusts its velocity to move towards it. In addition, the particles communicate with each other and identify the most valuable position encountered by any of the particles, the *swarm optimum*. Each particle in the swarm also adjusts its velocity to move towards the swarm optimum. Code Segment 9.8 illustrates how particles are used to share information and generate states to explore.

$$\vec{v}_{new} = \omega\vec{v}_{old} + \Lambda\left(\vec{x}_{opt,i} - \vec{x}\right) + \Gamma\left(\vec{x}_{opt,s} - \vec{x}\right) \tag{9.5}$$

Equation 9.5 is the classic formula for calculating each particle's velocity after each time step. The new velocity is a weighted sum of the old velocity, a vector in the direction of the individual optimum, and a vector in the direction of the swarm optimum. These weights can be thought of as the particles inertia and attraction to both the individual and the swarm optimums. Suppose that $\Gamma = 0$, $\omega = 0$, and that the individual optimum is at a local maximum. In this case, the trajectory of the particle will converge upon the maximum like a greedy search. However if the inertia is non-zero, convergence is delayed and the particle can potentially move outside of this local maximum and discover a better optimum. If $\Gamma \neq 0$ than the particle will be pulled outside of this local maximum in the direction of the swarm optimum, potentially discovering a new swarm optimum along the way. Thus the swarm behavior performs local optimization but can also search a large portion of the state space. ω, Γ, and Λ are chosen so that particles do not prematurely converge on local optimums nor the swarm optimum; the parameters are $\omega = 0.5$, $\Lambda = 0.375$,

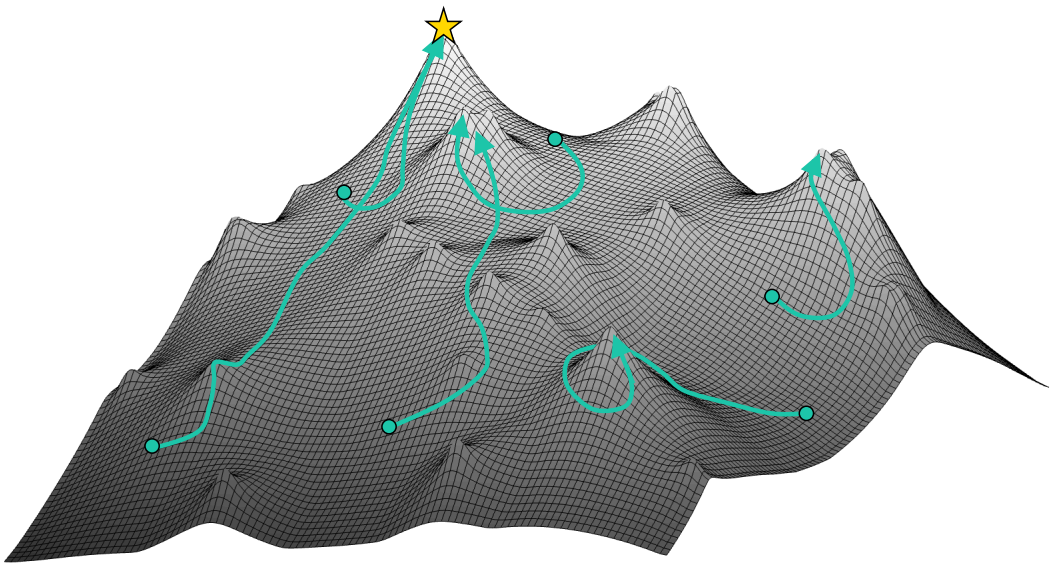


Figure 9.7: Particle swarm initializes particles evenly throughout the space with random initial velocities. The particles then propagate, being pulled towards the best state they have identified so far and the best state identified globally. The curves indicate the trajectory of each particle. As can be seen, particles converge on local maxima, but can also escape them by their own inertia and the pull of the global optimum state.

and $\Gamma = 0.125$ in the experiments presented in this work. Psuedo-code for particle movement is shown in Code Segment 9.8.

Without prior knowledge of the solution space, particles are initialized with a random position and velocity. Thus particles initial explore a large part of the state space. Over time the search becomes more localized as particles converge towards the swarm optimum. In the end, particles converge on local maxima. The relative independence of the particles distributes resources across different locations in the state space, resulting in more robust global exploration at the cost of less efficient local search. In a sense particle swarm is a hybrid of different approaches; it takes a global to local approach like simulated annealing, but it allows more dynamic exploration than search based approaches.

Particle swarm must be adapted in order to be used with discrete optimization problems because particles cannot have continuous valued positions nor velocities. In meta-gap optimization for example, switches can only be on or off and thus positions can only be 0 or 1. Fortunately, the concept behind particle swarm still applies if the state space can be mapped onto a metric space. In this scenario, addition and multiplication operators in Equation 9.5 are defined by the metric space. There

```
def moveParticle():                                                    1
    individual_distance = individual_optimum - position                2
    individual_pull = individual_weight * local_distance               3
                                                                       4
    swarm_distance = global_best - position                            5
    swarm_pull = swarm_weight * global_distance                        6
                                                                       7
    velocity = inertia*velocity + individual_pull + swarm_pull         8
    position = position + velocity                                     9
```

Code Segment 9.8: Particle Swarm Pseudo-Code: moveParticle().

are multiple methods of achieving this mapping for a discrete problem, such as defining operations based on crisp sets [7].

In this implementation, the discrete state space is mapped to a continuous *probability* space [8]. The position of a particle in dimension i is the probability that switch i is on, $P(b_i = 1)$. Velocities are treated as weighted estimates of whether individual switches are on or off in the optimal state, $P(b_i = 1|v_i)$. Thus, positions are updated by the velocity using Bayes Law as confidence in a switch being on or off changes. These probabilities are used to assign the particle to a discrete state whenever they are to be measured[4].

Variable Neighborhood Search

Another approach to discrete optimization problems is to treat them as a search where the graph representing the state-space is traversed from one state to the next. Greedy search, a discrete version of gradient ascent, is the most straightforward search-based approach. As described in Section 9.1, each N-bit state is connected to N other states in its 1-neighborhood by a single bit-flip. Greedy search measures each of these N neighboring states, identifies the best neighboring state, and moves there if it is better than the current state. The main drawback of greedy search is that it gets stuck in local maxima and thus is unlikely to find the optimal solution.

Variable neighborhood search (VNS) modifies greedy search in order to get out of local maxima [9]. As shown in Source Code 9.9, VNS performs a greedy search until it detects a local maxima when no state in the 1-neighborhood improves on the current state. At this point VNS increases the size of the search space to a larger

[4]In this sense, the approach used is analogous to quantum mechanics, where a particle's position is not determined until it is measured. We restrain ourselves from referring to the approach as the more buzzword friendly *quantum particle swarm* because that is the extent of their similarities.

```
def getNextStates():
    if batch_number == 0:
        next_states = getInitialStates()
        return next_states
    best_neighbor_state = getBestState(previous_states)
    current_value = getStateValue(current_state)
    best_neighbor_value = getStateValue(best_neighbor_state)
    # Improvement, perform greedy search
    if best_neighbor_value > current_value:
        k = 1
        current_state = best_neighbor_state
    # Local Maximum, expand search space
    else:
        k = 3
    next_states = []
    for i in range(states_per_batch):
        state = getUnexploredState(current_state,k)
        while state == None:
            k = k + 1
            state = getUnexploredState(current_state,k)
        next_states += [state]
    return next_states
```

Code Segment 9.9: VNS Pseudo-Code: getNextStates().

neighborhood, in this implementation the 3-neighborhood, and explores states with 3 bit-flips. This neighborhood is explored until either an improvement is found or it has been exhasted, at which point VNS increases the size of the search space again. Because the size of the k-neighborhood exponentially increases with k, states to be measured are randomly selected without replacement from the k-neighborhood. Once an improvement has been identified, the neighborhood is reduced to the 1-neighborhood again and greedy search continues.

Because multiple neighbors must be explored before making a decision, VNS is well adapted to operate on small batches. By making the batch size equal to N, every neighbor in the 1-neighborhood is explored each batch. However exploring the full 1-neighborhood in a single batch becomes impractical for very large N, in which case states are randomly selected from the neighborhood without replacement. Because the size of the k-neighborhood grows exponentially with k, random sampling is also used when a local maxima is reached. If the k-neighborhood is ever completely explored, then additional states are selected from the $k+1$-neighborhood to fill the batch.

VNS takes a local to global approach, rapidly reaching local maxima and only then

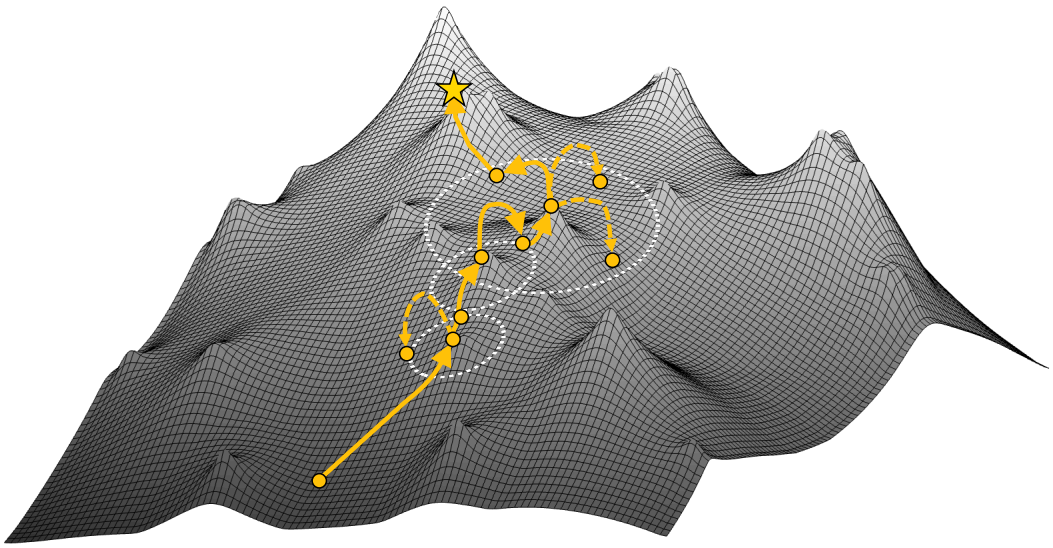


Figure 9.8: Variable Neighborhood Search behaves like a greedy search until it encouters a local maxima. At this point, it randomly searches increasingly larger local neighborhoods until it identifies a better point. Having escaped the maxima it proceeds with another greedy search. The trajectory of the search is indicated by solid lines, while dashed lines indicate random states that are explored but do not escape the local maxima. The neighborhood explored by the algorithm at each local maxima is indicated by the dashed white lines. VNS is very efficient at searching a local space but does not cover a wide portion of the total space.

broadening the search. For some problems, local maxima are located relatively close to each other in state space as they share optimal solutions for a subset of the total problem. VNS performs well on these sorts of problems as it is able to quickly ascend to a local maxima and then find better maxima nearby. VNS also works efficiently in convex problems as it can reach the optimal state in N iterations from any starting position. However, it does not perform well on problems with sparse optima as it only explores more global searches after exhausting the local space.

Simulated Annealing

Simulated Annealing is a search algorithm that takes the opposite approach as VNS; it initially explores the global state space and then converges on a local search. The classic Simulated Annealing algorithm modifies greedy search to select new states probabilistically based on their relative performance [10]. Therefore, simulated annealing can escape local maxima and continue the search. The key to simulated annealing, and its connection to its namesake, is how the probabilities are calculated.

```
def getNextStates():
    if batch_number != 0:
        best_state = getBestState(previous_states)
        current_value = getStateValue(current_state)
        best_value = getStateValue(best_state)
        # Improvement, perform greedy search
        if best_state > current_value:
            current_state = best_state
    fraction_remaining = 1 -  batch_number/number_of_batches
    current_temperature =
    ↪  temperatureSchedule(fraction_remaining)
    neighborHoodSize = numberOfBitFlips(current_temperature)
    possible_states =
    ↪  getNeighborhood(current_state,neighborHoodSize)
    next_states = []
    for i in range(states_per_batch):
        next_states += [pickRandomState(possible_states)]
    return next_states
```

Code Segment 9.10: Simulated Annealing Pseudo-Code: getNextStates().

Annealing is the process by which materials like glass are slowly cooled so their internal molecular structure becomes more durable. When a material is hot, its internal structure can change as atoms break and form new bonds. As it is cooled, there is less energy to break bonds and the structure crystallizes. By slowly cooling the material, atoms are able to reorganize themselves to lower energy states, removing discontinuities from the crystal structure. Thus the annealed material has a more uniform internal structure that makes it more resistant to breaks.

Simulated annealing emulates this process by introducing a *temperature* parameter that alters the probability of selecting worse performing states as the algorithm runs. A commonly used probability function, inspired by the Boltzmann distribution, is given in Equation 9.6 with E_o and E representing the characterization values, or *energies*, of the current and potential states and T indicating the temperature. At high temperatures, the difference in performance between states minimally effects their relative probabilities and thus the search is more random. At lower temperatures, the difference in performance is more important and the search is guided. At $T \approx 0$, only states that improve performance will be selected and the algorithm performs a greedy search. As with metallurgical annealing, the function of how temperature changes over time has a large impact on the final result and is referred to as the *temperature schedule*. In this work, temperature linearly decreases from a high start

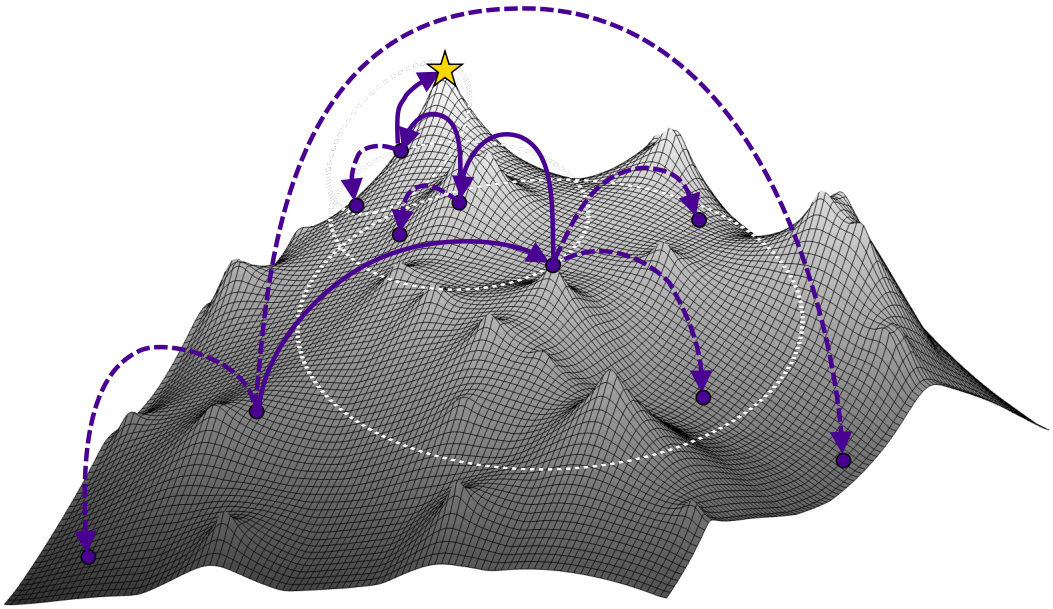


Figure 9.9: Simulated annealing randomly searches the space, moving to better states that are identified. Unlike random search however, the size of the neighborhood explored decreases over time until the algorithm converges to a greedy search. The solid lines indicate the trajectory of the search while dashed lines indicate states that are explored without improving performance. The size of the region explored over time is indicated by the dashed white circles.

value to zero over the course of the execution.

$$P(selected) = \begin{cases} \exp\left(\frac{E - E_o}{T}\right) & E \leq E_o \\ 1 & E > E_o \end{cases} \tag{9.6}$$

The critical feature of simulated annealing is not the specifics of Equation 9.6 or how states are selected, but the gradual reduction in the size of the search space. The random nature of the search at high temperatures allows it to overcome deep local optima and quickly span the state space. As the temperature cools, the search becomes more localized as shallow, but not deep, optima can be escaped. The final greedy search at the end guarantees that the algorithm will converge on a local optimum if given enough time.

Unfortunately, the classic simulated annealing approach is not well adapted to batched *in-situ* optimization. In order for the classic algorithm to globally search the state space, it must repeatedly make random decisions that alter only one bit. This is not possible in the batched measurement environment as decisions to move the state can only be performed at the end of a batch. In addition, while probabilistically

Table 9.3: Simulated Annealing Analogy

Terminology	⟷	Description
Energy	⟷	Optimization characteristic for an explored state
Temperature	⟷	Parameter determining how random the search is
Time	⟷	Batch number
Temperature Schedule	⟷	Change in temperature over time

selecting better performing states steers the search towards the optimum over a large number of decisions, it does not efficiently do so with a small number of decisions. Due to the limited number of batches, the classic simulated annealing algorithm would not be directed enough to converge.

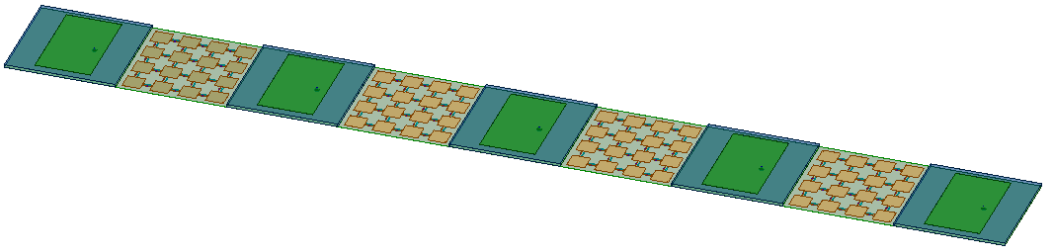
The implementation of simulated annealing shown in Section 9.10 achieves the same global to local behavior while taking a different approach that is more adapted to meta-gap optimization. Instead of randomly moving between states that are deterministically selected, the implemented search deterministically moves between states that are randomly selected. The temperature determines the size of the neighborhood from which each batch of states is randomly selected. If a better state is found, the search moves to the new state and continues. Like the classic approach, the search moves from global to local and finishes as a greedy search. The main difference in behavior is that this approach does not consider the relative magnitudes of the different states, however, this deterministic approach efficiently guides the search.

As a global to local search algorithm, simulated annealing performs well when the solution space is globally convex despite containing local optima. The algorithm initially is able to escape deep local optima and identify regions of the solution space that generally more optimal. As the temperature cools, the search becomes more confined until it finds a local maxima with an efficient greedy search. Thus the initial states explored are critical as they constrain the states explored later.

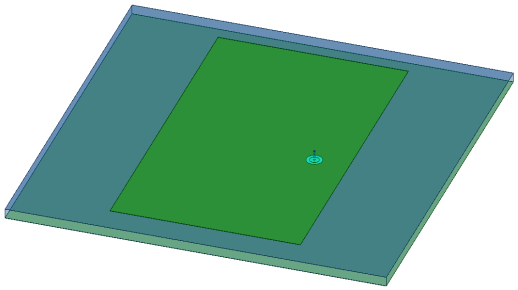
9.5 Simulated Statistical Experiments

Due to the stochastic nature of the employed meta-heuristics, they are best characterized by their performance over a large number of trials. For example it is *highly* unlikely, but not impossible, for random search to find the optimal state the first time it is run. Such a result would be misleading and suggest that random search is a the most efficient search algorithm.

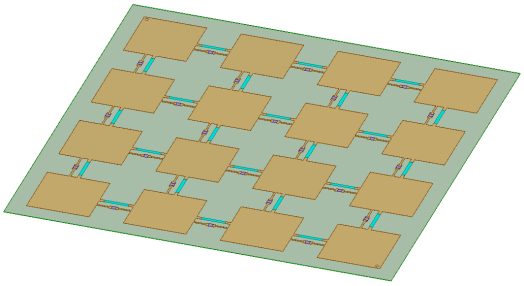
Unfortunately, as discussed in Section 2.6, the time required to measure a physical



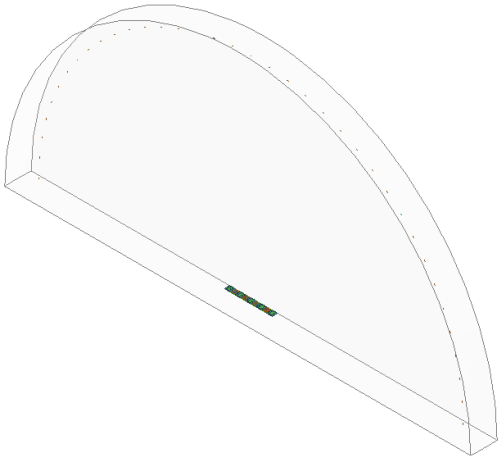
(a) 5 element λ-spaced array with meta-gaps



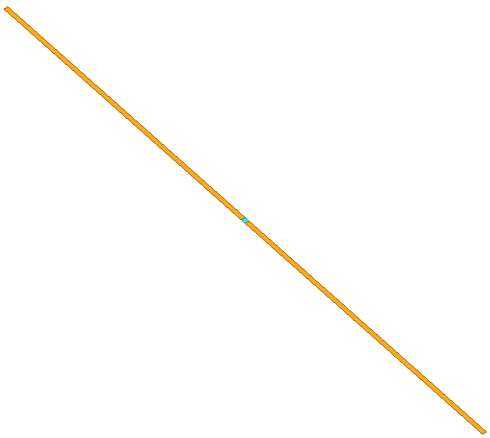
(b) Antenna model with excitation port



(c) Meta-gap model with control ports



(d) Simulation volume with array and far-field sense antennas



(e) Sense antenna

Figure 9.10: Finite Element Simulation Model

array prevents a large number of trials. Instead, the Lavaei-Babakhani method can be used to perform a large number of simulated trials. While deviations in a simulated model might give inaccurate characterization results, they do not fundamentally change the structure of the optimization problem. Thus, meta-heuristic algorithms should perform similarly in both simulation and *in-situ*.

The different optimization algorithms are run 1,000 times with random initial conditions. Optimization of main beam power, side lobe levels, and field of view are

treated separately with their own set of runs. Each run consists of 30 batches of 24 states each to match the experiments performed in Chapter 10. The meta-gap sheets are restricted the same switch settings in order to reduce the size of the search space to 2^{24} states. The results of the runs are used to calculate the average and standard deviation of the optimal state identified over time. These statistics provide insight into the convergence time of the algorithms, their relative expected performance, and the variance in runs.

Simulation Model

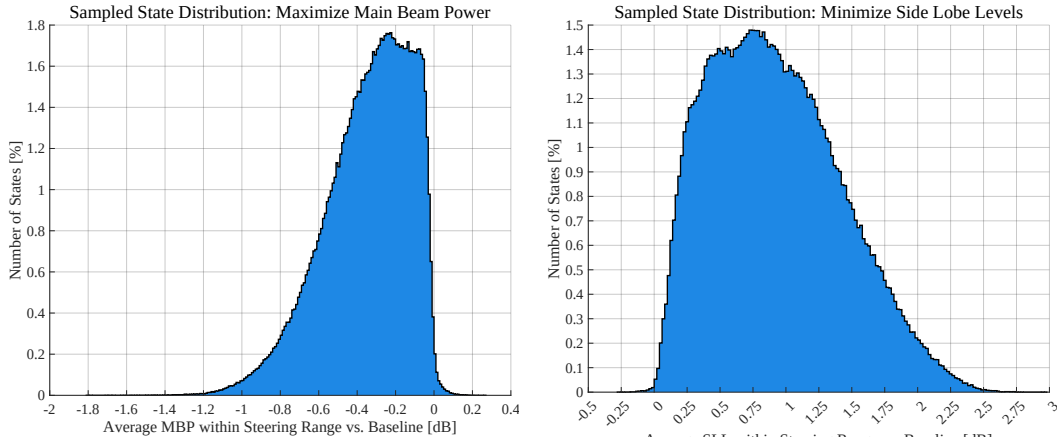
Figure 9.10 shows the simulation model used for statistical analysis. It is a five element λ-spaced linear array of patch antennas separated by four meta-gap sheets. Both the meta-gap sheets and patch antennas has the same metal patterns and dimensions as their physical equivalent. The meta-gap sheets also model the 30 nH choke inductors in parallel with switches. Figure 9.10d shows the array and the 37 short dipole sensor antennas located in the far-field every 5° in the steering-plane. This model contains 138 ports -5 excitation ports, 37 sensor ports, and 96 control ports- that are represented by teal rectangles. The total simulation volume including the array and sensor ports is $605\pi\lambda^3$. The array was simulated using HFSS's finite element solver over 15 hours using 23 cores and 218 GB of RAM.

It is important to note that the simulated problem is only a one dimensional array instead of the full two dimensional demonstration array from 10.1. A simulation of the full 2D array was not possible due to the significantly larger simulation volume required (closer to $7,100\pi\lambda^3$) and the larger number of ports (1058). Therefore, the simulated optimization problem is possibly quite different than the *in-situ* optimization problem and so caution is required when considering statistical results with respect to the physical array.

The 138 port s-parameter model derived from the FEM simulation enables the behavior of different meta-gap settings to be quickly calculated. This calculation method is encapsulated into a 'simulated range' that implements the same functions as the physical range presented in Section 10.2. Thus the same optimization framework given in Code Segment 9.1 is used to explore the simulated statistical performance of the algorithms presented in Section 9.4.

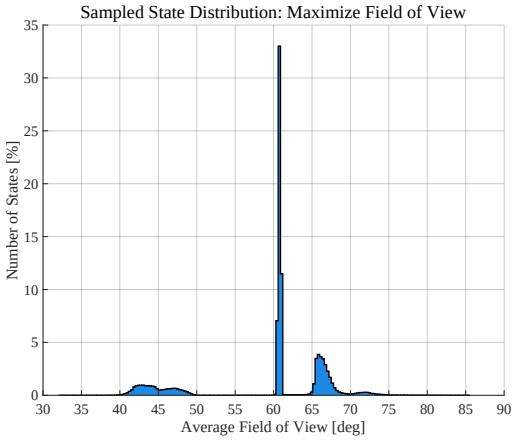
Distribution of States

As discussed in Section 9.4, the states measured by the random search algorithm provide a statistical sample of the underlying distribution of states. For each of the



(a) Main Beam Power

(b) Side Lobe Levels



(c) Field of View

Figure 9.11: Distribution of randomly sampled states in the simulation experiment.

optimization criteria, the random algorithm explored approximately 705 thousand unique states out of 16.8 million over the 1,000 runs.

Figure 9.11 contains histograms of state performance for the different optimization criteria and thus illuminates the underlying distribution of states. The distributions for main beam power and side lobe levels are closely related, with a long tail of states that perform worse than baseline and a sharp drop-off in states that perform better. For main beam power optimization, the sample mean and standard deviation are -0.35 dB and 0.23 dB, respectively, while they are 0.92 dB and 0.50 dB for side lobe level optimization. The distribution of states indicate that most meta-gap settings are detrimental to performance, but there a small number of states that offer performance improvements. This aligns with the intuition that random parasitic metal patterns in close proximity to an antenna will generally degrade its pattern and matching. The distributions also match the algorithmic results that suggest that

the two problems are closely related, with side lobe levels offering more variability.

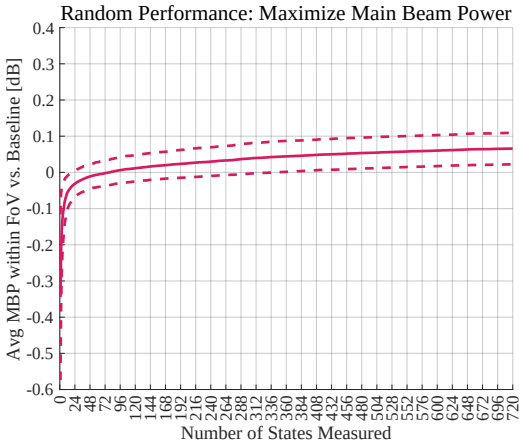
The histogram for the field of view optimization shown in Figure 9.11c is radically different. The distribution is tri-modal, with a sharp thin collection of states close to 60°, the ideal FoV for a λ-spaced array, and two wider loci around 67° and 45°. The high concentration of states indicates that most switch settings do not really effect the field of view. However, there exists sub-spaces that either enhance or suppress the grating lobes to some degree. The sample mean and standard deviation for field of view optimization are 59.7° and 7.8°. This standard deviation is misleading however as the variation is mostly attributed to the distinct modes with the center and right modes exhibiting far lower variability.

Individual Statistical Results

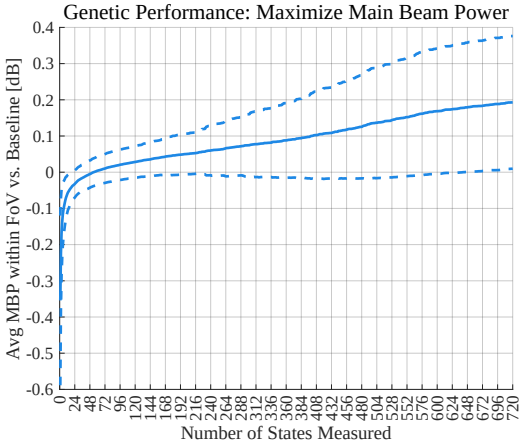
Figures 9.12,9.13, and 9.14 show the statistical performance of the five algorithms when optimizing the main beam power, side love levels, and field of view, respectively. Because the algorithms are stochastic, their performance varies between runs and so their performance is best visualized as a probability distribution. In each of these plots, the solid line represents the average characterization value of the optimal state over the course of a run and the dashed lines indicate one standard deviation in performance from this average. If run performance is assumed to be normally distributed than 68.2% of runs will fall between the two dashed lines. The relative performance of the algorithms are analyzed and compared in Section 9.6.

9.6 Simulated Statistical Analysis

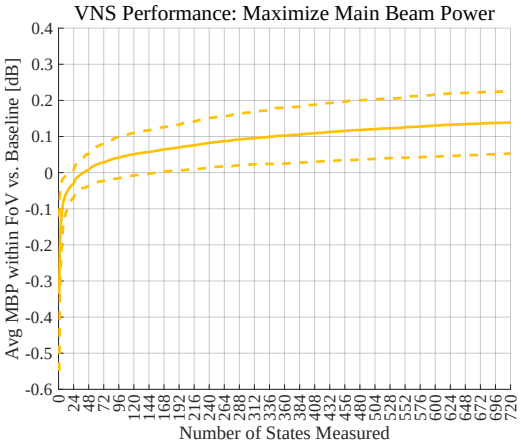
Figures 9.15 through 9.17 directly compare the average performance and standard deviations of the algorithms over time. It should be noted that the performance for the first batch of states is identical no matter the algorithm or optimization criteria. Because each algorithm is initialized by randomly selecting states, each algorithm is essentially a random search for its first batch, hence their identical performance. It is only after this first batch that the algorithms make different choices as to what states to explore and the performance begins to diverge.



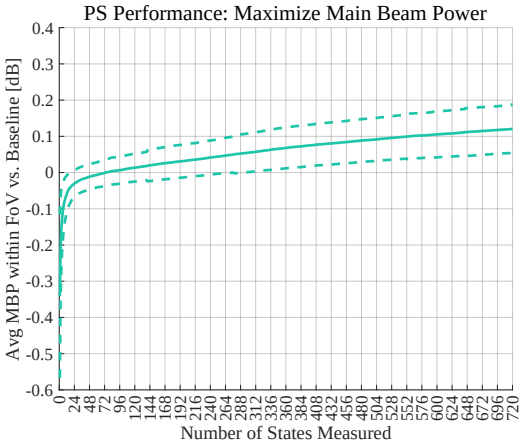
(a) Random Search



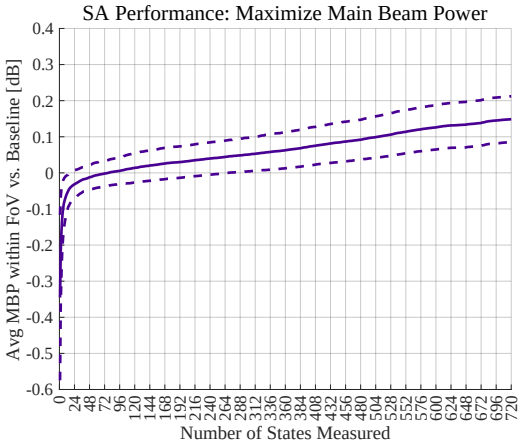
(b) Genetic Optimization



(c) Variable Neighborhood Search

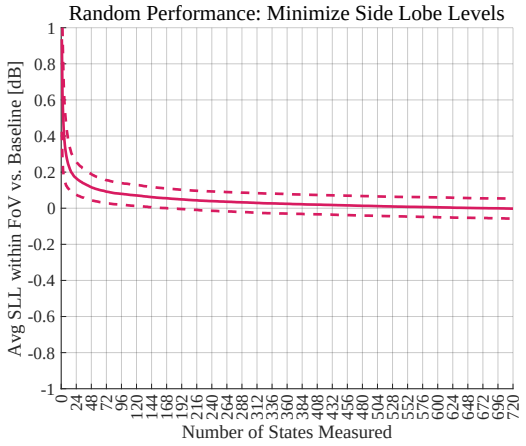


(d) Particle Swarm

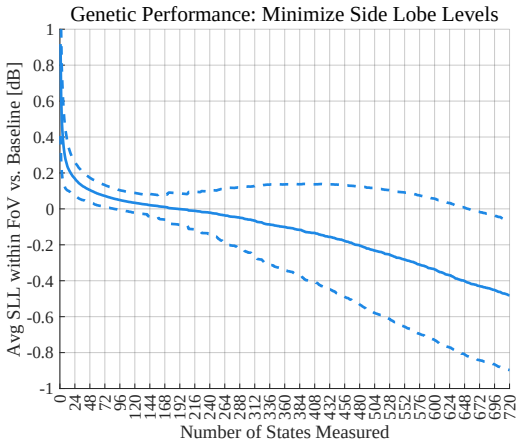


(e) Simulated Annealing

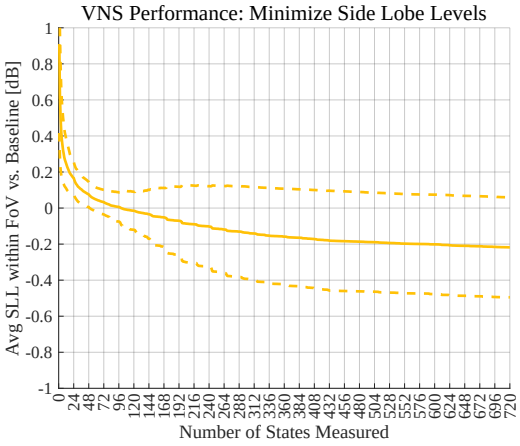
Figure 9.12: Simulated statistical performance of algorithms when optimizing main beam power. The average over 1,000 runs is represented by solid line. Dash lines are one standard deviation from average.



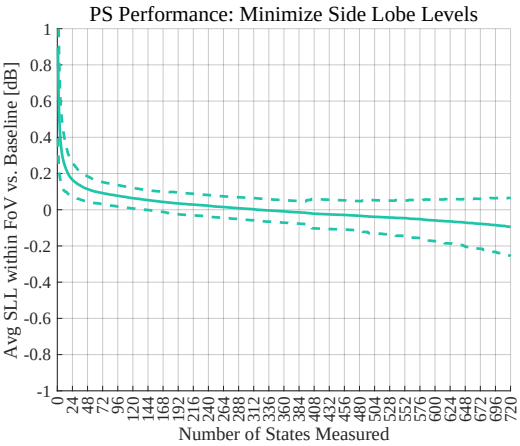
(a) Random Search



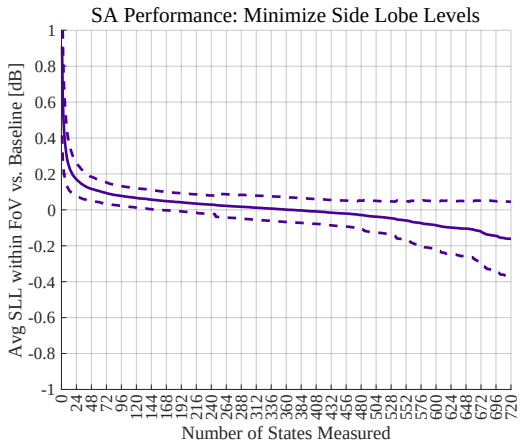
(b) Genetic Optimization



(c) Variable Neighborhood Search

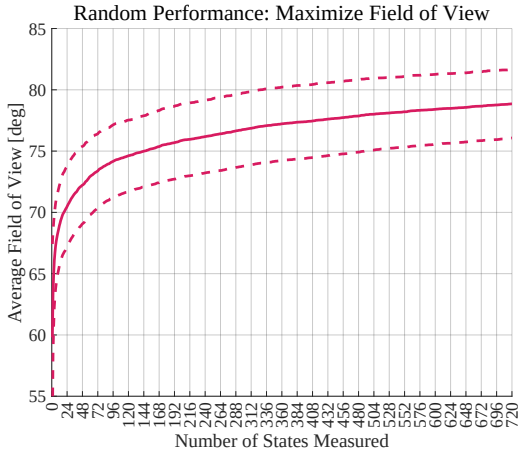


(d) Particle Swarm

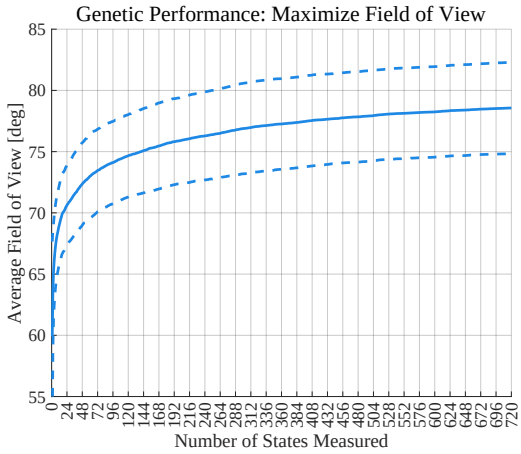


(e) Simulated Annealing

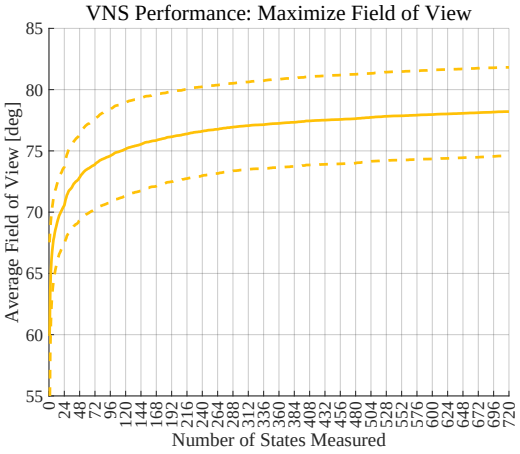
Figure 9.13: Simulated statistical performance of algorithms when optimizing side lobe levels. The average over 1,000 runs is represented by solid line. Dash lines are one standard deviation from average.



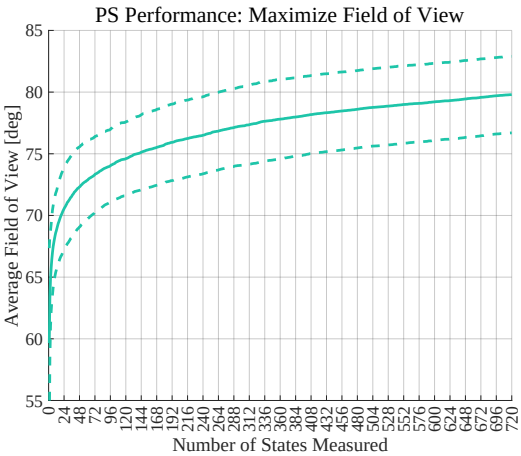
(a) Random Search



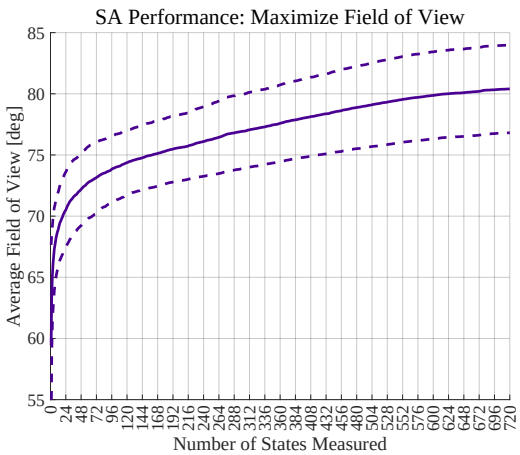
(b) Genetic Optimization



(c) Variable Neighborhood Search

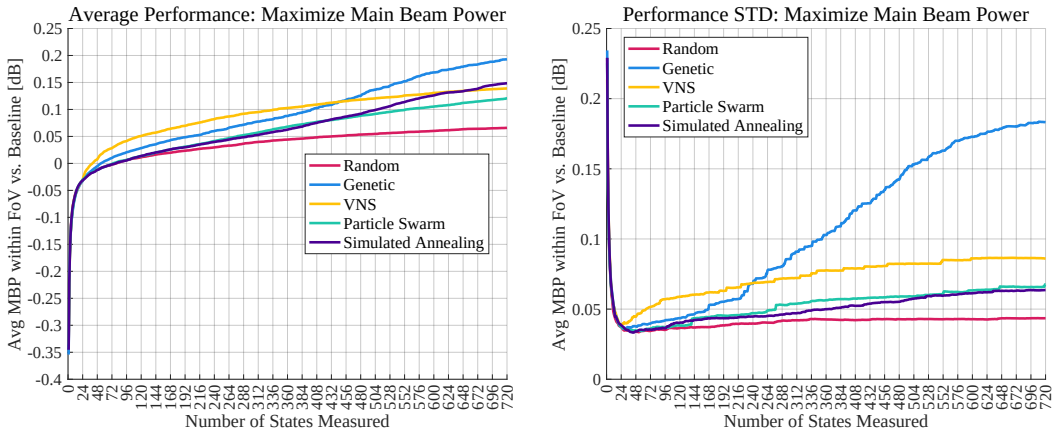


(d) Particle Swarm



(e) Simulated Annealing

Figure 9.14: Simulated statistical performance of algorithms when optimizing field of view. The average over 1,000 runs is represented by solid line. Dash lines are one standard deviation from average.



(a) Average optimization curve

(b) Deviation in performance

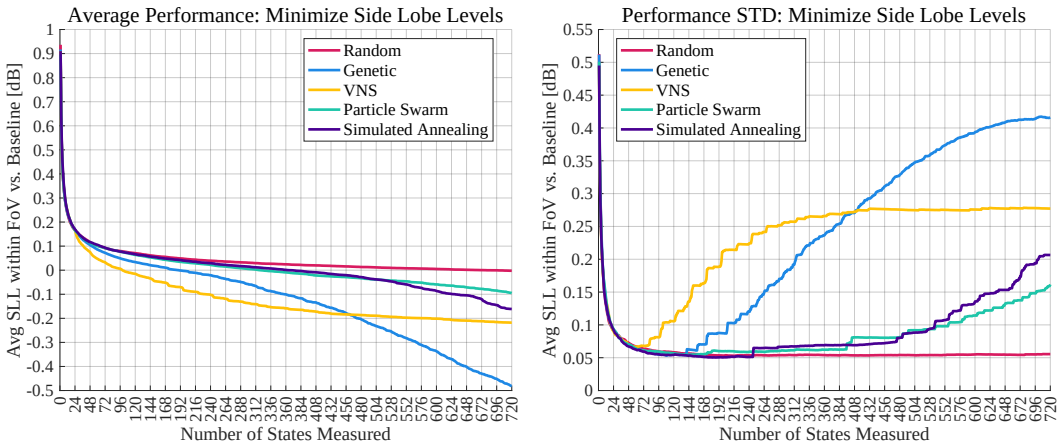
Figure 9.15: Comparison of different algorithm's statistical performance when maximizing the main beam power.

Maximize Main Beam Power Results

Figure 9.15a shows the algorithms' average performance over time when maximizing the average main beam power over the field of view. In terms of the value of the optimal state identified on average, genetic optimization performs the best and random search performs the worst. From a convergence perspective, VNS and random search both quickly approach their final value while the other algorithms exhibit slower but more consistent improvements.

Initially Variable Neighborhood Search outperforms the other algorithms as it initially takes a greedy approach and thus quickly identifies better performing states. However, it also quickly reaches a local optima, plateaus, and is surpassed by both Genetic and Simulated Annealing. Particle Swarm and Simulated Annealing have similar performance at first as they both broadly explore the space. However, towards the end of the run Simulated Annealing takes on more of a greedy approach and more efficiently searches the local space. Genetic optimization performs well throughout the run as it quickly identifies traits that benefit performance and recombines them for steady improvement.

The variance in performance shown in Figure 9.15b paints a different picture. When compared to the other approaches, genetic optimization exhibits significantly larger variation in the value of the optimal state identified. This makes sense as the relatively low mutation rate means that the algorithm mostly explores variations of the initial states; genes not present in the initial population are are unlikely to be introduced. Therefore the quality of the final result highly depends on whether good



(a) Average optimization curve

(b) Deviation in performance

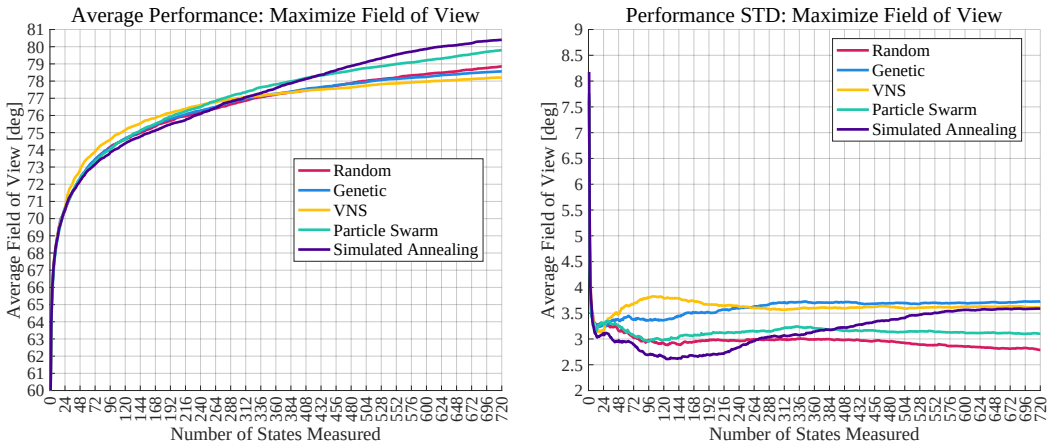
Figure 9.16: Comparison of different algorithm's statistical performance when minimizing the side lobe levels.

genes are present in the randomly selected initial population. VNS also demonstrates high variance as its local first approach also highly depends on the initially selected state. Particle swarm and simulated annealing are more consistent as they initially explore the full extent of the state space before settling in local minima. Finally, the performance of random search is the most consistent as each batch of each run behaves exactly the same. Thus the final performance is purely dictated by the underlying state distribution and the variance associated with Equation 9.2.

Minimize Side Lobe Level Results

Figure 9.15 shows the average and standard deviation of algorithm performance when minimizing the average side lobe level over the field of view. The results of this analysis generally matches that when maximizing the main beam power. Again genetic optimization performs better on average but has much higher variance. VNS and random converge quickly but plateau while simulated annealing and particle swarm gradually approach potentially better solutions. Simulated annealing outperforms particle swarm towards the end of the run. The variance of random search is the lowest and VNS is more variable than particle swarm or simulated annealing.

There are only three major differences in the MBP and SLL results. The first is that both the average improvement in optimization criteria and the variance in performance are generally larger for when minimizing the side lobe levels. The other two differences are more minor, when optimizing the side lobe levels, VNS outperforms simulated annealing on average and simulated annealing has higher



(a) Average optimization curve

(b) Deviation in performance

Figure 9.17: Comparison of different algorithm's statistical performance when maximizing the field of view.

variance than particle swarm.

Maximize Field of View Results

Figure 9.17a shows the average algorithm performance when maximizing field of view. The statistical behavior of the algorithms are very different when optimizing the FoV compared to the MBP or SLL. The most striking result is that, on average, the value of the optimal state identified by both VNS and genetic optimization is *worse* than that identified by a random search. On average, genetic optimization converges just like random search until the last few batches. However, the difference between the average performances is less than 0.5° and so it is possible that the statistical power of the experiment is not large enough for the difference to be statistically significant. VNS still converges the fastest, but is eventually surpassed by the random search by 1° on average, suggesting that a very broad search is required to consistently identify high performing states. Indeed simulated annealing performs the best on average closely followed by particle swarm.

The variance behavior shown in Figure 9.17b is more consistent with the MBP and SLL results. The order of most to least variablity in performance is the same– Genetic, VNS, Simulated Annealing, Particle Swarm, and then Random–, but the difference is smaller. Indeed, the variance of genetic optimization is approximately equal to that of VNS and simulated annealing.

One peculiar behavior is that while the variability of the final result is lowest in random, simulated annealing is more consistent for the first 11 batches. This can

Table 9.4: General Summary of Simulated Algorithm Performance

Algorithm	Random	Genetic	VNS	PS	SA
Performance [MBP/SLL]	Low	High	Medium	Medium	Medium
Performance [FoV]	Low	Low	Very Low	High	High
Variability	Low	High	Medium	Medium	Medium
Convergence	Fast	Slow	Fast	Slow	Slow
Dependence on Initial Conditions	None	High	Very High	Low	None

best be understood by considering the distribution of states given in Figure 9.11c; most of the states exist in a tight cluster while there are particular sub-spaces with better performance. Because the average performance of simulated annealing is also initially worse than random, it is probable that the initial results biases the algorithm's choices to the central mode with worse performance and lower variance. However, as time goes on there is a chance that simulated annealing can escape this subspace and increase its final performance. Thus on average the algorithm performs better than random in the end, but there is high variability as the algorithm is not guaranteed to escape the local optima.

Summary Analysis

In total, the statistical analysis indicates that the main beam power and side lobe level optimization problems are tightly linked and different than field of view optimization. For MBP and SLL, genetic optimization performs the best on average but exhibits a high degree of variance. However, genetic optimization is fairly ineffective when optimizing the field of view. On the other hand, simulated annealing and particle swarm generally perform similarly and reliably produce good performing states on all problems. That said, simulated annealing is subject to more variation on the field of view optimization problem. The behavior of variable neighborhood search is the same across all problems,it converges quickly as it aggressively optimizes the local area but then plateaus on a local minima. Its final performance is thus highly dependent on the initial conditions. VNS is expected to perform worse than a random search when optimizing the field of view. A general summary of the simulated algorithm performance is shown in Table 9.4.

References

[1] P. A. R. Ade, N. Aghanim, M. Arnaud, *et al.*, "Planck 2015 results," *Astronomy & Astrophysics*, vol. 594, A13, Oct. 2016, ISSN: 0004-6361. DOI: 10.1051/0004-6361/201525830.

[2] J. Lavaei, A. Babakhani, A. Hajimiri, and J. C. Doyle, "Solving large-scale hybrid circuit-antenna problems," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, pp. 374–387, 2 Feb. 2011, ISSN: 1549-8328. DOI: 10.1109/TCSI.2010.2072010. [Online]. Available: http://ieeexplore.ieee.org/document/5635369/.

[3] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 67–82, 1 Apr. 1997, ISSN: 1089778X. DOI: 10.1109/4235.585893. [Online]. Available: http://ieeexplore.ieee.org/document/585893/.

[4] K. Sörensen, "Metaheuristics-the metaphor exposed," *International Transactions in Operational Research*, vol. 22, pp. 3–18, 1 Jan. 2015, ISSN: 09696016. DOI: 10.1111/itor.12001.

[5] H. Bremermann, *Optimization through evolution and recombination*, M. Yovits, G. Jacobi, and G. Goldstein, Eds., 1962.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942–1948, ISBN: 0-7803-2768-3. DOI: 10.1109/ICNN.1995.488968. [Online]. Available: http://ieeexplore.ieee.org/document/488968/.

[7] W.-N. Chen, J. Zhang, H. Chung, W.-L. Zhong, W.-G. Wu, and Y.-h. Shi, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 14, pp. 278–300, 2 Apr. 2010, ISSN: 1941-0026. DOI: 10.1109/TEVC.2009.2030331.

[8] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm algorithm," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 1997, pp. 4104–4108, ISBN: 0-7803-4053-1. DOI: 10.1109/ICSMC.1997.637339.

[9] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, pp. 1097–1100, 11 Nov. 1997, ISSN: 03050548. DOI: 10.1016/S0305-0548(97)00031-2.

[10] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 4598 May 1983, ISSN: 0036-8075. DOI: 10.1126/science.220.4598.671.

Chapter 10

META-GAP EXPERIMENTS

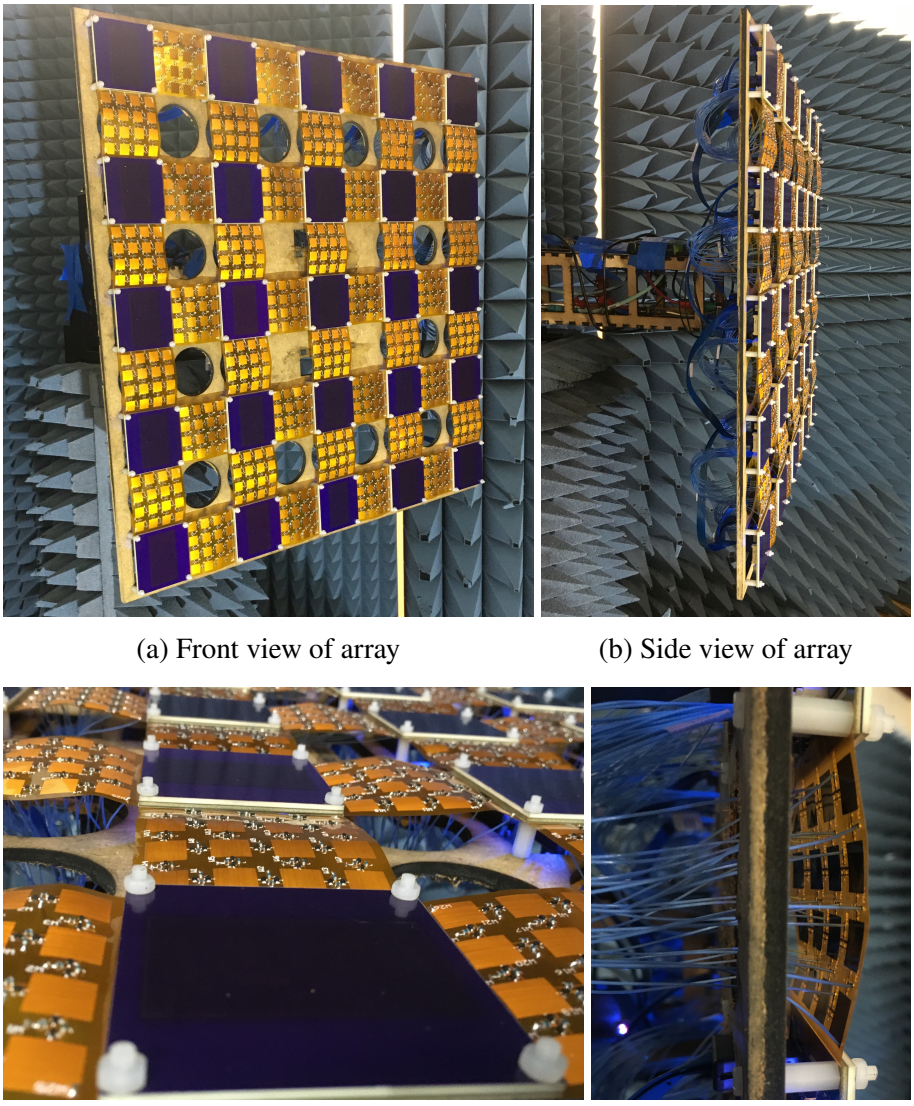
Music is a world within itself
with a language we all understand.
With an equal opportunity
for all to sing, dance, and clap their hands.

Stevie Wonder
Sir Duke

To experimentally test the performance of flexible meta-gaps, they are incorporated into a planar λ-spaced 2D demonstration array. The well established grating lobe behavior of this structure makes it a great test case for examining the behavior of meta-gaps. Using an automated test setup, the optimization algorithms presented in Chapter 9 are used to perform *in-situ* optimization of the array in a series of experiments. The results of these experiments provide insight into both the realized capabilities of meta-gaps and the underlying optimization problem.

The design of the demonstration array and the automated measurement setup are detailed in Sections 10.1 and 10.2, respectively. Then a set of baseline measurements of the array with its gaps filled with a ground plane are presented Section 10.3. These baseline measurements provide an important reference of comparison for later results.

Section 10.4 details the first set of *in-situ* experiments. The size of the search space in this experiment is reduced by restricting the switch settings on each meta-gap sheet to be identical. By constraining the problem to 2^{24} possible states, it is more likely that the optimization algorithms will identify good solutions that provide an initial picture of meta-gap performance. However, this restriction comes at the cost of potentially reducing the performance of the optimal state that can be identified. Based on the results from the first set of experiments and the statistical experiments presented in Chapter 9, the second experiment explores increasing the size of the search space and the associated trade-off between the quality of the optimal state and the ability to find it. Multiple potential switch mappings with different degrees of freedom are presented in Section 10.5. Using multiple tests, the performance



(a) Front view of array (b) Side view of array

(c) Closeup of meta-gaps in array (d) View of switch wires

Figure 10.1: Demonstration array to explore meta-gap performance. The spacing between elements in the 5-by-5 array is λ, with a meta-gap sheet in between each pair of neighboring tiles.

of these mappings are evaluated and an ideal mapping, the E&H Near-Field, is identified in Section 10.6. Section 10.7 repeats the exploration of optimization criteria and optimization algorithms on the E&H Near-Field mapping. Finally, the results of all the experiments are analyzed and conclusions are drawn in Section 10.8.

10.1 Demonstration Array

The full demonstration array is shown from multiple angles in Figure 10.1. The $\lambda-$spaced array consists of 25 radiating tiles separated by 40 meta-gap sheets. The tiles and sheets are held in place by nylon standoffs attached to a medium-density fibreboard (MDF) backbone, with sheets flush to the back of the tiles. As shown in Figures 10.1d and 10.1d, thin wires attached to the sheets pass through holes in the backbone and connect to headers on the tiles. As each sheet contains 24 switches, the total array contains 960 independently programmable switches. Both the tiles and the sheets are 6 cm squares, $\frac{\lambda}{2}$ at the operating frequency. The design of the tiles and the meta-gap sheets are described in Sections 6.4 and 8.5, respectively.

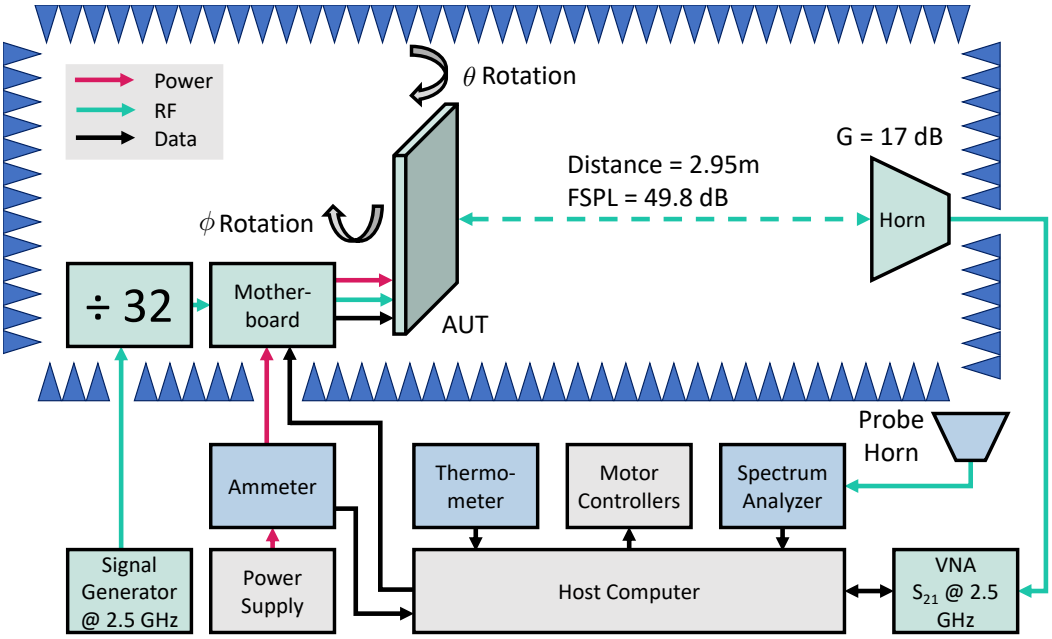
10.2 Measurement Approach

Using an automated range measurement system, meta-gap performance is experimentally explored by using the algorithms described in Section 9.4 to optimize the array's main beam power, side lobe levels, and field of view. As discussed in Section 2.6, states are measured in batches in order to reduce the total measurement time. The algorithms operate in batches of 24 states so that search based approaches, like variable neighborhood search and simulated, can evaluate every neighboring state in a single batch during the identical sheet experiments presented in Section 10.4. The algorithms are iterated for 30 batches, thus exploring a total of 720 states. The same batch size and number of batches are used for the experiments presented in Sections 9.5, 10.6, and 10.7 to facilitate the comparison of results.

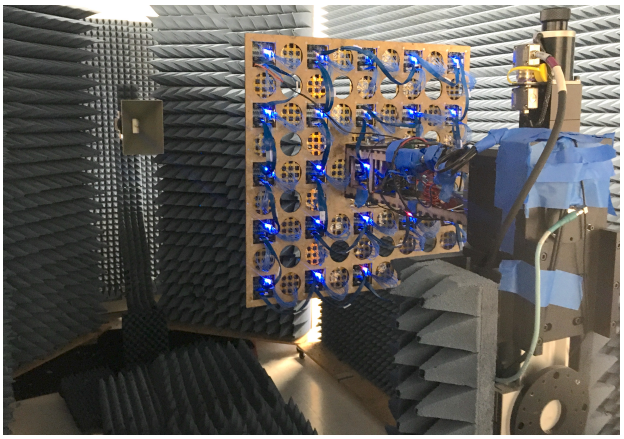
Meta-gap states are characterized by steering beams, measuring their patterns, and calculating the realized optimization criteria. For main beam power and side lobe level optimization, beams are steered every 5° within the field of view, $-30° \leq \theta \leq 30°$, in the E- and H-plane cuts. When optimizing the field of view itself, beams are steered every 5° from $-90° \leq \theta \leq 90°$. For all optimization criteria, the pattern of each beam is measured along the $E-$ and $H-$ planes from $-90° \leq \theta \leq 90°$ in 5° increments, providing information on pattern performance in two orthogonal axes. In total, optimizing the MBP and SLL requires measuring 676 points for each meta-gap setting, while optimizing the FoV requires 5,476 points.

RF Measurement Setup

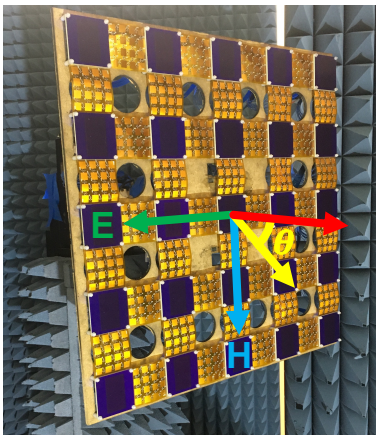
The radiation patterns of the Meta-gap array are measured using a similar experimental setup to the one used for the Shape-Changing Phased Array measurements described in Section 7.2. A diagram of the setup is shown in Figure 10.2a. A



(a) Diagram of measurement setup. Blue triangles indicate RF absorbers and the boundaries of the range. Green components are part of the RF measurement setup. Grey squares are other critical measurement devices. Blue shapes are auxiliary equipment that monitor the measurement environment.



(b) Measurement Range



(c) Coordinate system

Figure 10.2: Automated measurement setup for meta-gap optimization.

frequency generator (Stanford Research Systems, Inc. Model SG386) is used to provide a constant 2.5 GHz signal that is converted to a 78.125 MHz reference using a pair of frequency dividers (total division ratio of 32). This reference is buffered by the array motherboard and distributed through the array under test (AUT). Each tile up-converts the reference back to 2.5 GHz, adjusts its phase, and radiates a fixed power.

The AUT is mounted horizontally on a far-field scanner and rotated about the θ and ϕ axes shown in Figure 10.2c. The linearly polarized array array is oriented so that the array the $E-$ and $H-$ planes are along the $\phi = 0°$ and $\phi = 90°$ cuts. The combined radiated field is measured by a horn antenna 2.95 meters from the AUT. The output of the probe antenna is measured by a vector network analyzer (Agilent Technologies N5230A) set to measure S_{21} at 2.5 GHz. This measured S_{21} is thus a measure of the radiated power relative to a fixed reference[1].

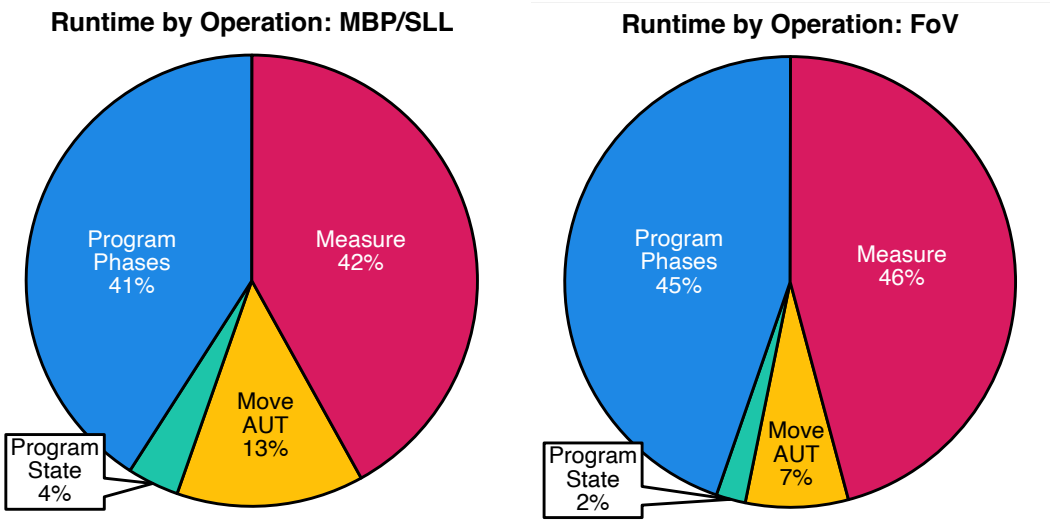
Automated Range Setup and Speed

The range measurement system was automated in order to manage the optimization and the high number of measurements required over several days. As can be seen in Figure 10.2a, a central computer interfaces with the range position motors, the VNA, and the array in order to program meta-gap states, steer beams, and measure patterns. The computer also connects to auxiliary equipment to monitor system performance and the measurement environment. A small probe horn and spectrum analyzer (Keysight N9918A) outside of the range detects leaked power and monitors the radiated spectrum and lock characteristic. A multimeter (Keysight 34461A) measures the current consumption of the array. Finally, the temperature in the room is measured with another multimeter (Keysight 34465A) connected to a thermocouple so thermal variation can be accounted for if necessary.

In order to interface with the optimization framework given in Code Segment 9.1, the system abstracts these controls into five functions. `programState(state)` decodes the abstract numerical representation of the meta-gap state and programs the tiles with the proper switch settings. `measure()` commands the VNA to measure the power at the current measurement point. `moveToAngle(beam_angle)` moves the array so that the probe antenna measures the desired (θ, ϕ) coordinate, automatically adjusting the probe polarization accordingly. `optimizePhase()` interfaces with both the array and the RF measurement equipment to steer beams and determine the optimum phase settings using the optimization algorithm described in Section 2.7. Finally, these optimal tiles phases are reprogrammed with `programPhases(phase_settings)` between measurement points.

On average, moving the array between measurement points takes 3.56 seconds,

[1]The output of the VNA is not used as a reference because the PLL at the core of the second generation tile expects a constant reference source on reset as it performs an automated calibration to optimize its lock range. Therefore, the pulsed nature of the vector network analyzer output introduces instability into the PLL lock behavior.



(a) MBP and SLL Measurements (b) Field of View Measurements

Figure 10.3: Breakdown of experiment run-time by operation.

programming the phases takes 17.9 milliseconds, programming the array state takes 42.7 milliseconds, and measuring the radiated power takes 5.9 milliseconds. Beam optimization is performed with 1° phase precision, thus requiring 10 measurements per tile. In total beam optimization takes 1.9 seconds. Given the time required for each function, the number of measurements per meta-gap state, and the batch size, the time required to measured each batch can be predicted using Equation 2.32. The total time required for each run is 23.4 hours for main beam power and side lobe level optimizations and 59.7 hours for field of view optimizations. The percentage of time spent performing each function is shown in Figure 10.3. The measurement speed is limited by the reliability of the array SPI channel and the time required to transfer measurement data from the VNA to the controlling computer.

Differential Measurements

Because the optimization algorithms run for an extended length of time, there is a risk that substantial low frequency noise and/or thermal drift will distort the relative performance of states measured at different times. For example, the main beam power optimization criteria is calculated using the absolute measured power over the steering range. If the same state is measured at two different times and the temperature increases between measurements, than the total power radiated by the tile's output PA can increase. This higher power throughout the radiation pattern would appear to be an increase in power concentrated in the main beam from the standpoint of the optimization algorithm. Therefore, the same state would

be deemed to have better performance. While this kind of repeatability challenge exists for all long-duration measurements, it is devastating in the context of an optimization. Measurement fluctuations could mislead the optimization algorithm into concluding that a worse state was optimal and redirect its search in the wrong direction.

To compensate for long-term temperature fluctuations, a baseline state in which every switch is turned off is remeasured in every batch and used to normalize the other patterns. Due to the interwoven nature of the batched measurements, different states in the batch happen in quick succession at a given measurement point, therefore, the difference between these measurements are mostly immune to low-frequency noise and temperature fluctuations. By normalizing each measured pattern to the same baseline state, the relative performance of states in different batches can be accurately compared. Finally, when the algorithm terminates, the top 10 performing states are remeasured in the same batch in order to directly compare their performance. As with the results in Chapter 9, the performance of meta-gap states is quantified relative to this "all off" state.

Using Single Data Points to Analyze Stochastic Algorithms

Due to the lengthy measurement time, each combination of algorithm, optimization criteria, symmetry, and initial states is only run once. Because the algorithms are stochastic, it is important to keep in mind that the experimental results are inherently variable. The measured optimization curves are a single run of stochastic system and so their performance is prone to algorithmic variance similar to those discussed in 9.6. For example, a result indicating that Particle Swarm performs better than Simulated Annealing is not definitive as there is always the possibility that the former had a unusually high performing run while the latter had an unusually low performing run. That said, there are clear trends between the different optimization criteria, the different *in-situ* experiments, and the statistical experiments in Chapter 9.

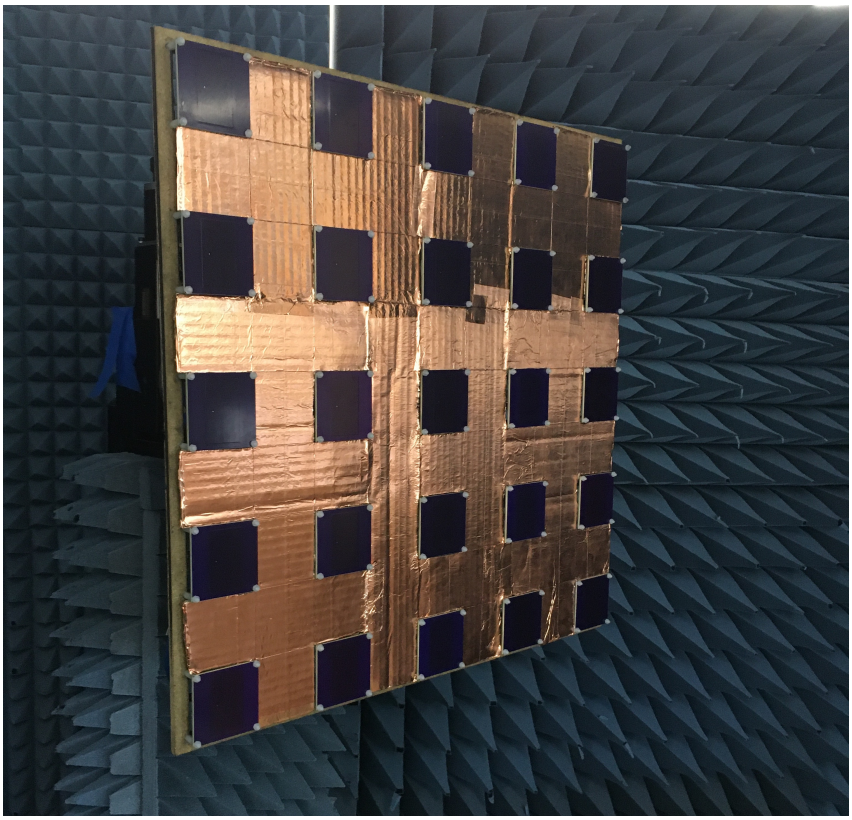
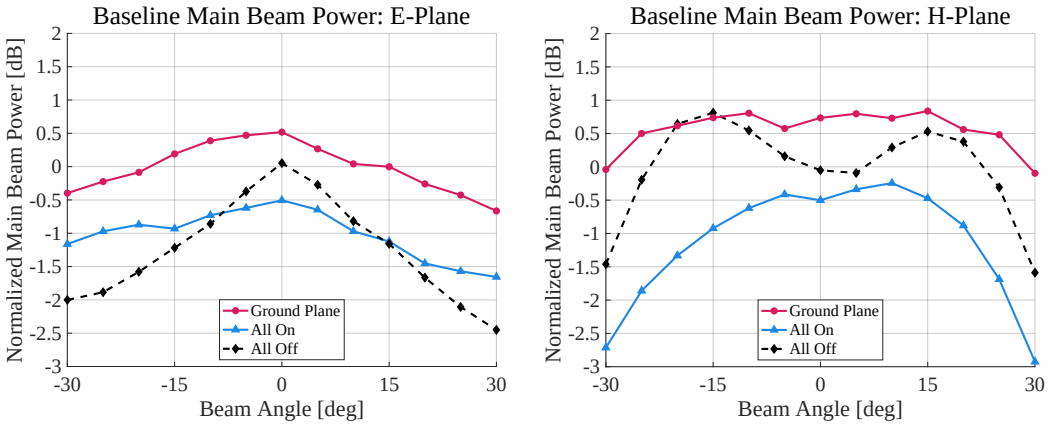


Figure 10.4: Photo of array with copper tape ground plane between tiles. This array serves as a baseline comparison for meta-gap performance.

10.3 Baseline Measurements

While useful for measurement purposes, the "all off" state used as a baseline is not indicative of the array performance without the meta-gap sheets. Although ideally the meta-gaps would be transparent in this state, in reality the metal grid, wires, switches, and inductors still interact with the electromagnetic field. Because it is highly likely that these interactions are parasitic and degrade array performance, the baseline is not an entirely fair comparison to general array performance. Thus, there is a risk of drawing incorrect conclusions about meta-gap performance; like concluding that carrying a 5 kg backpack improves the performance of a sprinter because they can run faster with it than when they are carrying a 25 kg backpack.

Of more interest is the relative performance of the meta-gap sheet compared to a simpler approach expected to improve array performance. An obvious solution is to fill the gaps in the array with a ground plane, effectively creating a uniform ground plane commonly found behind elements in an array. The ground plane creates isolation from the antennas and the backside electronics and a more uniform near-field environment. Thus a flexible ground plane is a much simpler structure



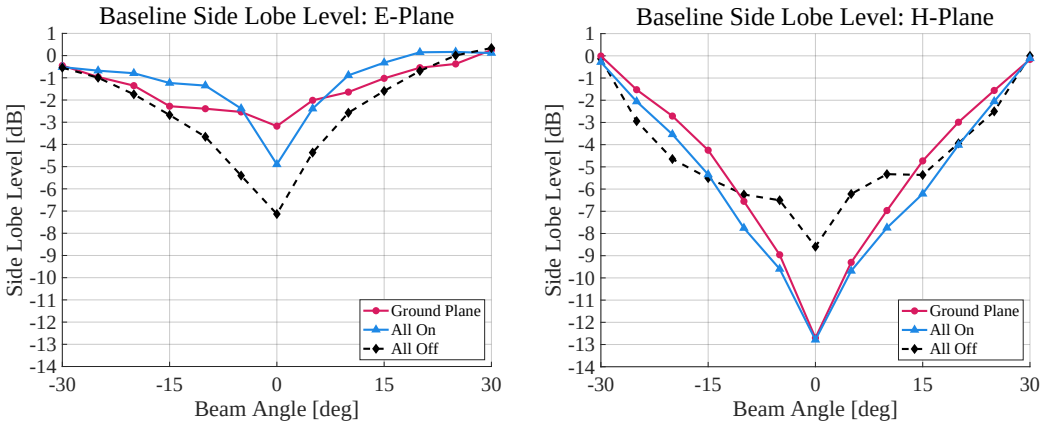
(a) $E-$plane cut (b) $H-$plane cut

Figure 10.5: Measured main beam power for the ground plane backed array and the "all off" and "all on" baselines. Power is normalized to the broadside power of the "all off" baseline. Reprinted with permission from the copyright holder, EuMA.

than meta-gaps that can be used to fill the gaps in the array to increase performance. While such a ground plane would make a superior baseline for measurements, the "all off" switch setting is used as a comparison instead because it can be rapidly switched to electronically unlike a true ground plane.

To enable comparisons between meta-gap performance and a ground plane, the gaps in the array is filled with a ground plane comprised of copper tape as shown in Figure 10.4. Performance of this ground-backed array is measured using the same method described in Section 10.2 with the exception that *absolute* power measurements are recorded instead of the power relative to the baseline. The same *absolute* power measurements are then taken for the meta-gap array in both the "all on" and "all off" baselines. These absolute measurements allow the performance of the ground plane backed array to be compared to the baseline, and thus also be compared to any meta-gap state measured relative to the baseline.

Figure 10.5 shows the main beam power measurements for the ground plane backed array and meta-gap array in the "all on" and "all off" states normalized by the peak broadside power of the "all off" baseline. As can clearly be seen, the "all off" baseline state reduces the main beam power at almost every steering angle. In fact, the average main beam power throughout the field of view for the ground plane backed array is 0.9 dB higher than the baseline state. The peak broadside main beam power is 0.5 dB higher in the ground plane backed array than the baseline state.



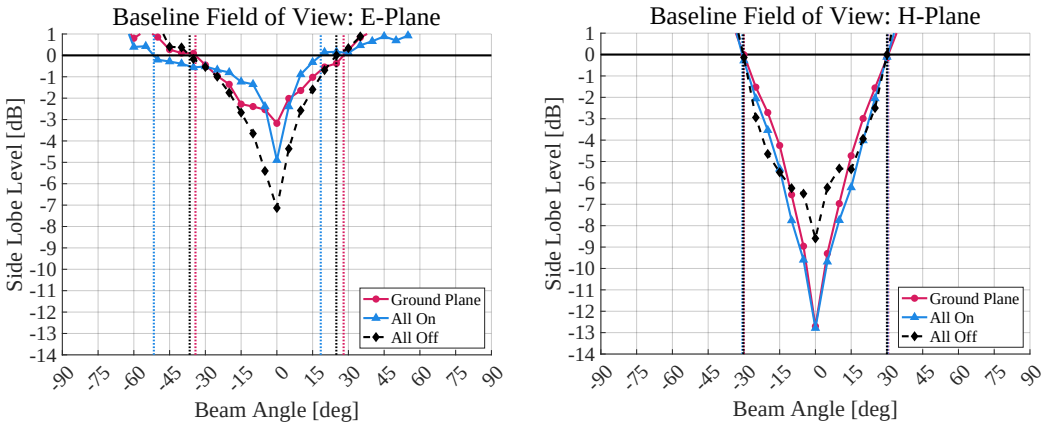
(a) $E-$plane cut (b) $H-$plane cut

Figure 10.6: Measured side lobe levels for the ground plane backed array and the "all off" and "all on" baselines. Reprinted with permission from the copyright holder, EuMA.

Also of interest is the performance of the "all on" state which ideally is equivalent to the ground plane backed array. It is notable that the general shape of the "all on" state resembles that of the ground plane array, although with at least 1 dB less main beam power at every angle. The degradation in performance due to the non-ideal switches and wires can thus be clearly seen.

Figure 10.6 shows the side lobe level measurements for the ground plane backed array and meta-gap array in the "all on" and "all off" states. The "all off" baseline actually provides better suppression of side lobes than the ground plane backed array, suppressing broadside side lobes by -7.8 dB, 4.6 dB more than the ground plane. In addition, the average side lobe level throughout the field of view for the ground plane backed array is 0.3 dB higher than the baseline state. Again the "all on" state performs similarly to that of the ground plane backed array, with slightly more side lobe suppression in the $E-$plane.

Finally, Figure 10.7 sows the field of view measurements for the ground plane backed array, the "all on" state, and the "all off" baseline. As expected average field of view between cuts for the ground plane backed array is $61.4° \approx 60°$. The "all off" baseline has a similar average field of view of $60.9°$. The $E-$plane field of view of the "all on" state is higher than the ground plane backed array, highlighting the non-ideality of meta-gaps states. The average field of view for the "all on" state is $65.5°$.



(a) $E-$plane cut (b) $H-$plane cut

Figure 10.7: Measured field of view for the ground plane backed array and the "all off" and "all on" baselines.

10.4 Identical Sheet Experiments

In the first set of experiments, each of the five algorithms are used to perform *in-situ* optimization of the main beam power, the side lobe levels, and the field of view. Initially each meta-gap sheet is programmed with the same switch settings in order to limit the size of the search space so decent solutions can be quickly found. This restriction is lifted in Sections 10.6 and 10.7.

Experimental results are presented in multiple formats. Algorithmic performance is evaluated using plots of the optimal optimization criteria identified over the course of the run. In addition, the average performance of states explored by the algorithm over time is presented using a moving average over two batches. This average provides another useful view of algorithm behavior. Plots of the array characteristics for the optimal states, and the "all off" baseline are also shown. Finally, images of the optimal switch settings identified are shown.

Main Beam Power Results

Figure 10.8 shows the algorithms' performance over time when maximizing the main beam power. Genetic optimization, particle swarm, and simulated annealing have similar final results and outperform both variable neighborhood search and random search. Meta-gaps are able to increase the average main beam power within the view of view by 1.13 dB compared to the all off baseline. Comparing this improvement to the ground plane measurements in Section 10.3 indicates a 0.23 dB improvement compared to the ground plane backed array. Interestingly even the random search is capable of improving the average main beam power by at least 0.89 dB compared to the baseline, only 0.01 dB lower than the ground plane backed antenna. Therefore it is relatively straightforward to identify meta-gap states that perform at least as well as the ground plane backed array.

The average value of the explored states provide additional insight. Critically the performance of the average state explored by the random search is stable. As discussed in Section 9.4, this stability and lack of a trend indicates that the experimental measurements do not have any major problems. In addition, while the final results of genetic, particle swarm, and simulated annealing are similar, the former's average state value increases much faster initially before leveling off. During this period, the algorithm quickly prunes lower performing genes. Simulated annealing and particle swarm however converge slower as they explore the full search space. Finally, the state of the VNS algorithm can be determined by its average state performance which increases when the algorithm performs a focused greedy search, and decreases once the algorithm reaches a local optimum and searches a wider space of states.

Figure 10.9 shows the array characteristic of the optimal state identified by each algorithm. As can be clearly seen, the meta-gaps improve the main beam power throughout the field of view. In addition, meta-gaps are able to correct the dip in power in the center of the $H-$plane. Broadside the improvement is even greater, with a 1.5 dB increase in main beam power compared to the baseline and thus a 1 dB improvement compared to the ground plane backed array.

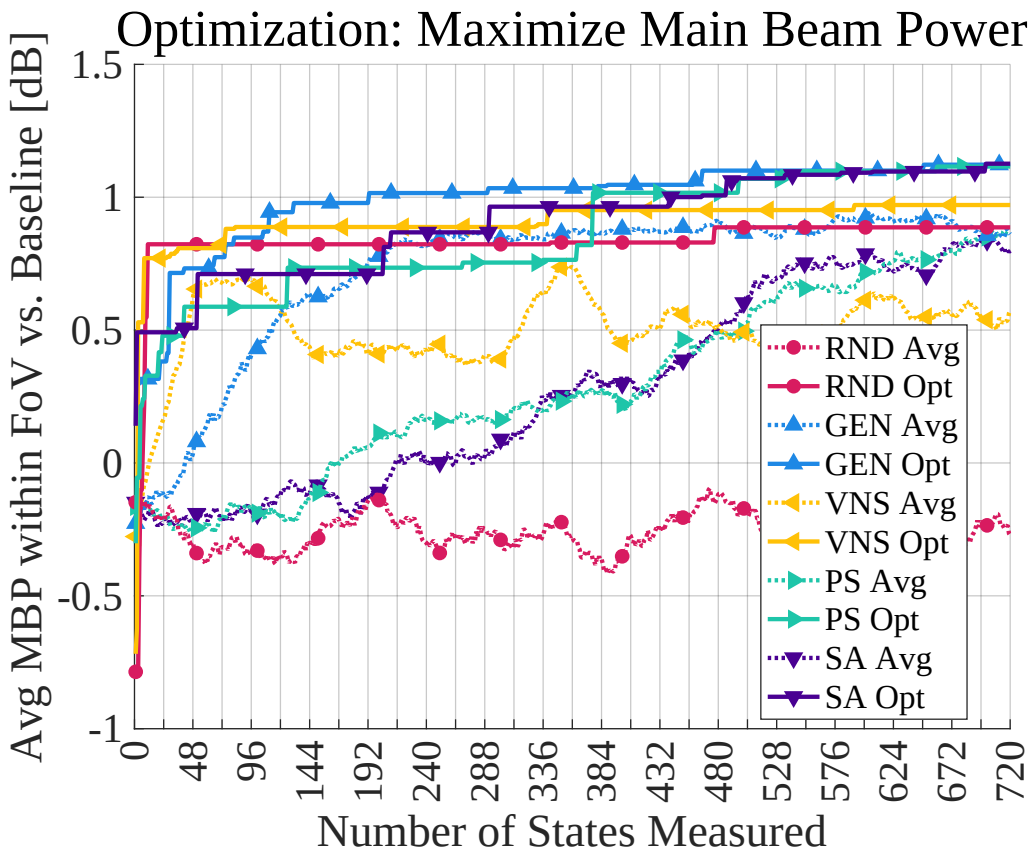
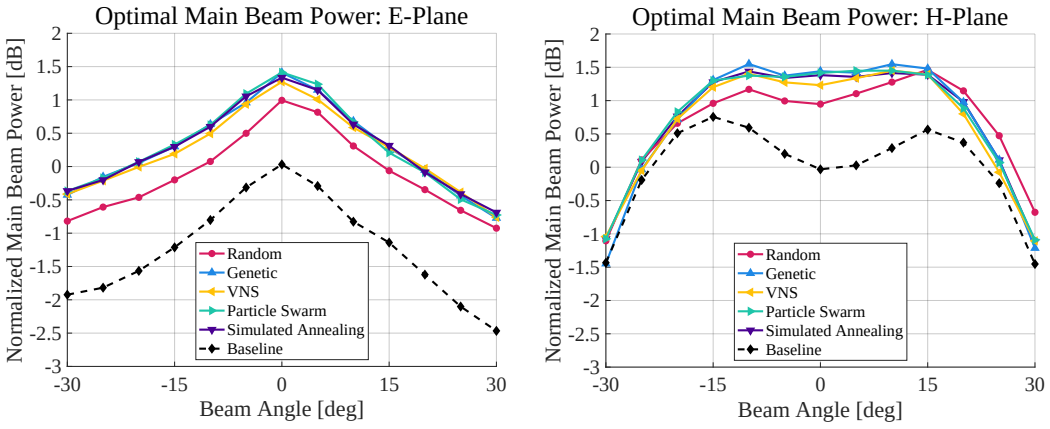


Figure 10.8: Optimization curves for the different algorithms when maximizing the main beam power subject to the identical sheet constraint.



(a) $E-$plane cut (b) $H-$plane cut

Figure 10.9: Measured main beam power of the optimal states identified by each algorithm when subject to the identical sheet constraint in addition to the "all off" baseline. Power is normalized to the broadside power of the "all off" baseline.

Side Lobe Level Results

Figure 10.10 shows the algorithms' performance over time when maximizing the side lobe levels. Again, genetic optimization, particle swarm, and simulated annealing outperform both variable neighborhood search and random search. Genetic optimization identifies the best performing state, which is capable of suppressing the side lobes by 1.7 dB on average throughout the field of view compared to the baseline state. This is equivalent to 2 dB more suppression than the ground plane backed array. Even random search is capable of suppressing the average side lobe level 1.3 dB more than the ground plane backed array.

The average explored state's side lobe levels mirror the average main beam power results; the random search is stable, VNS oscillates, genetic optimization decreases rapidly before plataueing, and particle swarm and simulated annealing improve gradually.

Figure 10.11 shows the array characteristic of the optimal state identified by each algorithm. Interestingly the side lobe levels are generally unaffected in the $E-$ plane, almost all of the improvement is due to reducing the relative side lobes in the $H-$ plane. However, the broadside side lobe-levels in the $E-$ plane are improved by 1 dB, indicating a 5.6 dB improvement over the ground plane backed array.

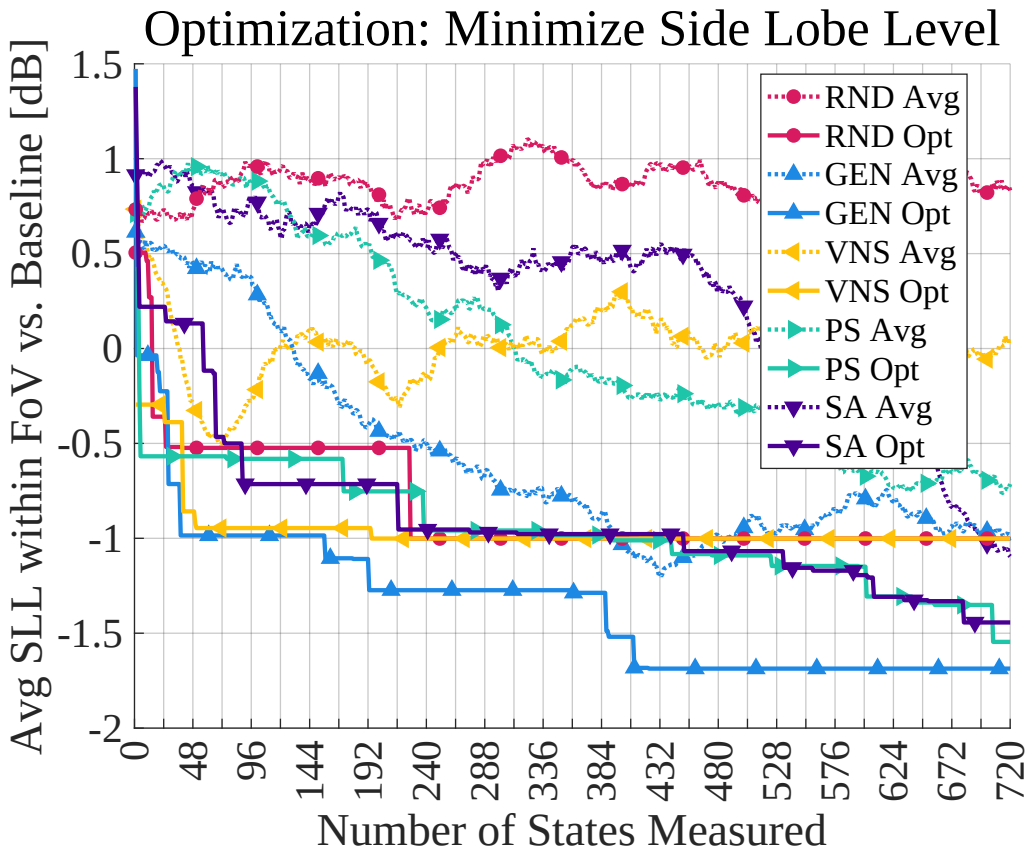
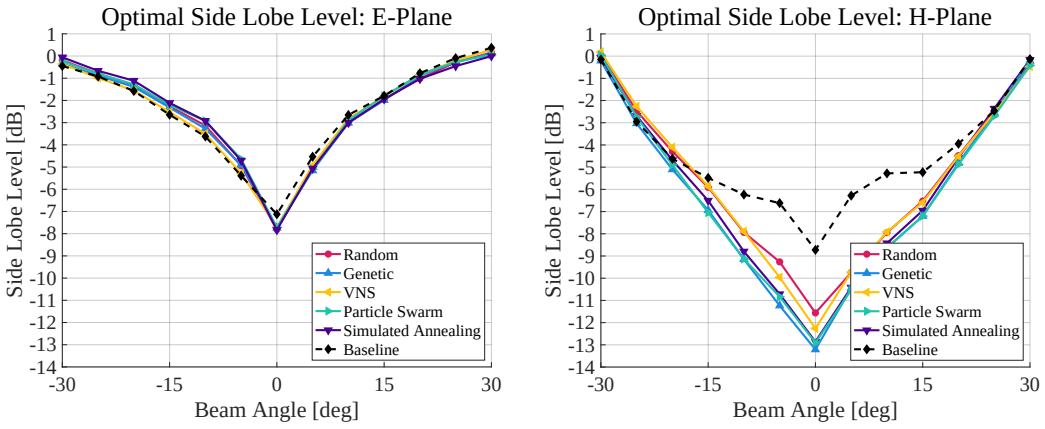


Figure 10.10: Optimization curves for the different algorithms when minimizing the side lobe levels subject to the identical sheet constraint.



(a) $E-$plane cut (b) $H-$plane cut

Figure 10.11: Measured side lobe levels of the optimal states identified by each algorithm when subject to the identical sheet constraint in addition to the "all off" baseline.

Field of View Results

Figure 10.12 shows the algorithms' performance over time when maximizing the field of view. In this optimization problem all of the algorithms performed similarly, with particle swarm identifying the best state. Each of the algorithms increased the average field of view between cuts to over 79.4°. The largest field of view identified is 83.5°, 23.5° higher than the theoretical value. Interestingly the average state behavior is relatively consistent between the algorithms, with the directed algorithms gradually making minor improvements and random search average being stable.

The field of view results are especially intersting when considering the field of view characteristic plots shown in Figure 10.12. The results are opposite those when suppressing the side lobe levels; the $H-$ plane is mostly unaffected while the $E-$ plane contains almost all of the field of view improvement. Thus algorithms are able to increase the FoV in the $E-$ plane to around a staggering 100°.

The shapes of the side lobe level curves in Figure 10.13a are especially curious as the field of view appears to be improved by *increasing* side lobe levels. The side lobe level is just barely negative for most of the expanded field of view, indicating the main lobe is only slightly higher than the peak side lobe. While this technically meets the definition of field of view given in Section 2.5, it does not follow the *spirit* of the concept and thus renders the results relatively impractical[2]. Nonetheless, the fact that field of view extension is even possible suggests that meta-gaps can effectively decrease the distance between antennas in phase space and thus change the aliasing behavior of the array.

[2]This behavior is not particularly surprising as exploiting the technicalities in a problem framing is the modus operandi of optimization algorithms.

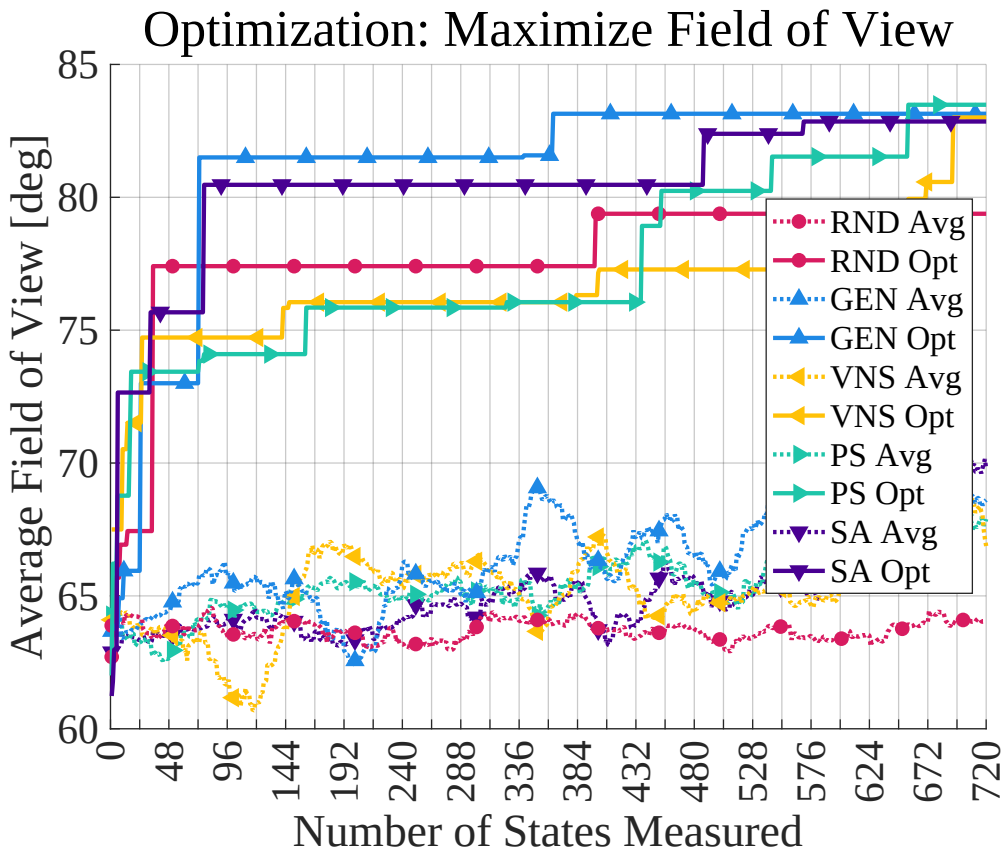
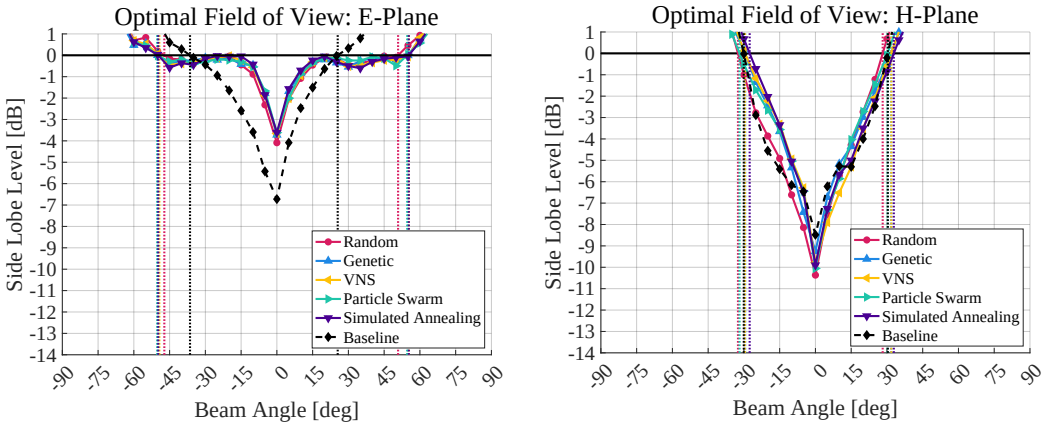


Figure 10.12: Optimization curves for the different algorithms when maximizing the field of view subject to the identical sheet constraint.



(a) $E-$plane cut (b) $H-$plane cut

Figure 10.13: Measured field of views of the optimal states identified by each algorithm when subject to the identical sheet constraint in addition to the "all off" baseline.

Optimal Switch Settings

Figures 10.14, 10.15, and 10.16 show diagrams of the optimal identical sheet switch settings identified by each algorithm when optimizing the main beam power, the side lobe levels, and the field of view, respectively. The meta-gap sheet is represented using a four-by-four grid of light copper-colored squares separated by either black, or darker copper-colored lines. These lines are indicate the status of each switch in the state; black lines indicate the switch is off while copper colored lines indicate it is on. Thus the shape of the formed conductor pattern can be visualized while still identifying the location of switches.

Each of the optimal sheets identified are likely to be local optima, and thus not the true ideal shape. However, it may be possible to identify traits of the globally optimal solution by considering the states as a group and identifying patterns. To facilitate this, Figures 10.14f, 10.15f, and 10.16f provide weighted heat-maps of the different states.

In these heat-maps, switch states are represented by an average of the switch settings in the different optimal states weighted by the relative performance of the state. This average indicates the relative agreement between optimal states. For example if all five states performed equally and switch A is on in three of the states, that the switch status would be represented by 0.6. If one of the states where switch A is off performed twice as well as the others, than the switch status would be represented by 0.5. These continuous status numbers are visualized using a color gradient between the light-copper conductor color, and black. Thus if a switch is on in all states, it's status would be 1 and it will be indistinguishable from the nearby conductor. If a switch is off in all states, than it will be a solid black line. Together the heat-map visualizes the best guess about the status of the switches in the optimal state.

Consider the optimal identified settings for main beam power shown in Figure 10.14. There is general agreement that the central four conductors should be connected in a square and that the to middle horizontal rows should be connected. The top and bottom rows of conductors should be completely disconnected however. While none of the states are symmetric about the horizontal or vertical axes, they are all very close to being symmetric, requiring only one or two switches to change to be symmetric about some axis. Indeed the weighted heat-map indicates that vertical symmetry is strongly preferred, and horizontal symmetry to a lesser extent.

The optimal sheets for side lobe level minimization shown in Figure 10.15 suggest similar conclusions. In fact, the genetic optimization state is symmetric about both

axes and shares a lot of features in common with most of the other states. This is indicated by the close resemblance between the genetic optimization state and the weighted heat-map. It is plausible that this genetic state is the local optima that both simulated annealing and particle swarm were approaching when the algorithm terminated.

Unlike the optimal main beam power and side lobe level states, the optimal states in the field of view optimization shown in Figure 10.16 do not share a lot in common. The only relatively clear pattern in the weighted heat-map, is the lack vertical connections to the top and bottom rows. There is strong disagreement for most of the remaining switches. Interestingly, the performance of these different geometric shapes is relatively consistent in Figure 10.12.

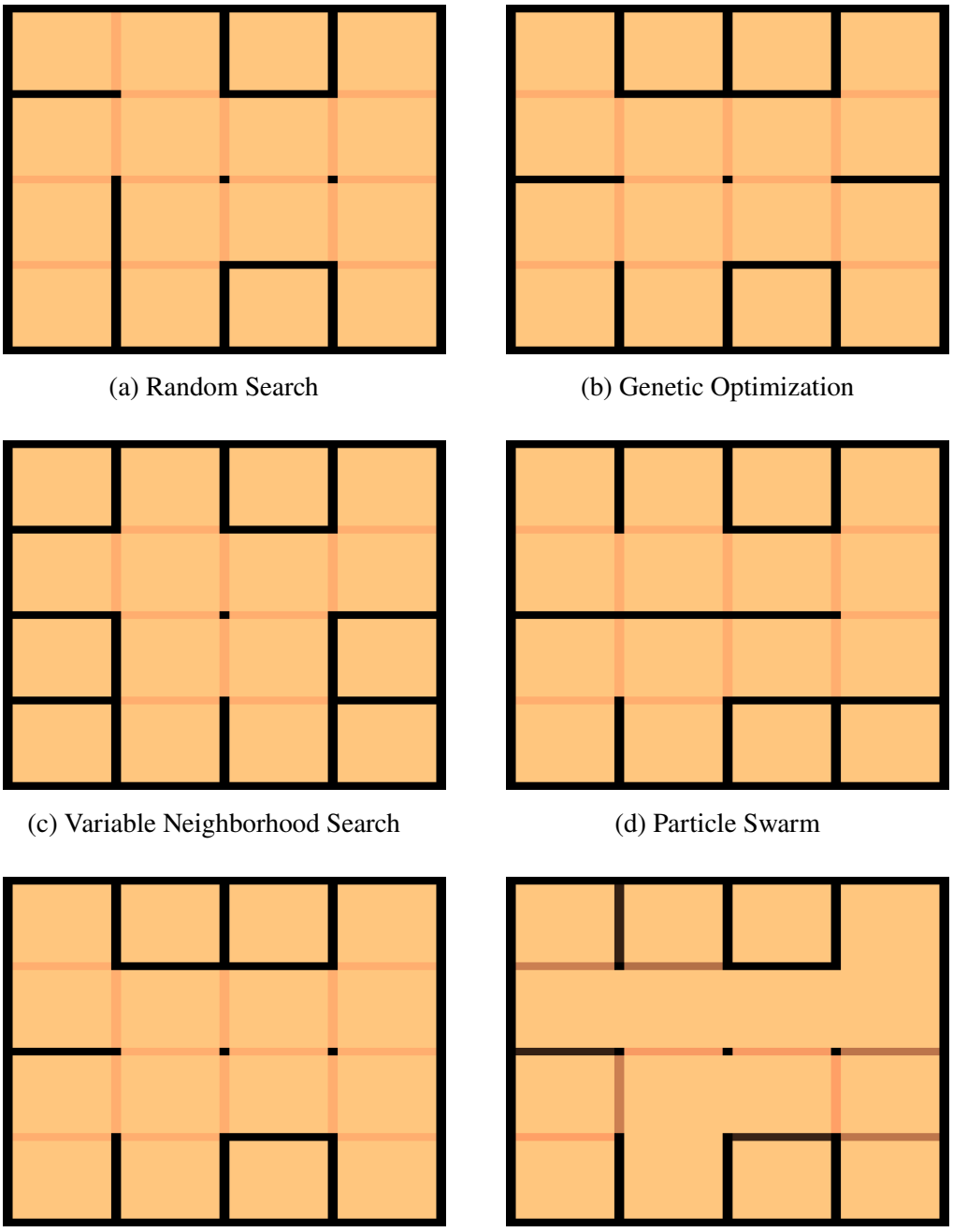
10.5 Electromagnetic Symmetry

Until this point, the switch settings of each sheet in the array have been identical in order to reduce the search space to 24 degrees of freedom. This is a highly restricted subspace of the full state space containing 960 degrees of freedom. Because the optimal states identified in Section 10.4 are a part of the larger state space, they set a minimum bound on the optimal array performance. By expanding the search space to include more degrees of freedom, it is possible to identify further improvements in performance.

However, fully opening up the search space to all 960 degrees of freedom is unlikely to result in significantly greater performance because the algorithms are less likely to find the optimal states in the significantly larger search volume. Fortunately by considering properties that the optimal solutions are likely to have, it is possible to restrict the search space without eliminating the best performing states.

There are several intuitive properties that the optimal states are expected to have. Because the array is completely symmetric about the $E-$ plane axis, and symmetric performance is desirable, it is highly likely that the optimal state is also symmetric about the $E-$ plane axis. In addition, the array is *mostly* symmetric about the $H-$ plane axis as the only asymmetry is the location of the antenna drive port. Thus another reasonable assumption is that the optimal solution will be symmetric about the $H-$ plane axis as well. Enforcing symmetry in both the $E-$ and $H-$ planes reduces the degrees of freedom from 960 to 248.

It is also possible to make more aggressive restrictions based on electromagnetic reasoning, though whether they must be true for the optimal state is less obvious.



(a) Random Search

(b) Genetic Optimization

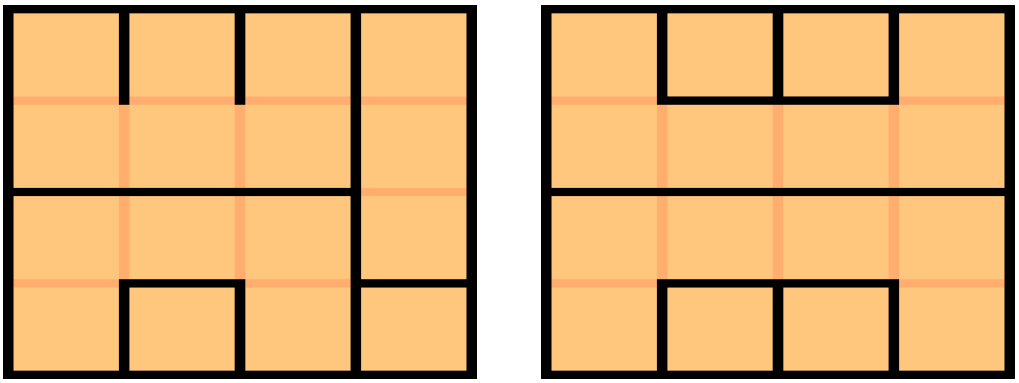
(c) Variable Neighborhood Search

(d) Particle Swarm

(e) Simulated Annealing

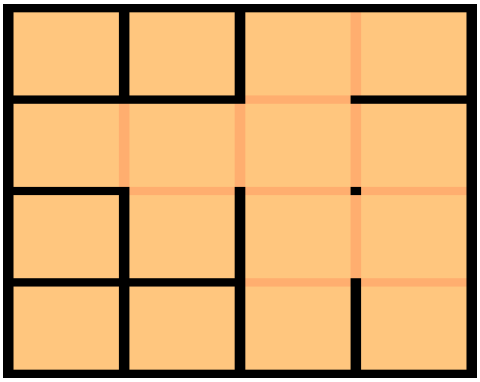
(f) Weighted Heatmap

Figure 10.14: Optimal identical sheet switch settings identified by different algorithms when optimizing main beam power subject to the identical sheet constraint.

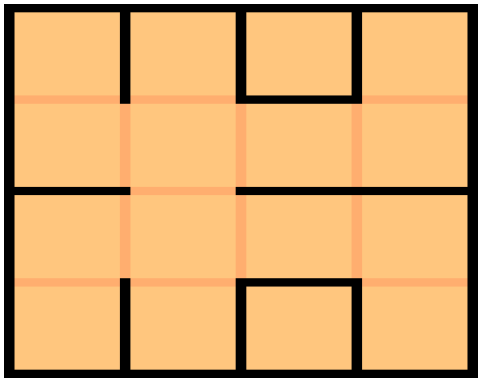


(a) Random Search

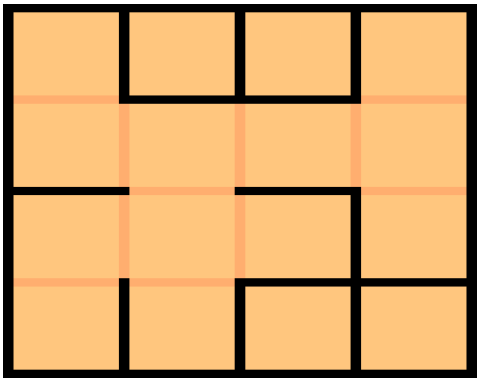
(b) Genetic Optimization



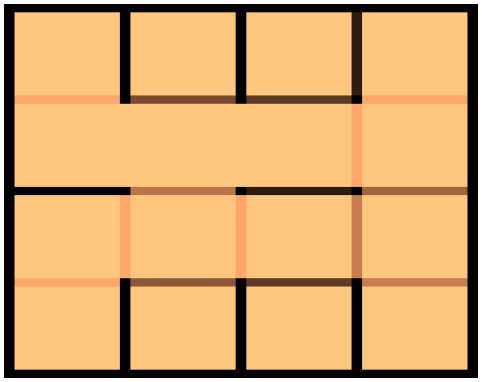
(c) Variable Neighborhood Search



(d) Particle Swarm

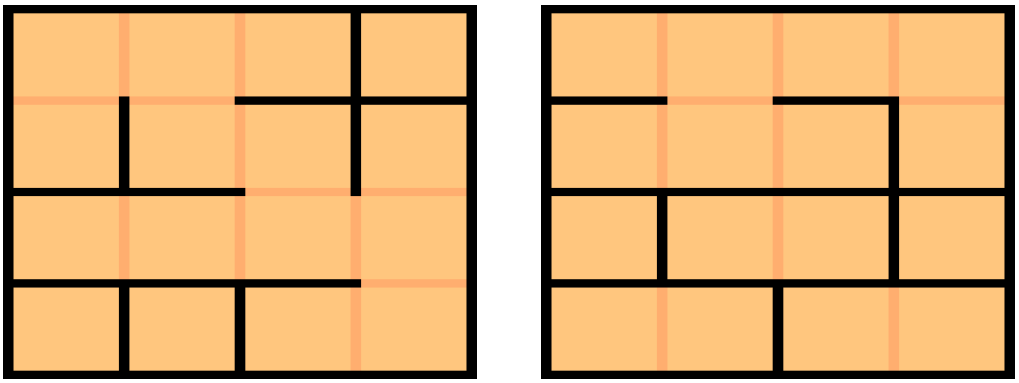


(e) Simulated Annealing



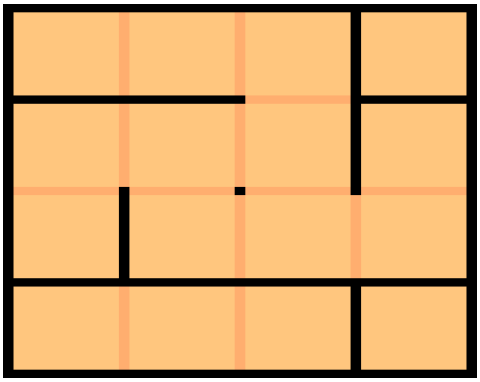
(f) Weighted Heatmap

Figure 10.15: Optimal identical sheet switch settings identified by different algorithms when optimizing side lobe levels subject to the identical sheet constraint.

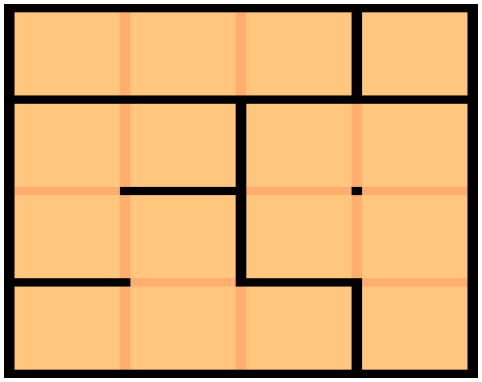


(a) Random Search

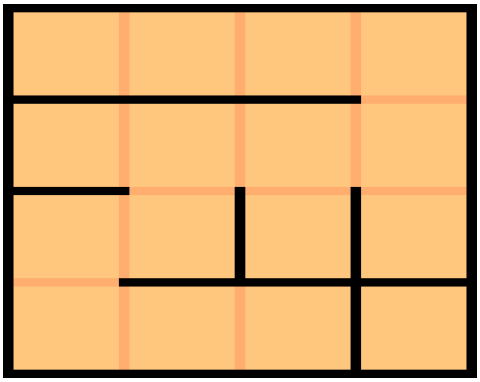
(b) Genetic Optimization



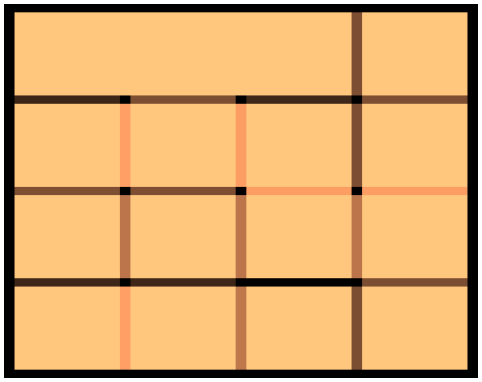
(c) Variable Neighborhood Search



(d) Particle Swarm



(e) Simulated Annealing



(f) Weighted Heatmap

Figure 10.16: Optimal identical sheet switch settings identified by different algorithms when optimizing field of view subject to the identical sheet constraint.

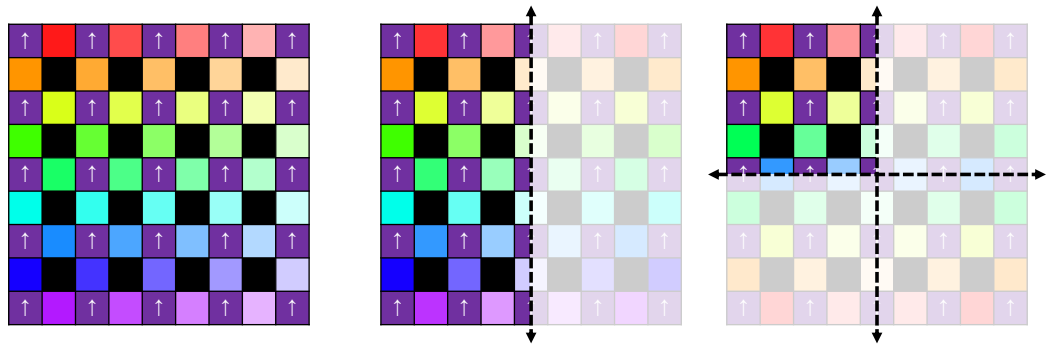
For example, elements at the edge of an array perform differently than those in the center due to the different electromagnetic environment. Therefore, enforcing edge sheets to have the same settings, but allow them to differ from interior sheets, is a reasonable restriction. Another option is to consider the relative location of sheets to their neighboring antennas. For a linearly polarized antenna, the near-field environment is different in the $E-$ and $H-$ planes. Therefore it is possible that in the optimal state, sheets located next to antennas in the $E-$ plane behave similarly to each other but differently than sheets located next to antennas in the $H-$ plane. It is also possible that all of these restrictions are applicable at the same time.

Conceptually, each set of restrictions and symmetries result in a mapping between switch settings and degrees of freedom. For example, the labeling of switches on each meta-gap sheet shown in Figure 9.1 is a mapping between the 960 switches in the array to the 24-bit number representing the state. This mapping is referred to as the *Identical Sheet* mapping and is the only one that has been explored thus far.

Figure 10.17 includes diagrams of seven mappings of different sizes based on the electromagnetic reasoning above. Each diagram is simplified representation of the array using a grid of squares. The dark purple squares with arrows represent the locations and polarizations of the array antennas. The black squares are the empty gaps between meta-gap sheets. The remaining squares are meta-gap sheets with colors representing their switch settings. Thus, squares with identical colors have the same[3] switch settings. For example, Figure 10.17g depicts the Identical Sheet mapping as each of the meta-gap sheets are the same color of orange. Figure 10.17a depicts the opposite extreme, the *Independent Sheet* mapping, in which each switch is independent as illustrated by the rainbow of colors.

The remaining five mappings each enforce at least one axis of symmetry, indicated by a dashed black line. Switches on either side of the dashed line are mirror images of each other which is further emphasized by partially fading the array. The switches of sheets that that lie along the axis of symmetry are also mirrored, reducing the number of independent switches per sheet from 24 to 14. As can be seen, the *E-Plane Symmetry* mapping shown in Figure 10.17b enforces symmetry across the $E-$plane axis while the *E&H-Plane Symmetry* mapping in Figure 10.17c enforces symmetry across both axes. The E-Plane Symmetry mapping has 488 degrees of freedom while the E&H-Plane Symmetry mapping has 248.

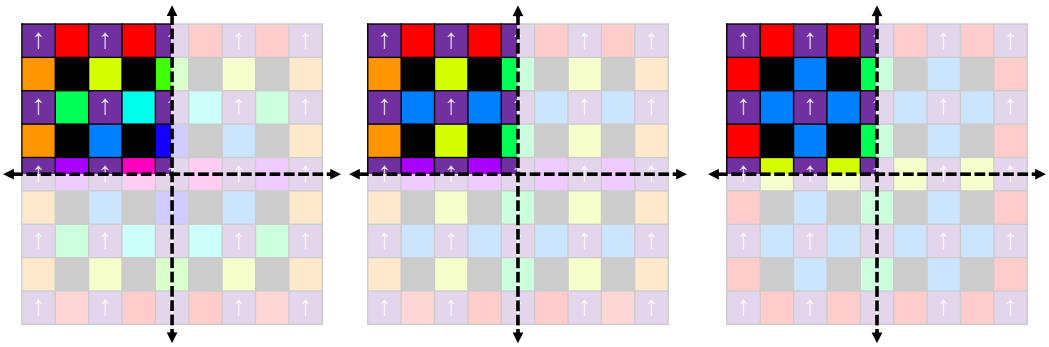
[3]Or mirrored.



(a) Independent Sheets: 960 bits

(b) E-Plane Symmetry: 488 bits

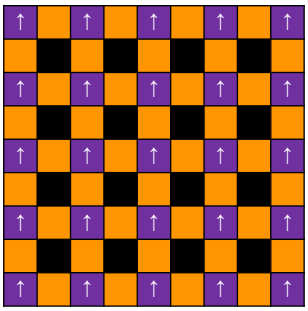
(c) E&H-Plane Symmetry: 248 bits



(d) Identical Edges: 200 bits

(e) E&H Near-Field: 124 bits

(f) Interior vs. Edge: 76 bits



(g) Identical Sheets: 24 bits

Figure 10.17: Diagram of mappings used to reduce degrees of freedom. Purple squares with arrows indicate location and polarization of tiles. Black squares are empty gaps. Dashed lines indicate lines of enforced symmetry. Identical colors indicate identical, although potentially mirrored, switch settings.

In addition to $E-$ and $H-$plane symmetry, Figures 10.17d, 10.17e, and 10.17f all introduce additional restrictions based on the local electromagnetic environment. The *Identical Edge* mapping ensures that sheets along the edge of the array are identical in the $E-$ and $H-$ planes, while the internal sheets are allowed to vary, resulting in 200 degrees of freedom. The *E&H Near-Field* mapping further reduces the degrees of freedom to 124 by ensuring that sheets along $E-$ or $H-$ planes in the near-field of an antenna have the same switch settings. Finally, the *Interior vs. Edge* mapping only allows variation between the interior[4] and edge sheets, thus only having 76 degrees for freedom.

In should be noted that, except for the Identical Sheet mapping, the more restrictive mappings are subspaces of the less restrictive mappings. Therefore, an optimal state in the Interior vs. Edge mapping is also a state in the E-Plane Symmetry mapping. The Identical Sheet mapping would be a subset of all of the mappings if it enforced symmetry along the $E-$ and $H-$ planes. However, the optimal identical sheet states shown in Figures 10.14, 10.15, and 10.16, and especially the heatmaps, are nearly symmetric across both axis. Therefore, it is plausible that similar symmetric states would have increased performance. Such states would be located in a subspace of every other mapping.

10.6 Electromagnetic Symmetry Experiments

Given the restrictions described in Section 10.5, we are left with the problem of assessing which restrictions are applicable and what mapping enables high performing states to be quickly identified. These questions are addressed experimentally by comparing the performance of random search, genetic optimization, and simulated annealing when optimizing the main beam power on each of the proposed mappings. As another test, the optimal main beam power states identified in the identical sheet experiments are used as initial seeds for variable neighborhood search on the different mappings. By comparing the results of this different experiments, the optimal mapping is identified.

Random Search Results

Figure 10.18 shows optimization curves when using a random search to maximize the main beam power under different mappings. As can clearly be seen, the mappings with more degrees of freedom perform those that are more restricted. In addition,

[4]Interior elements along the lines of symmetry are allowed to vary so that interior elements do not have to be internally symmetric about $E-$ and $H-$ plane axes.

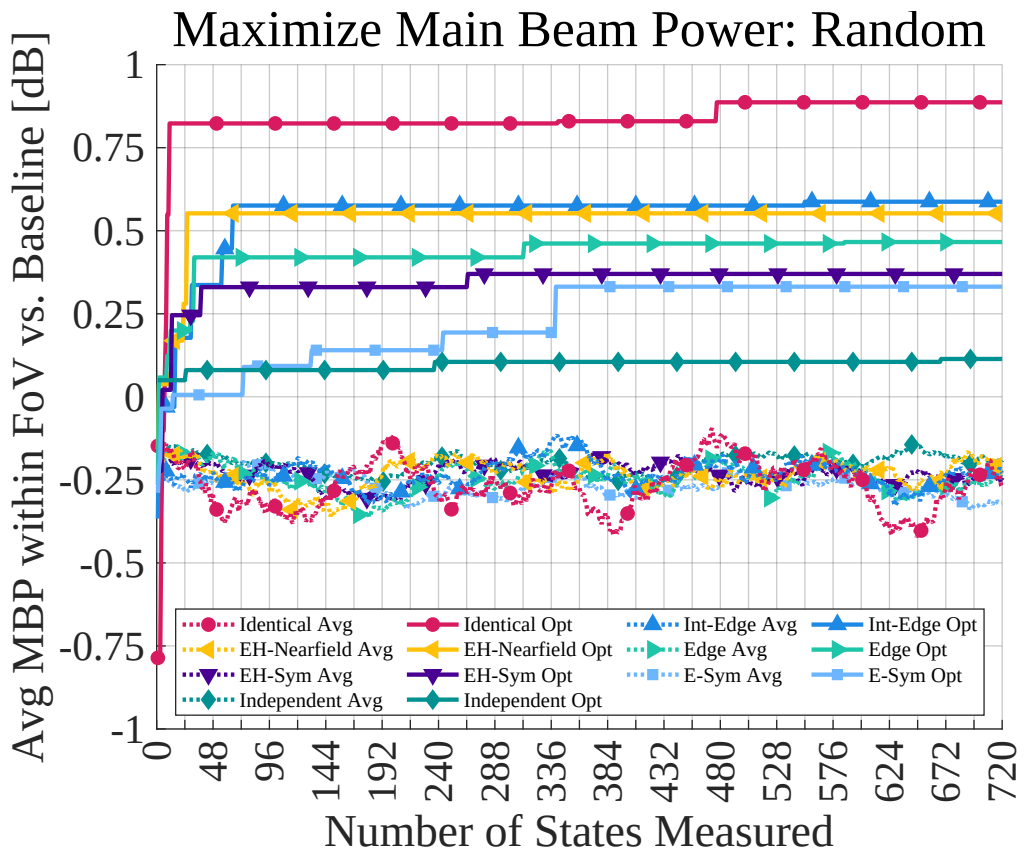


Figure 10.18: Optimization curves for the different mappings when maximizing the main beam power using random search.

the average value of the explored states appears to be approximately the same for different mappings.

Random search is also used to get a statistical sample of the state-space for the different mappings. Unfortunately, the sample size is not large enough to generate an accurate representation of the underlying distribution. However, Table 10.1 list the mean and standard deviation of the explored state values for the different mappings. As can be seen, the average state value for each mapping is approximately -0.24 dB, with the independent mapping performing the best with an average of -0.2 dB and the identical sheet mapping performing the worst with an average of -0.27 dB. However, the different mappings exhibit different variances; there is a clear trend of lower variance with higher degrees of freedom. This matches the trend of worse performance with lower degrees of freedom as the performance of random search is dependent on the variance of the state-space.

Considering the different mappings, these results are unsurprising. Even with en-

Table 10.1: Statistical Measures of State Performance over 720 States for Different Mappings

Mapping	μ **[dB]**	σ **[dB]**
Identical	-0.27	0.44
Interior vs. Edge	-0.23	0.27
E&H Near-Field	-0.24	0.29
Identical Edge	-0.24	0.25
E&H-Plane Symmetry	-0.23	0.22
E-Plane Symmetry	-0.27	0.17
Independent	-0.20	0.13

forced symmetry, a random set of switches is more likely to harm array performance than enhance it. While it is possible that the different mappings improve the average performance, such a different was small enough that it could not be identified with the size of random samples. The higher variability of more restricted mappings also matches intuition; fewer degrees of freedom mean that more switches must change between states, increasing the variability on array performance between states.

Genetic Optimization Results

Genetic optimization was the best performing algorithm when optimizing the main beam power in both the statistical experiments in Chapter 9 and the identical sheet experiments in Section 10.4. Therefore, it is expected that genetic optimization will identify high performing states in each of the proposed mappings. Thus the genetic optimization results should help identify which mapping provides optimal results in a limited time-frame.

The optimization curves when using genetic optimization to maximize the main beam power under different mappings are shown in Figure 10.19. Unlike the random search results, the identical sheet mapping does not give the best results. The genetic algorithm identifies a better performing state under both the interior vs. edge mapping and the E&H near-field mapping, the former performing the best.

In addition, both the optimal and average curves for the two mappings do not appear to have plateaued before the algorithm terminated. Therefore, even better mappings might have been identified if they ran for longer. This is in stark contrast with the optimization curve for the identical sheet mapping which stalls after approximately 12 batches. There are multiple explanations for this difference in convergence. Because the mutation rate *per bit* is the same for all runs mappings with a larger number of bits will mutate more per batch, thus reducing the rate of convergence.

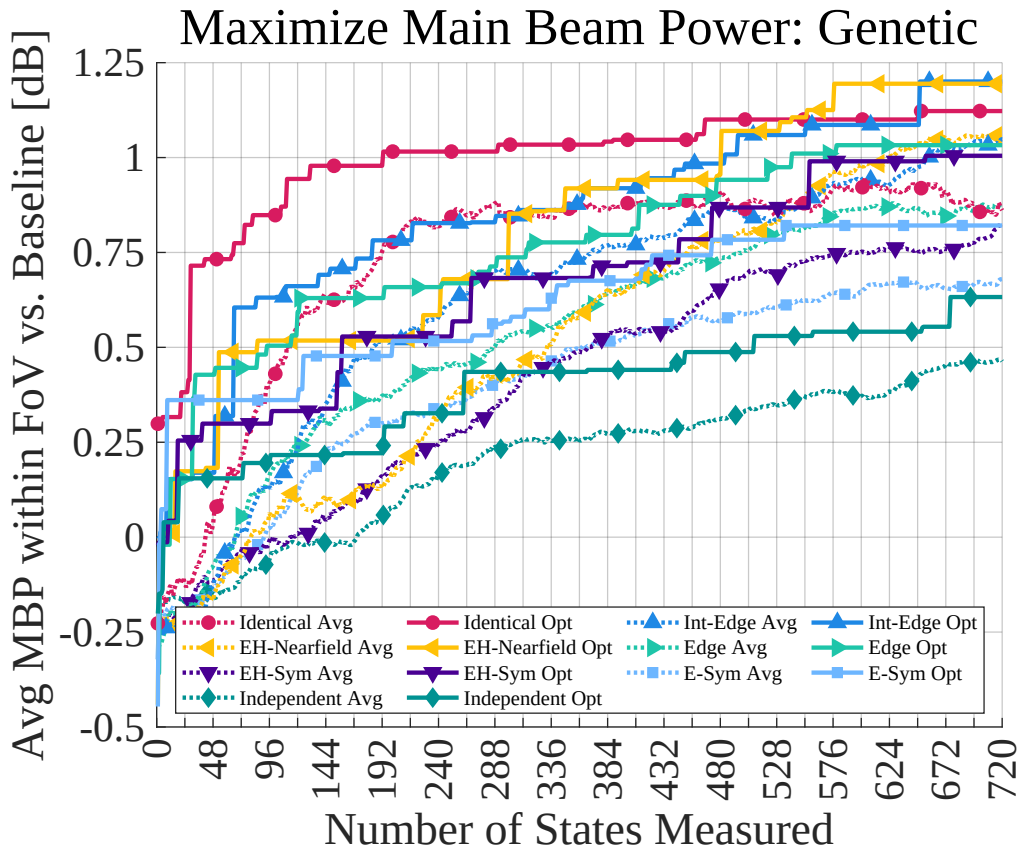


Figure 10.19: Optimization curves for the different mappings when maximizing the main beam power using genetic optimization.

The larger state size also means that there are more "genes" that need to be sorted through. The identical sheet average state value initially increases quite quickly as lower performing genes are eliminated.

The mappings with even higher degrees of freedom perform worse than the identical sheet mapping. While these algorithms also do not appear to have fully converged, their rate of improvement is much slower. While these mappings must logically identify a state at least as good as the ones identified by the interior vs. edge and the E&H near-field mappings, they are unable to do so within a reasonable amount of time. Thus these mappings are not ideal for time-constrained experiments or applications.

Simulated Annealing Results

The statistical experiments in Chapter 9 indicated that while genetic optimization is high performing on average, it also is high variance. It is thus possible that the single run results reported in Figure 10.19 are anomalous; the conclusion that

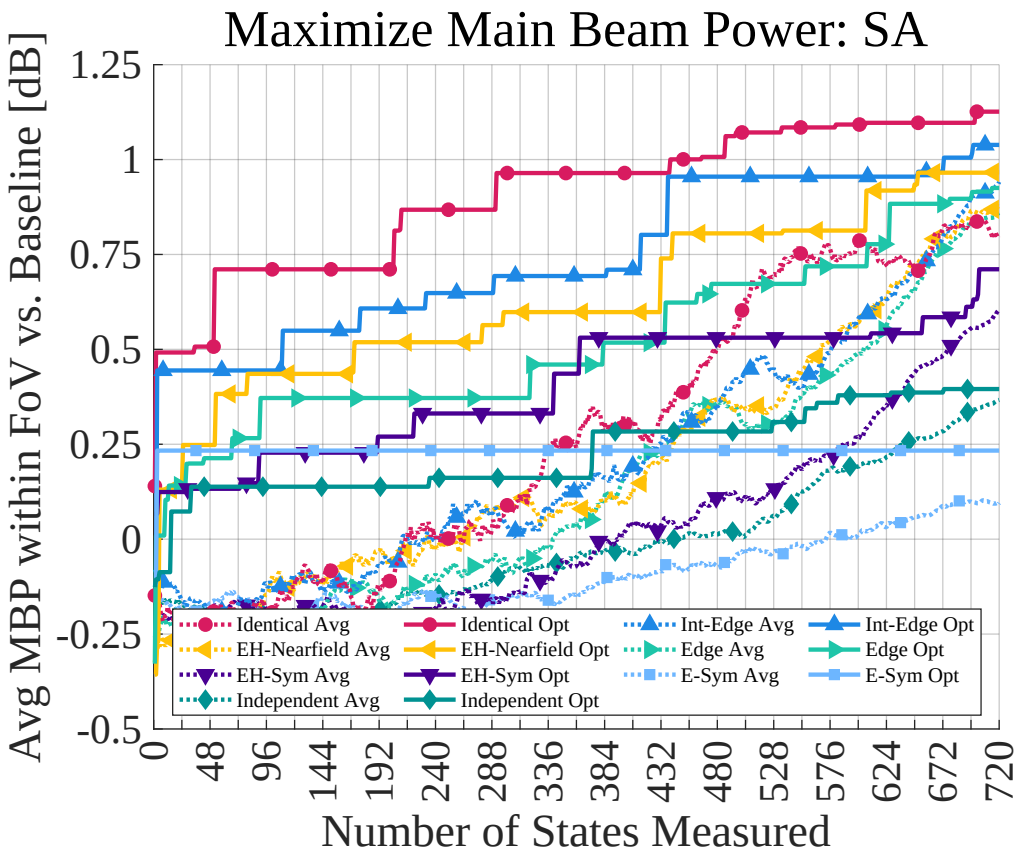


Figure 10.20: Optimization curves for the different mappings when maximizing the main beam power using simulated annealing.

the Interior vs. Edge mapping is better than the E&H Near-Field mapping would be incorrect if the former result was significantly above average and the latter was significantly below average. To reduce the risk of incorrect conclusions due to statistical uncertainty, simulated annealing is also used to explore the mappings; the algorithm identifies good solutions with lower variance than most of the other directed algorithms.

Figure 10.20 shows the simulated annealing optimization results for different mappings. Unlike genetic optimization, none of the mappings resulted in a better performing state than the identical sheet mapping. However the general order of the mappings' performance is the same[5], indicating that the the interior vs. edge and E&H near-field mappings are indeed able to identify better solutions than the others.

[5]The only exception being a particularly poor performance by the E-Plane Symmetry mapping.

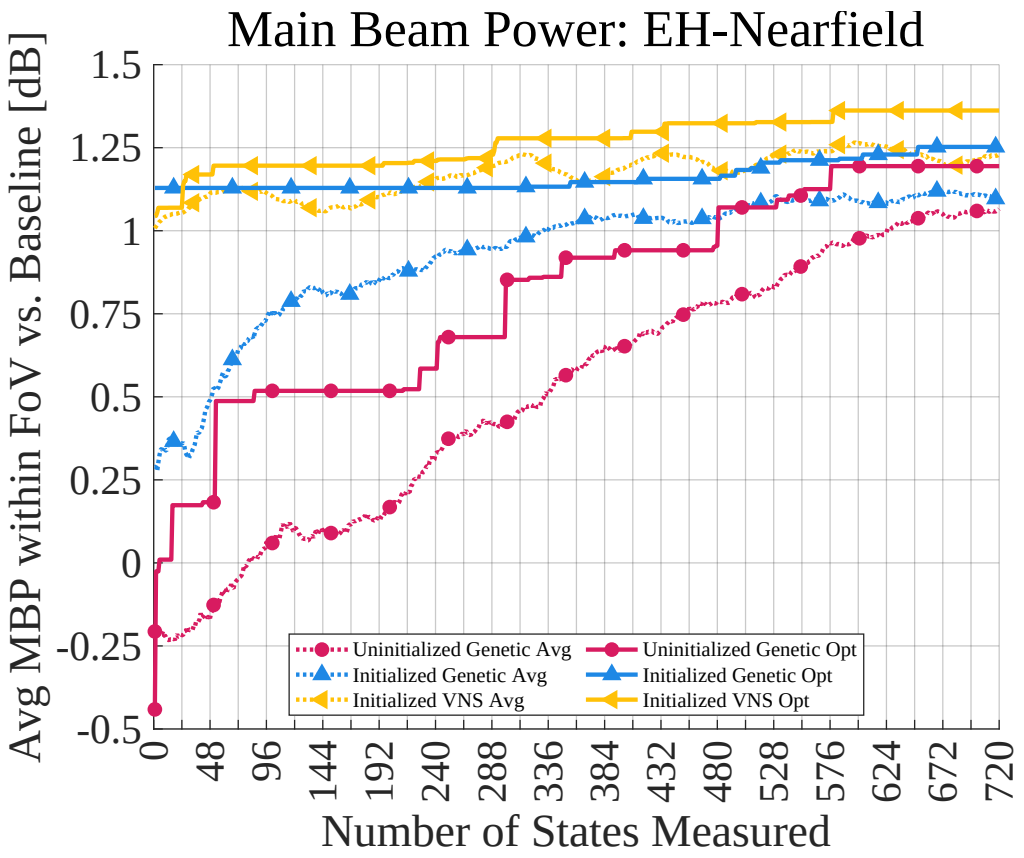


Figure 10.21: Effect of initialization on optimization. Optimization curves of the main beam power subject to the E&H near-field mapping using the VNS and genetic optimization algorithms initialized with the optimal states identified by the identical sheet experiments in Section 10.4. The optimization curve of the uninitialized genetic optimization is included for comparison.

Seeded Variable Neighborhood Search Results

All of the experiments thus far have assumed no knowledge of the problem prior to launching the algorithm. Because it is not clear what regions of the state-space or individual states might be high performing, the initial state(s) are selected at random so the algorithm is not biased towards one part of the search space. However, the identical sheet experiments in Section 10.4 have identified particular states that perform well. Therefore, it may be advantageous to bias the search by initializing algorithms with the states shown in Figure 10.14. As discussed is Section 10.5 while the identical sheet mapping is not strictly a sub-space of the other mappings, the optimal states identified are fairly symmetric about the $E-$ and $H-$ planes. Therefore, it is likely that "symmetricized" versions of the states will exhibit similar performance.

One way to conceptualize the effect of initialization and the different mappings is to think about the optimization problem as a discrete numeric search over a continuous function. This underlying function remains the same when changing the mapping, however, the coarseness of the discritization changes. Thus a search on a low degree of freedom mapping can quickly explore the full space, but will miss fine details, as each step covers a large distance. On the other hand, a search on a high degree of freedom mapping can accurately characterize the fine details of the underlying function at the cost of requiring a large number of steps to explore the full space.

With this conceptualization it is clear that the identical sheet mapping provides a very course exploration of the underlying function, identifying well performing regions but completely ignoring finer details. Therefore by initializing algorithms on higher degree of freedom mappings with specific states, they are able to increase the fineness of their search in high performing regions and likely identify even better solutions. However, the same coarseness trade-off exists in this more localized region; increasing the fineness too much and it is impossible to adequately explore the region about the initialized state. By comparing the relative performance of the initialized algorithms on different mappings, the optimal increase in degrees of freedom over the identical sheet mapping can be identified.

Initialization has a different effect on the various algorithms. Initialization is meaningless for random search as it simply explores random states. Similarly, initialization has no meaning in this work's implementation of simulated annealing as the algorithm is essentially random at high temperatures. However by reducing the initial temperature, the algorithm can be biased towards that region. This is equivalent to partially increasing the temperature after the an initial run converges and allowing the algorithm to settle a second time. In particle swarm, the particles can be initialized to start at well known positions, also biasing their search to a local region. For genetic optimization, the population is initialized with genes that are known to perform well. However, other genes must be introduced to population as well to enable further refinement. Finally, initializing variable neighborhood search simply starts the search where a previous one finished, essentially extending the algorithm's run-time. It is clear that initialization will effect algorithms differently, thus potentially changing their relative performance.

It is logical to start the initialed exploration with genetic optimization because it has exhibited the best performance thus far. Figure 10.21 shows the optimization curves for both initialized (blue) and uninitialized (red) genetic optimizations on the

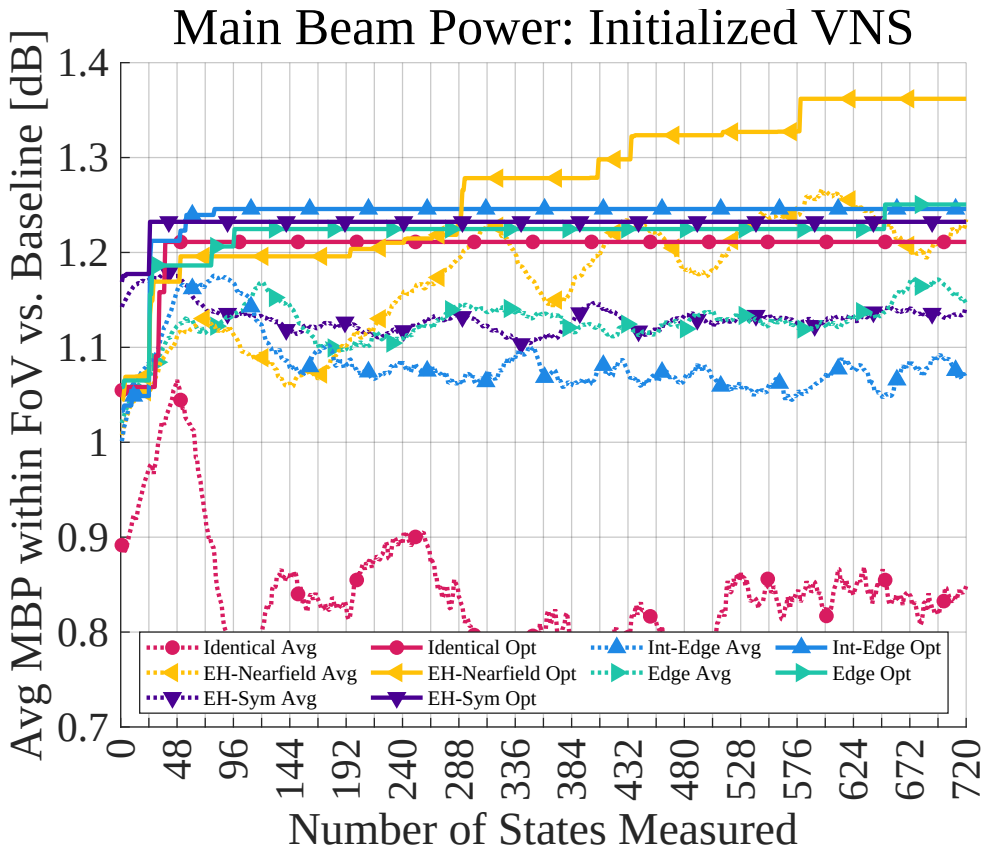


Figure 10.22: Optimization curves for the different mappings when maximizing the main beam power using VNS initialized with the optimal states identified by the identical sheet experiments in Section 10.4.

E&H Near-Field mapping. To initialize the genetic optimization, the five optimal performing states shown in Figure 10.14 are included in the initial population. The remaining 19 states are generated by randomly changing up to half of the bits in these five states. Thus the initial population is weighted towards the optimal states yet contains genetic variation.

As can clearly be seen, the initialized genetic algorithm only performs slightly better than the uninitialized version. For the first half of the run, the performance of the average state increases rapidly while optimal state remains constant. Thus the algorithm appears to spend a lot of resources pruning poor performing genes introduced in the initialization process. In addition, the algorithm only makes modest improvements even after this pruning process.

Figure 10.21 also includes the optimization curve for an initialized variable neighborhood search (yellow) on the E&H Near-Field mapping. In contrast to genetic optimization, VNS makes rapid and sustained improvement. This efficient perfor-

mance makes sense because of the algorithm's local first approach. If the advantage of initialization is the ability to increase the fineness of the search and localize it in a high-performing subspace, than a local approach is best suited to exploit this boon.

Given the synergistic relationship of initialization and VNS, the pair are used to explore the different mappings in order to identify the optimal mapping. Figure 10.22 shows the optimization curves for variable neighborhood search maximizing the main beam power on various mappings when initialized with a "symmetricized" version of the optimal identical sheet state. As can clearly be seen, the E&H Near-Field mapping resulted in significantly better performance than any other mapping with a 1.36 dB improvement over the baseline, or 0.46 dB improvement over a ground plane backed array. The remarkable performance of this mapping suggests that it could be an outlier and that on average the mapping performance is not quite as high. However, it is unlikely that the performance is substantially worse.

The general trend of the results match that of the genetic optimization; adding degrees of freedom improves performance initially before it starts to degrade. Due to the clearness of this trend, the E-Plane Symmetry and Independent Sheet mappings were not explored. However, the identical sheet mapping was also initialized and explored because part of the advantage of initialization for VNS is that it effectively expands the duration of the search. Therefore, enhancements greater than this second optimization of identical sheet state are likely due to the change in mapping and not the extended run-time. All mappings explored perform better than this second run.

Conclusion

The four experiments above examine the behavior of mappings in different manners. Simulated annealing's decent average, low-variance behaviour compliments the good average, high-variance behavior of genetic optimization and the poor average, low-variance behavior of random search. The initialized variable neighborhood search experiments examined the local characteristics of the mappings near optimal points. Taken together the four provide evidence for a more robust conclusion, especially because they are generally in agreement.

All of the experiments demonstrate that at some point, additional degrees of freedom reduce performance. This degraded performance is likely due to the sheer size of search spaces; while better states exist, they are near impossible to find in a limited amount of time. The genetic optimization and initialized VNS experiments however

indicate that small increases in the number of degrees of freedom can result in better performance. In the genetic optimization experiment, the Interior vs. Edge and E&H Near-Field mappings resulted in nearly identical performance. The initialized VNS experiment clearly indicated that the E&H Near-Field mapping is optimal. These results are not contradicted by the random search and simulated annealing experiments as these two mappings were the second and third best performing in both.

In three of the tests, the Interior vs. Edge and E&H Near-Field mappings had similar performance with the former slightly outperforming the latter. However, the E&H Near-Field mapping performed significantly better in the initialized VNS experiment. While this result may have been an outlier, it is the strongest differentiator between the two mappings in any of the experiments. Thus while it is likely that both mappings offer performance improvements, the E&H Near-Field mapping is the most promising given the data.

10.7 Symmetric E&H Near-Field Experiments

In the final set of experiments, the E&H Near-Field mapping is thoroughly explored by optimizing the main beam power, the side lobe levels, and the field of view using each of the five optimization algorithms. The experiments in Section 10.6 demonstrated that better performing states can be found using the E&H Near-Field mapping than with the identical sheet mapping. However it is unclear whether the mapping fundamentally changes the underlying optimization problem in addition to the characteristics of the optimal states. Thus a deep characterization of the optimization problem is repeated with the new mapping. As with the experiments in Section 10.4, the algorithms are not initialized in order to emulate the practical scenario where little is known about the array in advance.

Main Beam Power Results

Figure 10.23 shows the algorithms' performance over time when maximizing the main beam power. While genetic optimization still performs the best, VNS outperforms simulated annealing and particle swarm unlike in the identical sheet experiments. The optimal state identified in the E&H Near-Field mapping increases the average main beam power within the view of view by 1.2 dB compared to the all off baseline, a 0.07 dB improvement over the identical sheet mapping. This corresponds with a 0.3 dB enhancement compared to the ground plane backed array.

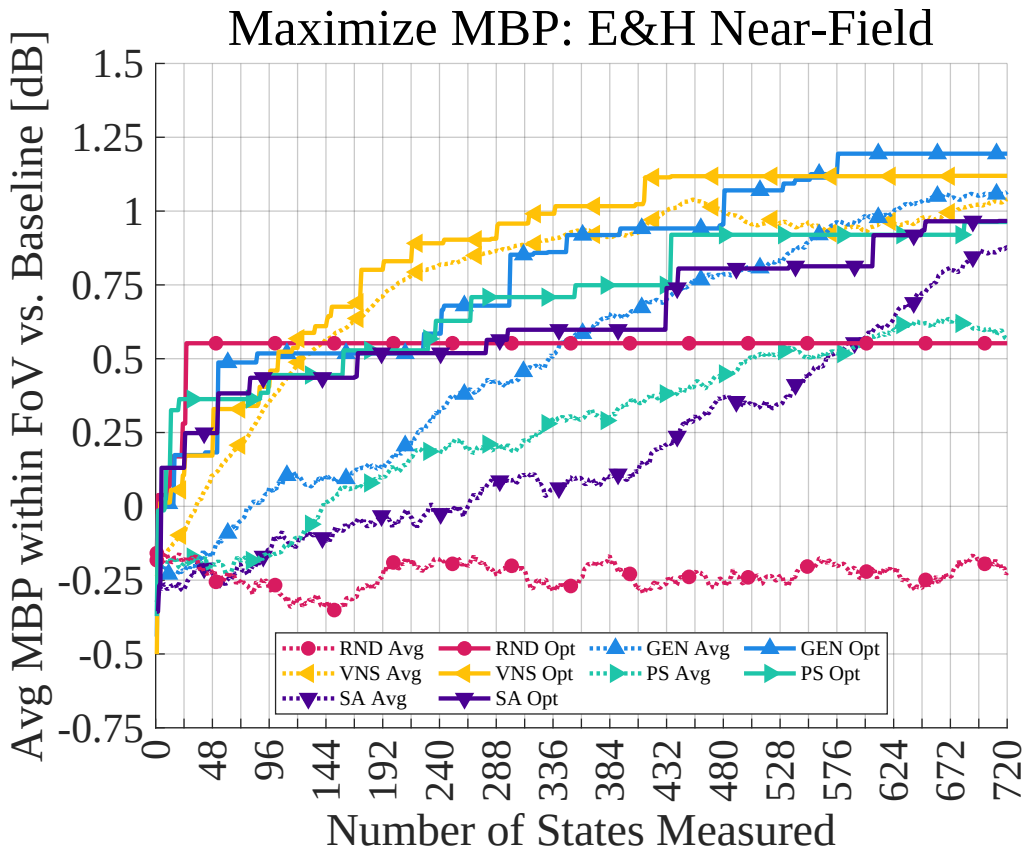
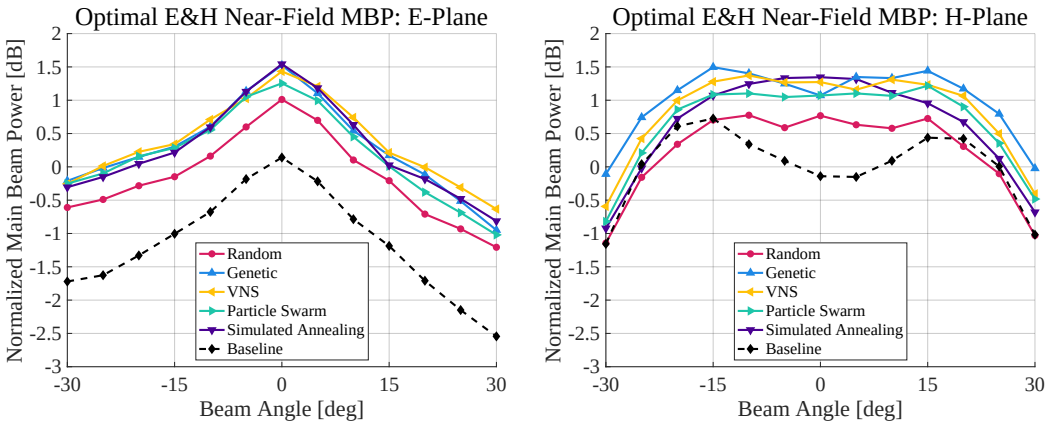


Figure 10.23: Optimization curves for the different algorithms when maximizing the main beam power subject to the E&H near-field mapping.



(a) $E-$plane cut (b) $H-$plane cut

Figure 10.24: Measured main beam power of the optimal states identified by each algorithm when to the E&H near-field mapping in addition to the "all off" baseline. Power is normalized to the broadside power of the "all off" baseline.

The increased difficulty of the search is demonstrated by the optimal state identified by the random search; the optimal E&H Near-Field state performs 0.34 dB worse than the optimal identical sheet state. In general, the average values of explored states behave similarly in both mappings. However, both genetic optimization and VNS take longer to converge. Indeed VNS does not begin to oscillate until over halfway through the run, indicating that it takes a long time to identify a local maxima.

Figure 10.24 shows the array characteristic of the optimal state identified by each algorithm. The results are similar to that in the identical mapping except that they appear to be more symmetric. The optimal improvement in broadside main beam power over the baseline is similar to that in the ground plane backed array.

Side Lobe Level Results

Figure 10.25 shows the algorithms' performance over time when maximizing the side lobe levels. As with the main beam power optimization, VNS and genetic optimization outperform simulated annealing and particle swarm. However, the difference between the optimal state identified in simulated annealing and particle swarm is larger. While genetic again performs the best, the optimal state it identifies performs exactly the same as the one identified with the identical sheet mapping, suppressing the side lobes by 1.7 dB on average throughout the field of view compared to the baseline. The general behavior of the average states explored are also similar to that in the identical sheet matching. However, like in the main beam power optimizations, both VNS and genetic optimization take longer to converge and VNS does not exhibit as many oscillations.

The optimal array characteristics shown in Figure 10.26 are also quite similar to those identified by the identical sheet optimization. Again the improvements are concentrated in the $H-$plane with the $E-$ plane being mostly unaffected. While the broadside side lobe levels in Figure 10.26b is approximately 1 dB *higher* than those in Figure 10.11b, the baseline levels are also 1 dB higher, suggesting that change is a function of some other aspect of the measurement than the meta-gaps.

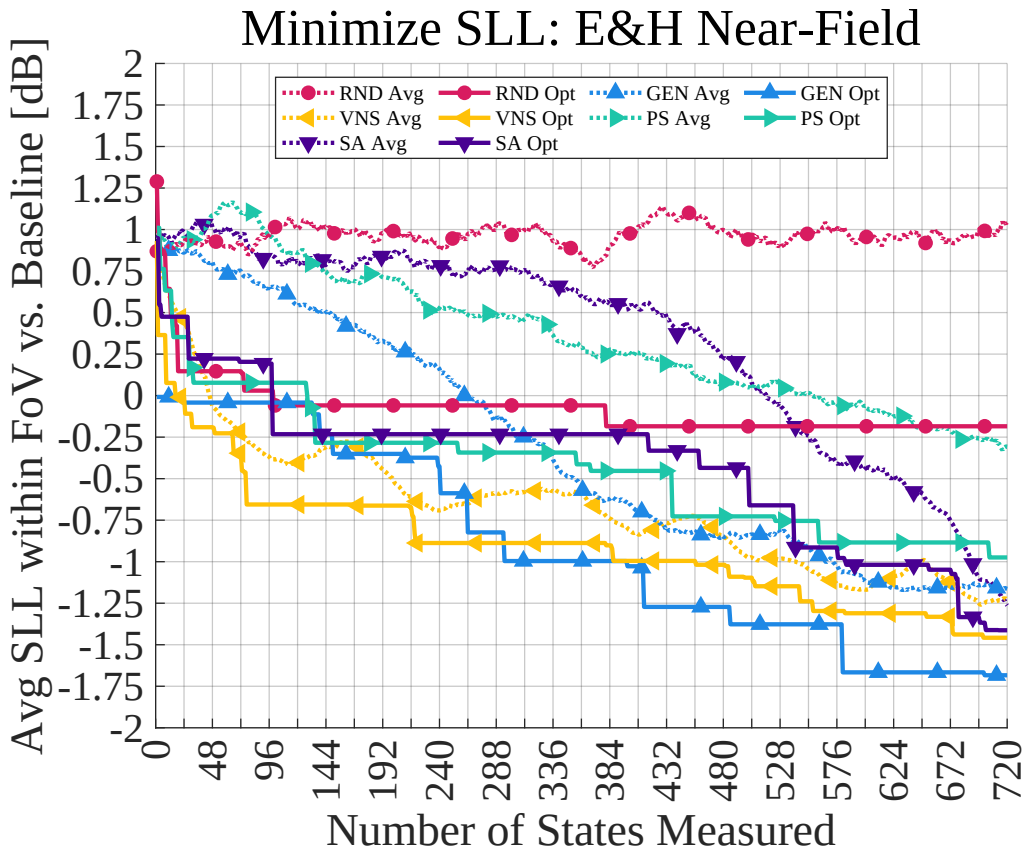
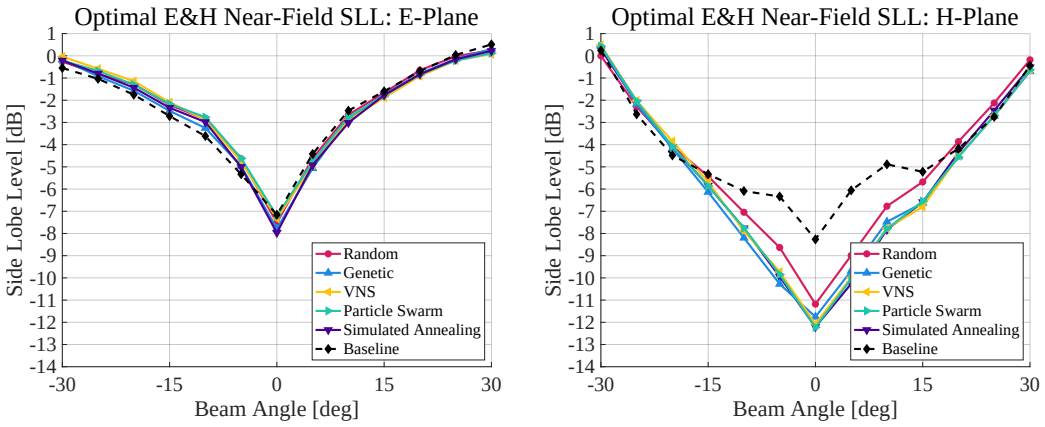


Figure 10.25: Optimization curves for the different algorithms when minimizing the side lobe levels subject to the E&H near-field mapping.



(a) $E-$plane cut. (b) $H-$plane cut.

Figure 10.26: Measured side lobe levels of the optimal states identified by each algorithm when subject to the E&H near-field mapping in addition to the "all off" baseline.

Field of View Results

Figure 10.27 shows the algorithms' performance over time when maximizing the field of view. The performance of the algorithms when optimizing the Field of View under the E&H Near-Field mapping is very different than under the identical sheet mapping. Whereas in the E&H Near-Field mapping, where each algorithm made similar improvements over the random search, in the E&H Near-Field mapping only VNS made substantial improvements over the random search. In fact, simulated annealing identified a *worse* solution than the random search. The optimal field of view identified by VNS in the E&H Near-Field mapping, 76.8°, is actually worse that the optimal field of view identified by random search in the identical sheet experiment. Thus, increasing the degrees of freedom strictly resulted in worse performance for field of view optimization.

The algorithm's convergence rates and average state behavior provide insight into the cause of this worse performance. The convergence rate of the genetic optimization and simulated annealing are similar to that of the random search, quickly identifying a decent performing state and only making slight improvements over time. Only particle swarm appears to follow its more typical gradual performance increase over time. The convergence of VNS is interesting as it makes essentially all of its improvements in one batch. The performance of the average state explored shows similar results, with particle swarm and simulated annealing making no average improvement over time, genetic optimization making only marginal improvement, and VNS only increasing for around two batches. These convergence rates and average state performance indicate that each of the algorithms, except VNS, essentially behave like a random search. VNS on the other hand, essentially searches randomly until it stumbles upon a narrow local maxima that it quickly optimizes.

These results suggest that the search space is relatively flat with small sparse local optima. It is likely that the worse performance by simulated annealing, and the better performance of VNS, were one-off events as the former failed to identify a good state and the latter fortunately found a local optima. The average statistical performance of the different algorithms is most likely fairly consistent. Assuming that the E&H Near-Field mapping does not alter the general form of the state distribution from that shown in Figure 9.11c, it is likely that the larger degrees of freedom greatly increased the number of states in the narrow central region without similarly increasing the number of states in the enhanced region; significantly more low-performing states were added than high-performing states, making the latter more difficult to find.

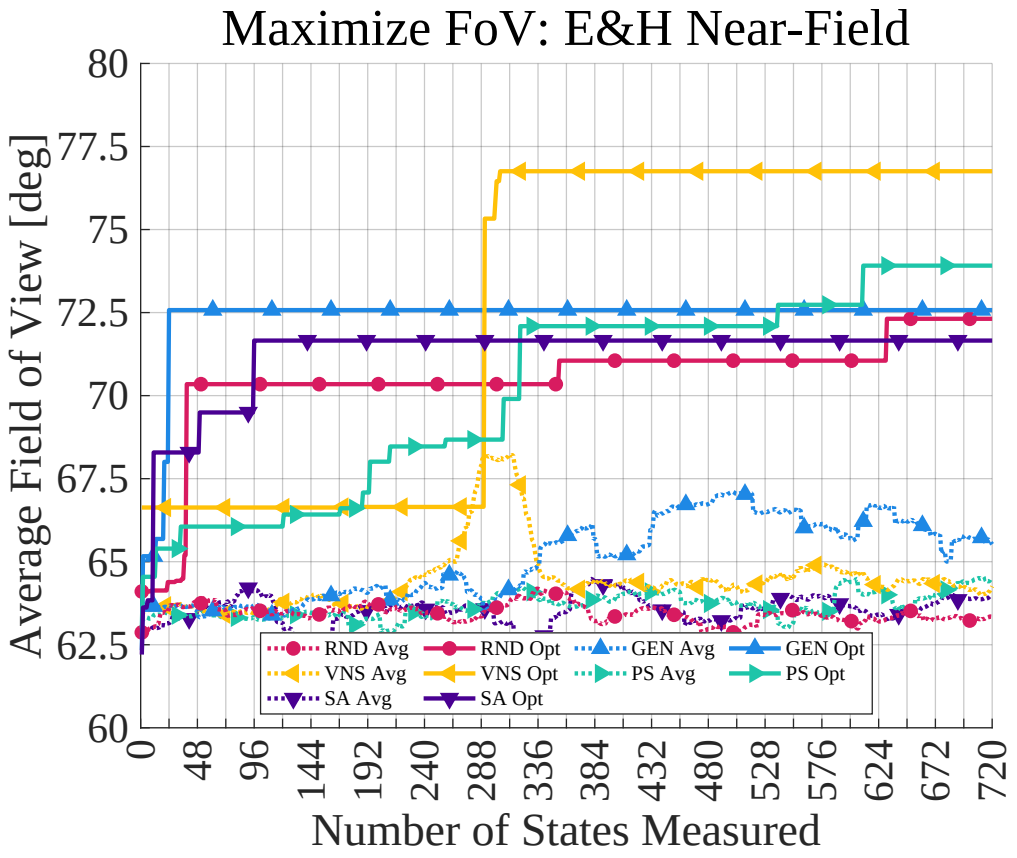
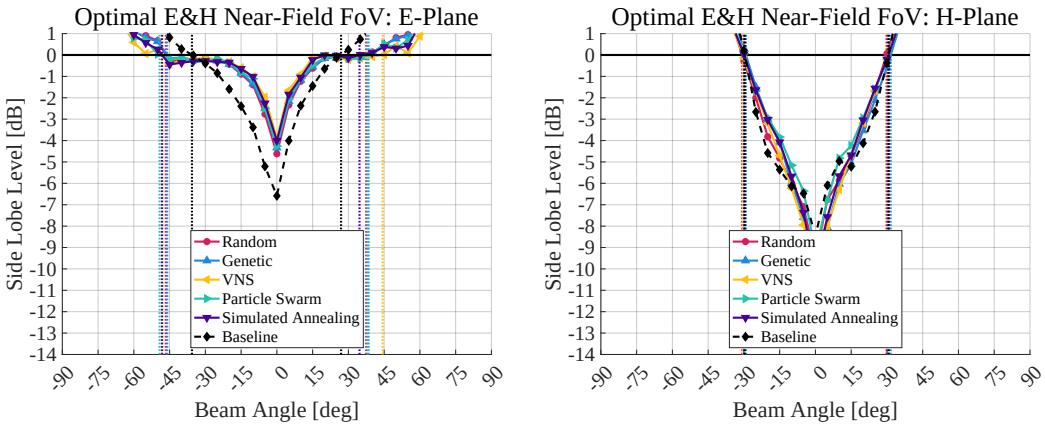


Figure 10.27: Optimization curves for the different algorithms when maximizing the field of view subject to the E&H near-field mapping.



(a) $E-$plane cut.

(b) $H-$plane cut.

Figure 10.28: Measured field of views of the optimal states identified by each algorithm when subject to the E&H near-field mapping in addition to the "all off" baseline.

Despite the difference in algorithm performance, the characteristic plots shown in Figure 10.28 resemble those of the identical sheet experiments. Again the $H-$ plane is mostly unaffected while the $E-$ plane contains almost all of the field of view improvement. Thus, the mechanism by which the field of view is enhanced remains the same; it just is not as efficiently exploited as in the identical sheet mapping.

10.8 Analysis

The experiments in Chapter 9 and Sections 10.4, 10.6, and 10.7 demonstrated the capabilities of meta-gaps and highlighted interesting properties about the associated optimization problem. From a performance perspective, meta-gaps are able to increase the average main beam power within the field of view by 0.46 dB and decrease the average side lobe levels within the field of view by 2 dB, compared to a ground plane backed array. The field of view itself can be increased by 23.5° compared to the theoretical 60° field of view for a $\lambda-$spaced array. These measured performance improvements represent a *minimum bound* on the possible achievable improvements as it is unlikely any of the optimal states identified are the true global optima. Given that only around 43,000 of the 2^{960} different states were characterized, it is likely that even greater performance is possible.

However, another conclusion is that finding optimal solutions is very difficult due to the vastness of the search space. All of the experiments demonstrate that at some point, additional degrees of freedom reduce the performance of the state that can be found within a limited time-frame. Indeed, the identical sheet mapping produced just as good, if not better, results as the E&H Near-Field mapping despite it being a fairly arbitrary and sub-optimal restriction.

Structure of the Optimization Problem

The experiments give insight into the underlying structure of the optimization problem. An important result is that the main beam power and side lobe levels are closely related, with algorithms behaving similarly on both problems; the similarity of the simulated statistical distributions is shown in Figures 9.11a and 9.11b. Thus, it is likely that algorithms or approaches that work well on one of the problems will work well on the other.

The relative performance of the different algorithms on the identical sheet experiments suggests that the MBP/SLL problem has many sparsely-spaced local minima in a relatively gradual global basin. VNS takes a local approach, performing a greedy search and then expanding its search space once it reaches a local optimum.

Because VNS quickly identifies a decent solution and then plateaus, it is likely that local minima are relatively shallow and far apart. SA and PS have the opposite strategy, covering a wide range of spaces initially before converging locally. These algorithms both gradually approach an optimum, indicating that there is some sort of global basin. The existence of this gradual basin is supported by the fact that a random search is able to quickly identify a decent solution, despite random switch settings degrading performance on average.

The relative performance of the algorithms on the E&H near-field mapping supports this idea. In this mapping, VNS takes longer to converge but results in better solutions than particle swarm or simulated annealing. As the fineness of the search increases, it takes longer for local optimal to be explored and it becomes more difficult to search the global space. Thus, particle swarm and simulated annealing are unable to either cover as much space or converge on a local optima in their final states as they can in a course search. VNS excels, as it dedicates more of its resources to exploring the same local optima with a finer precision.

The improved performance of the genetic algorithm over the other approaches in both mappings suggests that MBP/SLL problem solutions are comprised of features with intrinsic value. This matches intuition about the electromagnetic problem. Certain patterns of conductors, like closed loops or lines of a specific length, will have a pronounced effect on the electromagnetic fields. Randomly turning switches on, in general, should decrease performance as switches absorb power. Changing a small number of switches can make some improvement as the state resembles a particular pattern, but many switches have to change to switch between patterns. It seems likely that there exists a smaller set of basis solutions that could greatly reduce the search space.

Another conclusion is that the field of view optimization problem is quite different than the MBP/SLL problem. The FoV problem is much flatter with most states having little effect, and only pockets of minor improvement. This is demonstrated by the very gradual average-state enhancement in the identical sheet mapping and the almost non-existent average-state enhancement in the E&H near-field mapping. In fact, the problem under the E&H near-field mapping does not contain enough global variation for the algorithms to take advantage of, as their convergence behaviors resemble that of random search. Only VNS is able to make improvement because it ascends to a local maxima. In addition, the genetic optimization does not offer substantial improvement over the other methods, and the optimal identical sheet

states shown in Figure 10.16 do not appear to contain any shared building blocks or symmetry. Therefore, the field of view optimization problem is quite unlike the MBP/SLL problem and likely does not contain a set of basis solutions.

Possibility of Enhanced Algorithmic Performance

The possible existence of a set of basis states for the MBP/SLL optimization problems shows much promise for reducing the optimization time, and thus enhancing the meta-gap performance. While the binary switching optimization problem is NP-Hard *in general*, particular constraints can reduce the problem complexity, potentially making it convex[6]. If a set basis can be identified, then genetic optimization can become significantly more efficient, as it can be initialized with only the features that are required to produce high-quality solutions. In addition, depending on the size of the basis set, it could be explored efficiently through brute force. Depending on the properties of the basis set, it could be possible to design the meta-gaps to *only* contain basis states, reducing the number of switches and their parasitic effect on the radiated fields. Finally, just the existence of some sort of basis set suggests that machine learning could be used to efficiently optimize meta-gaps without specifically identifying the exact basis states.

There are other techniques beyond the basis set that could be used to increase meta-gap performance. Initialization proved to be an effective was to increase algorithm and meta-gap performance. It is thus possible that some sort of hybrid method that combines genetic optimization on a more restricted space with VNS on a broader space could increase performance. The fact that genetic optimization converged quickly in the identical sheet experiments suggests that the full 720 states are not necessary to find an appropriate initialization for VNS, further enhancing the speed of the combined approach.

Alternatively, the experiments in Chapter 9 demonstrated that simulated optimization is significantly more efficient than *in-situ* optimization. And while there are certainly errors introduced by imprecise modeling, it is unlikely that these errors drastically change the optimization problem. Therefore it is possible that optimal states could be identified in simulation and then used to initialize the *in-situ* optimization. This combined simulated-realized approach would combine the accuracy of the *in-situ* optimization with the speed enhancements of simulation.

[6]This is easy to see by imagining trying to numerically optimize a convex function by flipping bits in the binary representation of the argument on and off. The problem is clearly convex, despite there being an exponential number of choices

Finally, the experiments revealed that the random search converges rapidly and can quickly identify states with at least minor improvements over the ground plane backed array. Therefore, a straightforward and short random optimization could be useful in time-constrained applications.

If the optimization time can be enhanced, then significantly improved meta-gap performance is likely possible if states are optimized *for every beam*. While the single states explored in this work must improve array performance *on average*, separate states for each beam would allow asymmetry that directly harms beams pointed in a different directions. This asymmetry helps reduce grating lobes and increase the field of view as the array becomes *fundamentally different* for each beam direction, so the main beam power in one direction can be enhanced without increasing the side lobe levels in another. Despite its potential benefits, the current long optimization times of a single state precludes such an approach[7].

E- and H- Plane Asymmetry

One interesting property of the optimal field of view and side lobe characteristics is that their improvements are concentrated a single plane. The field of view optimization increases the side lobe levels in the $E-$plane at small beam angles in order to suppress them at larger angles, but the $H-$plane side lobe levels are generally unaffected. For the side lobe level optimization, only the $H-$plane side lobes are substantially reduced. These results indicate that the meta-gaps interact differently with side lobes in the two planes.

This asymmetry makes sense from an electromagnetic standpoint as the excitation fields are in different orientations in the two planes for a linearly polarized patch antenna. In a patch antenna, the magnetic fields circle the patch while most electric fields couple from either end to the ground plane below. Some of the electric fields curl from one end of the patch to the other, mostly above the patch surface, but also parallel to the patch to a lesser degree. Thus, the electric and magnetic fields in the near-field are weaker in the $E-$plane than in the $H-$plane and have different orientations. In the $H-$plane, the electric and magnetic fields are aligned to readily excite current along the polarization direction, introducing secondary fields that can strongly alter the radiated pattern. It is therefore unsurprising that the side lobes are easily suppressed in the $H-$ plane and not in the $E-$ plane, given the linearly

[7]While optimizations could be performed in parallel, the array would have to be reprogrammed with each measurement. As state programming is a time intensive operation, total optimization time would increase drastically

polarized excitation. However, it is less clear why the field of view optimization results are concentrated in the $E-$plane given the stronger near-field coupling in the $H-$plane.

Finally, an important note is that these results do not indicate that it is *impossible* to suppress the side lobes in the $E-$plane or extend the steering range in the $H-$plane. Rather, they only indicate that it is more efficient to focus the changes in the $H-$ and $E-$planes, respectively, in order to improve the average steering range or field of view.

Chapter 11

CONCLUSION

Time isn't holding up.
Time isn't after us.
Same as it ever was.
Same as it ever was.

Talking Heads
Once in a Lifetime

In this thesis, I set out to answer the questions posed by shape-changing phased arrays. Chapter 2 demonstrated the complexity of characterizing a single phased arrays and introduced a methodology to enhance the measurement of multiple arrays. This measurement technique is critical for characterizing the novel degrees of freedom introduced by shape-change and meta-gaps as they fundamentally alter array performance. Together Chapters 3 and 4 provided a strong theoretical basis for the trade-off between maximum gain and steering range that can be broken by shape-changing phased arrays. Chapter 5 presented both the benefits and drawbacks of shape-change, showing that the spacing between elements in a shape-changing array must change in order to allow the array to morph between geometries with different Gaussian curvatures.

Chapter 6 detailed the design of an array architecture based on independent radiating tiles. The flexibility of these tiles enables rigid shape-change by allowing the surface area to increase. Then the world's first shape-changing phased array was characterized in Chapter 7. The array was shown to able to increase its average -3 dB steering range from 88° to 138° at the cost of a 6 dB loss in broadside gain.

To combat the effect of increased spacing, the concept of a meta-gap is introduced in Chapter 8. These re-programmable structures are a solution to the difficult electromagnetic deign problem of optimizing far-field performance using passive parasitic structures under varying excitations. As these structures present an optimization problem that is, in abstract, an NP-Hard problem, Chapters 9 and 10 explored the optimization problem using meta-hueristic algorithms on both simulated and physical meta-gap structures. The experiments concluded that there is likely a set of

underlying configurations that form some sort of a basis which could reduce the complexity of the optimization problem. The *in-situ* experiments demonstrated that meta-gaps can *at least* improve the average main beam power over the steering range by 0.46 dB, suppress the average side lobe levels over the steering range by 2 dB, and increase the field of view itself by 23.5° when compared to a ground plane.

However, this work is just the tip of the proverbial shape-changing iceberg as numerous important questions remain. Having established that the trade-off between gain and steering range can be broken, what else is possible with array geometry? Is it possible to steer beams or form arbitrary radiation patterns using shape-change alone? Can the nulls of elements be dynamically rearranged to increase the suppression of jammers?

Meta-gaps offer their own world of mysteries to explore. What is the full scope of the optimization problem? Can desirable states be decomposed into a set of basis states that can be recombined? Does a decent polynomial time heuristic exist given the particulars of the electromagnetic problem? If decent solutions can be identified rapidly enough, how much performance enhancement can be gained by allowing the meta-gap states to change with the steering angle? Can meta-gap performance be improved by switching to slower, but lower loss, switches like MEMS? Alternatively, how much can non-programmable fixed metal patterns increase performance by removing switching losses?

Unfortunately, all things must come to an end, so these questions will have to be addressed in another thesis or collection of papers. It is my hope that this work will, at the very least, inspire interest in shape-changing phased arrays and the possibilities of using geometry as a design variable. I believe that these ideas are the future of wireless networks as they remove the final limits on what surface current patterns can be synthesized.

Appendix A

PROJECTION ANALYSIS JACOBIAN CALCULATION

The equivalence of equations 3.29 and 3.30 requires multiple algebraic steps that are included below for completeness:

We begin with the definitions given in equations 3.26, 3.27, and 3.25.

$$\langle x, y, z\rangle = \vec{\sigma}(u, v) \tag{A.1}$$

$$\vec{A} \equiv \langle \cos(\phi)\cos(\theta), \sin(\phi)\cos(\theta), -\sin(\theta)\rangle \tag{A.2}$$

$$\vec{B} \equiv \langle -\sin(\phi), \cos(\phi), 0\rangle\,. \tag{A.3}$$

We will notionally refer to the $\vec{i}$,$\vec{j}$, and $\vec{k}$ components of these vectors using superscripts, such as $B^i = -\sin(\phi)$, $B^j = \cos(\phi)$, and $B^k = 0$. Solving for the determinant in Equation 3.30:

$$|J(\Phi)| = \left|\begin{bmatrix} \vec{A}\cdot\vec{\sigma}_u & \vec{A}\cdot\vec{\sigma}_v \\ \vec{B}\cdot\vec{\sigma}_u & \vec{B}\cdot\vec{\sigma}_v \end{bmatrix}\right| \tag{A.4}$$

$$|J(\Phi)| = \left[\left(\vec{A}\cdot\vec{\sigma}_u\right)\left(\vec{B}\cdot\vec{\sigma}_v\right)\right] - \left[\left(\vec{A}\cdot\vec{\sigma}_v\right)\left(\vec{B}\cdot\vec{\sigma}_u\right)\right]. \tag{A.5}$$

Plugging in $\vec{A}$ and $\vec{B}$ gives:

$$\begin{aligned}|J(\Phi)| = &\left(A^i\sigma_u^i + A^j\sigma_u^j + A^k\sigma_u^k\right)\left(B^i\sigma_v^i + B^j\sigma_v^j + B^k\sigma_v^k\right) \\ &- \left(A^i\sigma_v^i + A^j\sigma_v^j + A^k\sigma_v^k\right)\left(B^i\sigma_u^i + B^j\sigma_u^j + B^k\sigma_u^k\right).\end{aligned} \tag{A.6}$$

Recognizing that $B^k = 0$:

$$\begin{aligned}|J(\Phi)| = &\left(A^i\sigma_u^i + A^j\sigma_u^j + A^k\sigma_u^k\right)\left(B^i\sigma_v^i + B^j\sigma_v^j\right) \\ &- \left(A^i\sigma_v^i + A^j\sigma_v^j + A^k\sigma_v^k\right)\left(B^i\sigma_u^i + B^j\sigma_u^j\right)\end{aligned} \tag{A.7}$$

$$\begin{aligned}|J(\Phi)| = &\Big(A^iB^i\sigma_u^i\sigma_v^i + A^jB^i\sigma_u^j\sigma_v^i + A^kB^i\sigma_u^k\sigma_v^i \\ &+ A^iB^j\sigma_u^i\sigma_v^j + A^jB^j\sigma_u^j\sigma_v^j + A^kB^j\sigma_u^k\sigma_v^j\Big) \\ &- \Big(A^iB^i\sigma_v^i\sigma_u^i + A^jB^i\sigma_u^i\sigma_v^j + A^kB^i\sigma_v^k\sigma_u^i \\ &+ A^iB^j\sigma_v^i\sigma_u^j + A^jB^j\sigma_v^j\sigma_u^j + A^kB^j\sigma_v^k\sigma_u^j\Big)\end{aligned} \tag{A.8}$$

$$|J(\Phi)| = \Big(A^i B^i - A^i B^i\Big)\sigma_u^i \sigma_v^i + \Big(A^j B^i - A^i B^j\Big)\sigma_u^j \sigma_v^i + \Big(A^i B^j - A^j B^i\Big)\sigma_u^i \sigma_v^j + \Big(A^j B^j - A^j B^j\Big)\sigma_u^j \sigma_v^j + A^k B^i \sigma_u^k \sigma_v^i + A^k B^j \sigma_u^k \sigma_v^j - A^k B^i \sigma_u^i \sigma_v^k - A^k B^j \sigma_u^j \sigma_v^k \quad \text{(A.9)}$$

$$|J(\Phi)| = \Big(A^j B^i - A^i B^j\Big)\sigma_u^j \sigma_v^i + \Big(A^i B^j - A^j B^i\Big)\sigma_u^i \sigma_v^j + A^k B^i \sigma_u^k \sigma_v^i + A^k B^j \sigma_u^k \sigma_v^j - A^k B^i \sigma_u^i \sigma_v^k - A^k B^j \sigma_u^j \sigma_v^k \quad \text{(A.10)}$$

Plugging in the coefficient values gives:

$$|J(\Phi)| = \Big(-\sin^2(\phi)\cos(\theta) - \cos^2(\phi)\cos(\theta)\Big)\sigma_u^j \sigma_v^i + \Big(\cos^2(\phi)\cos(\theta) + \sin^2(\phi)\cos(\theta)\Big)\sigma_u^i \sigma_v^j + \sin(\theta)\sin(\phi)\sigma_u^k \sigma_v^i - \sin(\theta)\cos(\phi)\sigma_u^k \sigma_v^j - \sin(\theta)\sin(\phi)\sigma_u^i \sigma_v^k + \sin(\theta)\cos(\phi)\sigma_u^j \sigma_v^k \quad \text{(A.11)}$$

$$|J(\Phi)| = \cos(\theta)\sigma_u^i \sigma_v^j - \cos(\theta)\sigma_u^j \sigma_v^i + \sin(\theta)\sin(\phi)\Big(\sigma_u^k \sigma_v^i - \sigma_u^i \sigma_v^k\Big) + \sin(\theta)\cos(\phi)\Big(\sigma_u^j \sigma_v^k - \sigma_u^k \sigma_v^j\Big) \quad \text{(A.12)}$$

$$|J(\Phi)| = \cos(\theta)\Big(\sigma_u^i \sigma_v^j - \sigma_u^j \sigma_v^i\Big) + \sin(\theta)\sin(\phi)\Big(\sigma_u^k \sigma_v^i - \sigma_u^i \sigma_v^k\Big) + \sin(\theta)\cos(\phi)\Big(\sigma_u^j \sigma_v^k - \sigma_u^k \sigma_v^j\Big) \quad \text{(A.13)}$$

$$|J(\Phi)| = \langle \sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)\rangle \cdot \langle \sigma_u^j \sigma_v^k - \sigma_u^k \sigma_v^j, \sigma_u^k \sigma_v^i - \sigma_u^i \sigma_v^k, \sigma_u^i \sigma_v^j - \sigma_u^j \sigma_v^i\rangle \quad \text{(A.14)}$$

$$|J(\Phi)| = \langle \sin(\theta)\cos(\phi), \sin(\theta)\sin(\phi), \cos(\theta)\rangle \cdot \Big(\langle \sigma_u^i, \sigma_u^j, \sigma_u^k\rangle \times \langle \sigma_v^i, \sigma_v^j, \sigma_v^k\rangle\Big) \quad \text{(A.15)}$$

$$|J(\Phi)| = (\vec{\sigma}_u \times \vec{\sigma}_v) \cdot \langle \cos(\phi)\sin(\theta), \sin(\phi)\sin(\theta), \cos(\theta)\rangle. \quad \text{(A.16)}$$

Equation A.16 is the same as Equation 3.30, thus showing the equivalence of Equations 3.29 and 3.30.

Appendix B

MAXIMUM GAIN OF BASIC GEOMETRIC SHAPES

This appendix contains derivations of the array properties discussed in Chapter 4 using the mathematical tools developed in Chapter 3. While in some cases, such as a sphere, it would be simpler to calculate the cross sectional area from geometric reasoning, the full general method is shown in order to demonstrate how it can be used for more complex geometries. This general method is also used for the numeric calculations in Chapter 4.

For the following calculations it is often simpler to rewrite Equation 3.37 in terms of the surface normal and assume an observer located at θ_o, ϕ_o in spherical coordinates.

$$A_{cross} = \iint_R \cos(\psi) \left\| \vec{N}_\sigma \right\| du dv \tag{B.1}$$

$$\vec{O}(\theta_o, \phi_o) = \langle \cos(\phi_o) \sin(\theta_o), \sin(\phi_o) \sin(\theta_o), \cos(\theta_o) \rangle \tag{B.2}$$

ψ is the angle between the observer and the normal vector on the surface. Therefore:

$$\cos(\psi) = \frac{\vec{N}_\sigma \cdot \vec{O}}{\left\| \vec{N}_\sigma \right\| \left\| \vec{O} \right\|}. \tag{B.3}$$

Because $\vec{O}$ is a unit vector

$$\cos(\psi) = \frac{\vec{N}_\sigma \cdot \vec{O}}{\left\| \vec{N}_\sigma \right\|}. \tag{B.4}$$

Plugging equation B.4 into equation B.1 gives:

$$A_{cross} = \iint_R \vec{N}_\sigma \cdot \vec{O} du dv. \tag{B.5}$$

Equation B.5 is a much more convenient form for calculating the cross sectional area from a surface patch $\sigma(u, v)$. In order to account for portions of area that are occluded or facing away from the observer, R is the region of the domain where the integrand is positive.

The approximate relationship between maximum aperture and gain described in Equation 2.14 is repeated below for convenience.

$$G_{max} = \frac{4\pi}{\lambda^2} A_{cross} \tag{B.6}$$

B.1 Relative Gain of Spherical Array

Parameterization

Let us parameterize the sphere using traditional spherical coordinates:

$$u = \theta, v = \phi \tag{B.7}$$

$$\vec{\sigma}(\theta, \phi) = \langle R\cos(\phi)\sin(\theta), R\sin(\phi)\sin(\theta), R\cos(\theta)\rangle. \tag{B.8}$$

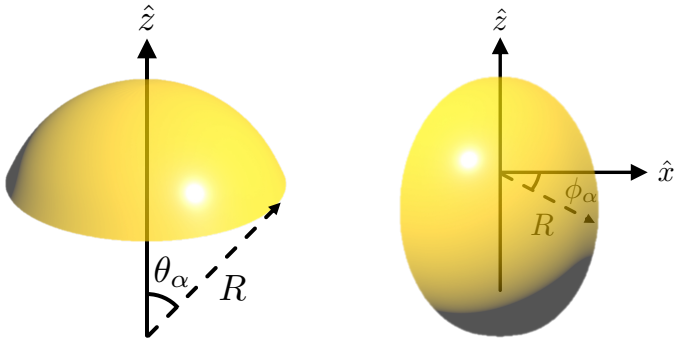
The surface tangent vectors are thus:

$$\vec{\sigma}_\theta = \langle R\cos(\phi)\cos(\theta), R\sin(\phi)\cos(\theta), -R\sin(\theta)\rangle \tag{B.9}$$

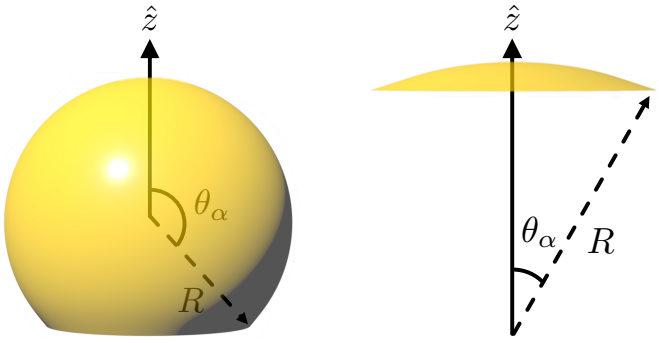
$$\vec{\sigma}_\phi = \langle -R\sin(\phi)\sin(\theta), R\cos(\phi)\sin(\theta), 0\rangle. \tag{B.10}$$

The surface normal vectors is thus:

$$\vec{N}_\sigma = \langle R^2\cos(\phi)\sin^2(\theta), R^2\sin(\phi)\sin^2(\theta), R^2\sin(\theta)\cos(\theta)\rangle. \tag{B.11}$$



(a) θ based parameterization (b) ϕ based parameterization



(c) Nearly Spherical (d) Nearly Planar

Figure B.1: Parameterization of a sphere. (a) Cap and (b) wedge parameterizations in order to normalize the surface area. θ_α is the minimum latitude of the spherical cap. Array is (c) nearly spherical for $\theta_\alpha \approx \pi$ and (d) effectively planar for $\theta_\alpha \approx 0$. These are equivalent to arrays with a small R and a large R, respectively.

As shown in Figure B.1a and B.1b, the domain of θ and ϕ determine the shape and extent of the array on the surface of the sphere. For example, if $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$ than the array covers the entire sphere. However, if $0 \leq \theta \leq \theta_\alpha$ and $0 \leq \phi \leq 2\pi$, than the array is a spherical cap. Similarly if $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq \beta$, than the array is a spherical wedge. For this analysis, we assume that the array is a spherical cap and adjust θ_α to normalize the surface area for different spherical radii as shown in Figures B.1c and B.1d.

Normalization of Area

The surface area of the spherical cap is:

$$A_{surface} = \int_0^{2\pi} \int_0^{\theta_\alpha} \left\|\vec{N}_\sigma\right\| d\theta d\phi. \tag{B.12}$$

Solving for the magnitude of the normal vector:

$$\left\|\vec{N}_\sigma\right\| = \sqrt{R^4 \cos^2(\phi) \sin^4(\theta) + R^4 \sin^2(\phi) \sin^4(\theta) + R^4 \sin^2(\theta) \cos^2(\theta)} \tag{B.13}$$

$$= R^2 \sqrt{\sin^4(\theta) + \sin^2(\theta) \cos^2(\theta)} \tag{B.14}$$

$$= R^2 sin(\theta) \sqrt{\sin^2(\theta) + \cos^2(\theta)} \tag{B.15}$$

$$= R^2 \sin(\theta). \tag{B.16}$$

Going back to the surface area integral

$$A_{surface} = \int_0^{2\pi} \int_0^{\theta_\alpha} R^2 \sin(\theta) d\theta d\phi \tag{B.17}$$

$$= R^2 \int_0^{2\pi} -cos(\theta) \Big|_0^{\theta_\alpha} d\phi \tag{B.18}$$

$$= R^2 \left(1 - \cos(\theta_\alpha)\right) \int_0^{2\pi} d\phi \tag{B.19}$$

$$A_{surface,sphere} = 2\pi R^2 \left(1 - \cos(\theta_\alpha)\right). \tag{B.20}$$

Thus we can calculate for the θ_α that normalizes the area to that of a planar array.

$$\cos(\theta_\alpha) = 1 - \frac{A_{plane}}{2\pi R^2} \tag{B.21}$$

Maximum Cross-Sectional Area

The maximum cross-sectional area of a spherical cap is when the observer is located along the z axis:

$$\vec{O} = \langle 0, 0, 1 \rangle. \tag{B.22}$$

The dot product between the surface normal and observer vector can be written as:

$$\vec{N}_\sigma \cdot \vec{O} = R^2 \cos(\theta) \sin(\theta) \tag{B.23}$$

$$\vec{N}_\sigma \cdot \vec{O} = \frac{R^2}{2} \sin(2\theta). \tag{B.24}$$

We can now solve for the cross sectional area given equation B.5. Note that the integrand is positive for all $\theta_\alpha \leq \frac{\pi}{2}$.

$$A_{cross,sphere} = \frac{R^2}{2} \int_0^{2\pi} \int_0^{\theta_\alpha} \sin(2\theta) d\theta d\phi \tag{B.25}$$

If $\theta_\alpha > \frac{\pi}{2}$ than the integration domain is limited to not include occluded area.

$$A_{cross,sphere} = \frac{R^2}{2} \int_0^{2\pi} \int_0^{\frac{\pi}{2}} \sin(2\theta) d\theta d\phi \tag{B.26}$$

Solving for the cross sectional area assuming $\theta_\alpha \leq \frac{\pi}{2}$ gives:

$$A_{cross,sphere} = \frac{R^2}{2} \int_0^{2\pi} \left[-\frac{1}{2} \cos(2\theta) \right] \Big|_0^{\theta_\alpha} d\phi \tag{B.27}$$

$$A_{cross,sphere} = \frac{R^2}{4} \left(1 - \cos(2\theta_\alpha)\right) \int_0^{2\pi} d\phi \tag{B.28}$$

$$A_{cross,sphere} = \frac{\pi R^2}{2} \left(1 - \cos(2\theta_\alpha)\right). \tag{B.29}$$

While if $\theta_\alpha > \frac{\pi}{2}$ than:

$$A_{cross,sphere} = \pi R^2. \tag{B.30}$$

Normalized Maximum Gain

Given Equations B.21 and B.29 we can calculate the cross sectional area of a spherical cap compared to a plane with the same surface area. We then can use this to compare the maximum gain of a spherical cap versus a planar array. Note that:

$$1 - \cos(2\theta_\alpha) = 1 - \left(2\cos^2(\theta_\alpha) - 1\right) = 2\left(1 - \cos^2(\theta_\alpha)\right). \tag{B.31}$$

Using this fact we can plug Equation B.21 into Equation B.29.

$$A_{cross,sphere} = \pi R^2 \left(1 - \left[1 - \frac{A_{plane}}{2\pi R^2}\right]^2\right) \tag{B.32}$$

$$A_{cross,sphere} = \pi R^2 \left(1 - \left[1 - 2\frac{A_{plane}}{2\pi R^2} + \frac{A_{plane}^2}{4\pi^2 R^4}\right]\right) \tag{B.33}$$

$$A_{cross,sphere} = \pi R^2 \left(\frac{A_{plane}}{\pi R^2} - \frac{A_{plane}^2}{4\pi^2 R^4}\right) \tag{B.34}$$

$$A_{cross,sphere} = A_{plane}\left[1 - \frac{A_{plane}}{4\pi R^2}\right]. \tag{B.35}$$

In the case where $\theta_\alpha > \frac{\pi}{2}$, the cross sectional area is fixed to πR^2 as the surface area above $2\pi R^2$ is occluded. To compare this cross sectional area to that of a plane with the same surface area, we will define χ to be the fraction of the spherical surface area that the array covers.

$$\chi = \frac{A_{plane}}{4\pi R^2} \tag{B.36}$$

We can thus solve for the radius of the sphere given χ.

$$R = \sqrt{\frac{A_{plane}}{4\pi\chi}} \tag{B.37}$$

Thus we can relate the cross sectional area to the fraction of area covered.

$$A_{cross,sphere} = \pi R^2 = \frac{A_{plane}}{4\chi} \tag{B.38}$$

Using χ we can create a piecewise expression for the cross sectional area:

$$A_{cross,sphere} = \begin{cases} (1-\chi)\, A_{plane} & 0 \le \chi \le 0.5 \\ \dfrac{1}{4\chi} A_{plane} & 0.5 \le \chi \le 1 \end{cases}. \tag{B.39}$$

Equations B.35 and B.39 quantify the visible array area reduction when a plane is mapped onto a sphere. For a large sphere $R^2 \gg A_{plane}$, the curvature is negligible and the cross-sectional area is unaffected. However $A_{cross,sphere}$ is cut in half if the array covers an entire hemisphere. For even larger arrays, the additional area is occluded as it falls on the opposite side and the cross-sectional area reduces to that of a sphere, πR^2. For a completely filled spherical array, $\chi = 1$ and $A_{cross,sphere} = \frac{A_{plane}}{4}$.

The maximum gain of the array mapped to the sphere is thus

$$G_{max,sphere} = G_{max,plane} - \left[\frac{A_{plane}}{\lambda R}\right]^2 \tag{B.40}$$

for spherical caps without occlusion and

$$G_{max,sphere} = \begin{cases} (1-\chi)\, G_{max,plane} & 0 \le \chi \le 0.5 \\ \dfrac{1}{4\chi} G_{max,plane} & 0.5 \le \chi \le 1 \end{cases} \tag{B.41}$$

in general.

B.2 Relative Gain of Cylindrical Array

Parameterization

For the purposes of this calculation, we only consider the curved portion of a cylindrical array and not the end-caps. We parameterize the array using standard cylindrical coordinates:

$$u = \phi, v = z \tag{B.42}$$

$$\vec{\sigma}(\phi, z) = \langle R\cos(\phi), R\sin(\phi), z\rangle. \tag{B.43}$$

The surface tangent vectors are thus:

$$\vec{\sigma}_\phi = \langle -R\sin(\phi), R\cos(\phi), 0\rangle \tag{B.44}$$

$$\vec{\sigma}_z = \langle 0, 0, 1\rangle. \tag{B.45}$$

The surface normal vector is thus:

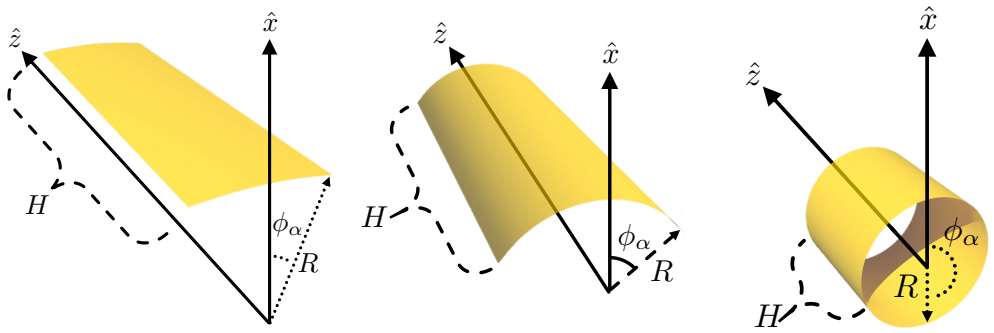
$$\vec{N}_\sigma = \langle R\cos(\phi), R\sin(\phi), 0\rangle. \tag{B.46}$$

As shown in Figure B.2, the domain of ϕ and z determine the shape and extent of the array on the surface of the cylinder. For example, if $0 \le \phi \le \pi$ than the array covers the full cylinder. The domain of z determines how long the array is along the cylinder. For the purpose of this analysis we will assume that $0 \le z \le H$ and that $-\phi_\alpha \le \phi \le \phi_\alpha$, with H determined by A_{plane} and ϕ_α.

Normalization of Area

The surface area of the cylindrical array is:

$$A_{surface} = \int_0^H \int_{-\phi_\alpha}^{\phi_\alpha} \left\|\vec{N}_\sigma\right\| d\phi dz. \tag{B.47}$$



(a) Nearly Planar (b) Partial Cylinder (c) Complete Cylinder

Figure B.2: Parameterization of a cylinder in order to normalize the surface area. H is the array length and ϕ_α is angular coverage of the cylinder. Array is (a) long and nearly planar for $\phi_\alpha \approx 0$, and (b) short and cylindrical for $\phi_\alpha \approx \pi$.

Solving for the magnitude of the normal vector:

$$\left\|\vec{N}_\sigma\right\| = \sqrt{R^2\cos^2(\phi) + R^2\sin^2(\phi)} = R. \tag{B.48}$$

Going back to the surface area integral

$$A_{surface} = \int_0^H \int_{-\phi_\alpha}^{\phi_\alpha} R d\phi dz \tag{B.49}$$

$$A_{surface} = 2\phi_\alpha R \int_0^H dz \tag{B.50}$$

$$A_{surface} = 2\phi_\alpha RH. \tag{B.51}$$

Thus we can calculate for the H that normalizes the area to that of a planar array.

$$H = \frac{A_{plane}}{2\phi_\alpha R} \tag{B.52}$$

Maximum Cross-Sectional Area

The maximum cross-sectional area of a cylindrical array as defined is when the observer is located along the x axis:

$$\vec{O} = \langle 1, 0, 0\rangle. \tag{B.53}$$

The dot product between the surface normal and observer vector can be written as:

$$\vec{N}_\sigma \cdot \vec{O} = R\cos(\phi). \tag{B.54}$$

We can now solve for the cross sectional area given equation B.5. Note that the integrand is positive for all $|\phi_\alpha| \leq \frac{\pi}{2}$.

$$A_{cross,cylinder} = \int_0^H \int_{-\phi_\alpha}^{\phi_\alpha} R\cos(\phi) d\phi dz \tag{B.55}$$

If $|\phi_\alpha| > \frac{\pi}{2}$ than the integration domain is limited to not include occluded area.

$$A_{cross,cylinder} = \int_0^H \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} R\cos(\phi) d\phi dz \tag{B.56}$$

Solving for the cross sectional area assuming $|\phi_\alpha| \leq \frac{\pi}{2}$ gives:

$$A_{cross,cylinder} = R\int_0^H \sin(\phi)\Big|_{-\phi_\alpha}^{\phi_\alpha} dz \tag{B.57}$$

$$A_{cross,cylinder} = R\left(\sin(\phi_\alpha) - \sin(-\phi_\alpha)\right)\int_0^H dz \tag{B.58}$$

$$A_{cross,cylinder} = 2R\sin(\phi_\alpha)H. \tag{B.59}$$

While if $\theta_\alpha > \frac{\pi}{2}$ than:

$$A_{cross,cylinder} = 2RH. \tag{B.60}$$

Normalized Maximum Gain

Given Equations B.52 and B.59, we can calculate the cross sectional area of a cylindrical array compared to a plane with the same surface area. We then can use this to compare the maximum gain of a cylindrical array versus a planar array. In the case where $|\phi_\alpha| \leq \frac{\pi}{2}$:

$$A_{cross,cylinder} = \frac{\sin(\phi_\alpha)}{\phi_\alpha} A_{plane} = \text{sinc}(\phi_\alpha) A_{plane}. \tag{B.61}$$

In the case where $\theta_\alpha > \frac{\pi}{2}$:

$$A_{cross,cylinder} = \frac{A_{plane}}{\phi_\alpha}. \tag{B.62}$$

We can thus create a piecewise expression for the cross sectional area:

$$A_{cross,cylinder} = \begin{cases} \sin(\phi_\alpha)\dfrac{A_{plane}}{\phi_\alpha} & 0 \leq |\phi_\alpha| \leq \frac{\pi}{2} \\ \dfrac{A_{plane}}{\phi_\alpha} & \frac{\pi}{2} \leq |\phi_\alpha| \leq \pi \end{cases}. \tag{B.63}$$

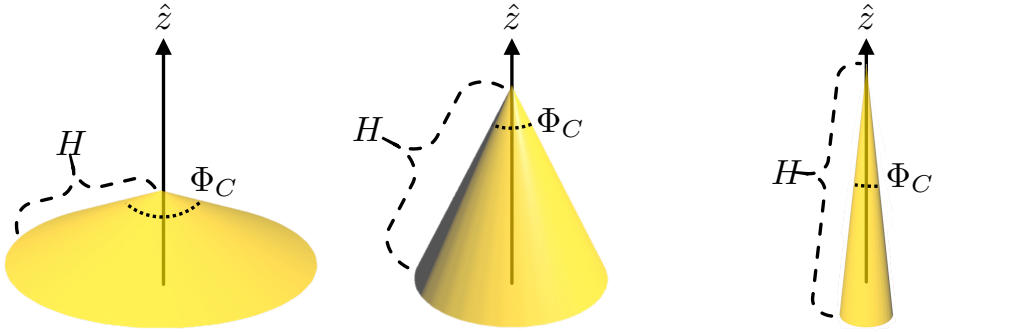
Equations B.61 and B.63 quantify the visible array area reduction when a plane is mapped onto a cylinder. For a cylinder with a large radius, $\phi_\alpha \approx 0$, the curvature is negligible and the cross-sectional area is unaffected. However, if the array is a half cylinder $A_{cross,sphere} = \frac{2}{\pi}$, a reduction of 36.4%. For a complete cylinder, $A_{cross,sphere} = \frac{1}{\pi}$ a reduction of 68.2%.

The maximum gain of the array mapped to the cylinder is thus

$$G_{max,cylinder} = \begin{cases} \text{sinc}(\phi_\alpha) G_{max,plane} & 0 \leq |\phi_\alpha| \leq \frac{\pi}{2} \\ \frac{1}{\phi_\alpha} G_{max,plane} & \frac{\pi}{2} \leq |\phi_\alpha| \leq \pi \end{cases}. \tag{B.64}$$

B.3 Relative Gain of Conic Array

Parameterization



(a) Nearly Planar (b) Circular Cone (c) Almost a vertical line

Figure B.3: Parameterization of a cone in order to normalize the surface area. H is the slant height and Φ_C is the apex angle. Array is (a) nearly planar for $\Phi_C \approx \pi$, and (b) tall and thin for $\Phi_C \approx 0$.

For the purposes of this calculation, we assume that the conic array is a complete right circular cone oriented around along the $z-$axis. Such a cone can be treated as a line revolved around the $z-$axis, here called the *slant line*, that goes through the origin. As shown in Figure B.3, we define the cone by the length of the slant line, H, and the angle formed at the apex of the cone, here called the *apex angle*, Φ_C. We parameterize the surface of the array by its location along the slant, h, and its angle with the $x-$axis, ϕ.

$$u = \phi, v = h \tag{B.65}$$

With this parameterization, the equations for the slant line are:

$$z = h \cos\left(\frac{\Phi_C}{2}\right) \tag{B.66}$$

$$x = h \sin\left(\frac{\Phi_C}{2}\right). \tag{B.67}$$

Thus the array is defined by:

$$\vec{\sigma}(\phi, h) = \langle h \sin\left(\frac{\Phi_C}{2}\right) \cos(\phi), h \sin\left(\frac{\Phi_C}{2}\right) \sin(\phi), h \cos\left(\frac{\Phi_C}{2}\right) \rangle. \tag{B.68}$$

The surface tangent vectors are thus:

$$\vec{\sigma}_\phi = \langle -h \sin\left(\frac{\Phi_C}{2}\right) \sin(\phi), h \sin\left(\frac{\Phi_C}{2}\right) \cos(\phi), 0 \rangle \tag{B.69}$$

$$\vec{\sigma}_h = \langle \sin\left(\frac{\Phi_C}{2}\right) \cos(\phi), \sin\left(\frac{\Phi_C}{2}\right) \sin(\phi), \cos\left(\frac{\Phi_C}{2}\right) \rangle. \tag{B.70}$$

The surface normal vector is thus:

$$\vec{N}_\sigma = \langle \frac{h}{2} \sin(\Phi_C) \cos(\phi), \frac{h}{2} \sin(\Phi_C) \sin(\phi), -h \sin^2\left(\frac{\Phi_C}{2}\right) \rangle. \tag{B.71}$$

As shown in Figure B.3, the domain of ϕ and z determine the shape and extent of the array on the surface of the cone. For example, if $0 \leq \phi \leq \pi$ than the array covers half of the cone. The domain of h determines how long the array is along the cone. For the purpose of this analysis we will assume that $0 \leq h \leq H$ and that $0 \leq \phi \leq 2\pi$, with H determined by A_{plane}. In other words, we assume that the array forms a complete cone of slant length H.

Normalization of Area

The surface area of the conic array is:

$$A_{surface} = \int_0^H \int_0^{2\pi} \left\| \vec{N}_\sigma \right\| d\phi dh. \tag{B.72}$$

Solving for the magnitude of the normal vector:

$$\left\| \vec{N}_\sigma \right\| = \sqrt{\frac{h^2}{4} \sin^2(\Phi_C) + h^2 \sin^4\left(\frac{\Phi_C}{2}\right)} \tag{B.73}$$

$$\left\| \vec{N}_\sigma \right\| = h \sqrt{\frac{1}{4} \sin^2(\Phi_C) + \frac{1}{4} (1 - \cos(\Phi_C))^2} \tag{B.74}$$

$$\left\| \vec{N}_\sigma \right\| = \frac{h}{2} \sqrt{\sin^2(\Phi_C) + 1 - 2\cos(\Phi_C) + \cos^2(\Phi_C)} \tag{B.75}$$

$$\left\| \vec{N}_\sigma \right\| = \frac{h}{2} \sqrt{2 - 2\cos(\Phi_C)} \tag{B.76}$$

$$\left\| \vec{N}_\sigma \right\| = \frac{h}{2} \sqrt{4 \sin^2\left(\frac{\Phi_C}{2}\right)} \tag{B.77}$$

$$\left\| \vec{N}_\sigma \right\| = h \sin\left(\frac{\Phi_C}{2}\right). \tag{B.78}$$

Going back to the surface area integral

$$A_{surface} = \int_0^H \int_0^{2\pi} h \sin\left(\frac{\Phi_C}{2}\right) d\phi dh \tag{B.79}$$

$$A_{surface} = 2\pi \sin\left(\frac{\Phi_C}{2}\right) \int_0^H h dh \tag{B.80}$$

$$A_{surface} = \pi H^2 \sin\left(\frac{\Phi_C}{2}\right). \tag{B.81}$$

Thus we can calculate for the H that normalizes the area to that of a planar array.

$$H = \sqrt{\frac{A_{plane}}{\pi \sin\left(\frac{\Phi_C}{2}\right)}} \tag{B.82}$$

Maximum Cross-Sectional Area

Here we assume that the maximum cross-sectional area of a conical array as defined is when the observer is located on the negative z axis:

$$\vec{O} = \langle 0, 0, -1 \rangle. \tag{B.83}$$

The dot product between the surface normal and observer vector can be written as: The surface normal vector is thus:

$$\vec{N}_\sigma \cdot \vec{O} = h \sin^2\left(\frac{\Phi_C}{2}\right). \tag{B.84}$$

We can now solve for the cross sectional area given equation B.5. Note that the integrand is positive over the entire domain.

$$A_{cross,cone} = \int_0^H \int_0^{2\pi} h \sin^2\left(\frac{\Phi_C}{2}\right) d\phi dh \tag{B.85}$$

$$A_{cross,cone} = 2\pi \sin^2\left(\frac{\Phi_C}{2}\right) \int_0^H h dh \tag{B.86}$$

$$A_{cross,cone} = \pi H^2 \sin^2\left(\frac{\Phi_C}{2}\right) \tag{B.87}$$

Normalized Maximum Gain

Given Equations B.82 and B.87 we can calculate the cross sectional area of a conic array compared to a plane with the same surface area. We then can use this to compare the maximum gain of a conic array versus a planar array.

$$A_{cross,cone} = \sin\left(\frac{\Phi_C}{2}\right) A_{plane} \tag{B.88}$$

Equations B.88 quantifies the visible array area reduction when a plane is mapped onto a cone. For steep cones with small apex angles, most of the surface area is oriented away from the observer and the cross sectional area is greatly reduced, possibly to zero. In this case, the direction of maximum gain might be oriented away from the top of the cone. This scenario is discussed in Section 4.5.

The maximum gain of the array mapped to the cone is thus:

$$G_{max,cone} = \sin\left(\frac{\Phi_C}{2}\right) G_{max,plane}. \tag{B.89}$$

Appendix C

EXPECTED VALUE OF RANDOM SEARCH

The behavior of random search can be understood as a stochastic process with predictable mean performance. We start by representing the histogram of state performance using the probability distribution function $f_X(x)$. The probability of selecting a state with value of at most v is represented by the cumulative distribution function $F_X(v)$.

$$P(X \leq v) = F_X(v) = \int_{\infty}^{v} f_X(x)dx \tag{C.1}$$

In order for the maximum value found to be at most v after selecting k states requires the value of every selected state to be at most v. Because states are independently selected at random, we know that:

$$P(X_k \leq v) = \prod^{k} P(X \leq v) = F_X(v)^k. \tag{C.2}$$

Equation C.2 is itself a cumulative distribution function

$$F_{X_k}(v) = F_X(v)^k \tag{C.3}$$

describing the probability that k selected states will have a maximum value of at most v. Therefore we can calculate the expected maximum value of a set of k selected states with Equation C.4.

$$E[X_k] = \int_{-\infty}^{\infty} x dF_{X_k}(x) \tag{C.4}$$

We can look to integration by parts for an understanding of how to solve this integral. Suppose that instead of infinite limits, we instead had finite limits 0 and $b < \infty$. In this case we could write:

$$\int_0^b x dF_{X_k}(x) = xF_{X_k}(x)\Big|_0^b - \int_0^b F_{X_k}(x)dx \tag{C.5}$$

$$\int_0^b x dF_{X_k}(x) = bF_{X_k}(b) - \int_0^b F_{X_k}(x)dx \tag{C.6}$$

$$\int_0^b x dF_{X_k}(x) = -b\left[1 - F_{X_k}(b)\right] + b - \int_0^b F_{X_k}(x)dx \tag{C.7}$$

$$\int_0^b x dF_{X_k}(x) = -b\left[1 - F_{X_k}(b)\right] + \int_0^b dx - \int_0^b F_{X_k}(x)dx \tag{C.8}$$

$$\int_0^b x dF_{X_k}(x) = -b\left[1 - F_{X_k}(b)\right] + \int_0^b 1 - F_{X_k}(x)dx. \tag{C.9}$$

Now we consider what happens to the convergence of these integrals as we allow $b \to \infty$. The integral on the left hand side is simply the integral of a product of x and a probability distribution function. By definition, the integral of a pdf is equal to 1. Therefore if $F_{X_k}(x)$ has finite support, than the left hand integral must converge. The integral in the right hand side is also bounded because $1 - F_{X_k}(x)$ is bounded between 0 and 1 and also has finite support. Finally, by definition we have $F_{X_k}(\infty) = 1$, making the product on the right hand side equal to $\infty \times 0$, which is in general undefined. However, due to the convergence of the other terms in the expression we can safely conclude that

$$\int_0^\infty x dF_{X_k}(x) = \int_0^\infty \left[1 - F_{X_k}(x)\right] dx \tag{C.10}$$

in the sense that if one side converges than so does the other [1]. We can examine an integral with a infinite negative bound in a similar way:

$$\int_a^0 x dF_{X_k}(x) = xF_{X_k}(x)\Big|_a^0 - \int_a^0 F_{X_k}(x)dx \tag{C.11}$$

$$\int_a^0 x dF_{X_k}(x) = -aF_{X_k}(a) - \int_a^0 F_{X_k}(x)dx. \tag{C.12}$$

One again we examine the convergence behavior as $a \to -\infty$. The argument for the convergence of the left hand integral is the same, whereas the right hand integral converges by definition. Finally, because $F_{X_k}(-\infty) = 0$ we can conclude using the same reasoning as above that:

$$\int_{-\infty}^0 x dF_{X_k}(x) = -\int_{-\infty}^0 F_{X_k}(x)dx \tag{C.13}$$

in the sense that if one side converges than so does the other. In light of Equations C.10 and C.13, we revisit Equation C.4.

$$\int_{-\infty}^\infty x dF_{X_k}(x) = \int_{-\infty}^0 x dF_{X_k}(x) + \int_0^\infty x dF_{X_k}(x) \tag{C.14}$$

$$\int_{-\infty}^\infty x dF_{X_k}(x) = \int_0^\infty \left[1 - F_{X_k}(x)\right] dx - \int_{-\infty}^0 F_{X_k}(x)dx \tag{C.15}$$

Therefore we can conclude that the expected value is

$$E[X_k] = \int_0^\infty \left[1 - F_{X_k}(x)\right] dx - \int_{-\infty}^0 F_{X_k}(x) dx. \tag{C.16}$$

Finally we can use Equation C.2 to write the expected value returned by the random search after examining k states in terms of the underlying distribution of state performance, as described by the cumulative distribution function $F_X(x)$.

$$E[X_k] = \int_0^\infty \left[1 - F_X(x)^k\right] dx - \int_{-\infty}^0 F_X(x)^k dx. \tag{C.17}$$

References

[1] W. Feller, *An Introduction to Probability Theory and Its Applications*, 2nd ed. John Wiley & Sons, Inc., Jan. 1971, vol. 2, pp. 150–160, ISBN: 9780471257097.

Appendix D

META GAP OPTIMIZATION CODE

Optimizer.py

```
# parent class used to specify interface
class Optimizer(object):
    # intializes the optimizer with its required data
    # return type: self
    def __init__(self,number_top_states,characterizer,verbose=False):
        self.verbose = verbose
        self.number_top_states = number_top_states
        self.characterizer = characterizer
        self.top_states = [-1 for state in range(number_top_states)]
        self.top_values = [float("-inf") for state in range(number_top_states)]

        # dictionary storing the state and evaluated value of every state
        self.explored_states = {}

    # ABSTRACT:determines whether algorithm should proceed with next batch of states
    # returns type: bool
    def isFinished(self):
        raise Exception("Optimizer isFinished is not defined")

    # ABSTRACT: provides the next batch of states to be measured
    # if there IS a sort of batch counter checked by is finished, it is updated here
    # returns type: list of states
    def getNextStates(self):
        raise Exception("Optimizer getNextStates is not defined")


```

```
                                                                    27
    # characterize each of the states explored compared to          28
    ↪  the baseline_state
    # store characterization value in a dictionary                  29
    # update the list of top states                                 30
    def evaluateMeasurements(self,states,baseline_state):           31
        if self.verbose:                                            32
            print "Evaluating measurements"                         33
        for state in states:                                        34
            state_value = self.characterizer.characterize(          35
            ↪  state,baseline_state)
            self.explored_states[state] = state_value               36
            self.compareWithOptimalList(state,state_value)          37
                                                                    38
    # maintains a list of top states by comparing new state         39
    ↪  with optimals
    # returns type: None                                            40
    def compareWithOptimalList(self, state, value):                 41
        if state >= 0:                                              42
            optimal_idx = self.number_top_states - 1                43
            larger_value = False                                    44
            while not larger_value:                                 45
                stored_value = self.top_values[optimal_idx]         46
                stored_state = self.top_states[optimal_idx]         47
                if value >= stored_value:                           48
                    self.top_values[optimal_idx] = value            49
                    self.top_states[optimal_idx] = state            50
                    if optimal_idx + 1 <                            51
                    ↪  self.number_top_states:
                        self.top_values[optimal_idx+1] =            52
                        ↪  stored_value
                        self.top_states[optimal_idx+1] =            53
                        ↪  stored_state
                else:                                               54
                    larger_value = True                             55
                optimal_idx = optimal_idx - 1                       56
                if optimal_idx < 0:                                 57
                    larger_value = True                             58
                                                                    59
                                                                    60
                                                                    61
                                                                    62
                                                                    63
```

```
    # returns the optimal state from the optimizer
    # returns type: state
    def getOptimalState(self):
        optimum_state = self.top_states[0]
        optimum_value = self.top_values[0]
        if self.verbose:
            print "Optimal State:", optimum_state,", Value
            ↪  =", optimum_value
        return optimum_state

    # returns the list of  optimal states
    # returns type: [state]
    def getOptimalList(self):
        return self.top_states
```

RandomOptimizer.py

```
import Optimizer
import random

# this optimizer generates random states and takes the best one
# has the option of a using a tabu list, this is VERY INNEFICIENT if most of the search space will be explored
class RandomOptimizer(Optimizer.Optimizer):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    # use_taby: boolean indicated whether previously explored states should be removed from pool of potential states
    def __init__(self, num_batches, states_per_batch, num_bits, number_top_states, characterizer,use_tabu = False, verbose=False):
        super(RandomOptimizer,self).__init__( number_top_states,characterizer,verbose)
        self.verbose = verbose

        self.num_batches = num_batches
        self.states_per_batch = states_per_batch

        self.num_bits = num_bits
        self.maximum_state =  2**num_bits-1

        self.batch_counter = 0
        self.current_states = []

        self.use_tabu = use_tabu

    def isFinished(self):
        if self.verbose:
            print "Checking if Finished"
        return self.batch_counter >= self.num_batches
```

```
    def getNextStates(self):                                    36
        if self.verbose:                                        37
            print "Getting Next States"                         38
        self.current_states = []                                39
        for i in range(self.states_per_batch):                  40
            random_state = random.randint(0,                    41
            ↪  self.maximum_state)
            if self.use_tabu:                                   42
                while random_state in                           43
                ↪  self.explored_states.key():
                    random_state = random.randint(0,            44
                    ↪  self.maximum_state)
            self.current_states += [random_state]               45
        self.batch_counter = self.batch_counter + 1             46
        return self.current_states                              47
```

GeneticAlgorithm.py

```
import Optimizer
import random

# this optimizer performs a generational genetic optimization over a given number of generations
# it uses uniform crossover
# generation size MUST be an even number
class GeneticAlgorithm(Optimizer.Optimizer):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    def __init__(self, mutation_prob, num_generations, generation_size, num_bits, number_top_states, characterizer, initial_population = None, verbose=False):
        super(GeneticAlgorithm,self).__init__(number_top_states,characterizer,verbose)
        self.verbose = verbose

        self.num_generations = num_generations
        self.generation_size = generation_size
        self.maximum_state =  2**num_bits-1
        self.num_bits = num_bits

        self.initial_population = initial_population

        self.generation_number = 0
        self.population = []

        self.p_mutate = mutation_prob

############### INTERFACE FUNCTIONS ###############

    def isFinished(self):
        if self.verbose:
            print "Checking if Finished"
        return self.generation_number >= self.num_generations
```

```
    def getNextStates(self):
        if self.verbose:
            print "Getting Next States"
        if len(self.population) == 0:
            print "Initializing Population..."
            self.population = self.getInitialPopulation()
        else:
            print "Selecting Parents..."
            parents = self.selection()
            #print parents
            print "Performing Recombination..."
            children = self.recombination(parents)
            #print children
            print "Mutating Children..."
            self.population = self.mutation(children)
        self.generation_number += 1
        return self.population

############### Child Class Helper Functions ###############

    # return a list of uniformly random states
    def getInitialPopulation(self):
        if self.initial_population[0] is None:
            population = [random.randint(0,
            ↪  self.maximum_state) for i in
            ↪  range(self.generation_size)]
        else:
            population = self.initial_population
            num_needed = self.generation_size -
            ↪  len(population)
            if num_needed < 0:
                population =
                ↪  population[:self.generation_size]
            if num_needed > 0:
                for idx in range(num_needed):
                    # pick a random existing state
                    state = random.sample(population, 1)
                    # convert to binary
                    bin_state = bin(state[0])
                    state_bit_count = len(bin_state)-2
                    if state_bit_count < self.num_bits:
                        missing_bits = self.num_bits -
                        ↪  state_bit_count
```

```
                        bin_state = bin_state[:2] +
                        ↪  '0'*missing_bits + bin_state[2:]
                    # flip up to half its bits
                    mutated_state = '0b'
                    # get potential bit flip idx
                    bits_to_flip =
                    ↪  random.sample(range(self.num_bits),
                    ↪  self.num_bits/2)
                    for bit_idx in bits_to_flip:
                        state_bit = bin_state[2+bit_idx]
                        # randomly flip bit
                        coin_flip = random.random()
                        if coin_flip >= 0.5:
                            if state_bit == '0':
                                mutated_state += '1'
                            else:
                                mutated_state += '0'
                        else:
                            mutated_state += state_bit
                    mutated_state = int(mutated_state,2)
                    population += [mutated_state]
        return population

    def selection(self, method ='ranked'):
        population_fitness = [self.explored_states[state]
        ↪  for state in self.population]
        # Sort the population from most to least fit
        ordered_population = sorted(zip(population_fitness,
        ↪  self.population),reverse=True)
        if method == 'ranked':
            for i in range(len(ordered_population)):
                ordered_population[i] =
                ↪  (ordered_population[i][1],len(␣
                ↪  ordered_population)-i)
        if method == 'fitness':
            min_fitness = min(population_fitness)
            for i in range(len(ordered_parents)):
                ordered_population[i] =
                ↪  (ordered_population[i][1],
                ↪  ordered_population[i][0] + min_fitness +
                ↪  1)
        total_weight = 0
        for member in ordered_population:
```

```
            total_weight += member[1]
        parents = []
        next_generation_size = 0
        while next_generation_size < self.generation_size:

            parent_a = ordered_population[0][0]
            parent_b = ordered_population[0][0]
            lotto_number_a = random.uniform(0, total_weight)
            # not the most efficent way to do this but I
            ↪ think it works
            while parent_a == parent_b:
                lotto_number_b = random.uniform(0,
                ↪ total_weight)
                lotto_draw = 0
                for member_idx in
                ↪ range(len(ordered_population)):
                    parent_candidate =
                    ↪ ordered_population[member_idx][0]
                    parent_candidate_weight =
                    ↪ ordered_population[member_idx][1]
                    lotto_draw += parent_candidate_weight
                    if lotto_draw <= lotto_number_a:
                        parent_a = parent_candidate
                    if lotto_draw <= lotto_number_b:
                        parent_b = parent_candidate

            parent_pair = (parent_a,parent_b)
            rev_parent_pair = (parent_b,parent_a)
            parents += [parent_pair]
            next_generation_size = next_generation_size + 2
        return parents

    # uniform random recombination
    def recombination(self,parents):
        children = []
        for parent_pair in parents:
            parent_a = bin(parent_pair[0])
            parent_b = bin(parent_pair[1])
            a_bits = len(parent_a)-2
            b_bits = len(parent_b)-2
            if a_bits < self.num_bits:
                missing_bits = self.num_bits - a_bits

```

```
                parent_a = parent_a[:2] + '0'*missing_bits + parent_a[2:]
            if b_bits < self.num_bits:
                missing_bits = self.num_bits - b_bits
                parent_b = parent_b[:2] + '0'*missing_bits + parent_b[2:]
            kids = ['0b','0b']
            for bit_idx in range(2,2+self.num_bits):
                bit_a = parent_a[bit_idx]
                bit_b = parent_b[bit_idx]
                kid_a_idx = random.randint(0,1)
                kid_b_idx = 1 - kid_a_idx
                kids[kid_a_idx] += bit_a
                kids[kid_b_idx] += bit_b
            children += kids
        return children

    def mutation(self,children):
        population = []
        for kid in children:
            mutant_kid = None
            # repeat mutate process until mutant kid is not in the tabu_list
            # this will force mutations when there are only a couple of states left
            while mutant_kid is None or mutant_kid in self.explored_states.keys():
                mutant_kid = '0b'
                for bit_idx in range(2,2+self.num_bits):
                    kid_bit = kid[bit_idx]
                    coin_flip = random.random()
                    if coin_flip < self.p_mutate:
                        if kid_bit == '0':
                            mutant_kid += '1'
                        else:
                            mutant_kid += '0'
                    else:
                        mutant_kid += kid_bit
                mutant_kid = int(mutant_kid,2)
            population += [mutant_kid]
        return population
```

VariableNeighborSearch.py

```
import Optimizer
import random

# this optimizer performs a generational genetic optimization over a given number of generations
# it uses uniform crossover
# generation size equals the number of bits in the state
class VariableNeighborSearch(Optimizer.Optimizer):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    def __init__(self, expandedNeighborhoodSize, batch_size, num_batches, num_bits, number_top_states, characterizer, initial_state = None, verbose=False):
        super(VariableNeighborSearch,self).__init__(number_top_states,characterizer,verbose)
        self.verbose = verbose

        self.num_batches = num_batches

        self.num_bits = num_bits
        self.maximum_state =  2**num_bits-1

        self.batch_counter = 0
        self.current_state = initial_state
        self.current_value = float("-inf")
        self.next_states = []

        self.batch_size = batch_size

        self.expandedNeighborhoodSize = expandedNeighborhoodSize


############### INTERFACE FUNCTIONS ###############

    def isFinished(self):
        if self.verbose:
            print "Checking if Finished"
        return self.batch_counter >= self.num_batches

    def getNextStates(self):
        if self.verbose:
```

```
            print "Getting Next States"
        if self.batch_counter == 0:
            if self.verbose:
                print "Selecting Initial States..."
            self.next_states = self.getInitialStates()
            self.batch_counter += 1
            return self.next_states
        if self.verbose:
            print "Comparing Choices..."
        best_neighbor_state = float("-inf")
        best_neighbor_value = float("-inf")
        for neighbor_state in self.next_states:
            neighbor_value =
            ↪  self.explored_states[neighbor_state]
            if neighbor_value > best_neighbor_value:
                best_neighbor_state = neighbor_state
                best_neighbor_value = neighbor_value
        if self.verbose:
            print "Checking for Local Maximum..."

        self.next_states = []
        if self.current_value < best_neighbor_value:
            if self.verbose:
                print "Improvement Detected: Moving State"
            self.current_state = best_neighbor_state
            self.current_value = best_neighbor_value
            neighborhoodSize = 1
        else:
            if self.verbose:
                print "Local Maximum Detected: Expanding
                ↪  Neighborhood"
            neighborhoodSize = self.expandedNeighborhoodSize
        leftover_states = self.batch_size
        # fill up the batch using randomly selected states
        ↪  from expanded neighborhood
        while leftover_states > 0:
            print "Selecting Neighbors from k Neighborhood"
            neighborhood = self.getNeighborhood(
            ↪  self.current_state,neighborhoodSize)
            possible_neighbors = [neighbor for neighbor in
            ↪  neighborhood if (neighbor not in
            ↪  self.explored_states.keys())]
            if  len(possible_neighbors) >  leftover_states:
```

```
                self.next_states +=
                ↪  random.sample(possible_neighbors,
                ↪  leftover_states)
            else:
                self.next_states += possible_neighbors
                neighborhoodSize += 1
            leftover_states = self.batch_size -
            ↪  len(self.next_states)
            if leftover_states > 0 and self.verbose:
                print "Neighborhood explored: Adding extra
                ↪  expansion"
        self.batch_counter += 1
        return self.next_states

############## Child Class Helper Functions ##############

    # return a list of uniformly random states
    def getInitialStates(self):
        if self.current_state is None:
            initial_states = [random.randint(0,
            ↪  self.maximum_state) for i in
            ↪  range(self.batch_size)]
        else:
            neighborhoodSize = 1
            initial_states = [self.current_state]
            leftover_states = self.batch_size - 1
            # fill up the batch using randomly selected
            ↪  states from expanded neighborhood
            while leftover_states > 0:
                print "Selecting Neighbors from k
                ↪  Neighborhood"
                neighborhood = self.getNeighborhood(
                ↪  self.current_state,neighborhoodSize)
                possible_neighbors = [neighbor for neighbor
                ↪  in neighborhood if (neighbor not in
                ↪  self.explored_states.keys())]
                if  len(possible_neighbors) >
                ↪  leftover_states:
                    initial_states +=
                    ↪  random.sample(possible_neighbors,
                    ↪  leftover_states)
                else:
                    initial_states += possible_neighbors
```

```
                    neighborhoodSize += 1
                leftover_states = self.batch_size -
                ↪  len(initial_states)
                if leftover_states > 0 and self.verbose:
                    print "Neighborhood explored: Adding
                    ↪  extra expansion"
        return initial_states

    def state2bin(self,state):
        bin_state = bin(state)
        state_bit_count = len(bin_state)-2
        if state_bit_count < self.num_bits:
            missing_bits = self.num_bits - state_bit_count
            bin_state = bin_state[:2] + '0'*missing_bits +
            ↪  bin_state[2:]
        return bin_state

    def bin2state(self,bin_state):
        return int(bin_state,2)

    def binNeighborhood2Neighborhood(self,binNeighborhood):
        neighborhood = []
        for bin_state in list(binNeighborhood):
            neighborhood += [int(bin_state,2)]
        return neighborhood

    # flip the bit in the state indicated by the bit_idx
    def flipBit(self, bin_state, bit_idx):
        bit_idx = bit_idx + 2
        bit_value = bin_state[bit_idx]
        if bit_value == '0':
            bin_state = bin_state[:bit_idx] + '1' +
            ↪  bin_state[(bit_idx+1):]
        else:
            bin_state = bin_state[:bit_idx] + '0' +
            ↪  bin_state[(bit_idx+1):]
        return bin_state

    def getBinNeighborhood(self, bin_state, start_idx,
    ↪  num_to_flip):
        if num_to_flip == 0:
            return [bin_state]
        if num_to_flip == 1:
```

```
            neighborhood = []                                          141
            for bit_idx in range(start_idx, self.num_bits):            142
                new_state = self.flipBit(bin_state,bit_idx)            143
                neighborhood += [new_state]                            144
            return neighborhood                                        145
        else:                                                          146
            neighborhood = []                                          147
            for bit_idx in range(start_idx, self.num_bits):            148
                new_state = self.flipBit(bin_state,bit_idx)            149
                neighborhood +=                                        150
                ↪  self.getBinNeighborhood(new_state,
                ↪  bit_idx+1, num_to_flip-1)
        return neighborhood                                            151
                                                                       152
    def getNeighborhood(self, state, number_of_steps):                 153
        bin_state = self.state2bin(state)                              154
        bin_neighborhood = self.getBinNeighborhood(⌋                   155
        ↪  bin_state,0,number_of_steps)
        return self.binNeighborhood2Neighborhood(⌋                     156
        ↪  bin_neighborhood)
```

ParticleSwarm.py

```
import Optimizer
import random

# this optimizer performs a generational genetic optimization over a given number of generations
# it uses uniform crossover
# generation size equals the number of bits in the state
class ParticleSwarm(Optimizer.Optimizer):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    def __init__(self, inertia, global_pull, noisy_movement, num_batches, states_per_batch, num_bits, number_top_states, characterizer, initial_states = None, verbose=False):
        super(ParticleSwarm,self).__init__(number_top_states,characterizer,verbose)
        self.verbose = verbose

        self.num_batches = num_batches

        self.num_bits = num_bits
        self.maximum_state =  2**num_bits-1

        self.batch_counter = 0

        self.inertia = inertia
        self.global_pull = global_pull
        self.noisy_movement = noisy_movement

        self.num_particles = states_per_batch
        self.particles = []
        for i in range(self.num_particles):
            # randomize initial positions
            if initial_states[0] is None:
                self.particles += [Particle(self.num_bits, inertia, global_pull, noisy_movement)]
            # initial positions are assigned so particles are likely to be at initial positions
            else:
                position = []
                if i < len(initial_states):
                    discrete_position = initial_states[i]
```

```
                else:
                    rand_samp =
                    ↪  random.sample(initial_states, 1)
                    discrete_position = rand_samp[0]
                bin_position = bin(discrete_position)
                bit_count = len(bin_position)-2
                if bit_count < self.num_bits:
                    missing_bits = self.num_bits - bit_count
                    bin_position = bin_position[:2] +
                    ↪  '0'*missing_bits + bin_position[2:]
                for bit_idx in range(2,self.num_bits+2):
                    position_bit = bin_position[bit_idx]
                    if position_bit == '1':
                        position += [0.9]
                    else:
                        position += [0.1]
                self.particles +=
                ↪  [Particle(self.num_bits,inertia,
                ↪  global_pull, noisy_movement,
                ↪  initial_position = position)]


############### INTERFACE FUNCTIONS ###############

    def isFinished(self):
        if self.verbose:
            print "Checking if Finished"
        return self.batch_counter >= self.num_batches

    def getNextStates(self):
        if self.verbose:
            print "Getting Next States"

        if self.batch_counter != 0:
            if self.verbose:
                print "Updating Local and Global
                ↪  Maximums..."

            global_optimum_state = self.top_states[0]
            for i in range(self.num_particles):
                particle = self.particles[i]
                observed_state = self.next_states[i]

```

```
                observed_value =
                ↪  self.explored_states[observed_state]
                particle.updateLocalBest(
                ↪  observed_state,observed_value)
                particle.updateGlobalBest(
                ↪  global_optimum_state)

            if self.verbose:
                print "Moving particles..."
            for particle in self.particles:
                particle.moveParticle()
        if self.verbose:
            print "Observing Particle Locations..."

        self.next_states = []
        for particle in self.particles:
            self.next_states += [particle.observeState()]

        self.batch_counter += 1
        return self.next_states

############### Child Class Helper Functions ###############

class Particle(object):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    def __init__(self, num_bits, inertia, global_pull,
    ↪  noisy_movement, initial_position =
    ↪  None,initial_velocity = None):
        self.num_bits = num_bits

        self.noisy_movement = noisy_movement

        self.inertia = inertia
        self.attraction = 1.0 - inertia

        self.global_pull = global_pull
        self.local_pull = 1.0 - global_pull

        self.position = QuantumPosition(
        ↪  self.num_bits,bit_probabilities=initial_position)
```

```
        self.velocity =
        ↪  QuantumVelocity(self.num_bits,velocities =
        ↪  initial_velocity)

        self.global_best = None
        self.local_best = None
        self.local_best_value = float("-Inf")

    def observeState(self):
        observed_position = self.position.observe()
        return observed_position.getState()

    def moveParticle(self):
        local_distance = self.local_best - self.position
        if self.noisy_movement:
            local_distance.addNoise()
        local_force = local_distance * self.local_pull

        global_distance = self.global_best - self.position
        if self.noisy_movement:
            global_distance.addNoise()
        global_force = global_distance * self.global_pull

        self.position = self.position + self.velocity
        self.velocity = self.velocity*self.inertia +
        ↪  (local_force + global_force)*self.attraction

    def updateLocalBest(self,local_state,local_value):
        if local_value >= self.local_best_value:
            self.local_best_value = local_value
            self.local_best =
            ↪  DiscretePosition(self.num_bits,state =
            ↪  local_state)

    def updateGlobalBest(self,global_state):
        self.global_best = DiscretePosition(␣
        ↪  self.num_bits,state=global_state)

# location of particle in probability space
class DiscretePosition(object):
    def __init__(self, num_bits, coordinates = None, state =
    ↪  None):
        self.num_bits = num_bits
```

```
        if coordinates is not None:
            self.coordinates = coordinates
        elif state is not None:
            self.coordinates = []
            for bit_idx in range(self.num_bits):
                # is bit a 1 or 0
                state_bit = (state>>bit_idx) & 1
                self.coordinates += [state_bit]
        else:
            self.coordinates = []
            for i in range(num_bits):
                self.coordinates += [random.randint(0,1)]

    def __sub__(self, other):
        if str(type(other)) == "<class
        ↪  'Optimizer.ParticleSwarm.DiscretePosition'>":
            velocities = []
            for bit_idx in range(self.num_bits):
                e1 = self.coordinates[bit_idx]
                e2 = other.coordinates[bit_idx]
                # if they are the same, don't change
                ↪  anything
                if e1 == e2:
                    velocities += [0]
                # if they are different, indicate
                ↪  directional difference to equal e1
                else:
                    velocities += [2*(e1-0.5)]
            # return a quantum velocity indicating the
            ↪  desired direction
            return QuantumVelocity(
            ↪  self.num_bits,velocities=velocities)
        elif str(type(other)) == "<class
        ↪  'Optimizer.ParticleSwarm.QuantumPosition'>":
            return other.__sub__(self)*-1
        else:
            exception_name = "Subtraction not defined for
            ↪  Type DiscretePosition and Type " +
            ↪  str(type(other))
            raise Exception(exception_name)

    def getState(self):
        state = 0
```

```
        for bit_idx in range(len(self.coordinates)):
            bit_value = self.coordinates[bit_idx]
            state += bit_value * 2**(bit_idx)
        return state

class QuantumPosition(object):
    def __init__(self, num_bits, bit_probabilities = None):
        self.num_bits = num_bits
        if bit_probabilities is None:
            self.bit_probabilities = []
            for i in range(self.num_bits):
                self.bit_probabilities += [random.random()]
        else:
            self.bit_probabilities = bit_probabilities

    def observe(self):
        bit_states = []
        for bit_probability in self.bit_probabilities:
            random_draw = random.random()
            if random_draw <= bit_probability:
                bit_states += [1]
            else:
                bit_states += [0]
        return DiscretePosition(self.num_bits, coordinates =
        ↪  bit_states)

    # applies bayes rule using quantum velocity to change
    ↪  probability
    def __add__(self, other):
        if str(type(other)) != "<class
        ↪  'Optimizer.ParticleSwarm.QuantumVelocity'>":
            exception_name = "Addition not defined for Type
            ↪  QuantumPosition and Type " +
            ↪  str(type(other))
            raise Exception(exception_name)
        new_bit_probabilities = []
        for bit_index in range(self.num_bits):
            p_on = self.bit_probabilities[bit_index]
            p_off = 1.0 - p_on
            p_velocity_given_on =
            ↪  0.5*(1+other.velocities[bit_index])
            p_velocity_given_off = 1.0 - p_velocity_given_on

```

```
            p_velocity = p_velocity_given_on*p_on +
            ↪  p_velocity_given_off*p_off
            p_on_given_velocity =
            ↪  p_velocity_given_on*p_on/p_velocity
            new_bit_probabilities += [p_on_given_velocity]
        return QuantumPosition(self.num_bits,
        ↪  bit_probabilities= new_bit_probabilities)

    def __sub__(self, other):
        if str(type(other)) == "<class
        ↪  'Optimizer.ParticleSwarm.DiscretePosition'>":
            velocities = []
            for bit_idx in range(self.num_bits):
                self_location =
                ↪  self.bit_probabilities[bit_idx]
                other_location = other.coordinates[bit_idx]
                difference = self_location - other_location
                velocities += [difference]
            # return a quantum velocity indicating the
            ↪  desired direction
            return QuantumVelocity(
            ↪  self.num_bits,velocities=velocities)
        elif str(type(other)) == "<class
        ↪  'Optimizer.ParticleSwarm.QuantumPosition'>":
            velocities = []
            for bit_idx in range(self.num_bits):
                self_location =
                ↪  self.bit_probabilities[bit_idx]
                other_location =
                ↪  other.bit_probabilities[bit_idx]
                difference = self_location - other_location
                velocities += [difference]
            # return a quantum velocity indicating the
            ↪  desired direction
            return QuantumVelocity(
            ↪  self.num_bits,velocities=velocities)
        else:
            exception_name = "Subtraction not defined for
            ↪  Type DiscretePosition and Type " +
            ↪  str(type(other))
            raise Exception(exception_name)


```

```
# a quantum velocity is a number between -1 and 1
# this represents how much weight to adjust a quantum position
# using bayes rules
# -1 implies that the quantum position P(0) will become 1
#  1 implies that the quantum position P(1) will become 1
# thus we set a max speed to ensure this never happens
class QuantumVelocity(object):
    def __init__(self, num_bits, velocities = None):
        self.max_speed = 1 - 10**(-6)
        self.num_bits = num_bits
        if velocities is None:
            self.velocities = []
            for i in range(num_bits):
                # random number between -1 and 1
                velocity = 2*(random.random()-0.5)
                #cap speed
                self.velocities += [velocity]
            self.velocities = self.bound_velocities(self.velocities)
        else:
            # bound velocity to max_seed
            velocities = self.bound_velocities(velocities)
            self.velocities = velocities

    def __add__(self, other):
        if str(type(other)) != "<class 'Optimizer.ParticleSwarm.QuantumVelocity'>":
            exception_name = "Addition not defined for Type DiscreteVelocity and Type " + str(type(other))
            raise Exception(exception_name)
        if self.num_bits != other.num_bits:
            raise Exception("Cannot add velocities with different dimensions")
        new_v = []
        for bit_idx in range(self.num_bits):
            v1 = self.velocities[bit_idx]
            v2 = other.velocities[bit_idx]
            v3 = v1 + v2
            new_v += [v3]
        new_v = self.bound_velocities(new_v)

```

```
            return QuantumVelocity(self.num_bits,velocities =
            ↪  new_v)

        def __mul__(self, other):
            if str(type(other)) in ["<type 'int'>","<type
            ↪  'float'>"]:
                new_velocities = []
                for velocity in self.velocities:
                    new_v = other * velocity
                    new_velocities += [new_v]
                # cap to between -self.max_speed and
                ↪  self.max_speed
                new_v = self.bound_velocities(new_velocities)
                return QuantumVelocity(self.num_bits,velocities
                ↪  = new_velocities)
            else:
                exception_name = "Multiplication not defined for
                ↪  Type DiscreteVelocity and Type " +
                ↪  str(type(other))
                raise Exception(exception_name)
            pass

        def addNoise(self):
            for i in range(len(self.velocities)):
                ideal_v = self.velocities[i]
                self.velocities[i] = ideal_v * random.random()

        def bound_velocities(self, velocities):
            velocities = [min(self.max_speed,v) for v in
            ↪  velocities]
            velocities = [max(-1*self.max_speed,v) for v in
            ↪  velocities]
            return velocities
```

SimulatedAnnealing.py

```
import Optimizer
import random
import math

# this optimizer performs a generational genetic optimization over a given number of generations
# it uses uniform crossover
# generation size equals the number of bits in the state
class SimulatedAnnealing(Optimizer.Optimizer):
    # num_batches: how many batchs to iterate over
    # states_per_batch: how many states per batch
    def __init__(self, temperature_schedule, num_batches, states_per_batch, num_bits, number_top_states, characterizer, initial_state = None, verbose=False):
        super(SimulatedAnnealing,self).__init__(number_top_states,characterizer,verbose)
        self.verbose = verbose

        self.num_batches = num_batches
        self.states_per_batch = states_per_batch

        self.num_bits = num_bits
        self.maximum_state =  2**num_bits-1

        self.batch_counter = 0

        self.temperatureSchedule = temperature_schedule

        self.random_initialization = True
        self.current_state = 0
        if initial_state is not None:
            self.random_initialization = False
            self.current_state = initial_state

        self.current_value = float("-inf")
        self.next_states = []

        # calculate a threshold for a neighborhood size
        # that could being fully explored
        neighborHoodSize = 0
        finished = False
        while not finished:
```

```
            neighborHoodSize += 1                                          39
            num_neighbors =                                                40
            ↪  len(self.getUnexploredNeighborhood(,
            ↪  0,neighborHoodSize))
            finished = num_neighbors > self.num_batches *                  41
            ↪  self.states_per_batch
        self.neighborhood_threshold = neighborHoodSize                     42
                                                                           43
                                                                           44
################ INTERFACE FUNCTIONS ###############                      45
                                                                           46
    def isFinished(self):                                                  47
        if self.verbose:                                                   48
            print "Checking if Finished"                                   49
        return self.batch_counter >= self.num_batches                      50
                                                                           51
    def getNextStates(self):                                               52
        if self.batch_counter != 0:                                        53
            if self.verbose:                                               54
                print "Comparing States..."                                55
            best_neighbor_state = float("-inf")                            56
            best_neighbor_value = float("-inf")                            57
            for neighbor_state in self.next_states:                        58
                neighbor_value =                                           59
                ↪  self.explored_states[neighbor_state]
                if neighbor_value > best_neighbor_value:                   60
                    best_neighbor_state = neighbor_state                   61
                    best_neighbor_value = neighbor_value                   62
            if self.current_value < best_neighbor_value:                   63
                if self.verbose:                                           64
                    print "Improvement Detected: Moving                    65
                    ↪  State"
                self.current_state = best_neighbor_state                   66
                self.current_value = best_neighbor_value                   67
                                                                           68
        remaining_batches = self.num_batches -                             69
        ↪  self.batch_counter
        fraction_remaining =                                               70
        ↪  (remaining_batches-1)/float(self.num_batches-1)
        current_temperature =                                              71
        ↪  self.temperatureSchedule(fraction_remaining)
                                                                           72
                                                                           73
```

```
        # if an inital state is provided, halve the
        ↪  temperature so that at most half the bits can
        ↪  flip
        # without temperature reduction, the initial state
        ↪  would be meaningless as all bits can flip
        if not self.random_initialization:
            current_temperature = current_temperature / 2.0
        if self.verbose:
            print "Temperature =", current_temperature
        neighborHoodSize =
        ↪  self.numberOfBitFlips(current_temperature)

        if self.verbose:
            print "Getting Next States"
        self.next_states = []
        # there is a relatively decent chance that the
        ↪  neighborhood has been
        # fully, or nearly fully, explored. Thus we
        ↪  explicitly calculate
        # all possible neighbors and randomly pick from that
        ↪  list
        if neighborHoodSize <= self.neighborhood_threshold:
            if self.verbose:
                print "Small Neighborhood, performing
                ↪  explicit selection..."
            possible_states =
            ↪  self.getUnexploredNeighborhood(
            ↪  self.current_state,neighborHoodSize)

            remaining_spots = self.states_per_batch -
            ↪  len(self.next_states)
            while len(possible_states) <= remaining_spots:
                if self.verbose:
                    print "Adding all possible neighbors of
                    ↪  distance", neighborHoodSize
                # if all of neighborhood can be explored,
                ↪  explore all of it
                self.next_states += possible_states
                remaining_spots = self.num_bits -
                ↪  len(self.next_states)
                # expand the neighborhoodSize
                neighborHoodSize += 1
```

```
                possible_states =
                ↪  self.getUnexploredNeighborhood(
                ↪  self.current_state,neighborHoodSize)

            if self.verbose:
                print "Randomly adding neighbors of
                ↪  distance", neighborHoodSize
            # fill up the rest of the batch using randomly
            ↪  selected states from expanded neighborhood
            self.next_states +=
            ↪  random.sample(possible_states,
            ↪  remaining_spots)

        # it is impossible for the neighborhood to be fully
        ↪  explored
        # thus we find new states by random selection.
        ↪  Without exploration guarentee
        # this might not terminate
        else:
            if self.verbose:
                print "Large Neighborhood, performing random
                ↪  selection..."
            while len(self.next_states) <
            ↪  self.states_per_batch:
                possible_state = self.flipRandomBits(
                ↪  self.current_state,neighborHoodSize)
                if possible_state not in
                ↪  self.explored_states.keys():
                    self.next_states += [possible_state]

        # for state in self.next_states:
        #     print self.state2bin(self.current_state),
        ↪  self.state2bin(state)
        self.batch_counter += 1
        return self.next_states


############### Child Class Helper Functions ###############

    def state2bin(self,state):
        bin_state = bin(state)
        state_bit_count = len(bin_state)-2
        if state_bit_count < self.num_bits:
```

```
            missing_bits = self.num_bits - state_bit_count
            bin_state = bin_state[:2] + '0'*missing_bits +
            ↪  bin_state[2:]
        return bin_state

    def bin2state(self,bin_state):
        return int(bin_state,2)

    # temperature is between 0 and 100, 0 means only 1 bit
    ↪  flips, 100 means all
    def numberOfBitFlips(self,current_temperature):
        degrees_per_bit = 100.0/(self.num_bits)
        fractional_bits =
        ↪  current_temperature/degrees_per_bit
        num_flips = int(math.ceil(fractional_bits))
        #cannot flip more bits than there are
        num_flips = min(self.num_bits,num_flips)
        # make sure at least one bit flips at 0 degrees
        return max(num_flips,1)

    # flip the bit in the state indicated by the bit_idx
    def flipBit(self, bin_state, bit_idx):
        bit_idx = bit_idx + 2
        bit_value = bin_state[bit_idx]
        if bit_value == '0':
            bin_state = bin_state[:bit_idx] + '1' +
            ↪  bin_state[(bit_idx+1):]
        else:
            bin_state = bin_state[:bit_idx] + '0' +
            ↪  bin_state[(bit_idx+1):]
        return bin_state

    # flip a random set of bits in the state
    def flipRandomBits(self, state, number_of_bits):
        bin_state = self.state2bin(state)
        # randomly select which bits can flip
        bits_to_flip = random.sample(range(self.num_bits),
        ↪  number_of_bits)
        for bit_idx in bits_to_flip:
            # randomly flip bit
            coin_flip = random.random()
            if coin_flip >= 0.5:
                bin_state = self.flipBit(bin_state,bit_idx)
```

```
        return self.bin2state(bin_state)

    def getUnexploredNeighborhood(self,state,neighborHoodSize):
        bin_state = self.state2bin(state)
        bin_neighborhood = []
        for num_flips in range(1,neighborHoodSize+1):
            bin_neighborhood += self.getBinNeighborhood(bin_state, 0,num_flips)
        neighborhood = self.binNeighborhood2Neighborhood(bin_neighborhood)
        possible_states = []
        for neighbor in neighborhood:
            # prune all neighbors that have been, or have already selected to be, explored
            if neighbor not in self.explored_states.keys() and neighbor not in self.next_states:
                possible_states += [neighbor]
        return possible_states

    def getBinNeighborhood(self, bin_state, start_idx, num_to_flip):
        if num_to_flip == 0:
            return [bin_state]
        if num_to_flip == 1:
            neighborhood = []
            for bit_idx in range(start_idx, self.num_bits):
                new_state = self.flipBit(bin_state,bit_idx)
                neighborhood += [new_state]
            return neighborhood
        else:
            neighborhood = []
            for bit_idx in range(start_idx, self.num_bits):
                new_state = self.flipBit(bin_state,bit_idx)
                neighborhood += self.getBinNeighborhood(new_state, bit_idx+1, num_to_flip-1)
        return neighborhood


    def binNeighborhood2Neighborhood(self,binNeighborhood):
        neighborhood = []
        for bin_state in list(binNeighborhood):
```

```
            neighborhood += [int(bin_state,2)]
        return neighborhood


    @staticmethod
    def linearCooling(fraction_remaining):
        return 100*fraction_remaining

    @staticmethod
    def linearCoolingHoldZero(fraction_remaining):
        fraction_at_zero = 1/6.0
        start_temp = 100.0/(1 - fraction_at_zero)
        temp =
        ↪  start_temp*(fraction_remaining-fraction_at_zero)
        temp = max(temp,0)
        temp = min(temp,100)
        return temp
```