Optimization and Modeling for Optical Phased Array

by

Lei Yuan

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University

Ottawa, Ontario

Abstract

The rapid growth of technology these years has brought up the demands on LiDAR. Research on LiDAR has shown a higher detecting range, resolution, and detection speed than microwave radar. As the essential component of LiDAR, the optical phased array plays a fundamental role in the beamforming for the LiDAR.

Unlike microwave phased array antennas, optical phased array antennas naturally have grating lobes in the radiation pattern due to the large element spacing in the array. The grating lobes are undesirable since they dissipate energy from the main lobe, reducing the power efficiency of the optical phased array. Eliminating and reducing the grating lobes then become a pressing problem to solve.

This thesis presents three types of optical phased arrays: the rectangular, circular, and randomly distributed configurations, to achieve the maximum grating lobes suppression. Using a specific antenna design as the base element, we design and optimize three types of the arrays by implementing the genetic algorithm. We conclude that the rectangular phased arrays have a minor performance on sidelobe suppression compared to the other two types of phased arrays. The randomly distributed phased arrays will have a similar performance on sidelobe suppression compared to the circular phased arrays when over 400 antennas are placed, but it utilizes less device footprint. The circular phased arrays utilize the most device footprint, but it has the narrowest 3 dB beamwidth.

Acknowledgements

Firstly, I would like to deeply thank my thesis supervisor, Professor Winnie Ye, for her invaluable support throughout my master's studies. I greatly appreciate her for leading me to the correct path once I had no clue about my thesis research.

I am grateful for all the discussions and help from NRC team, which guided me in many ways. I also sincerely appreciate all the help and productive discussions with my fellow classmates, colleagues along the way. I am grateful to Qiankun Liu, Ilyas Kandid, and Shahrzad Khajavi for involving me in tremendous conversations that resolve the crucial problems throughout my thesis research.

Finally, I would like to truly thank my parents for their selfless support throughout my years of studies. I am sincerely thankful to my loving partner Yun Zhou for her kind caring and creating a cozy atmosphere that allows me to focus deeply on my studies and not have to worry about trifles.

List of Abbreviation

FWHM	Full Width at Half Maximum.	25
GA	Genetic Algorithm.	2
OPA	Optical Phased Array.	1
SLL	Sidelobe Level.	24
SWG	Subwavelength Grating.	6
ULA	Uniform Linear Array.	15

Table of Contents

List of Tables

List of Figures

Chapter 1:

Introduction

1.1 Motivation

The optical phased array (OPA) has recently drawn attention due to the rapid growth in wireless technology. As the essential component for the LiDAR, it controls the direction of the beam radiating out from the LiDAR by changing the relative phased between the antennas. Compared to the microwave phased array, OPA has the following advantages: higher resolutions, wider bandwidth, better anti-interference to EM fields [1]. Nevertheless, the evolution of technology always has several problems accompanying it. The optical antennas usually will have a larger size than the operating wavelength, which inevitably causes the grating lobes (multiple high-power signals radiating out from the LiDAR) for the optical phased array.

This thesis proposes the optimization methodologies for the commonly used optical phased array, which is rectangular, circular, and rarely used randomly distributed phased array. Section 1.2 and 1.3 provide background regarding the antennas used for all three types of the phased array and the algorithm implemented for the optimization,

respectively.

1.2 Fundamentals of Optical Phased Array

The concept of OPA came from the phased array in the microwave, so that the OPA can be seen as a realization of the microwave phased array in optical wavelength. The OPA is used to transmit or receive light waves to achieve telecommunication. By controlling the optical properties, it is able to steer the light beam without mechanically moving the hardware. However, grating lobes generated by the OPA will dissipate power and distribute it in the unwanted zone in the far-field pattern, which also causes the wrong signal readings for the receiver OPA. Thus, several algorithms were investigated and implemented to eliminate the grating lobes, in which the genetic algorithm is the commonly used algorithm [2].

1.3 Genetic Algorithm

Genetic Algorithm (GA) is a type of stochastic optimization, and it was proposed by Alan Turing in 1950, which is a so-called “learning machine” that is used to analog the principles of evolution. Somehow, the algorithm is not widespread until the early 1970s when Professor John Henry Holland contributes his tremendous work on this algorithm.

Genetic algorithm (GA) nowadays is used to solve various types of problems, and it utilizes the principle of evolution. The flow chart illustrates the process of the GA.

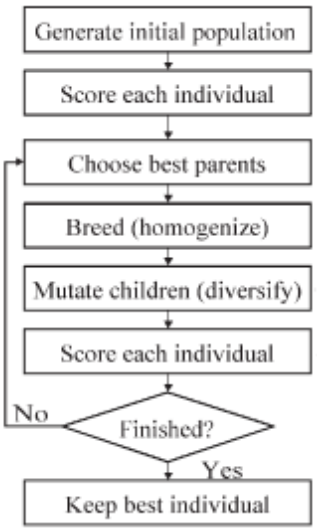


Figure 1.1: Flowchart of the GA [2]

At first, the algorithm will have multiple chromosomes (binary encoded), with each chromosome standing for a possible solution to the problem. The set of all current solutions at a given point during the algorithm is called the generation. When the starting generation is going to jump into the next generation, natural selection is performed, where it uses a fitness function to decide how good the given solution is, and it will return the fitness value for each chromosome. Generally speaking, the chromosome that has a higher score in fitness value is more likely to be picked to perform the reproduction. The reproduction consists of crossover and mutation. The crossover will choose two chromosomes with relatively higher fitness scores as the "parent" and swap some part of the binary code to form a new chromosome (child). The mutation will then reverse some binary code of the child's chromosome to obtain better results for the problem. Thus, the optimal results can be obtained after the number of

generations [2].

1.4 LiDAR: The Next-Generation Radar

As the OPA technology is getting mature, it has found more applications in the market, such as space communication, autonomous vehicle, etc.[3-6] However, there is a critical issue that limits its application range, which is the grating lobes due to the large element spacing. Grating lobes are caused by the element spacing in an OPA being larger than half of the operating wavelength [7]. For instance, if the operating wavelength of the OPA is 1550 nm, the element spacing will have to be less than 775 nm to avoid grating lobes. For some antenna designs, their size is about to reach 4000nm width and 7000nm height [8]. Thus, placing this type of antenna within 775nm is impossible.

Previously, many researchers have tried to use amplitude weighting and phase weighting to reduce the grating lobes. These methods break the uniform weighted power of each radiating element for the planar OPA, which enlarges the power on the main lobe. As a result, sidelobes around the main lobe are indeed reduced, yet these weighting functions did not lessen the grating lobes since the grating lobe is caused by the periodically constructive interference performed by the radiating elements, which is the consequence of the large element spacing.

In 2007, Professor Stuart Yin at the University of Pennsylvania suggested non-uniform spacing in one-dimensional OPA [9], and more researchers started to delve into

the non-uniform spacing of the elements. Many simulations and test results showcase that non-uniform spacing effectively reduces the grating lobes by breaking the periodicity of the electric field [10-14]. Thus, non-uniformed spacing is a feasible solution to reduce the grating lobes without extra hardware. Through the years, the technology of optical phased array has been growing rapidly with various developed research [15-20], the optical phased array will likely be the next-generation radar.

1.5 Diffraction Gratings

The diffraction grating is an optical component with either periodic gratings or apodized gratings, and it is often used in integrated optics. For example, in a transmission grating, the incident light can be diffracted into a specific discrete direction (so-called diffraction order, m) depending on the wavelength and pitch length, because the equation for the transmission grating is

$$d(sin\alpha - sin\beta) = m\,\lambda \tag{1.1}$$

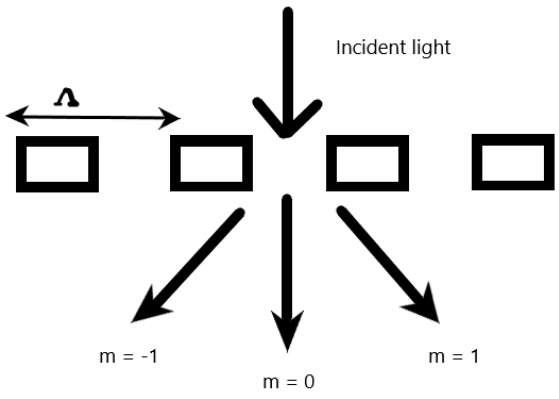


Fig. 1.2: Example of transmission grating

where d is the distance between the pitch, α is the angle between the incident light and the normal to the grating (the incident angle), β is the angle between the diffracted light and the normal to the grating (the diffraction angle), m is the diffraction order, and λ is the wavelength of light.

Another application of the diffraction grating is the grating coupler as shown in Figure 1.4 below.

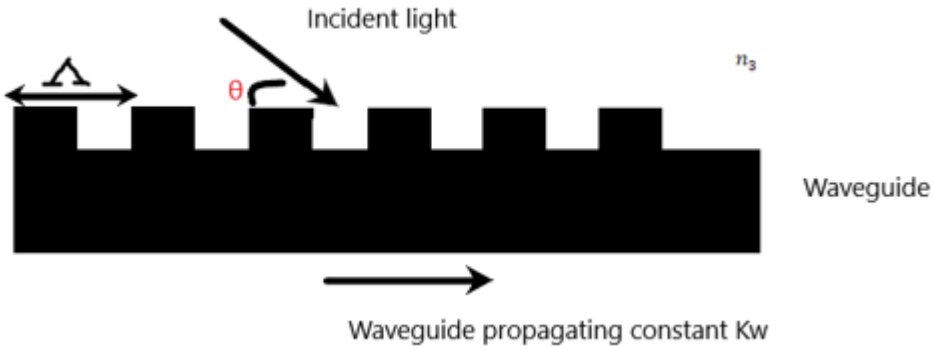


Fig. 1.3: Example of grating coupler

For incident light to couple into the waveguide, the incident light must have the same phase velocities along the propagation direction (phase-matching condition).

$$K_w = K_a - \frac{2\pi}{d} \tag{1.2}$$

The term on the right-hand side can also be expressed as

$$K_0 N - \frac{2\pi}{d} = K_0 n_3 sin\theta \tag{1.3}$$

Where K_w is the waveguide propagation constant, K_a is the propagation constant without waveguide, K_0 is the propagation constant in free space, N is the effective index, and n_3 is the refractive index of free space. Thus, the effective index can be modified by adjusting the pitch length of the gratings to coupling light.

We can also design an optical antenna with subwavelength gratings so that it

radiates the light in a particular direction depending on the wavelength of the source as shown in Figure 1.3 below.

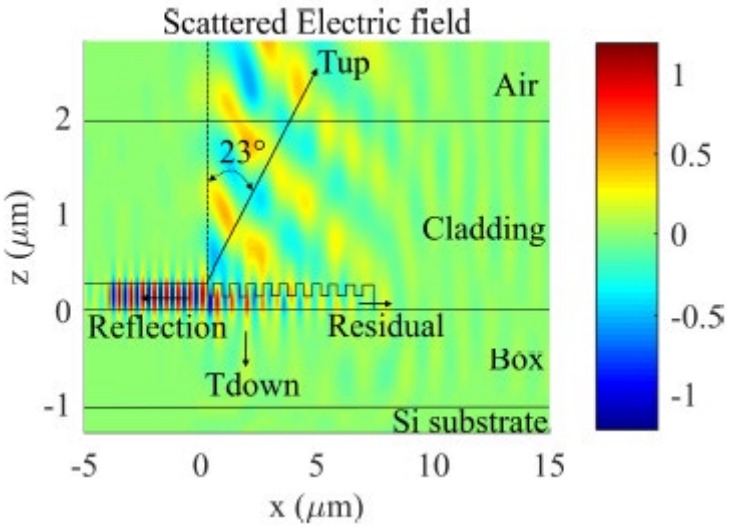


Fig. 1.4: 3D FDTD simulation of electric field distribution of the optimized antenna [8]

1.6 Thesis Objectives and Background

This thesis is focused on optimizing rectangular, circular, and randomly distributed phased arrays by applying the GA. The main objective is to eliminate the appearance of grating lobes due to the large spacing of optical antennas and to achieve better sidelobe suppression than the original phased array design with a given set of numbers of antennas elements.

Although research showed that an aperiodic phased array could eliminate the grating lobes and suppress the sidelobes effectively, many papers on aperiodic optimizations are focused on the linear phased array [12-14], and only a small number of papers were focused on the optimization of rectangular and circular phased array

[21-26]. Thus, we proposed novel optimization methodologies for the rectangular, circular, and random phased array are proposed in this thesis.

The primary metrics used to assess the performance of an antenna are the high diffraction efficiency, low back-reflection, broadband operation, etc. Research shows an L-shaped grating will effectively improve the diffraction efficiency and grating directionality [8].

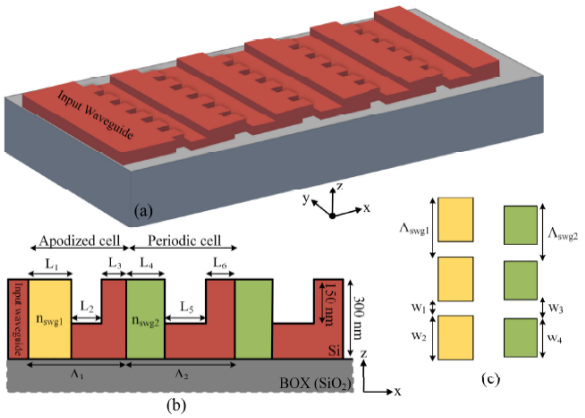
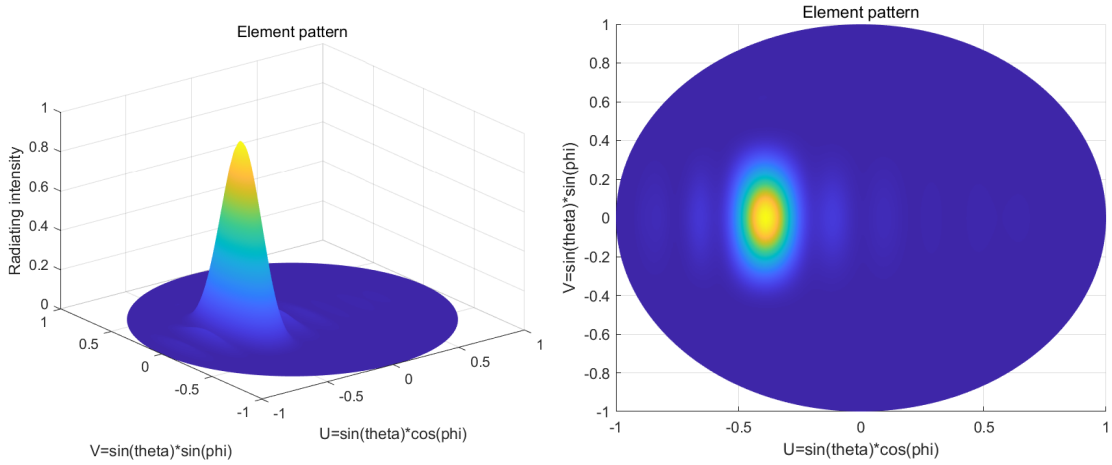


Fig. 1.5: a) Schematics of the antenna. b) Cross-section of the structure with grating periods. c) Top view of the subwavelength grating structures utilized in apodized yellow cell and periodic green cells with light come in via input waveguide, and propagate along the x-axis [8].

The L-shaped subwavelength grating (SWG) antenna designed and proposed by Khajavi [8] has a high diffraction efficiency of 0.89 and directionality of 0.94, where directionality by definition is how well an antenna focuses its radiated power in a particular direction to the exclusion of other directions. The L-shaped grating antenna is wavelength sensitive; it diffracts light upwardly at a specific angle with a corresponding wavelength. Furthermore, the L-shaped grating periods break the vertical symmetry of the antenna, where it leads most of the power radiated upward as

desired [8]. The antenna has a 23° diffraction angle (θ) in the far-field radiation pattern ,as shown in the figure below. This particular antenna design is used as the fundamental element in all the phased arrays proposed in this thesis.



a) *b)*

Figure 1.6: a) Element pattern of the antenna. b) Top view of the element pattern

For a systematic comparison, we simulate and compare the total far-field radiation pattern with a similar number of antennas arranged based on the three types of the proposed arrays: the rectangular, circular, and randomly distributed configuration. Due to the physical limitations of the optimal element placements in circular phased arrays (chapter 3, section 3.2), the total numbers of the antenna elements in 5 circular configurations are carefully selected as 126, 225, 324, 423, 513 because they are the optimal structures with the best performance on sidelobe suppression. In order to provide a valid comparison of the three types of phased arrays, we choose the same numbers of antenna elements in the optimized rectangular and randomly distributed array.

1.7 Thesis Organization

The thesis is organized as follows:

In chapter 2, the simulation of the rectangular phased array will be performed. Section 2.1 introduces the background and the condition for the appearance of the grating lobes for the rectangular phased array and the synthesis of the array factor with examples. Section 2.2 reveals an optimization example for the rectangular phased array. Section 2.3 shows the simulation results for the rectangular, and the summary comes within section 2.4.

In chapter 3 of the thesis, the simulation of the circular phased array will be performed as the same as for the rectangular phased array. The appearance of the grating lobes will be explained in section 3.1. In section 3.2, the optimization results will be revealed. In section 3.3, there will be a brief summary of the chapter.

In chapter 4, we examine the random phased array. The random phased array does not have a certain shape compared to either rectangular or circular phased array. All of the antennas were randomly distributed and optimized by the GA afterward.

In chapter 5, a conclusion is made upon the simulation results presented in this thesis, including the actual implementation feasibility and future works.

Chapter 2 :

Rectangular Optical Phased Array

2.1 Fundamentals

The rectangular phased array is also called the planar phased array, and its original idea of the phased array was proposed by physicist Karl F. Braun who has demonstrated beamforming in 1905. Many types of phased arrays were proposed and investigated afterward, and the planar phased array was one of them. All of the antennas for the phased array have the same specification. For the rectangular phased array, antennas are usually placed in a plane and separated by a uniform distance between each antenna in the x and y direction, as shown in the figure below.

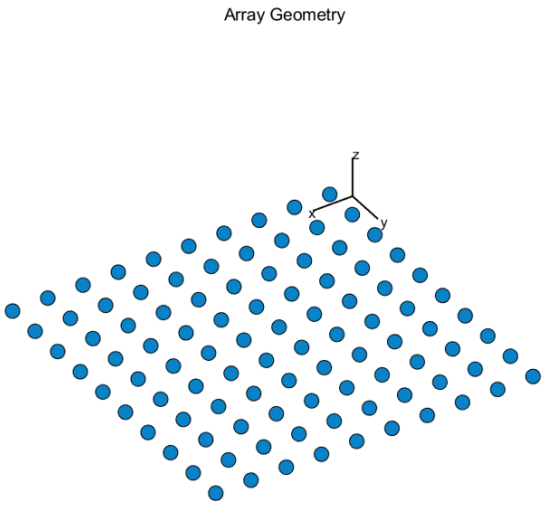


Figure 2.1: Uniformly spaced planar phased array

The actual antenna can be seen as a single point source to simplify the simulation process because the total field pattern from an array is the multiplication of the element pattern (the pattern which a single antenna produced) and the array factor. Due to the fact that all of the emitters have the same specification, simulating just the array factor would make computing easier than calculating the total field pattern for every single optimization process.

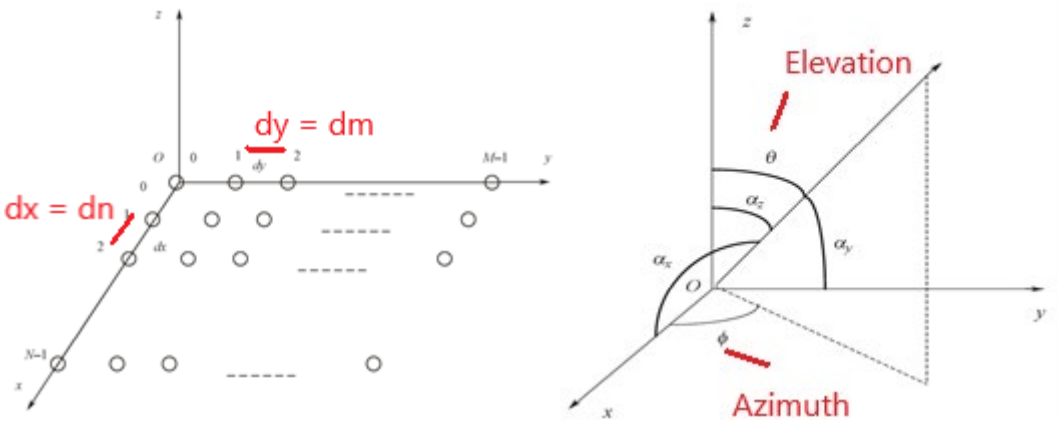
2.1.1 Synthesis of the Array Factor of Rectangular Array

Arrays of antennas are used to form a radiated power at a desired angular sector. The number, geometrical distribution, and the amplitudes and phases weighting of the antenna must be carefully designed to achieve the desired results [27]. The fundamental principle of the phased array to form a directed and desired beam is that the different distribution of antennas will make the different translational phase shift in the radiation

vector. Some of them have constructive interference, and some of them have destructive interference. The array factor can be characterized by the number of elements, the excitation of the elements (a_n) such as the magnitude of the electric field, and the geometric arrangement of the elements. The general equation for the array factor can be written as below.

$$F(k) = a_0 e^{jk*d_0} + a_1 e^{jk*d_1} + a_2 e^{jk*d_2} + \cdots \quad (2.1)$$

Applying equation 2.1 to the rectangular phased array. Assuming the phased array has the uniform amplitude weighting, which is 1, as shown in the figure below.



a) *b)*

Figure 2.2: a) Rectangular phased array's geometrical arrangement. b) *Coordinate system of the rectangular phased array* [28].

Equation 2.1 represents the total array factor which the term e^{jk*d} stands for the

relative phase between every two emitters. The wave vector can be expressed as $\frac{2\pi}{\lambda} * cos\theta$ which the term λ stands for wavelength and θ represents the angle of arrival with respect to the z-axis. Thus, the relative phase among the antennas placed on the rectangular phased array [28] can be expressed as

$$\begin{cases} \Delta\Phi_x = \frac{2\pi}{\lambda} d_n cos\alpha_x \\ \Delta\Phi_y = \frac{2\pi}{\lambda} d_m cos\alpha_y \end{cases} \tag{2.2}$$

Where *dn* and *dm* stand for the element spacing regarding to the x-axis and y-axis respectively. According to the geometrical knowledge in figure 2.2 (b), the terms for the angle of arrival [28] can be extended as

$$\begin{cases} cosa_x = sin\theta sin\Phi \\ cos\alpha_y = sin\theta sin\Phi \\ cos\alpha_z = cos\theta \end{cases} \tag{2.3}$$

From equation 2.1, the array factor is the summation of the relative phase with respect to one central emitter of all of the emitters multiply its each amplitude weights, assume the weights are all uniform which is 1 for all of the antennas, the array factor of the regular rectangular phased array can be found in terms of θ and ϕ as

$$F(\theta,\phi) = \sum_{y=0}^{Y-1} e^{j\frac{2\pi}{\lambda}dy(sin\theta sin\phi - sin\theta_0 sin\phi_0)} * \sum_{x=0}^{X-1} e^{j\frac{2\pi}{\lambda}dx(sin\theta cos\phi - sin\theta_0 cos\phi_0)} \tag{2.4}$$

The θ and ϕ in equation 2.4 represent the elevation and azimuth angle in the

celestial coordinate system, where θ_0 and ϕ_0 are the steering angles.

After the formula of the array factor is expressed, we can implement the formula into MATLAB. The operating wavelength of the OPA is at 1550 nm, and in order to avoid the grating lobes, the element spacing between each antenna has been all set to half of the operating wavelength, and there are 100 emitters uniformly distributed by 10 columns and 10 rows. The array factor in the UV space can be found.

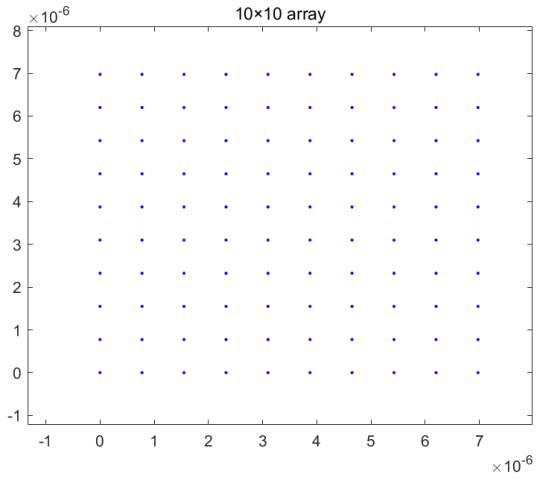
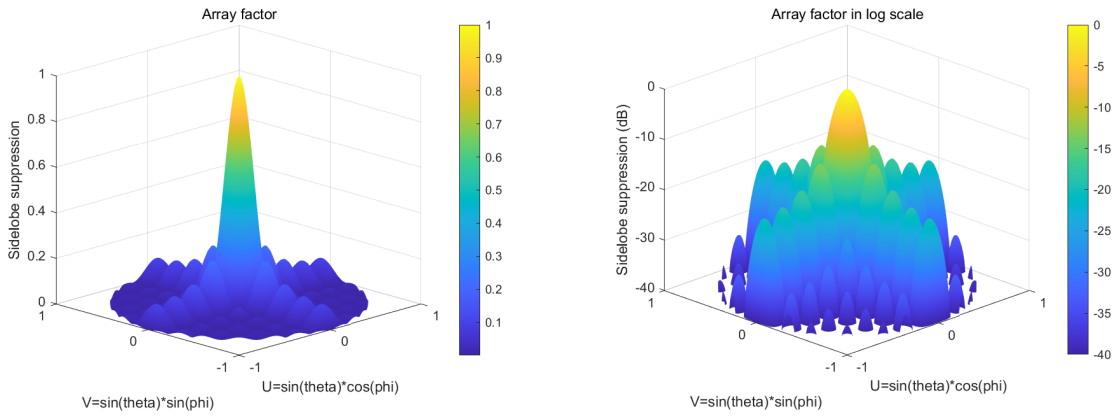


Figure 2.3: Geometry of 10 by 10 rectangular phased array with element spacing of half wavelength



a) *b)*

Figure 2.4: a) Array factor in the uv space, b) Array factor in log scale

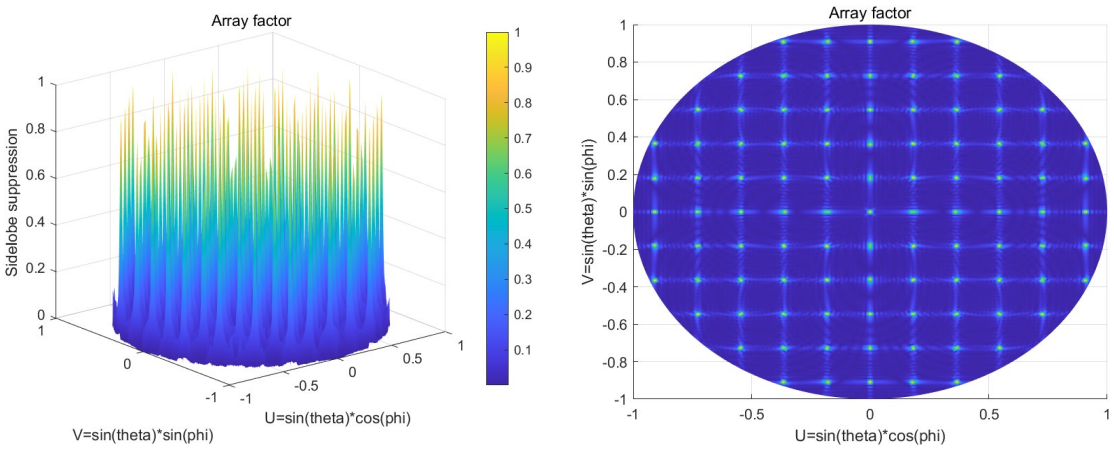
From the results in figure 2.5, we can see that none of the grating lobes appeared with

element spacing of half of the operating wavelength.

2.2 Implementation of a Rectangular Phased Array

The L-shaped SWG (subwavelength grating antenna) introduced in figure 1.5 of chapter 1.6 will be implemented in this design since it achieved high diffraction efficiency of 0.89 and directionality of 0.94. However, the antenna itself has the size of 7.6μm by 4.5μm

[8], which exceeds the operating wavelength at 1.55μm. Therefore, placing this size of the optical emitter at half of the operating wavelength is impossible. In order to safely place all the optical emitters without overlap, the minimum distance for both x-direction and y-direction will need to be set at 12μm since antennas will be rearranged within a 4μm x 4μm design space with respect to the central point of the antenna for the optimization.



a) *b)*

Figure 2.5: a) Array factor of large element spacing phased array in the uv space. b) Top view of the array factor

From the results in figure 2.5, we can see that many grating lobes appear in the far-field

distribution due to the large spacing of the emitters. The rectangular array is a two-dimensional linear array, the condition for the grating lobes of the uniform linear array (ULA) can also be applied to the rectangular array where equation 2.10 in section 2.2.2 explained the appearance of the grating lobes for the ULA in detail. Thus, figuring out the conditions for grating lobes to appear in a linear array would help us to understand the grating lobes in the rectangular array.

2.2.1 Uniform Linear Array

A ULA is consisted of N radiating elements on an axis, and each element is separate by a given distance.

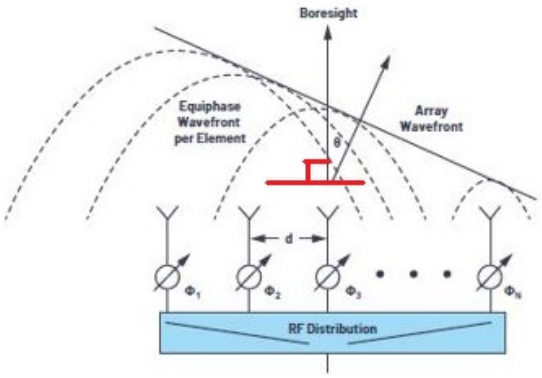


Figure 2.6: Example of uniform linear array [7].

Based on the equation 2.1, we know that the array factor is the sum of relative phase between the antennas, so the array factor of a ULA [14] can be derived as

$$F(\theta) = \sum_{n=1}^{N} a_n e^{j(2\pi nd*cos\theta)/\lambda} \quad (2.5)$$

where a_n is the excitation of each antenna, such as the magnitude of the electric field, d is the distance between the antenna, and λ is the operating wavelength. For better understanding the grating lobes, the following examples illustrate how grating lobes appear with increasing element spacing of a 10 element ULA.

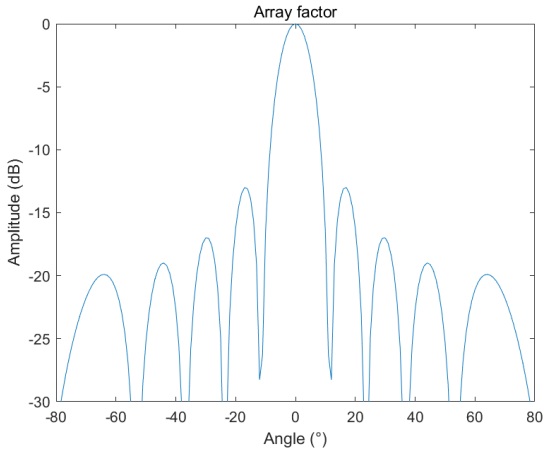


Figure 2.7: Uniform linear array with element spacing of 0.5 wavelength.

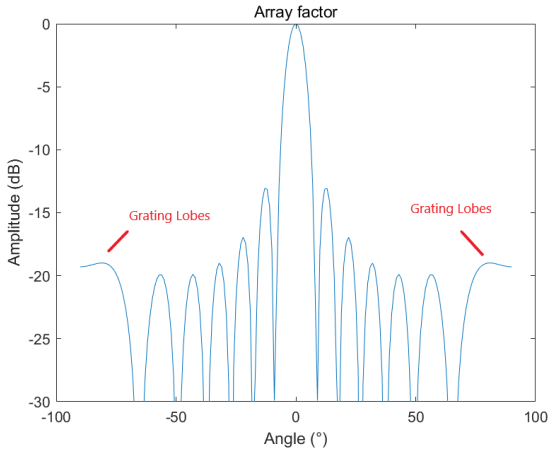


Figure 2.8: Uniform linear array with element spacing of 0.66 wavelength.

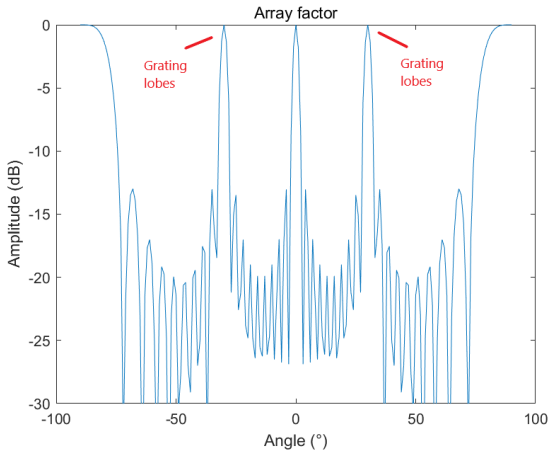


Figure 2.9: Uniform linear array with element spacing of 2 wavelength

From figure 2.9 to 2.10, the grating lobes start to appear when the element spacing is slightly above half of the operating wavelength. The grating lobes then became more obvious when the element spacing is at 2 times the operating wavelength. The following section mathematically explained the conditions for the grating lobes.

2.2.2 Grating lobes of Uniform Linear Array

Grating lobes can be seen as an analog to the digital sampling system. The Nyquist-Shannon sampling theorem (Nyquist) stated that there would be aliasing of the signal when the sampling frequency is greater than half of the original signal's frequency. Thus, we can analog the digital sampling system as the spatial sampling system for the phased array, in which each element's wavefront is sampled. The Nyquist theorem can be extended to this case, where the distance between adjacent elements need to be less or equal to half of the operating wavelength to avoid spatial aliasing (grating lobes) [29].

Before we dig into the rectangular phased array, we shall focus on the linear phased array since the rectangular phased array is a two-dimensional linear phased array.

Assume we have a ULA, and we have the trigonometry between the two elements, as shown in the figure below. The mechanical boresight is the angle that is perpendicular to the line formed by multiple antennas.

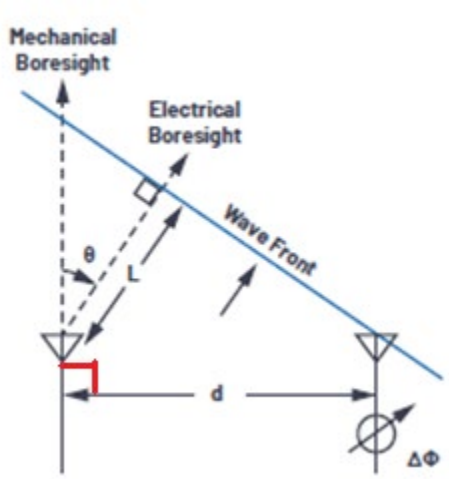


Figure 2.10: Trigonometry of the linear phased array [7].

From figure 2.10, the phased shift ΔΦ required to steer the beam to an angle θ is

$$\Delta\Phi = \frac{2\pi d sin\theta}{\lambda} \tag{2.6}$$

Inversing the formula 2.6 will give us the function of phase shift with respect of the beam angle

$$\theta = arcsin\left(\frac{\Delta\Phi}{2\pi} \times \frac{\lambda}{d}\right) \tag{2.7}$$

The equation 2.7 is a repeating function which it repeats itself every 2π, so the equation 2.7 could be rewrite to

$$\theta = arcsin\left(\frac{m \times 2\pi + \Delta\Phi}{2\pi} \times \frac{\lambda}{d}\right) \quad (2.8)$$

where m = 0, ± 1, ±2 ….

Mathematically speaking, to avoid the appearance of the grating lobes, it is necessary to obtain a single real solution for equation 2.10. The arcsin function only produces real solutions for the argument between -1 and 1, so if we obtained multiple real solutions for the arguments between the range of -1 and 1, that means there will be grating lobes.

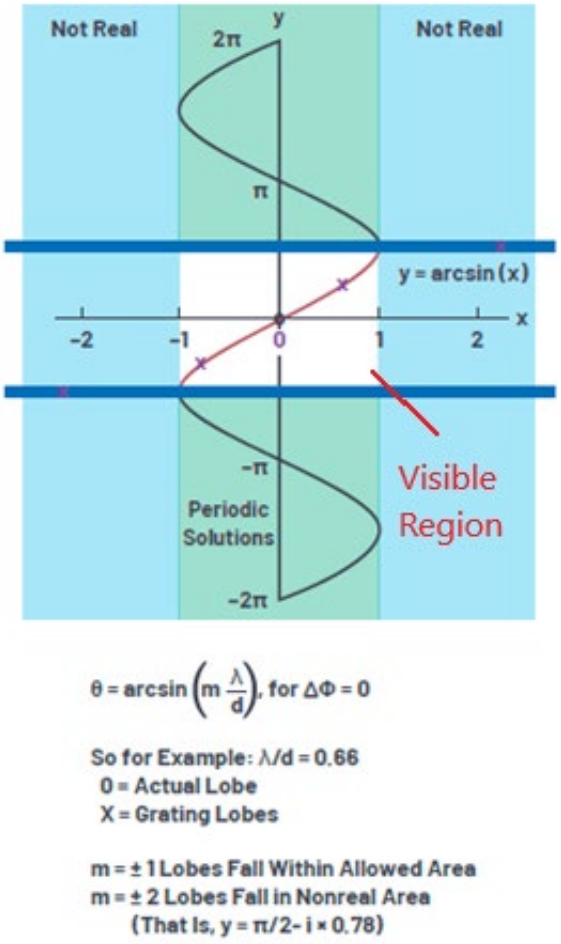


Figure 2.11: Example of arcsin function [29]

According to figure 2.12, when the λ/d is at 0.66 which the element spacing is greater the half of the operating wavelength, there are multiple real solutions that fall in the allowed area (visible region), which indicates the appearance of the grating lobes [29].

2.3 Optimization of the Rectangular Phased Array

In section 2.2.2, the principle of grating lobes for the linear phased array was revealed, where this principle can be applied to the rectangular phased array because the rectangular phased array is a two-dimensional linear phased array.

In an equally spaced rectangular phased array, the element spacing is constant, so the fully constructive interference from different radiating elements would produce the grating lobes. However, breaking the rectangular symmetry would change the element spacing to a varying number, which the grating lobes will not result in particular constructive interference from emitters [21].

Therefore, applying the GA to redistribute the position vertically or horizontally for each antenna will break the symmetry of the rectangular phased array, which will eliminate the grating lobes.

The GA is an important tool for optimization and search problems. Yet, breaking the symmetry will indeed eliminate the grating lobes, but if we do not apply the GA, we will likely have a bad sidelobe suppression performance. The following example simulates the far-field pattern of the array factor after using GA. First, we need to decide the number of populations (chromosomes) that perform the optimization with the

minimum time where each chromosome represents the position of the antennas. The following figure illustrates the consistency of the optimization results with different population sizes. The crossover rate and mutation rate are kept at 0.995 and 0.0065,respectively, because the general rule for the GA is to use the high crossover rate and low mutation rate [2].

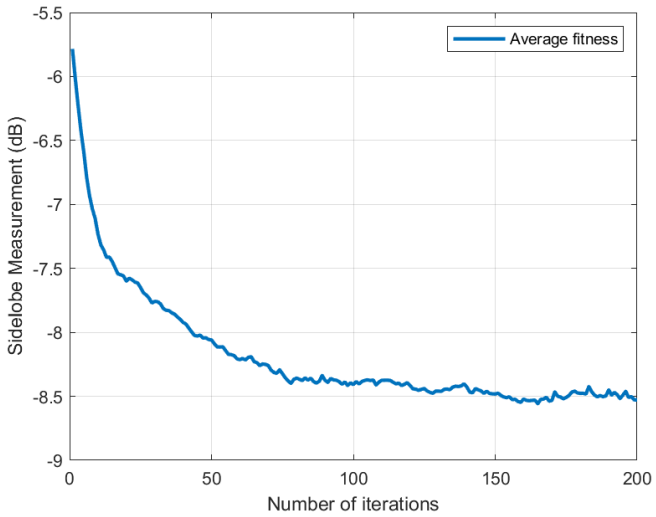


Figure 2.12: Sidelobe measurement with 200 populations

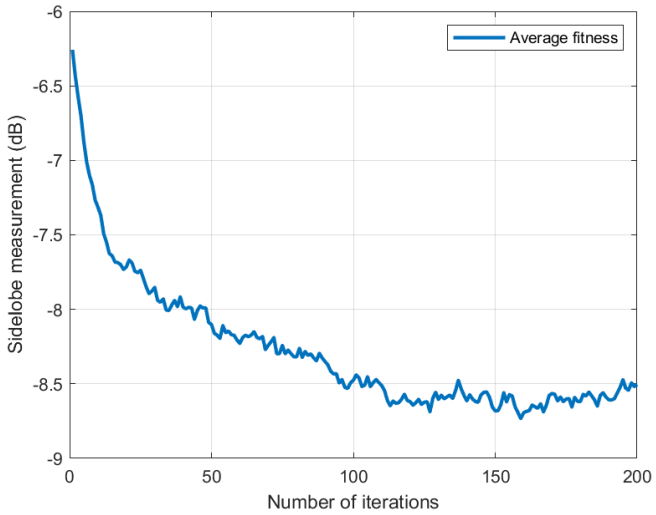


Figure 2.13: Sidelobe measurement with 100 populations

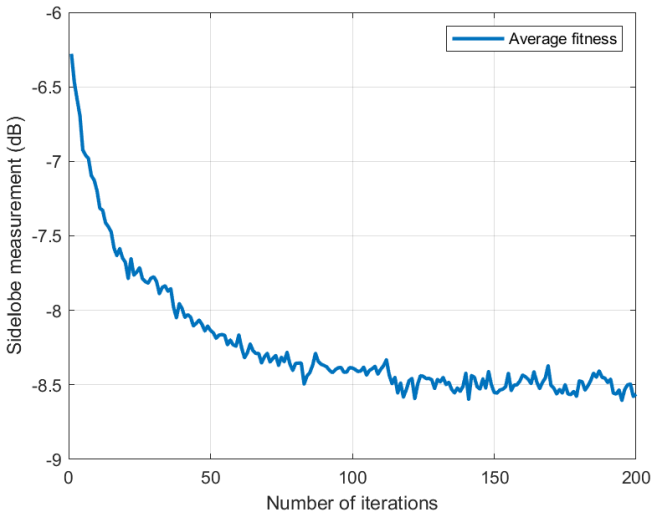


Figure 2.14: Sidelobe measurement with 50 populations

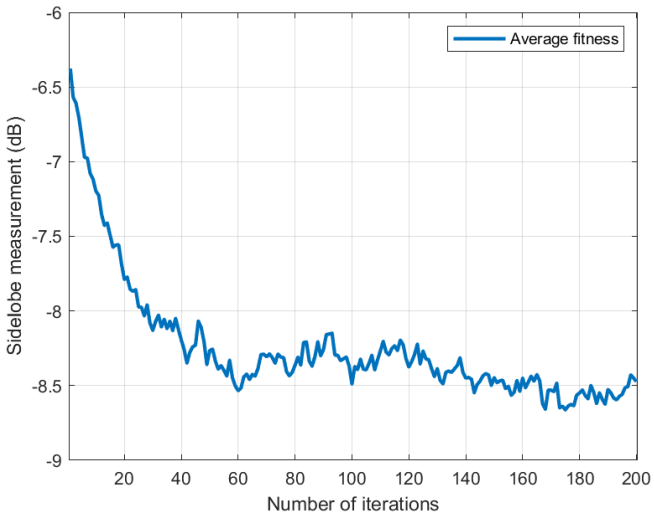


Figure 2.15: Sidelobe measurement with 30 populations

We can see that all the optimization with 30 populations performs almost the same among the other populations. Therefore, for reducing the simulation time, the number of populations will be chosen as 30 for all the optimizations performed in the thesis. The next step is to binary encode the chromosomes, which depends on the number of antennas. For instance, a regular rectangular phased array with 100 antennas will be

encoded into 2000 binary bits. In that, each antenna will have 10 bits resolution of the moving range for both x and y directions. If the moving range for each antenna in both x and y directions is from -2μm to 2μm (total in 4μm), then the resolution will be $\frac{4\mu m}{2^{10}}$, which is 0.000390625μm. Afterward, a cost function that measures the maximum sidelobe of a particular position of the antennas is developed (please see Appendix B, func_obj). Then, a set of chromosomes will go over the process accordingly: ranking (based on their performance on sidelobe suppression), selection, crossover, mutation, converting back into real vectors, remeasure the maximum sidelobe, and reinserting the offspring in the population. The ending condition for this regular rectangular phased array is at -10dB. Otherwise, the iteration will keep going 200 times.

Note that the following lists contain the genetic algorithm toolbox developed at the Department of Automatic Control and System Engineering of The University of Sheffield, UK.

- crtbp.m: Create an initial population (Appendix D)
- ranking.m: Performs ranking of individuals (Appendix E)
- sus.m: Performs selection with stochastic universal sampling (Appendix F)
- xovsp.m: Performs crossover or swapping of the bits between pairs of individuals based on the crossover rate (with lower-level function, Appendix G)
- mut.m: Performs mutations of individual based on the mutation rate (Appendix H)
- bs2rv.m: Decodes binary chromosomes into real vectors (Appendix I)
- reins.m: Reinserts offspring in the population (Appendix J)

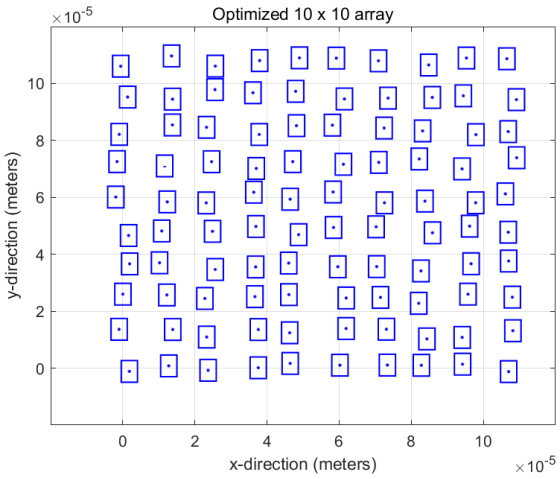


Figure 2.16: Redistributed rectangular phased array

The blue dots in figure 2.16 represent the single point source, and the blue rectangles stand for the actual size of the emitter. There are 100 antennas in total, and the original element spacing was set at 12μm to provide enough spacing to fulfill and rearrange the 100 antennas. Afterward, implement the GA to readjust the position for each antenna within the range of 2μm to ensure that antennas would not overlap. The array factor of redistributed phased array is shown in the figures below.

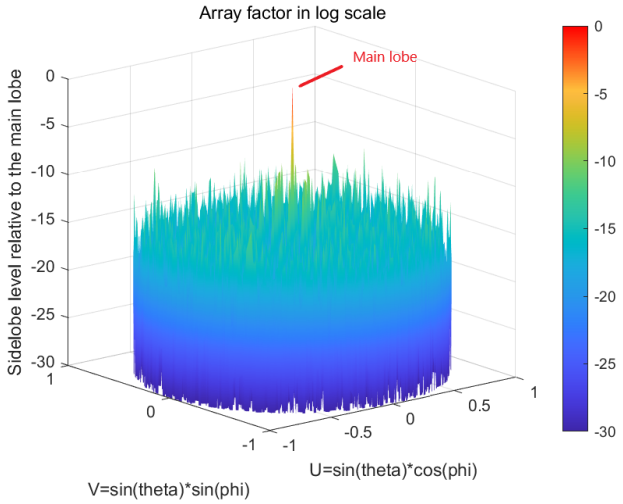


Figure 2.17: Optimized 10 by 10 rectangular phased array factor in log scale

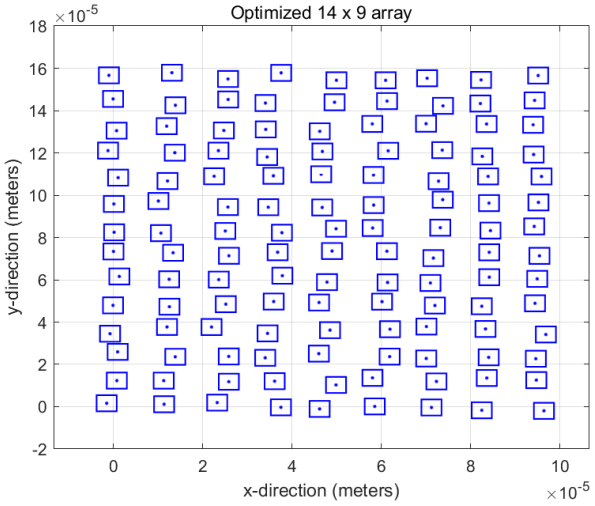
After redistributing the radiating elements, the grating lobes are eliminated, and the

maximum sidelobe is at -8.67dB. From the results, applying GA is able to eliminate the grating lobes and suppress the sidelobes of the array factor effectively. Thus, the following simulations for rectangular phased arrays with different numbers of antennas, which are 126, 225, 324, 423, 513 antennas, will be performed. Several parameters will be recorded in order to compare with the circular and the random arrays in later chapters, such as the minimum design space required to fully place a given number of antennas without overlap and sidelobe suppression in the total far-field pattern. The total far-field pattern can be obtained by multiplying the array factor and element pattern.

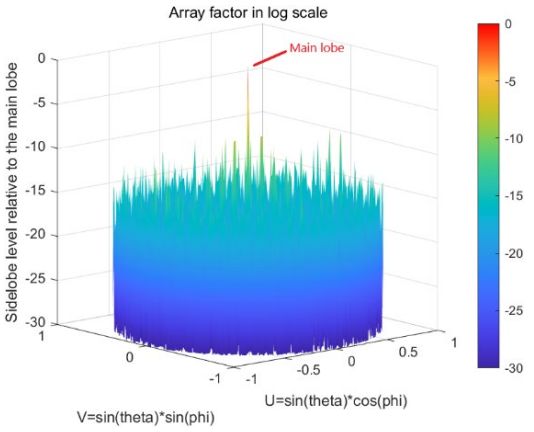
2.4 Simulation Results

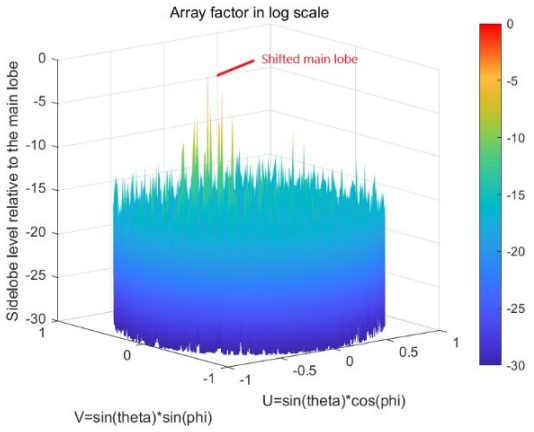
Based on the pattern multiplication rule, the total far-field pattern of an array is the product of the array factor and element pattern. In fact, the antenna itself has a 23° diffraction angle θ, the main lobe of the array factor then must be shifted 23° of θ angle to have two maxima (generated from phased array and antenna) multiply each other to obtain the total maxima. If we do not steer the main lobe of the array factor, the total pattern will have a null at the original position of the main lobe of the array factor. For all the simulations, the main lobe of the array factor would be steered 23° of θ angle first, then multiplied it to the element pattern to obtain the optimized total pattern.

2.4.1 Rectangular Phased Array with 126 (14 by 9) Antennas



a)





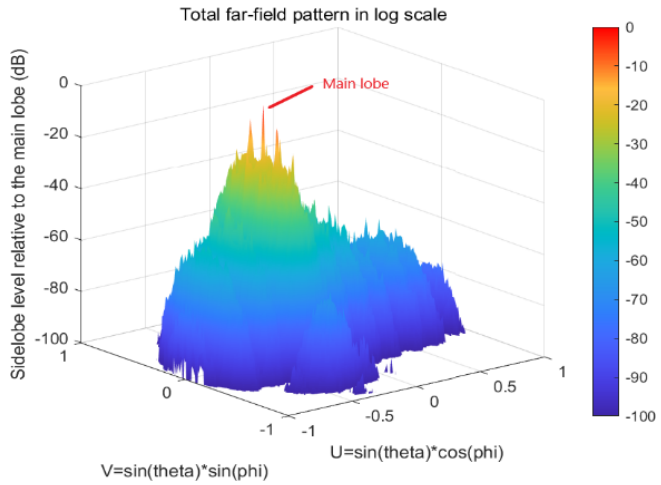
b) *c)*

Figure 2.18: a) Geometry of optimized 14 columns by 9 rows rectangular phased array. b) Array factor in log scale. c) Array factor in log scale with main lobe steered 23° in θ angle

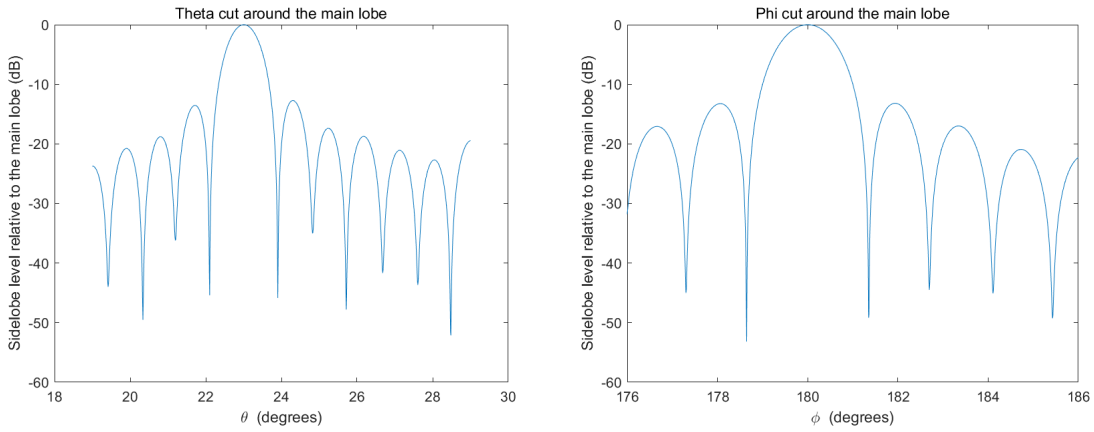
Figure 2.18 b) illustrates the radiation pattern of the array factor, where the array factor is independent of the antenna design, so it can be seen as a scaler function of a single element pattern. Figure 2.18 c) is the array factor with a scanning angle of 23° in θ, so it is dependent on the relative phase difference of antennas to steering the beam. The minimum chip size to fulfill 126 antennas is around 105 by 165 microns. The maximum

sidelobe level (SLL) relative to the main lobe in the array factor without beam steering is -7.63dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -2.23dB

Total far-field pattern



a)



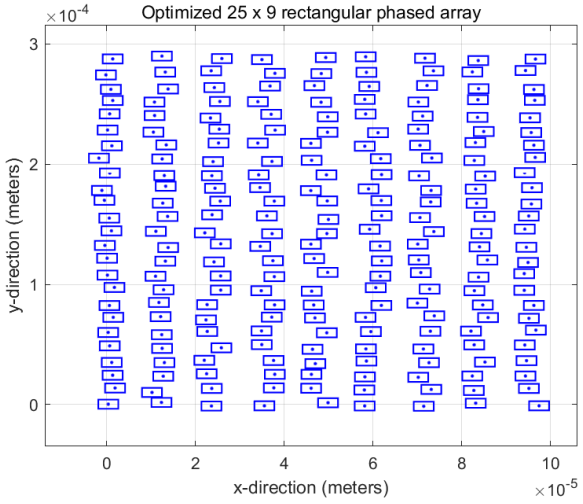
b) *c)*

Figure 2.19: a) Total far-field pattern in log scale b) θ cut of 10 degrees around the main lobe. c) ϕ cut of 10 degrees around the main lobe.

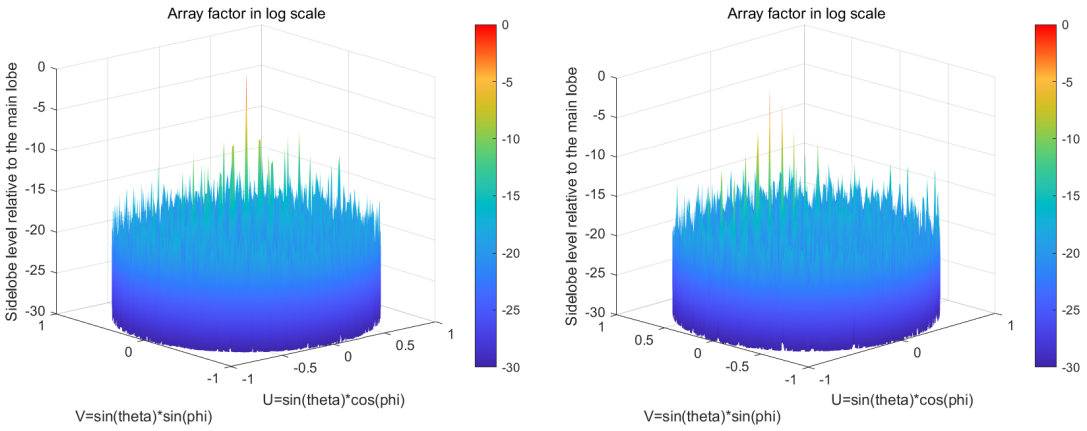
Figure 2.19 a) illustrates the total far-field radiation pattern, where it is dependent on the antenna design. The maximum SLL relative to the main lobe in the total far-field

pattern is -8.36dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the full width at half maximum (FWHM) with a higher solution. The FWHM is then calculated from figure 2.16 b and c, which is 0.81° in θ angle and 1.21° in ϕ angle.

2.4.2 Rectangular Phased Array with 225 (25 by 9) Antennas



a)

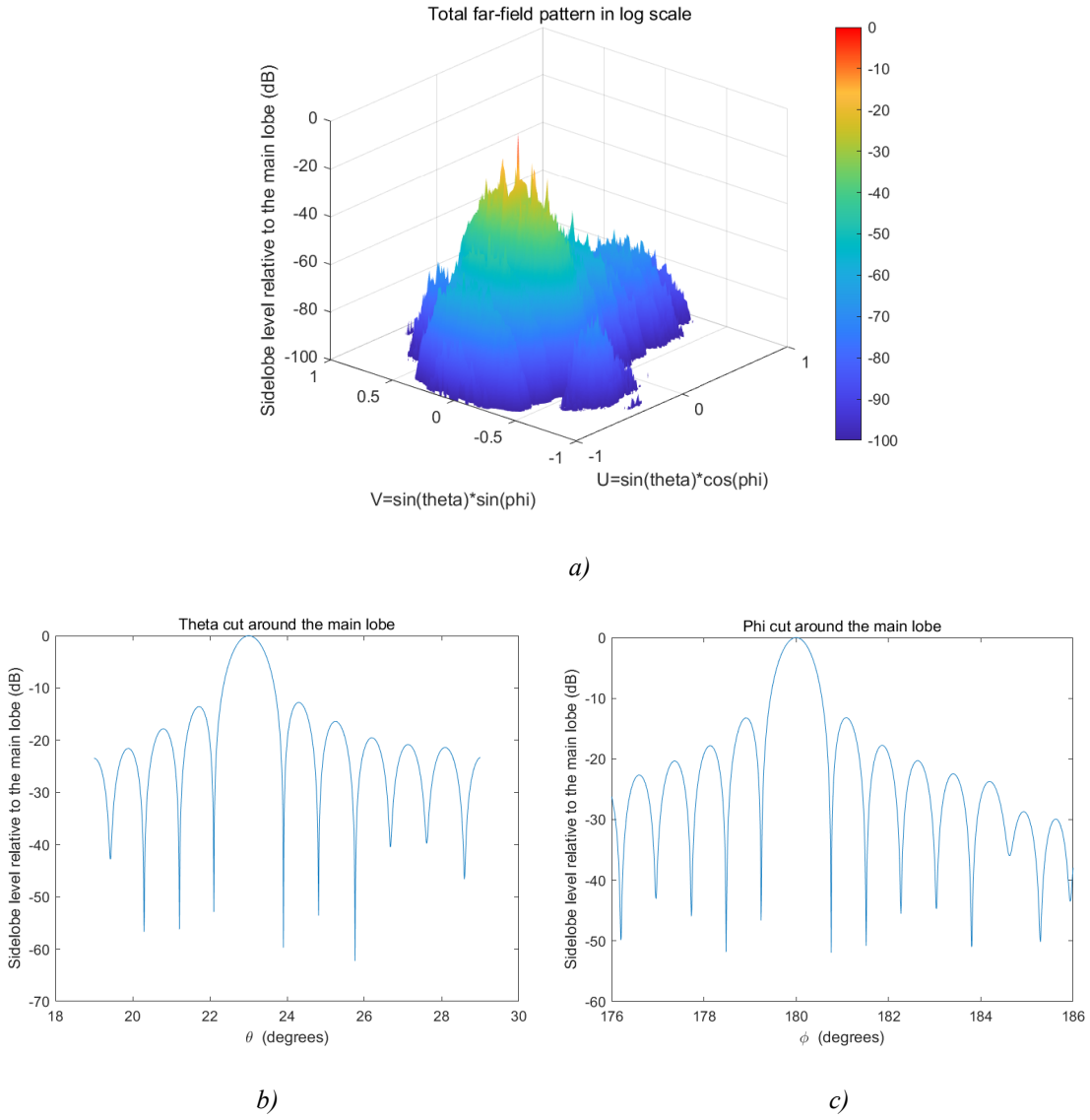


b) *c)*

Figure 2.20

The chip size is around 105 by 290 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -8.13dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -2.23dB.

Total far-field pattern



a)

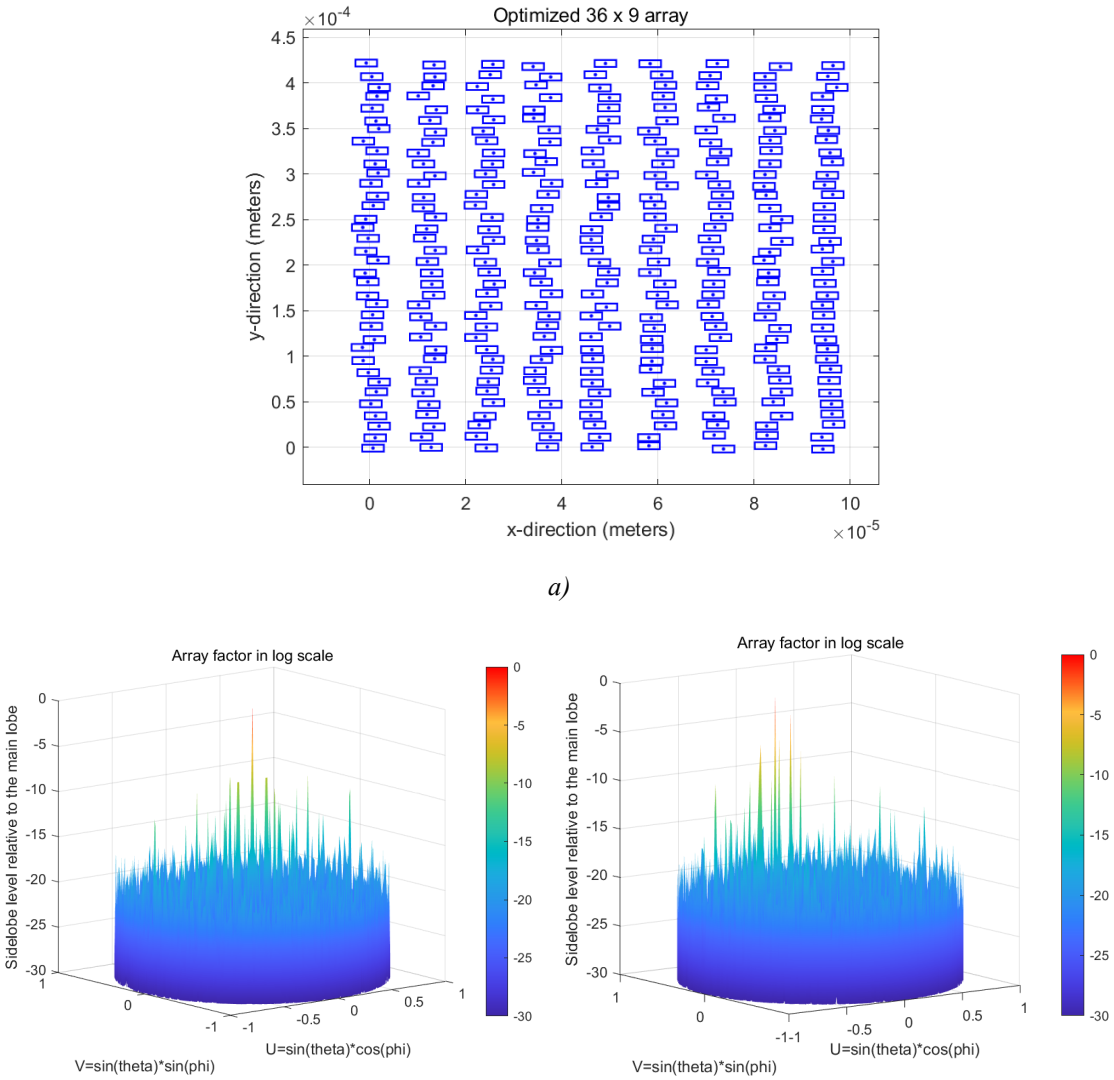
b)

c)

Figure 2.21

The maximum SLL relative to the main lobe in the total far-field pattern is -12.77dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 2.18 b and c, which is 0.80° in θ angle and 0.69° in ϕ angle.

2.4.3 Rectangular Phased Array with 324 (36 by 9) Antennas



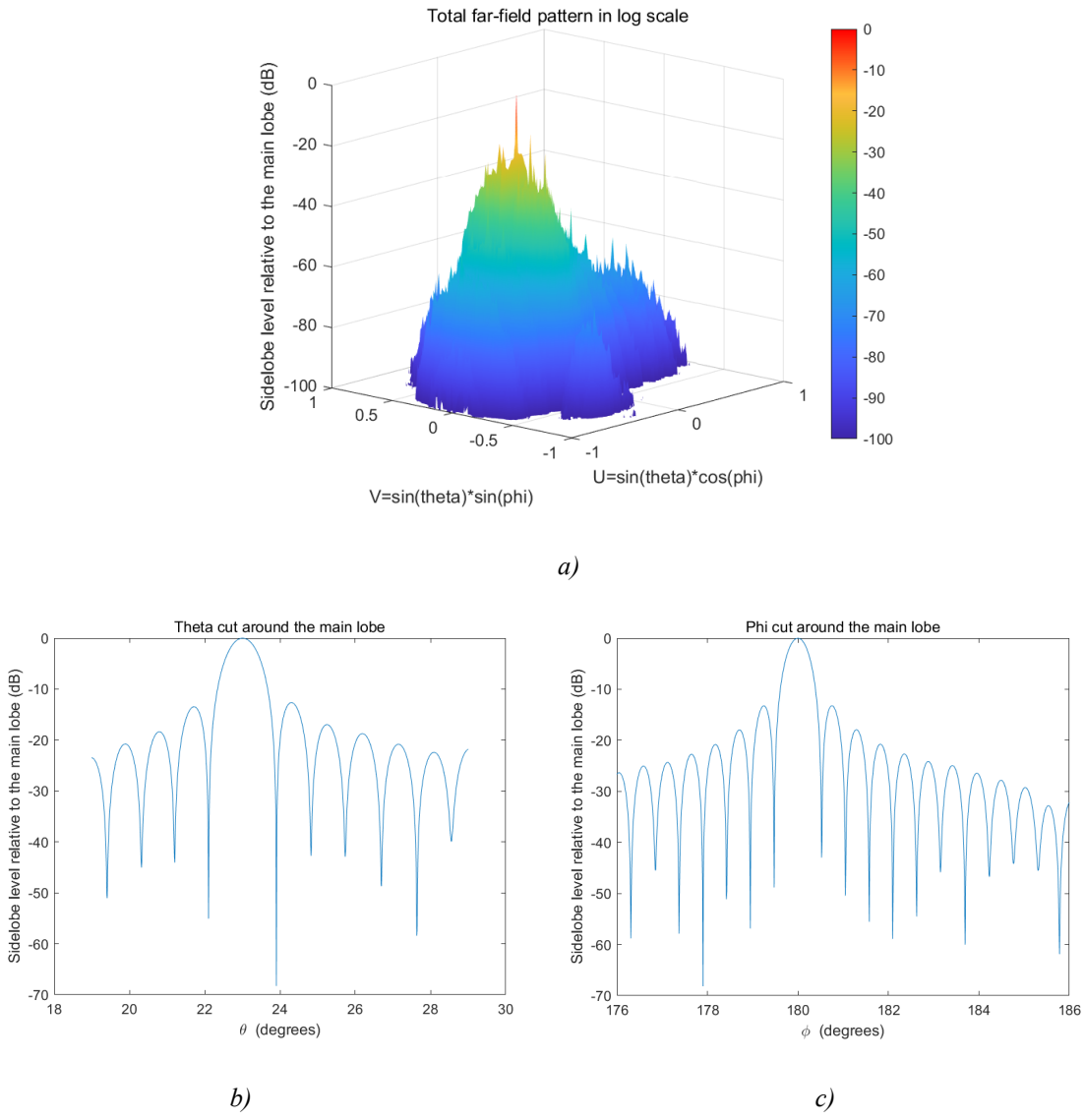
a)

b) *c)*

Figure 2.22

The chip size is around 105 by 435 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -8.39dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -2.29dB.

Total far-field pattern



a)

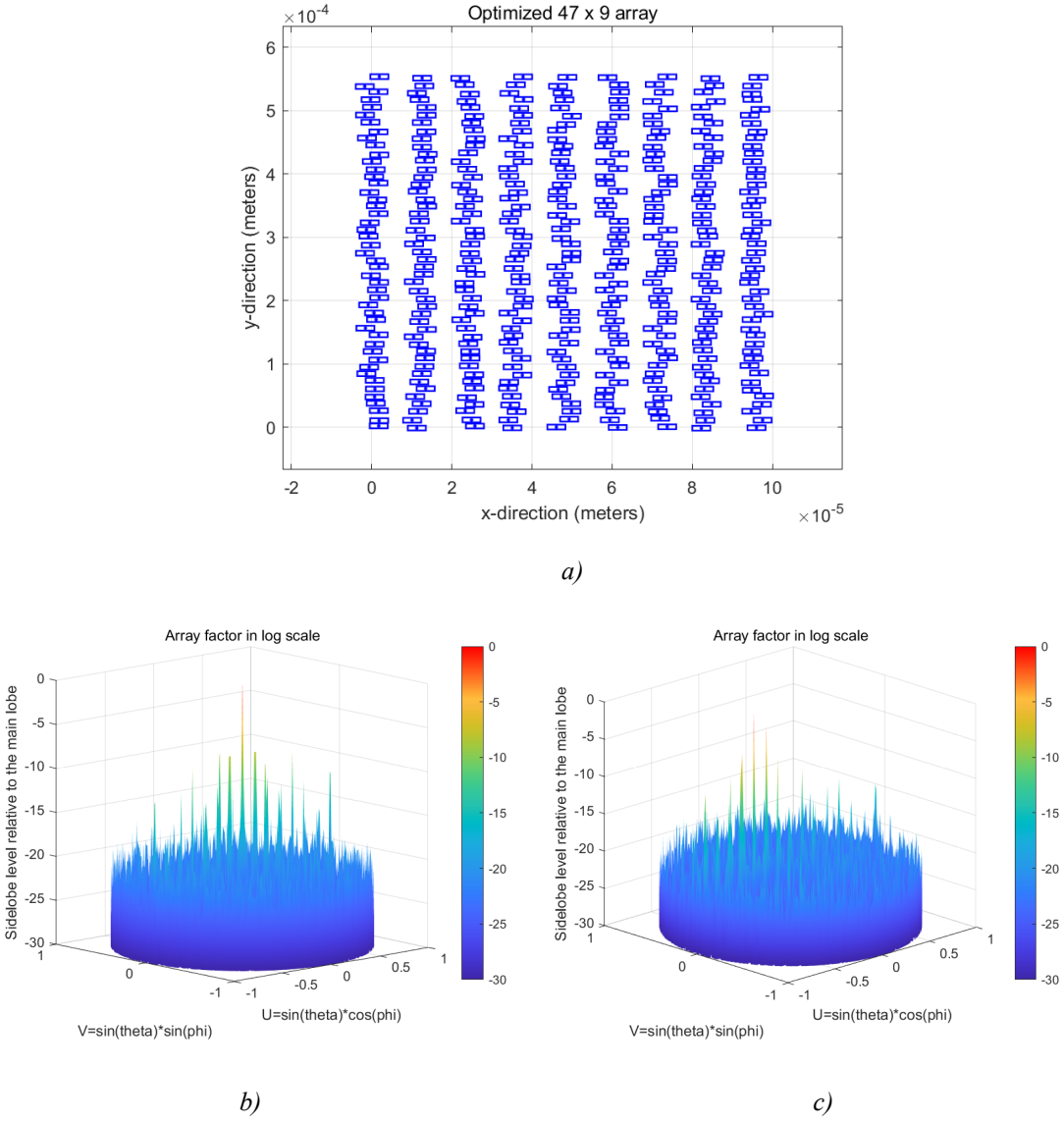
b)

c)

Figure 2.23

The maximum SLL relative to the main lobe in the total far-field pattern is -14.71dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 2.20, which is 0.80° in θ angle and 0.46° in ϕ angle.

2.4.4 Rectangular Phased Array with 423 (47 by 9) Antennas



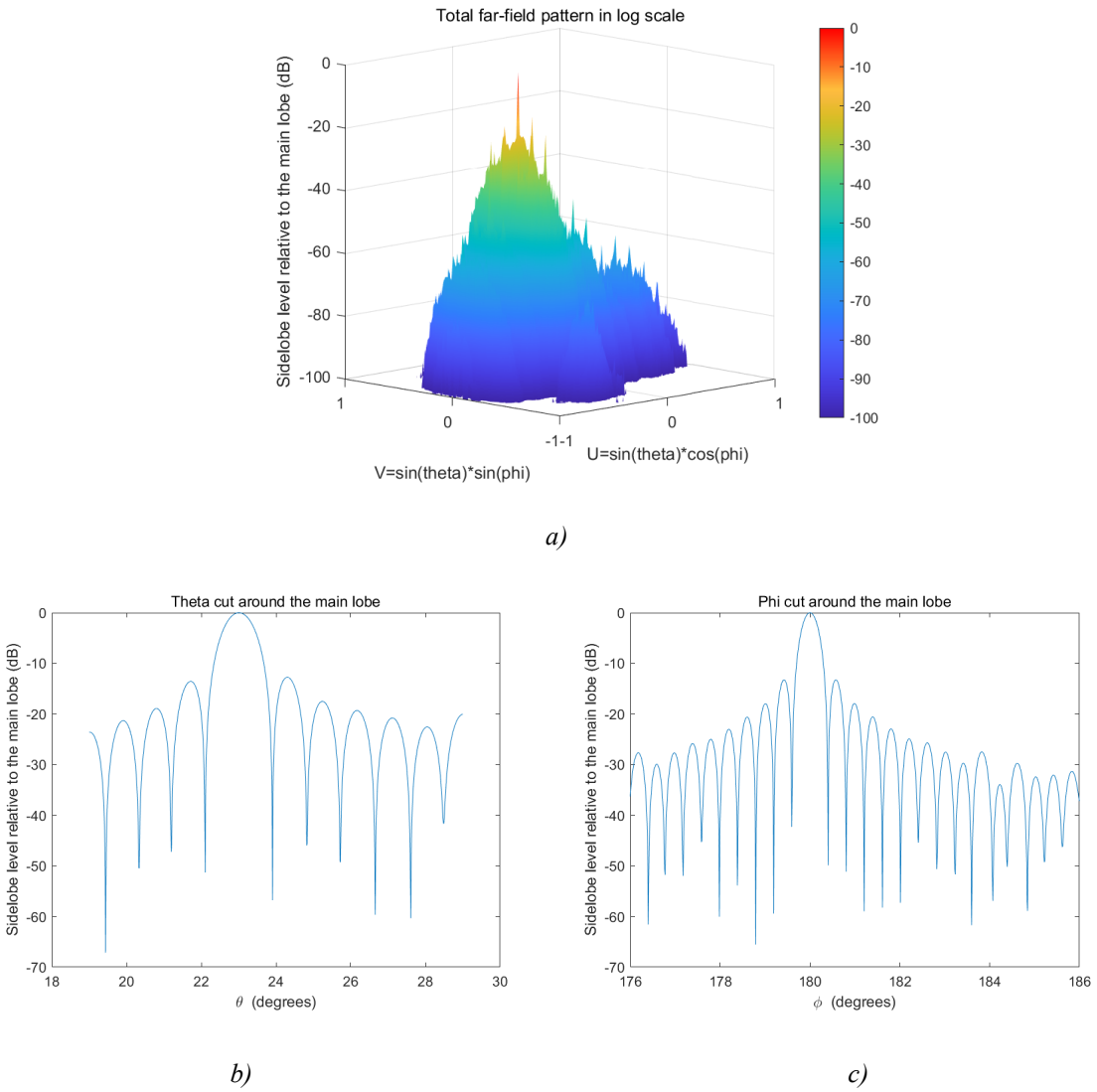
a)

b) *c)*

Figure 2.24

The chip size is around 105 by 558 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -8.89dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -2.21dB.

Total far-field pattern



a)

b)

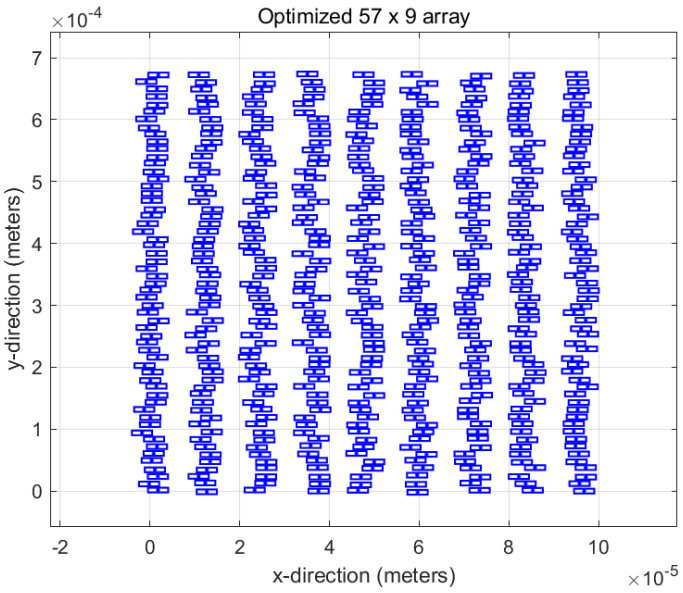
c)

Figure 2.25

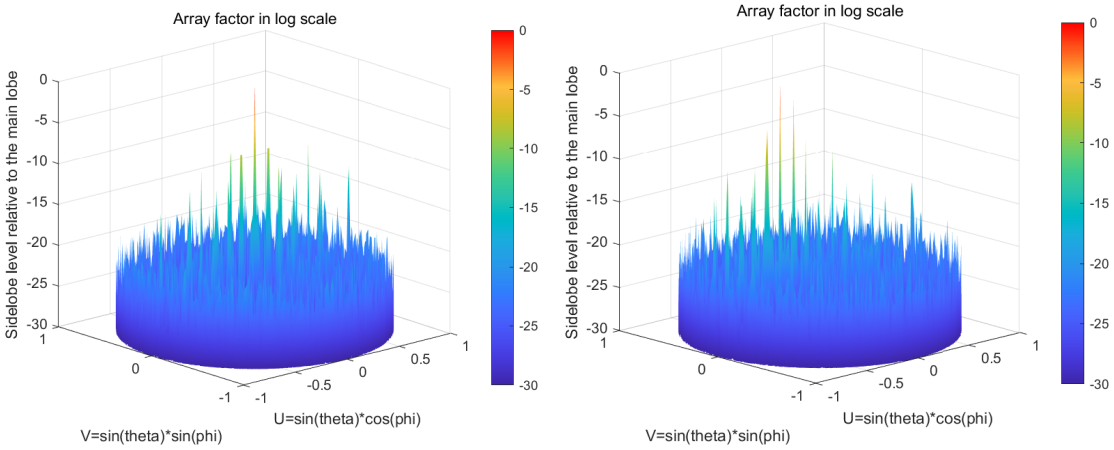
The maximum SLL relative to the main lobe in the total far-field pattern is -14.83dB

with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 2.22, which is 0.80° in θ angle and 0.35° in ϕ angle.

2.4.5 Rectangular Phased Array with 513 (57 by 9) Antennas



a)

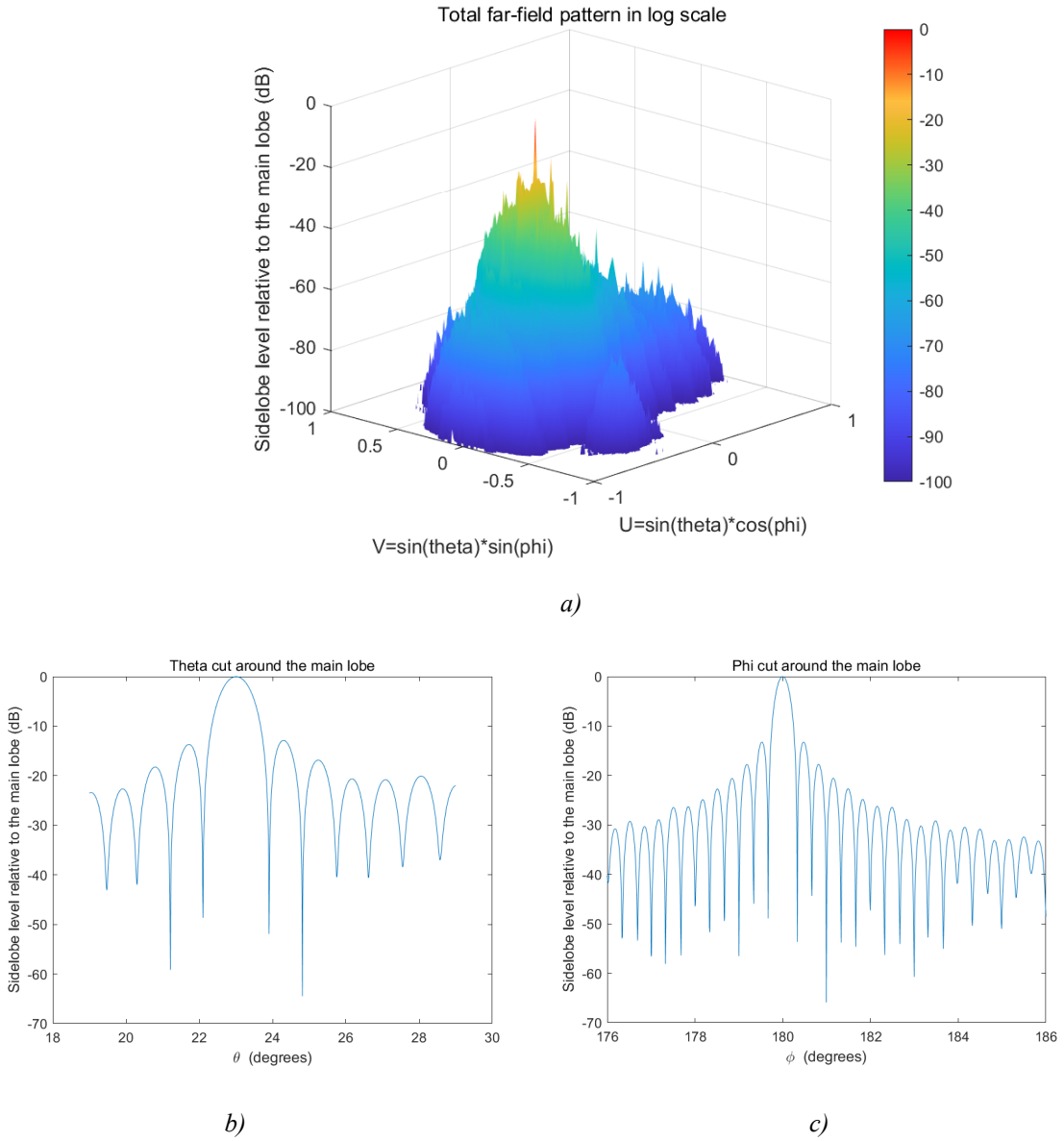


b) *c)*

Figure 2.26

The chip size is around 105 by 690 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -8.91dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -2.36dB.

Total far-field pattern



a)

b)

c)

Figure 2.27

The maximum SLL relative to the main lobe in the total far-field pattern is -15.19dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 2.24, which is 0.79° in θ angle and 0.30° in ϕ angle.

2.5 Summary

Applying the GA to redistribute the antennas can eliminate the grating lobes and further suppress the SLL since the symmetry of the rectangular phased array is broken.

By repeating the above-mentioned process for the rectangular arrays with a given number of antennas, we can obtain the several parameters shown in table 2.1 below. The chip size stands for the minimum design space required to fully place the given number of antennas, where it is used to compare the footprint of three different types of arrays with a given number of antennas. For the rectangular phased array, the FWHM at θ angle slightly decreased with an increasing number of antennas, the FWHM at ϕ significantly decreased with an increasing number of antennas due to the effective array length. The FWHM is related to the effective array length *Lx* and *Ly* in the x-direction and y-direction, where the beamwidth is equal to a number divided by the *Lx* and *Ly* [30]. Hence when the effective length increases significantly, the beamwidth will get narrower. Based on the geometry of the rectangular arrays, the *Lx* in the x-direction is slightly varied, so the FWHM in θ angle slightly changed. Additionally, the SLL is

further suppressed with an increasing number of antennas because a higher gain of the phased array is achieved with more antennas placed [30].

The simulation results will be used for the comparison with circular and random phased arrays in later chapters.

Table 2.1: Characteristics of the rectangular phased array

Number of antennas (Rows by Columns)	**Chip Size**	**Main lobe FWHM (θ)**	**Main lobe FWHM (ϕ)**	**SLL of total far-field pattern**
126 (14 by 9)	105 x 165 (microns)	0.81°	1.21°	-8.36(dB)
225 (25 by 9)	105 x 290 (microns)	0.80°	0.69°	-12.77(dB)
324 (36 by 9)	105 x 435 (microns)	0.80°	0.46°	-14.71(dB)
423 (47 by 9)	105 x 558 (microns)	0.80°	0.35°	-14.83(dB)
513 (57 by 9)	105 x 690 (microns)	0.79°	0.30°	-15.19(dB)

Chapter 3 :

Circular Optical Phased Array

3.1 Fundamentals

The circular phased array usually consists of multiple rings with numbers of antennas placed on each ring.

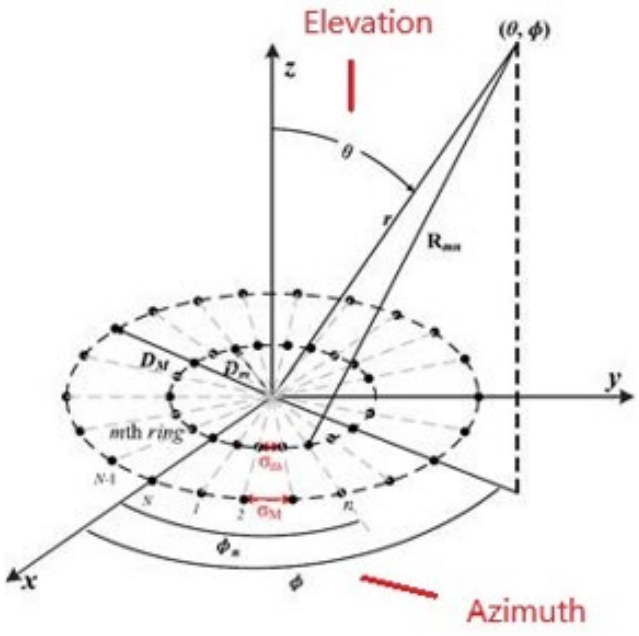


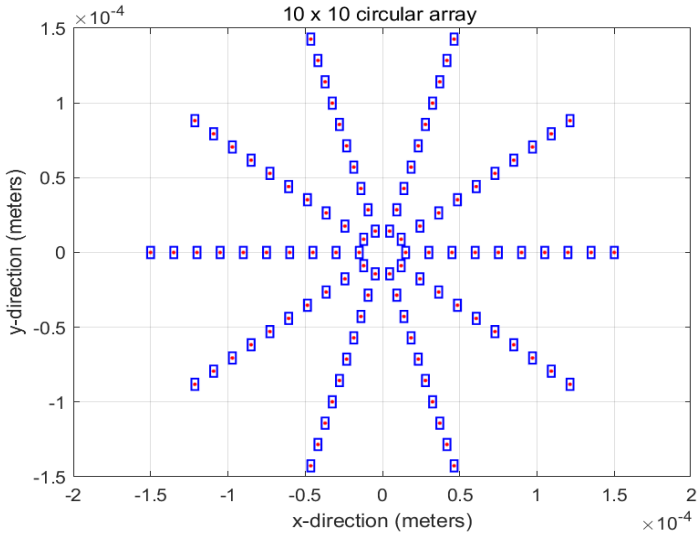
Figure 3.1: Example of circular phased array [25]

In figure 3.1, the antennas are placed on an x-y plane and equally spaced on each ring. The term D_m denotes the radius of the *m*th ring, and the term σ_m denotes the arc length (distance) between two adjacent antennas in the *m*th concentric ring. (θ, ϕ) denotes the far-field position in the spherical coordinate [25].

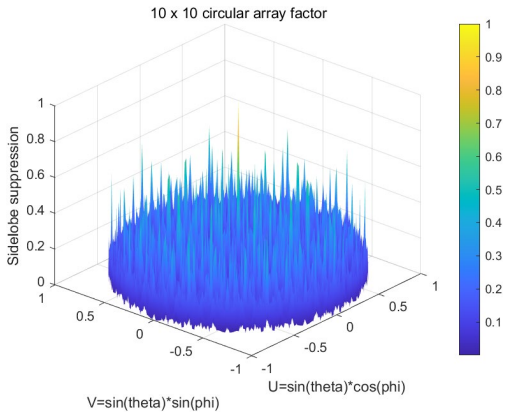
For the planar uniform circular array without a center antenna placed at the origin point, and assuming the amplitude weight is the same for each antenna, the array factor can be expressed as [24]

$$F(\theta,\phi) = \sum_{m=1}^{M}\sum_{n=1}^{N} e^{jkD_m \sin\theta \cos(\phi-\phi_n)} \quad (3.1)$$

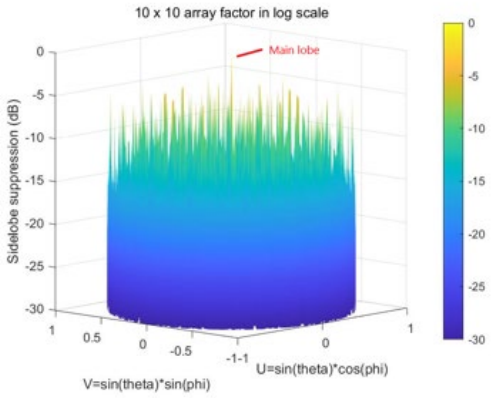
Where $\phi_n = 2n\pi/N$, it stands for the angular position of the element n at the *m*th ring, which N is the total antennas placed on the *m*th ring. Applying equation 3.1 to a 10 by 10 circular phased array which has 10 rings, and each ring has 10 antennas uniformly placed on it. Each ring and the central point of the antenna is at least separated by 15μm (the radius for each ring is 15 microns times m where m = 1,2,3…10) to provide sufficient spacing for further optimization.



a)



b)



c)

Figure 3.2: a) Regular 10 by 10 circular phased array. b) Array factor of the regular circular array in uv space. c) Array factor in log scale

The blue boxes in figure 3.2 a) represent the actual size of the antenna to ensure that antennas would not overlap on each other. The maximum SLL relative to the main lobe is -4.85dB.

Figure 3.2 b) showed that the circular phased array's geometrical arrangement promotes an unequal distribution of sidelobes compared to those of the regular

rectangular phased array, which naturally suppresses the grating lobes. However, the high-power sidelobes still exist in the far-field pattern of the array factor.

The geometry of the circular phased array affects the performance of sidelobe suppression [31]. Therefore, before applying GA to further optimize the circular array, it is necessary to investigate the impact on the sidelobe suppression caused by the geometries. After that, implementing a GA would then reduce the SLL furthermore.

3.2 Geometries of the Uniform Circular Phased Array

The geometry of the circular phased array plays an essential role in the sidelobe suppression performance. Assume the array has a radius of 15μm for the first inner ring, with an increment of 15μm in radius for each ring away from the adjacent ring. Varying N (number of antennas placed on each ring) and M (total number of rings) gives the following results.

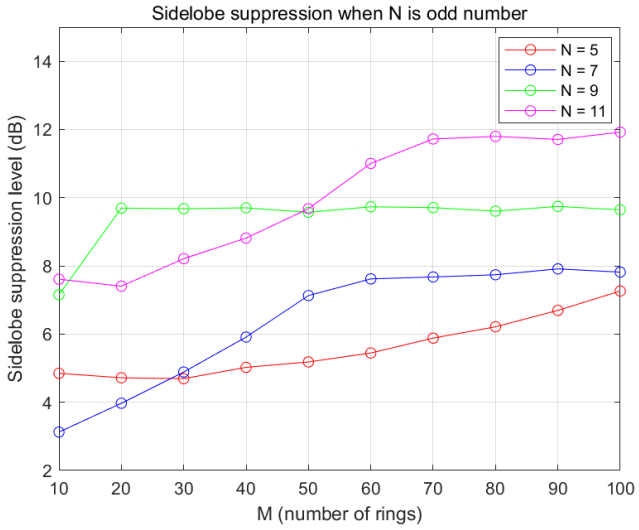


Figure 3.3: Sidelobe suppression with increasing M when N is odd

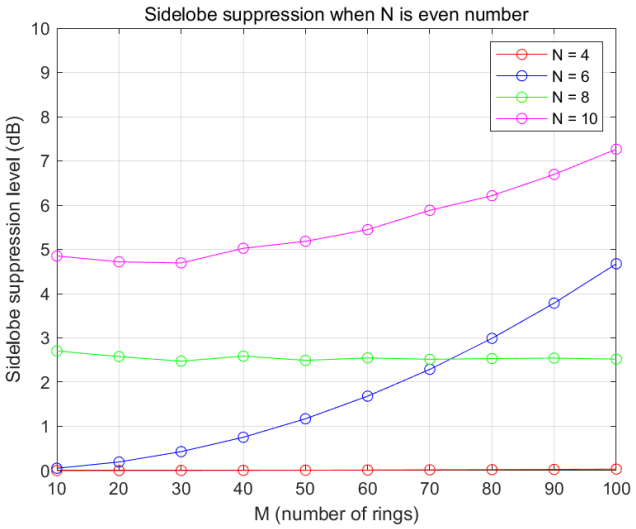


Figure 3.4: Sidelobe suppression with increasing M when N is even

Figure 3.4 illustrates that the circular phased array with N as an odd number has a better overall performance on sidelobe suppression than the circular phased array with N as an even number with a similar number of antennas are placed. This is mainly due to the bilateral symmetry of the circular phased array. When N is odd, the bilateral symmetry is broken, which results in reducing the amplitude of the grating lobes [31]. However, there are two special cases shown in figure 3.4, which are when N is equal to 4 and 8. The SLL does not significantly change no matter how M (total number of rings) adds up. In fact, any number N that is multiple integers of 4 could have this strange behavior, such as when N is equal to 12, 16, 20, 24, etc.

Hence, we shall first inspect the geometries of the circular arrays when N is equal to 4 and 8.

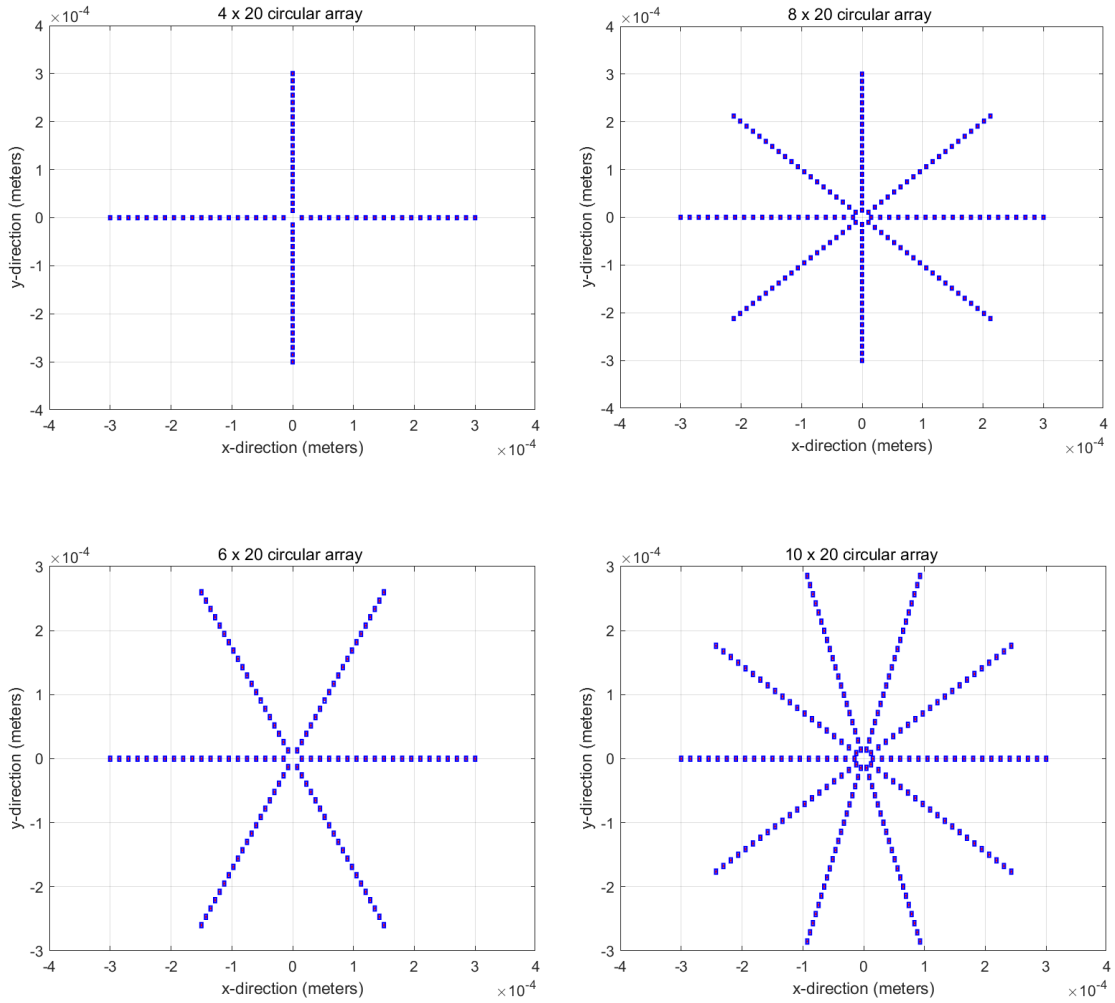
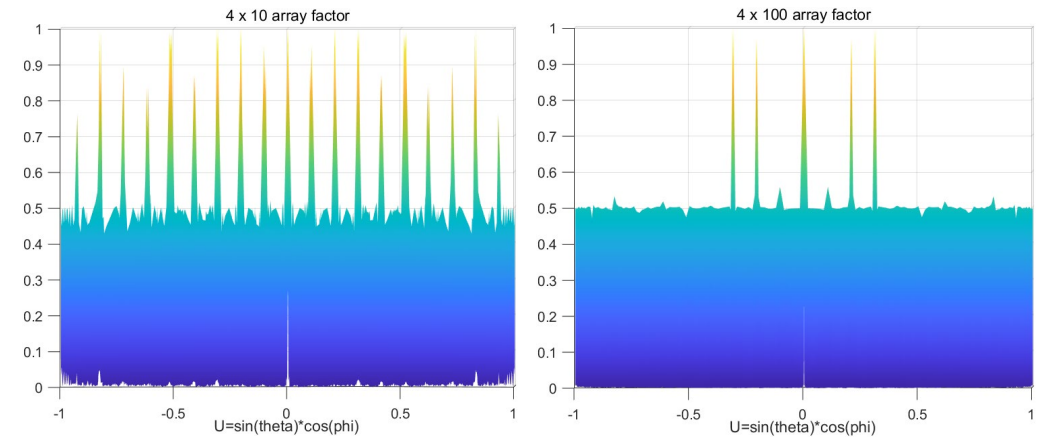
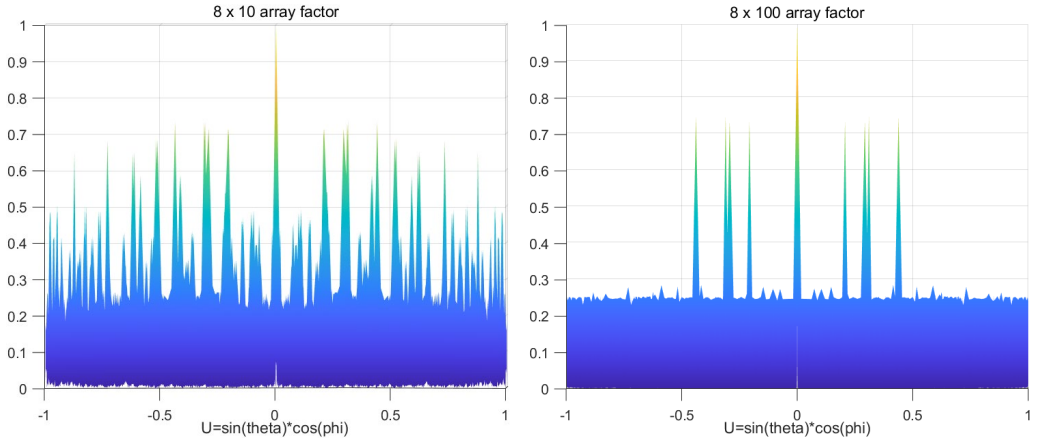


Figure 3.5: Different geometries of circular phased array

When N is equal to 4 and 8, the circular array will have the special geometry, in which they have an equal number of subarrays that lie along with the vertical, horizontal, and angular directions. This type of geometry is a bit similar to the "rectangular array." Thus, increasing M will no longer suppress the high-power maxima formed by the constructive interference at certain θ and ϕ angles, even though the average sidelobe is suppressed when M is increased from 10 to 100, as shown in the figures below.



a) *b)*



c) *d)*

Figure 3.6: a) Array factor of 4 by 10 circular array. b) Array factor of 4 by 100 circular array. c) Array factor of 8 by 10 circular array. d) Array factor of 8 by 100 circular array.

Theoretically, the circular phased array is more bilaterally symmetric when the N is an even number. There will be more constructive interference from different emitters if the circular phased array is bilaterally symmetric, which will lead to the appearance of the high-power maxima. Somehow, increasing M will suppress both the grating lobes and sidelobes. When N is an odd number, the bilateral symmetry is broken so that the magnitudes of the grating lobes (high-power maxima) are weighted down, resulting in

better average sidelobe suppression [31].

The following figure illustrates the sidelobes suppression performance to further investigate the special cases when N is the integers multiplied by 4.

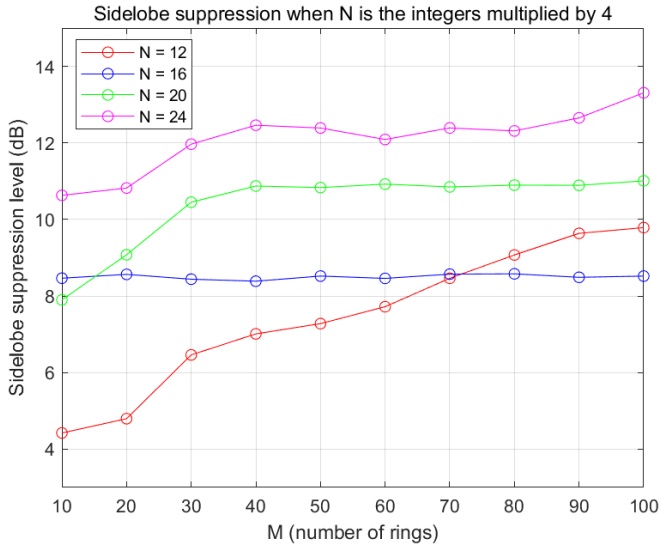


Figure 3.7: Sidelobe suppression with N is the integers multiplied by 4

As shown in figure 3.7, when N equals 16, adding more rings would no longer further suppress the sidelobes. However, when N increases, the geometry of the circular array is "less like" the rectangular array, such that adding more rings will start to further suppress the sidelobes.

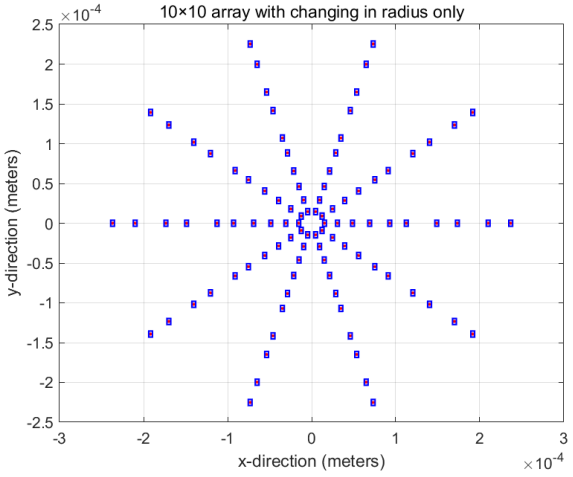
Overall, both M and N will significantly impact SLL for the uniform circular phased array. Yet, the circular phased array with N as an odd number has a better performance than the circular phased array with N as an even number. However, when the total number of antennas is the same, the circular phased array with higher N (number of antennas placed on each ring) will have better sidelobe suppression. Thus, N will be the dominating factor for the SLL.

3.3 Optimization of Circular Phased Arrays

A method similar to that described in section 2.2 was proposed to suppress the sidelobes of the circular phased array, which is to redistribute the radiating elements to form a non-uniform spacing between them. There are two ways to redistribute the radiating elements. The first one is to change the radius between each concentric ring [25], and the second is to change the angular distribution of the radiating elements [26]. For a fair comparison between the two methods, the total number of antennas is fixed to 100.

The following two examples showcase the capability of the suppression of the SLL, where the first method is to change the radial distances between the concentric rings while keeping the angular distribution of the antennas the same. The second method is to change the angular distribution of the antennas only without adjustments of the radii of the concentric rings.

3.3.1 10 by 10 Circular Phased Array (with modified radii only)



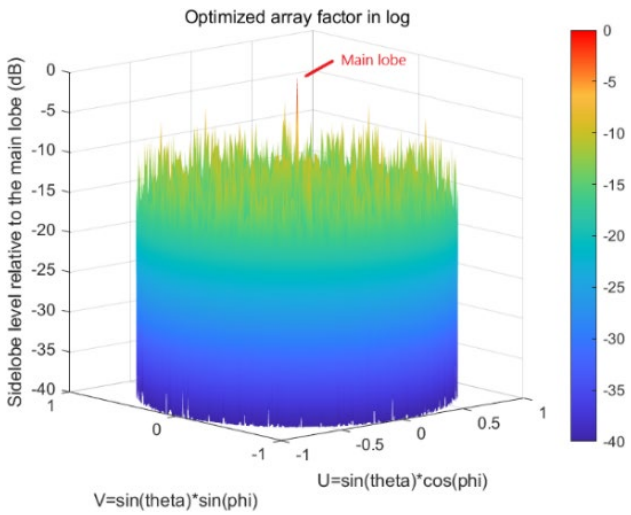
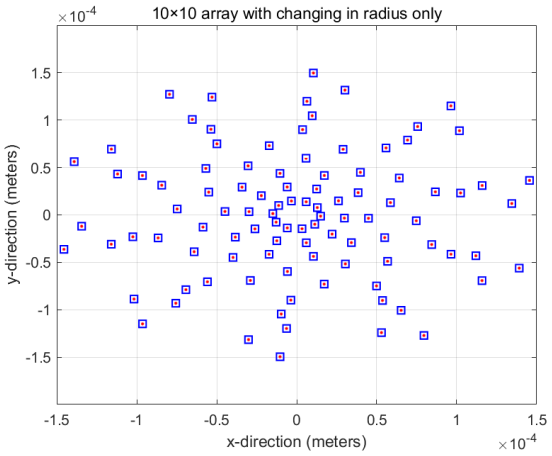


Figure 3.8: a) Geometry of the 10 by 10 circular phased array with modified radii. b) Array factor of modified circular array in log scale.

From figure 3.8 a), the radius for the first inner ring is kept at 15μm, and the radii for the outer rings are randomly changed from 11μm to 17μm through the GA. The maximum SLL relative to the main lobe is suppressed from -4.85dB to -6.55dB with modified in radii only.

3.3.2 10 by 10 Circular Phased Array (with modified angular distributions)



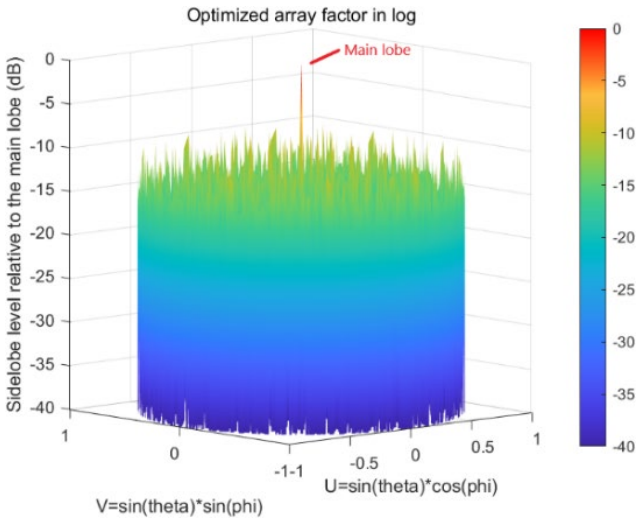


Figure 3.9: a) Geometry of the 10 by 10 circular phased array with modified angular distribution. b) Array factor of modified circular array in log scale

From figure 3.9 a), the radii between the rings are kept the same. To avoid the overlap between antennas, the angular position of the antenna is randomly changed from -0.2 to 0.2 through the GA to obtain an optimized result. Maximum SLL relative to the main lobe is suppressed down to -8.62dB with only changing the angular distribution of antennas.

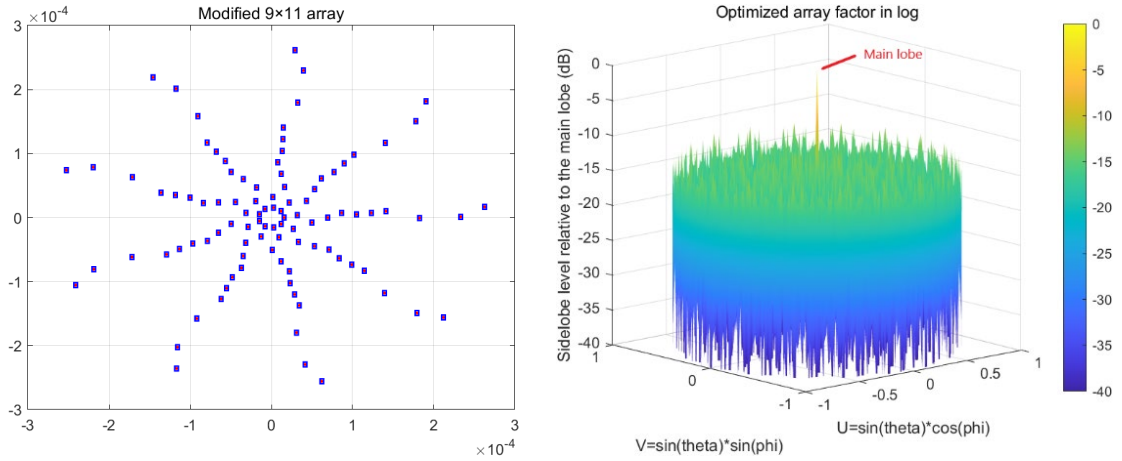
Comparing figure 3.8 and figure 3.9 to figure 3.2 (with the unmodified 10 by 10 circular phased array), changing the radius has less impact on the sidelobe suppression where it only suppressed the amplitude of the sidelobes from -4.85 dB to -6.55 dB, but changing the angular distribution significantly suppressed sidelobe down to -8.62dB.

Mathematically speaking, the non-uniform angular distribution of antennas changes the term $\cos(\theta - \theta_n)$, reducing the constructive interference for each subarray in certain angular directions. The sidelobes experience significant amplitude suppression. On the contrary, changing radius will not influence canceling the

constructive interference, making the angular distribution the primary factor to suppress the sidelobes.

In order to compare with the uniform 10 by 10 circular phased array, we aim for around 100 antenna elements in our optimization process. Assuming the antennas were initially separated by 15μm, each adjacent outer concentric ring has a distance of 15μm separated from each other. The first time in applying the GA is to find the optimal radius between the rings. The optimization process is very similar to the rectangular phased array. First, a group of chromosomes is created and binary encoded. Then we obtain the ranking (fitness) of the chromosomes through the evaluation function. Based on the ranking, the selection, crossover, and mutation are then performed to form another set of new chromosomes needed for the next iteration. After 200 iterations, the radius with the best performance on the sidelobe suppression is applied to the circular phased array. A similar process of finding the optimal angular distribution of the antennas is applied to the GA for the final optimization. The most optimal geometry found is a 9 (M) by 11 (N) circular phased array by performing the optimization process using the GA twice.

3.3.3 Optimized 9 by 11 Circular Phased Array



a) *b)*

Figure 3.10: a) Geometry of optimized 9 by 11 circular array. b) Optimized Array factor for 9 by 11 circular array in log scale.

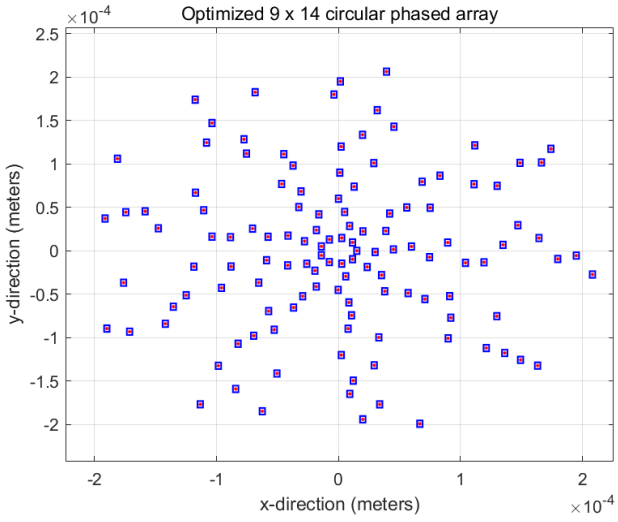
The sidelobe is further suppressed to -11.75dB from -4.85dB, which is the best result so far compared to the previous circular phased arrays.

Based on figure 3.3, the geometry with the best sidelobe suppression level will be selected to optimize with different antennas, which are 126, 225, 324, 423, 513 antennas, in order to compare with rectangular phased arrays and random phased arrays. For all the simulations, the main lobe of the array factor would be steered 23° of θ angle first, then multiplied it to the element pattern to obtain the optimized total pattern since the antenna itself has a 23° diffraction angle (θ).

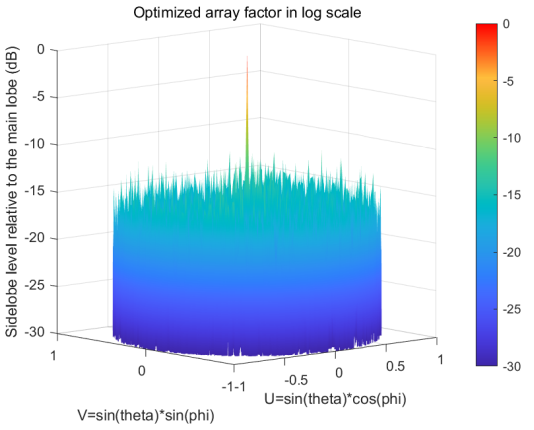
3.4 Simulation Results

For all the simulations, the main lobe of the array factor would be steered 23° of θ angle first, then multiply it to the element pattern to obtain the optimized total pattern.

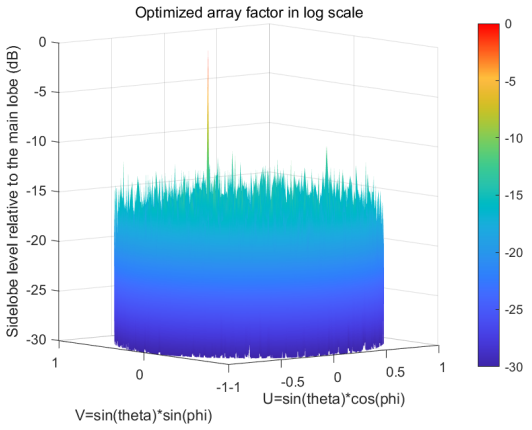
3.4.1 Circular Phased Array with 126 (9 by 14) Antennas



a)



b)

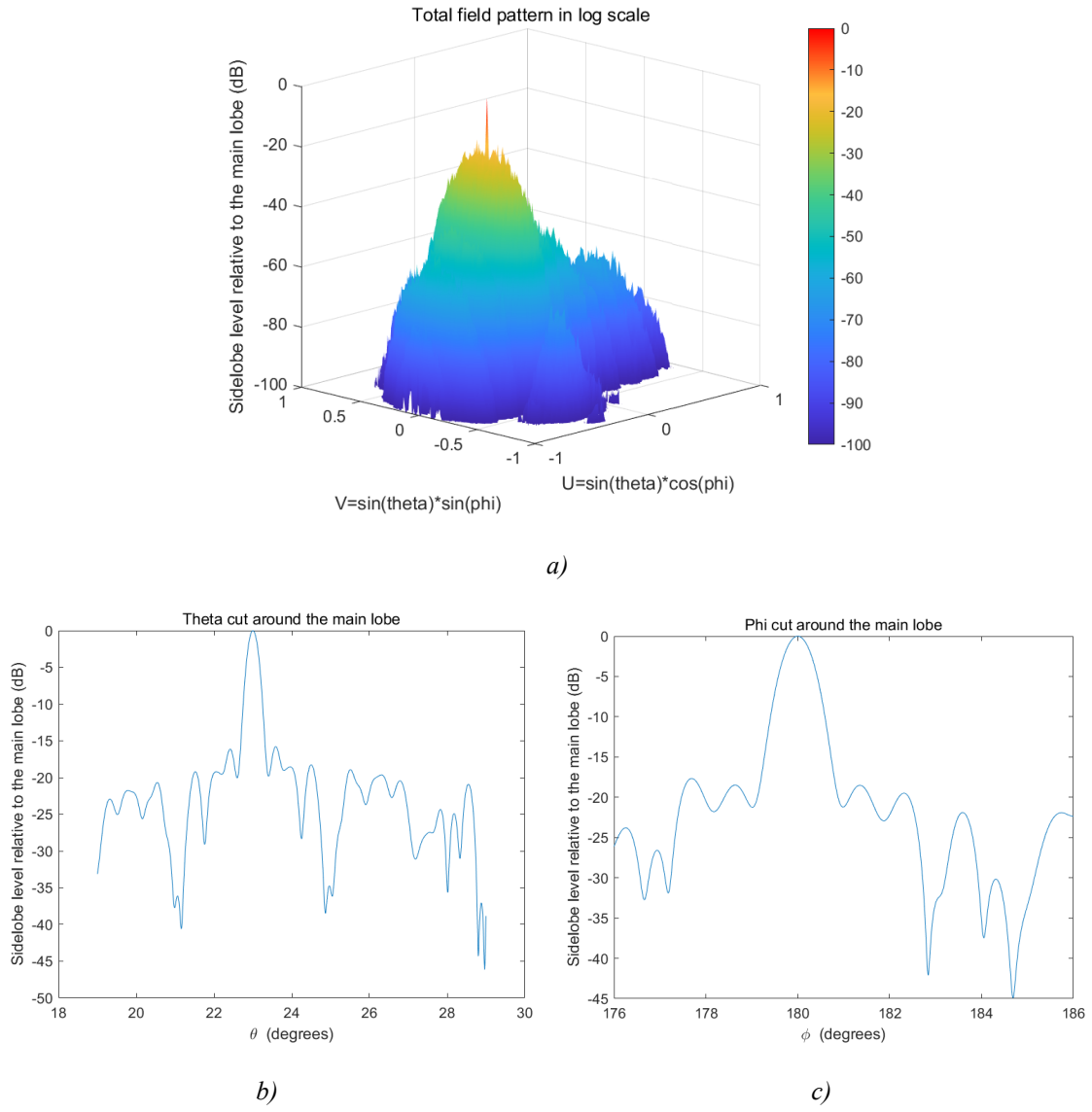


c)

Figure 3.11: a) Geometry of optimized 9 (N) by 14 (M) circular phased array. b) Far-field pattern of array factor in log scale. c) Far-field pattern of array factor in log scale with main lobe steered 23° in θ angle

The size of the chip is around 420 by 410 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -11.49dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -10.82dB.

Total far-field pattern



a)

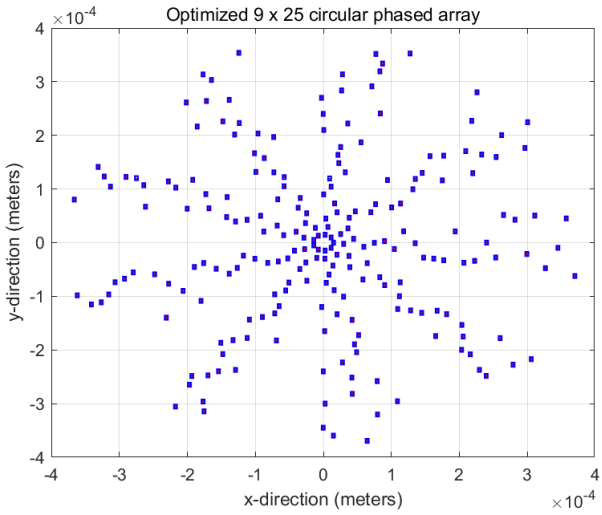
b)

c)

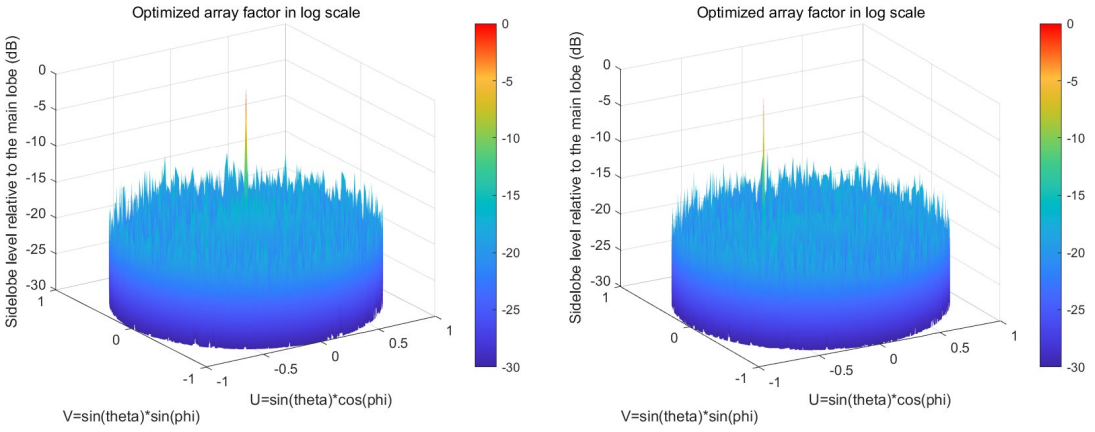
Figure 3.12: a) Total far-field pattern in log scale b) θ cut of 10 degrees around the main lobe. c) ϕ cut of 10 degrees around the main lobe.

The maximum SLL relative to the main lobe in the total far-field pattern is -18.52dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 3.12 b and c, which is 0.29° in θ angle and 0.66° in ϕ angle.

3.4.2 Circular Phased Array with 225 (9 by 25) Antennas



a)



b) *c)*

Figure 3.13

The size of the chip is around 760 by 760 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is -14.83dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -13.46dB.

Total far-field pattern

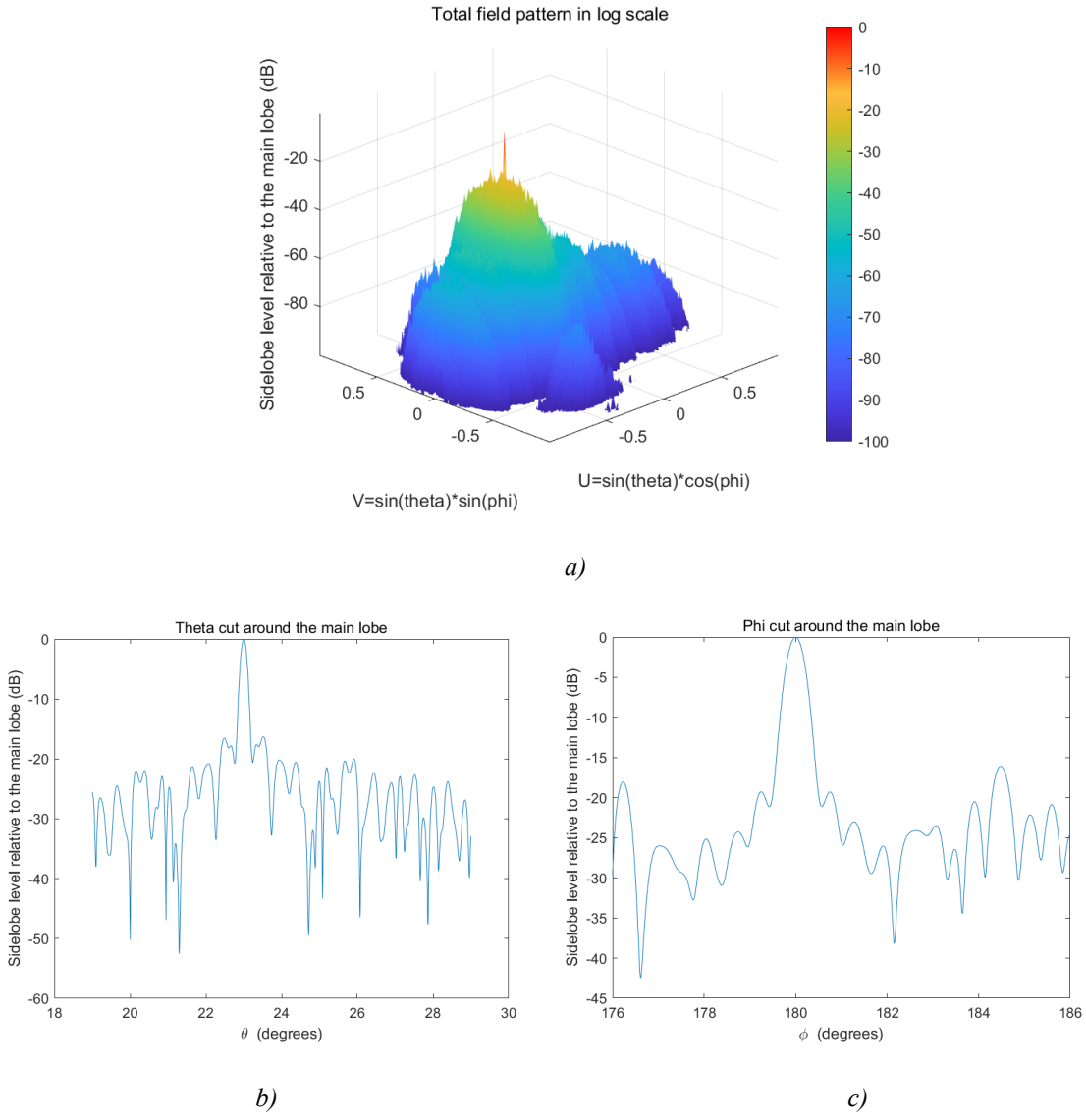
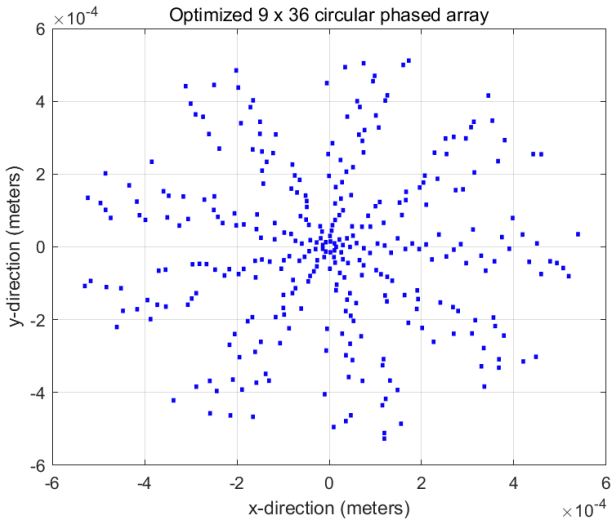


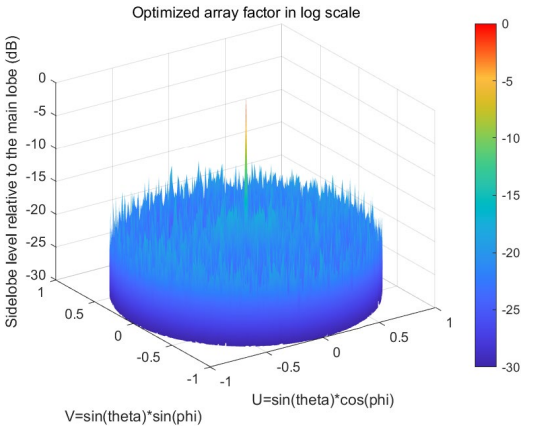
Figure 3.14

The maximum SLL relative to the main lobe in the total far-field pattern is -20.31dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 3.14 b and c, which is 0.15° in θ angle and 0.38° in ϕ angle.

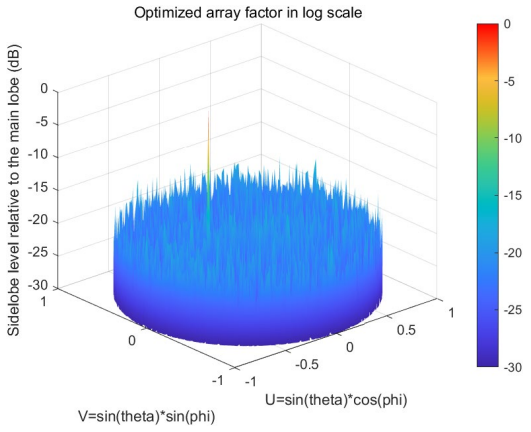
3.4.3 Circular Phased Array with 324 (9 by 36) Antennas



a)



b)

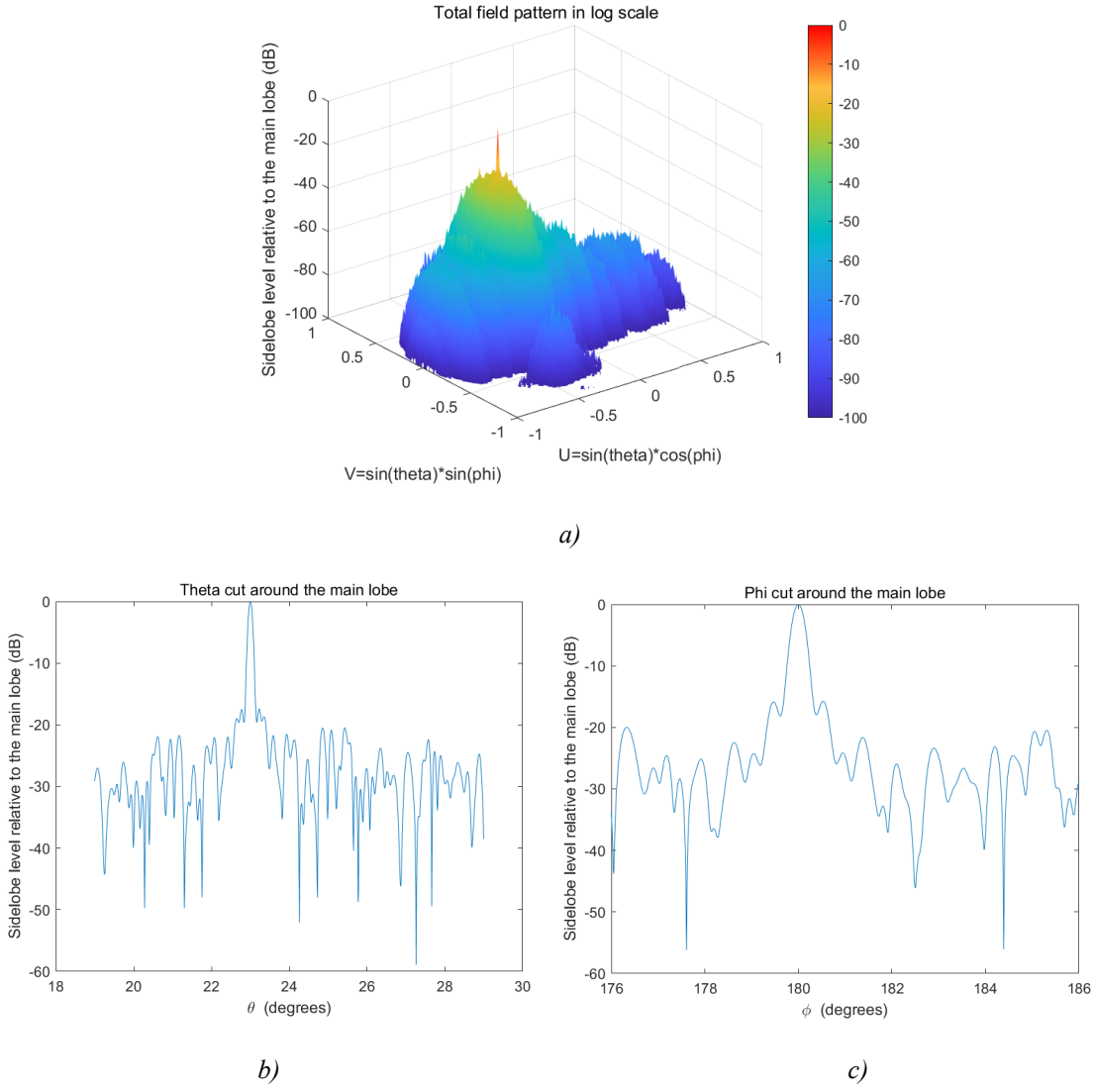


c)

Figure 3.15

The size of the chip is around 1120 by 1120 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -16.32dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -15.64dB.

Total far-field pattern



a)

b)

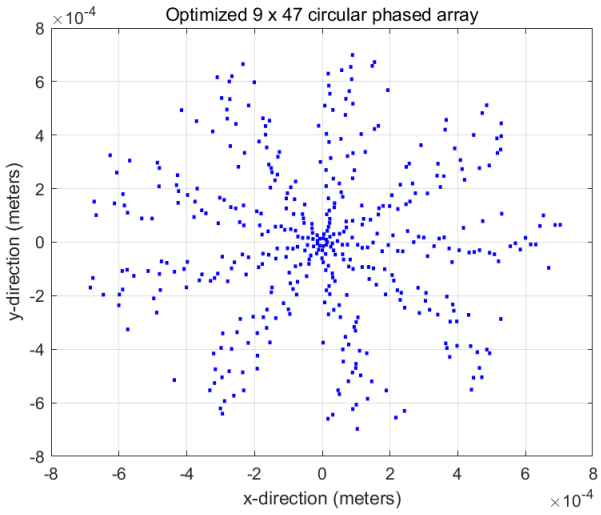
c)

Figure 3.16

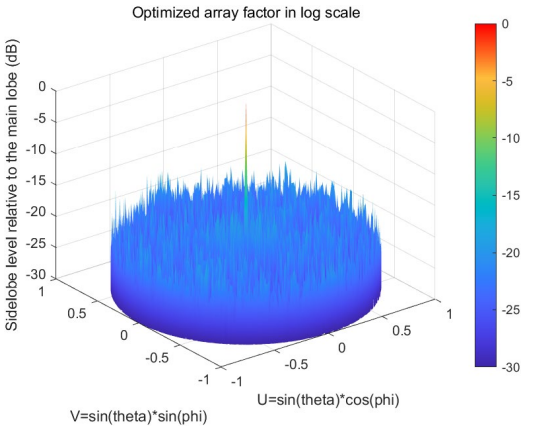
The maximum SLL relative to the main lobe in the total far-field pattern is -21.42dB

with the beam steered in 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees at recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 3.16 b and c, which is 0.11° in θ angle and 0.27° in ϕ angle.

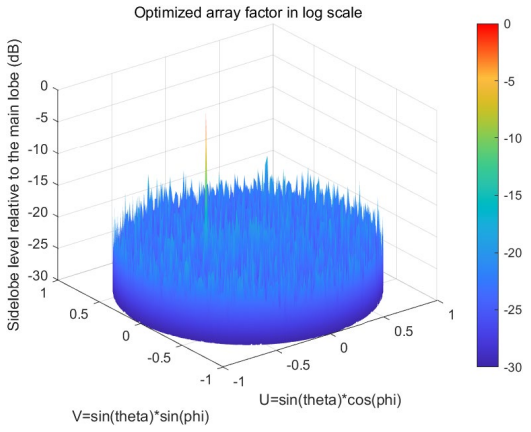
3.4.4 Circular Phased Array with 423 (9 by 47) Antennas



a)



b)

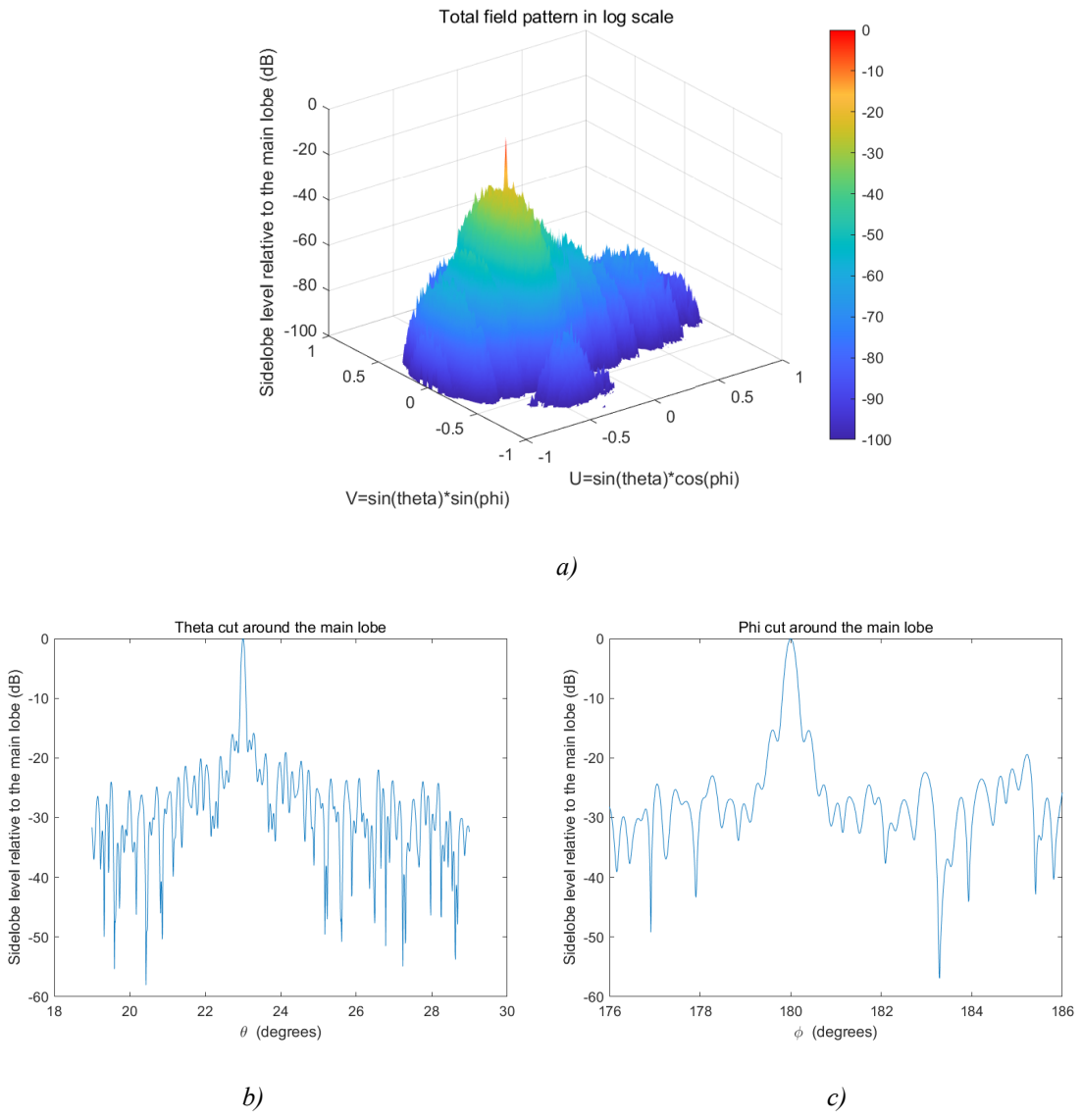


c)

Figure 3.17

The size of the chip is around 1440 by 1440 microns. The maximum SLL relative to the main lobe in the the array factor without beam steering is at -17.08dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -16.50dB.

Total far-field pattern



a)

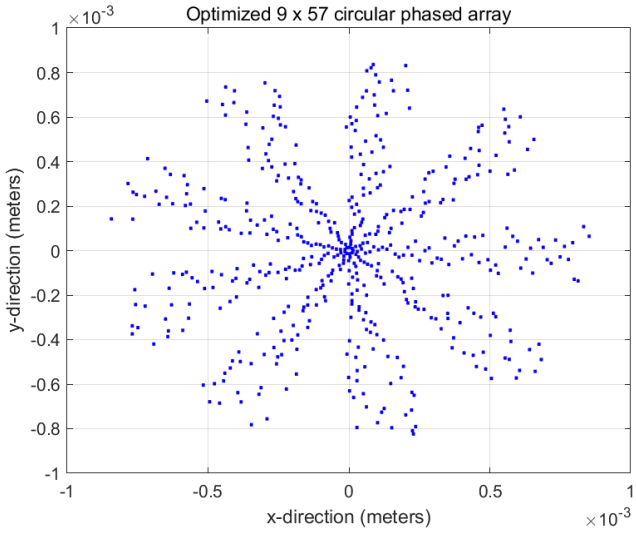
b)

c)

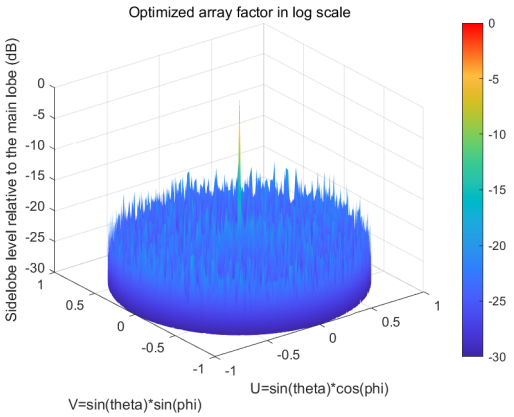
Figure 3.18

The maximum SLL relative to the main lobe in the total far-field pattern is -21.78dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 3.18, which is as 0.09° in θ angle and 0.20° in ϕ angle.

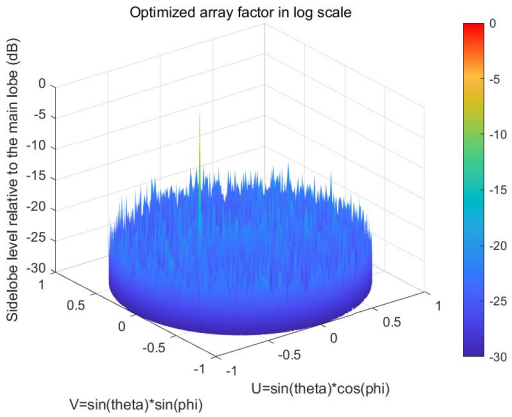
3.4.5 Circular Phased Array with 513 (9 by 57) Antennas



a)



b)

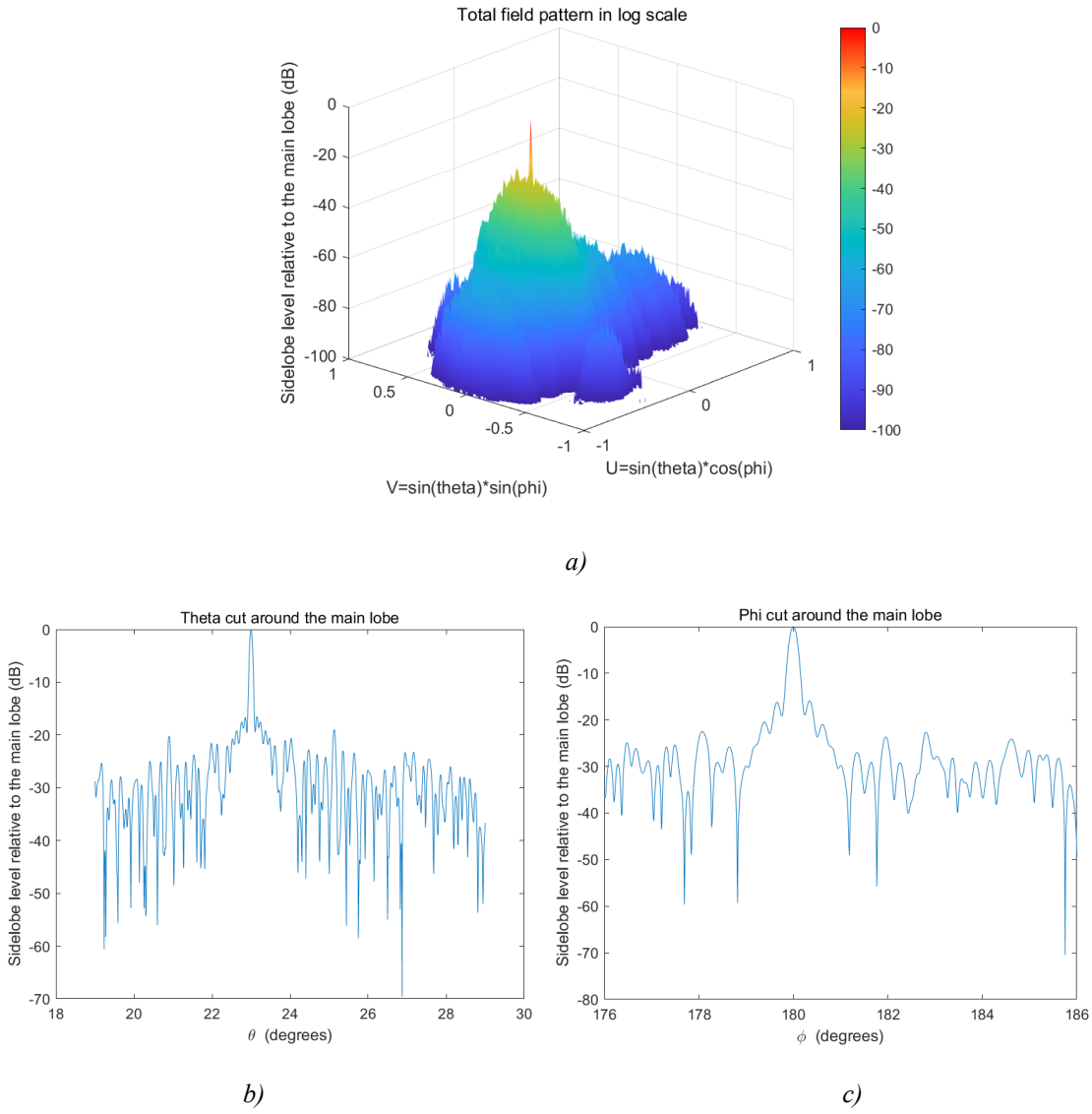


c)

Figure 3.19

The size of the chip is around 1650 by 1650 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -18.62dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -18.11dB.

Total far-field pattern



a)

b)

c)

Figure 3.20

The maximum SLL relative to the main lobe in the total far-field pattern is -22.93dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 3.20 b and c, which is 0.08° in θ angle and 0.17° in ϕ angle.

3.5 Summary

The geometry of the circular phased array can naturally suppress the SLL without any optimization. Circular phased arrays with a higher number of antennas on each ring (N) would have better sidelobe suppression, except for cases where N is equal to 4, 8, 16.

Overall, the circular arrays with an odd number of N outperformed the circular arrays with an even number of N on sidelobe suppression because when N is odd, the bilateral symmetry is broken, which results in a significant reduction in the amplitude of the grating lobe, and the power is distributed in a certain angular range [31]. Table 3.1 recorded several parameters from the simulation results. As the total number of antennas increases, the circular phased array would obtain better sidelobe suppression because a higher gain of the phased array is achieved with more antennas placed [30]. The width of FWHM also decreases with an increasing number of antenna elements because when the number of antenna elements increases, the effective length of the phased array also increases, and the beamwidth is equal to a constant divided by the effective length of the phased array *Lx* and *Ly* in the x-direction and y-direction [30]

hence, increasing the effective length of the phased array will decrease the beamwidth, and bring a higher resolution for the circular arrays. The performance on SLL suppression of the circular arrays is better than the rectangular arrays due to its geometrical arrangement, where it naturally suppresses the "grating lobe-like" sidelobes.

Although applying the GA would break the symmetry of the rectangular array, the symmetry of the rectangular array is not fully broken. In some way or another, the geometry still looks like a "rectangular", where the high-power maxima would still generate by constructive interference of the signal radiating out from the antennas. Therefore, the rectangular phased arrays have the least performance on SLL suppression compared to the circular and the random phased arrays.

Table 3.1: Characteristics of the circular phased array

Number of antennas (N by M)	**Chip Size**	**Main lobe FWHM (θ)**	**Main lobe FWHM (ϕ)**	**SLL of total far-field pattern**
126 (9 by 14)	420 x 410 (microns)	0.29°	0.66°	-18.52(dB)
225 (9 by 25)	760 x 760 (microns)	0.15°	0.38°	-20.31(dB)

324 **(9 by 36)**	1120 x 1120 (microns)	0.11°	0.27°	-21.42(dB)
423 **(9 by 47)**	1440 x 1440 (microns)	0.09°	0.20°	-21.78(dB)
513 **(9 by 57)**	1650 x 1650 (microns)	0.08°	0.17°	-21.93(dB)

Chapter 4:

Planar Random Phased Array

4.1 Fundamentals

Randomly distributed phased array is a different type of phased arrays compared to rectangular or circular phased arrays. Unlike the rectangular or circular phased arrays where antenna elements all have a certain shape where the antennas can be placed in precise positions, the random array does not have any specific position for the radiating elements, and the elements can be landed anywhere in the design spaces.

An article regarding the random phased array was first proposed in 2015. The article analyzed the directional pattern of a non-equidistant antenna array with a random distribution in a two-dimensional space, and in a more complex three-dimensional space [32-33].

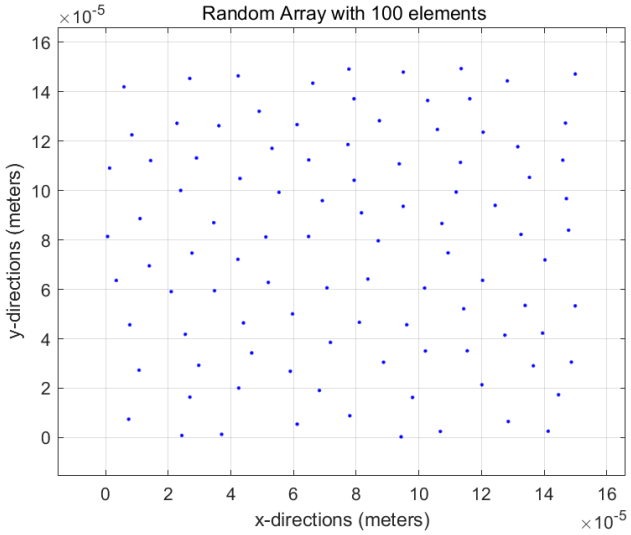


Figure 4.1: Example of two-dimensional random array

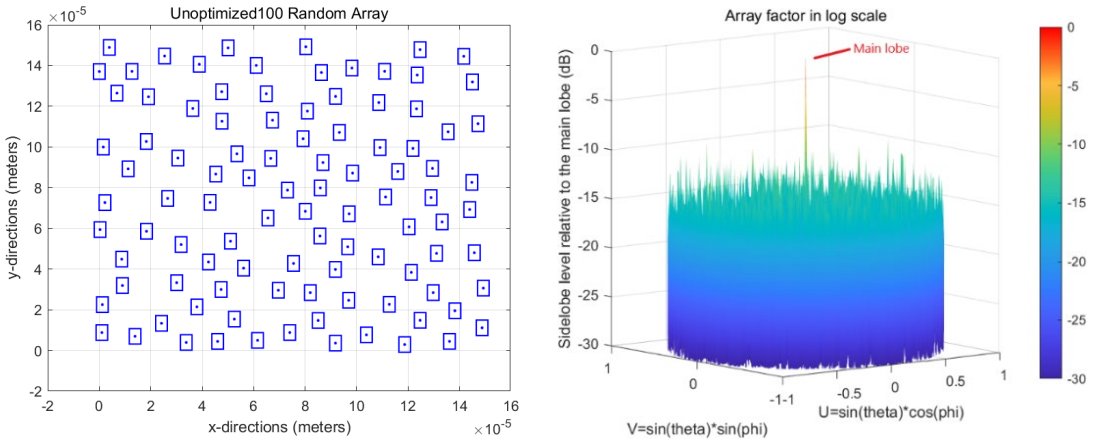
A random array's design process differs from rectangular and circular arrays. Since the random array does not follow a given shape where the radiating elements can be placed everywhere in a design space [32], the final layout can have a shape that looks like a rectangle, circle, star, ellipse, or any other geometry depending on the desired outcome. For simplicity, we choose the square-shaped design space to ensure a maximized geometric area for our designs. The number of antennas will determine the actual size of the design space. The rectangular and circular arrays will need to place the antennas in a specific position, so the size of the design space of random arrays would be more compact than the other two types of arrays. Presumably, the design space of random arrays would be smaller than the rectangular and circular arrays.

4.1.1 Synthesis of the Far-Field Pattern

For a random phased array, all of the radiating elements are randomly distributed in a plane. Unlike the rectangular and circular phased arrays, there will be no unique formula for the array factor of the random array.

Nevertheless, no matter how the geometry of the phased array changes, the far-field pattern of the array factor can always be expressed by the general equation 2.1 in chapter 2 because the beamforming of a phased array is achieved by arranging the radiating elements in such a way that the directing signal has constructive interference at certain angles. Theoretically, as soon we know the positions for each antenna, the relative phase between two adjacent antennas can be calculated to obtain the array factor of any geometry.

The simulation of an unoptimized 100 random phased array is performed. 100 antennas are distributed randomly one by one. Unlike the rectangular phased array in figure 2.10, the antennas can be placed anywhere on the chip. After one antenna is placed on the chip, the minimum distance of the antenna relative to all the other antennas would need to be checked. If it violates the minimum distance, the antenna would be re-positioned and rechecked until all 100 antennas are fully placed on the chip.



a) *b)*

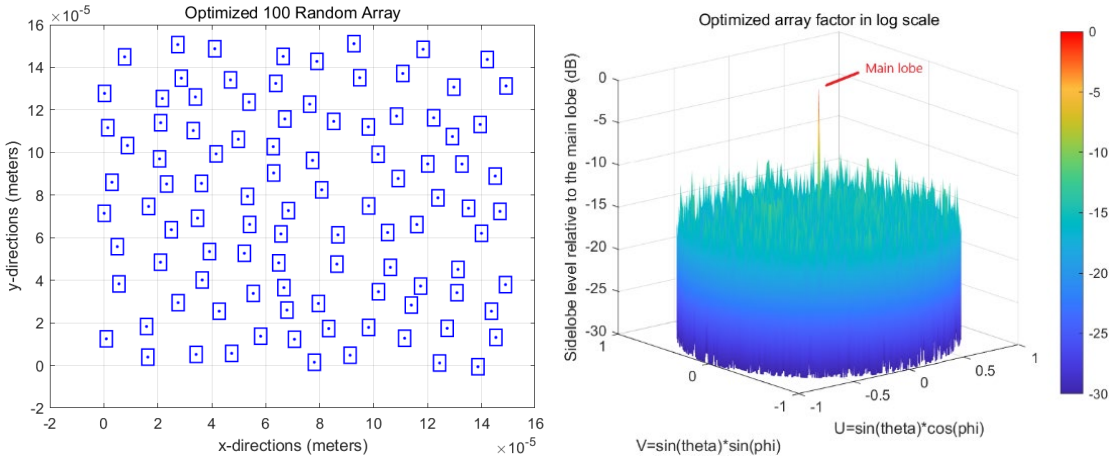
Figure 4.2: a) Geometry of unoptimized random array with 100 antennas. b) Unoptimized array factor. in log scale

Figure 4.2 illustrates that there are no grating lobes in the far-field. The maximum SLL relative to the main lobe is -9.32dB. The performance of sidelobe suppression is better than the unoptimized rectangular and circular phased array due to the random arrangement for the antennas. In fact, the random distribution totally breaks the geometry symmetry and periodicity, the signal power is more likely to be weighted to the main lobe. The following section will provide an example of the optimization of a random phased array.

4.2 Optimization of Random Phased Array

The process of optimization for random arrays is the same as rectangular arrays, which is implement the GA to change the distribution of the antennas within a limited range to avoid the overlap of the radiating elements. This will give better sidelobe suppression than the unmodified random array. Using the antenna element described in Section 1.6,

we obtain the optimized array factor, as shown in Figure 4.3 below.



a) *b)*

Figure 4.3: a) Geometry of optimized random array with 100 antennas. b) Optimized array factor in log scale

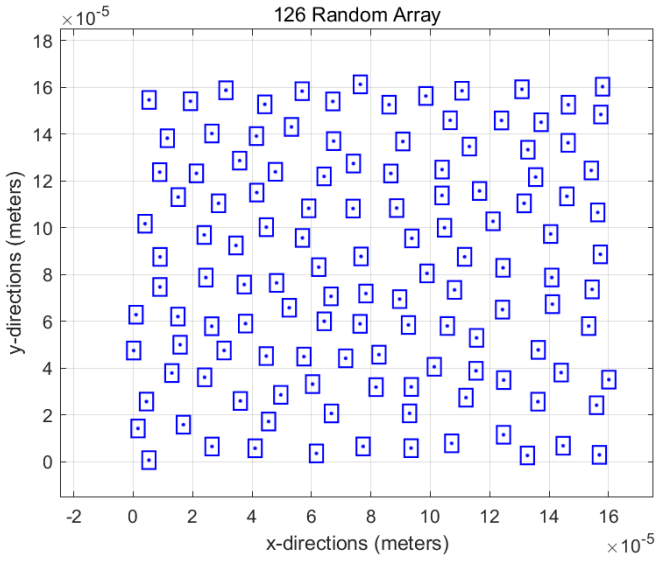
As shown in Figure 4.3, the sidelobe is further suppressed to -11.22 dB after the optimization. Simulations for the random phased array with a different number of antennas, which are 126, 225, 324, 423, 513 antennas, will be performed in the next section. Several parameters will be recorded in order to compare with corresponding circular and rectangular phased arrays, such as minimum design space (chip size) needed with a given number of antennas, FWHM in both θ and ϕ angles, and sidelobe suppression in the total far-field pattern. The total far-field pattern will also be obtained via the multiplication of the array factor and element pattern.

4.3 Simulation Results

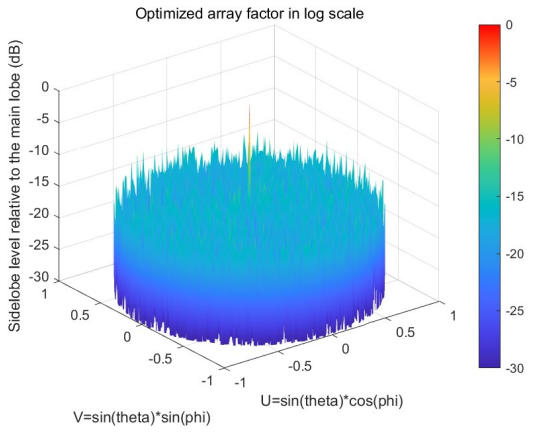
For all the simulations, the main lobe of the array factor would be steered 23° of θ

angle first, then multiply it to the element pattern to obtain the optimized total pattern.

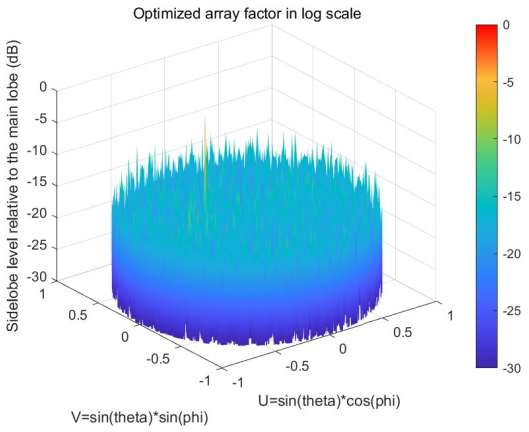
4.3.1 Random Phased Array with 126 Antennas



a)



b)

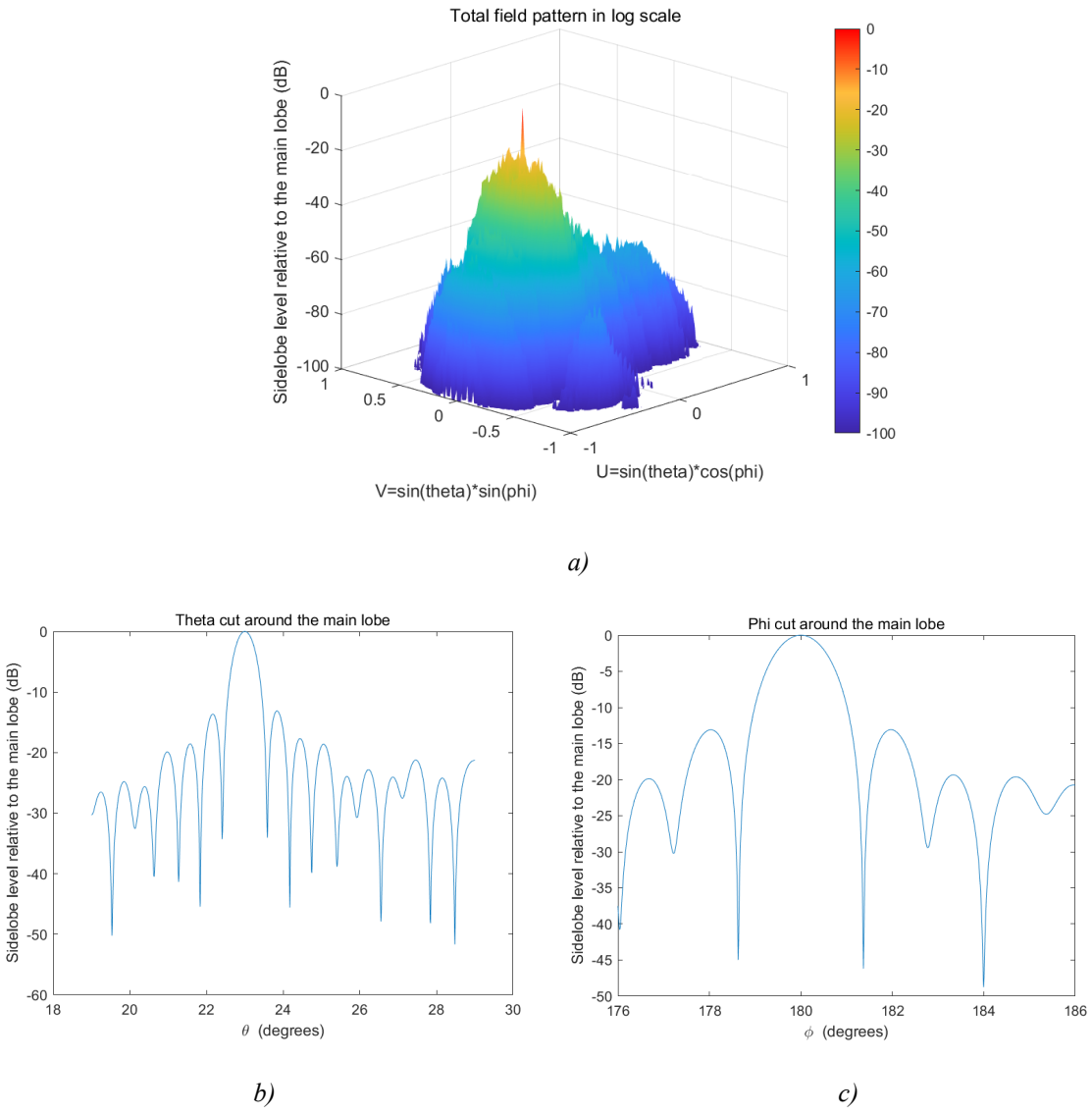


c)

Figure 4.4: a) Geometry of optimized random phased array with 126 antennas. b) Far-field pattern of array factor in log scale. c) Far-field pattern of array factor in log scale with main lobe steered 23° in θ angle

The size of the chip is around 165 by 165 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -12.64dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -12.58dB.

Total far-field pattern



a)

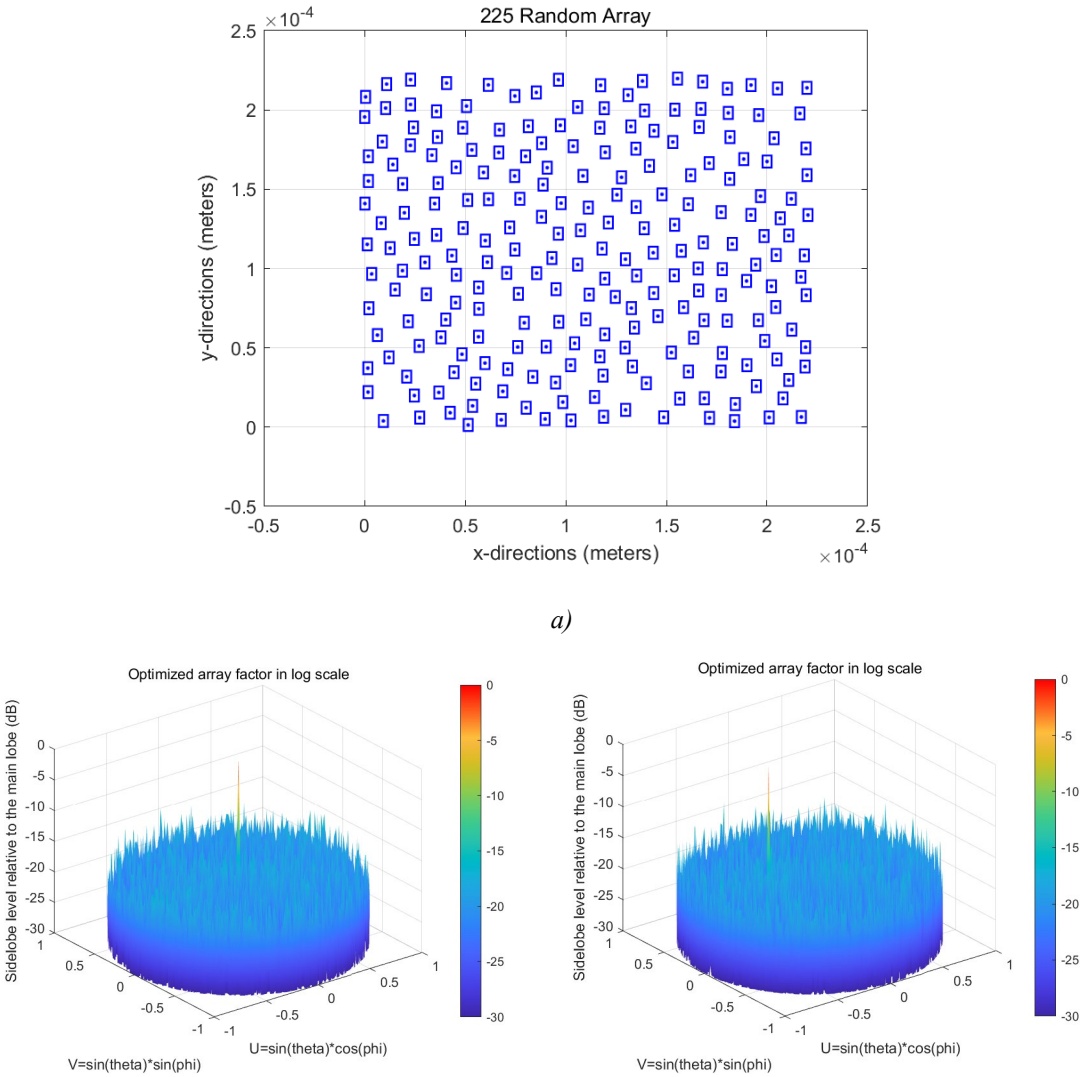
b)

c)

Figure 4.5: a) Total far-field pattern in log scale. b) θ cut of 10 degrees around the main lobe. c) ϕ cut of 10 degrees around the main lobe.

The maximum SLL relative to the main lobe in the total far-field pattern is -16.62dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 4.5 b and c, which is 0.51° in θ angle and 1.21° in ϕ angle.

4.3.2 Random Phased Array with 225 Antennas



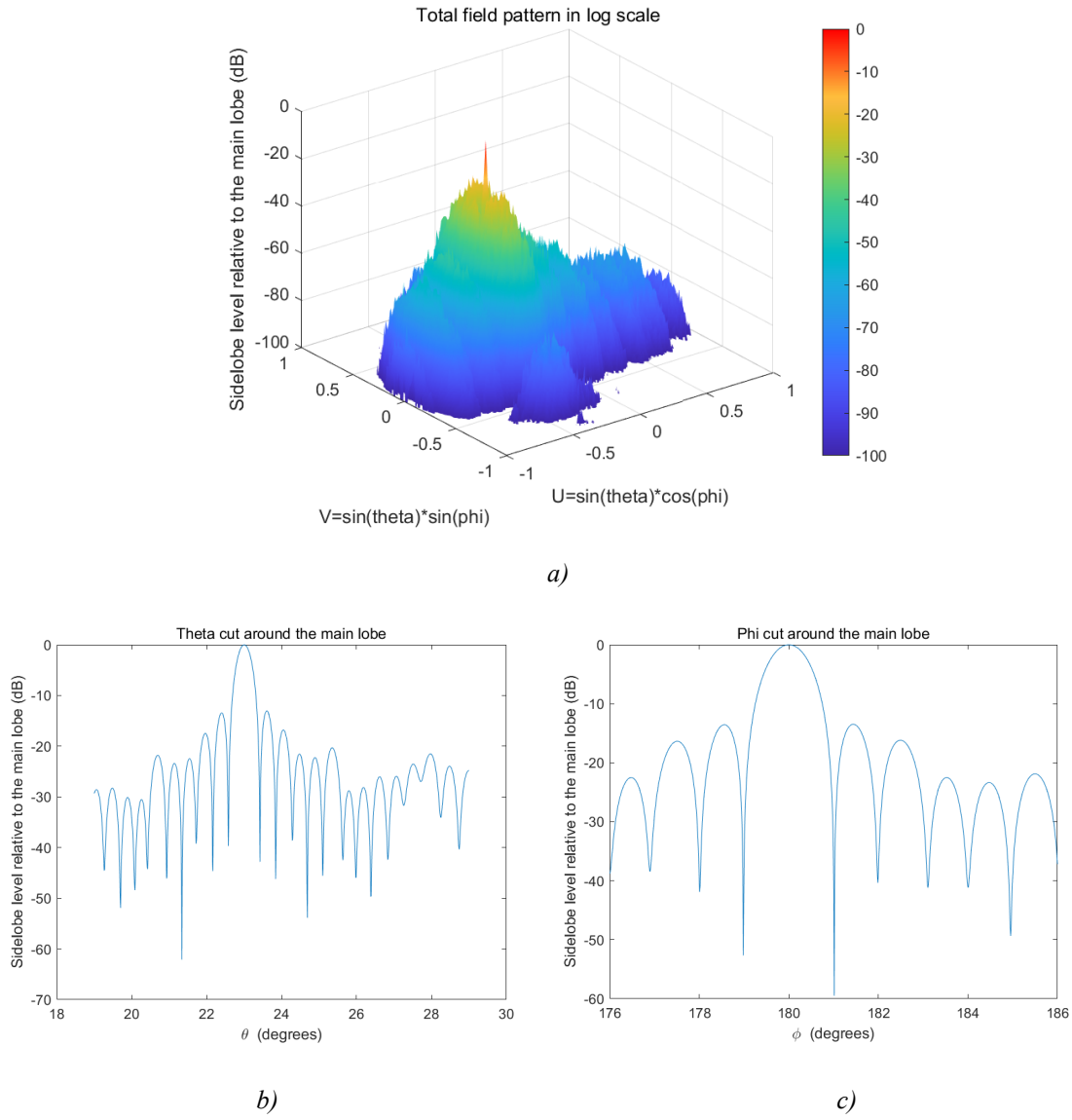
a)

b) *c)*

Figure 4.6

The size of the chip is around 235 by 235 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -14.60dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -14.03dB.

Total far-field pattern



a)

b) *c)*

Figure 4.7

The maximum SLL relative to the main lobe in the total far-field pattern is -17.44dB

with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 4.7 b and c, which is 0.37° in θ angle and 0.91° in ϕ angle.

4.3.3 Random Phased Array with 324 Antennas

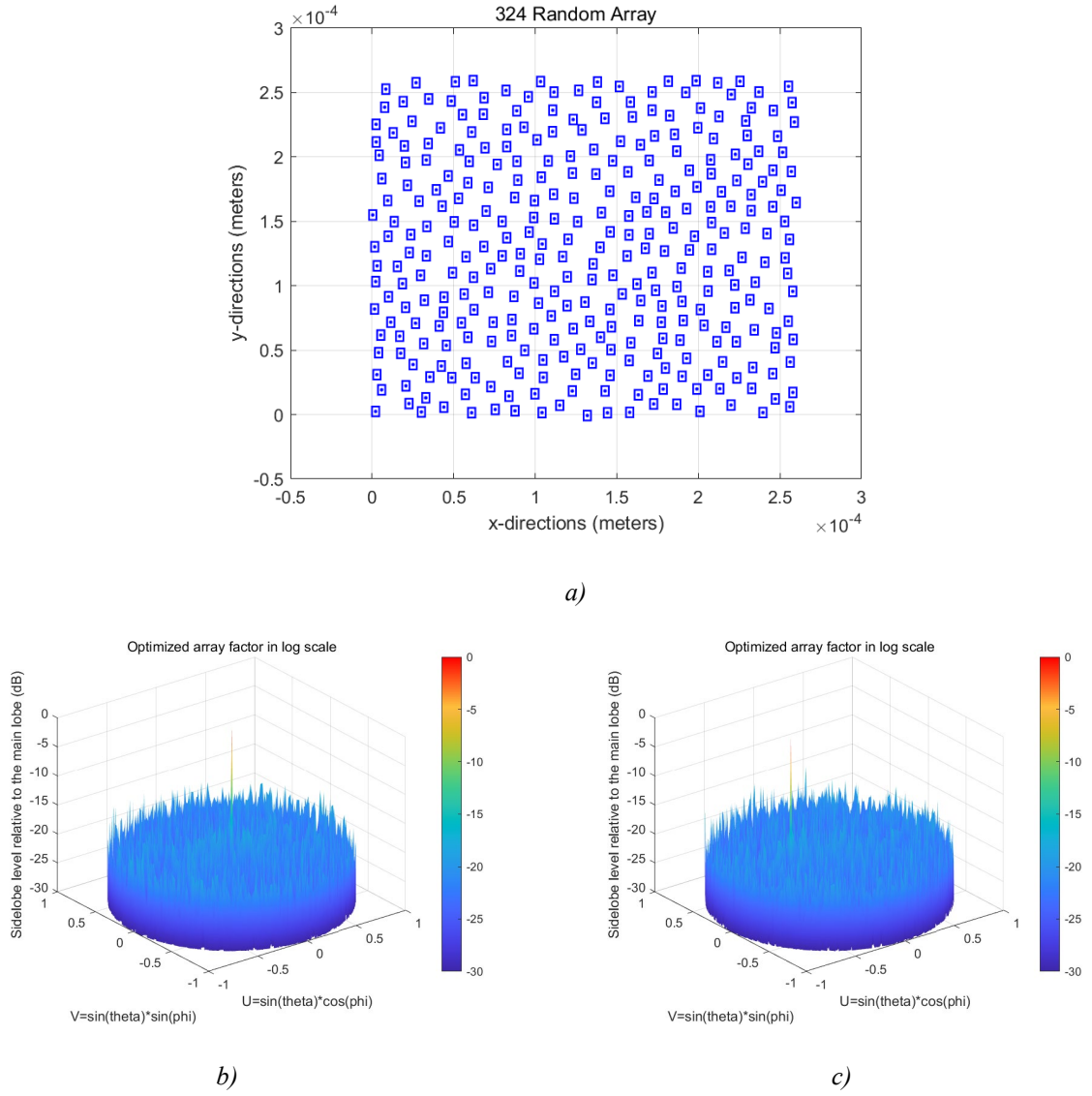
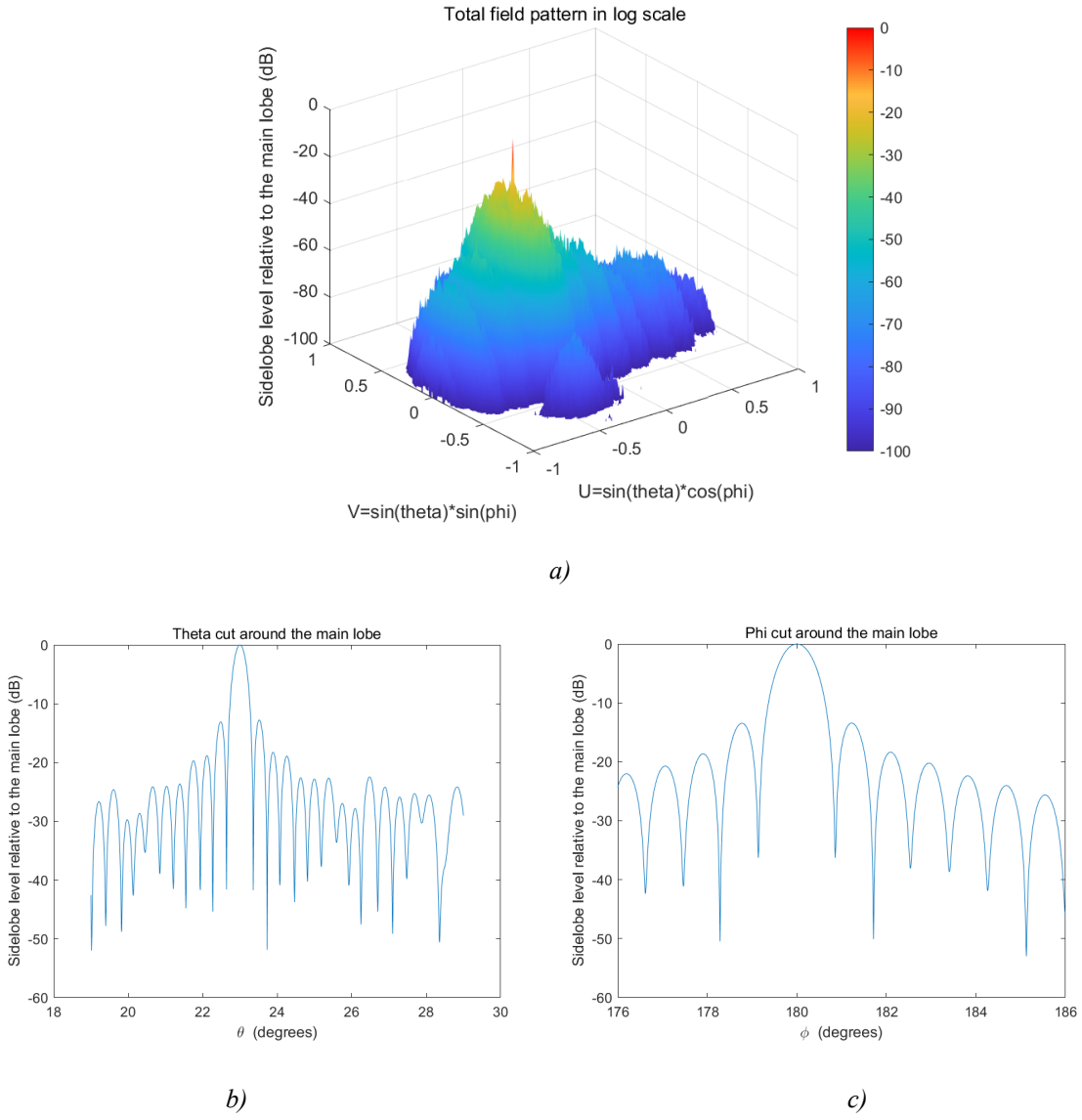


Figure 4.8

The size of the chip is around 260 by 260 microns. The maximum SLL relative to the

main lobe in the array factor without beam steering is at -15.99dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -14.97dB.

Total far-field pattern



a)

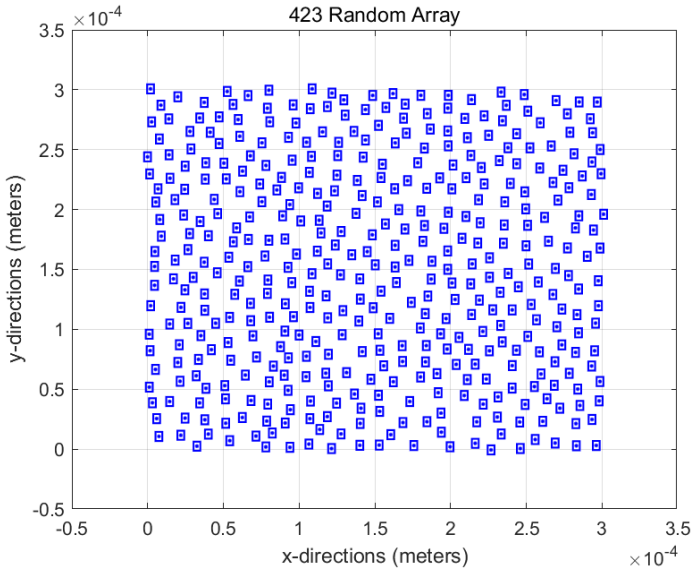
b) *c)*

Figure 4.9

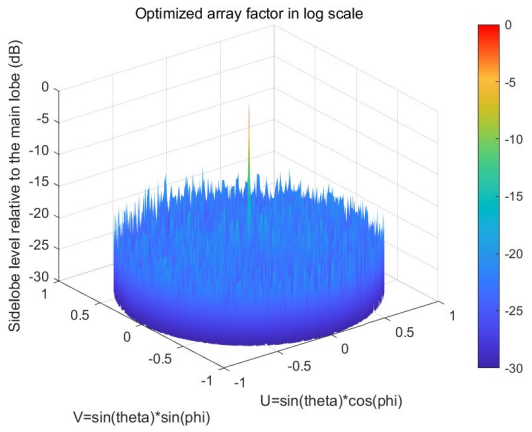
The maximum SLL relative to the main lobe in the total far-field pattern is -19.40dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to

obtain the FWHM with higher solution. The FWHM is then calculated from figure 4.9 b and c, which is 0.32° in θ angle and 0.75° in ϕ angle.

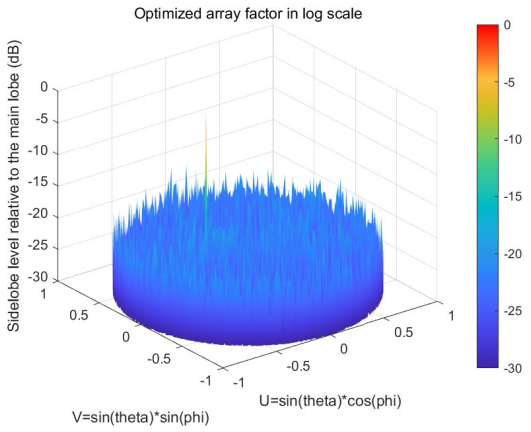
4.3.4 Random Phased Array with 423 Antennas



a)



b)

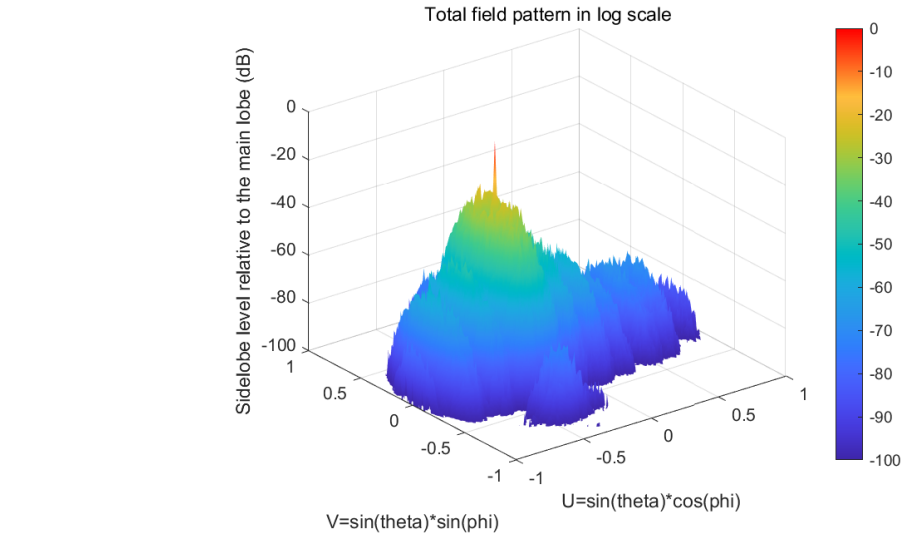


c)

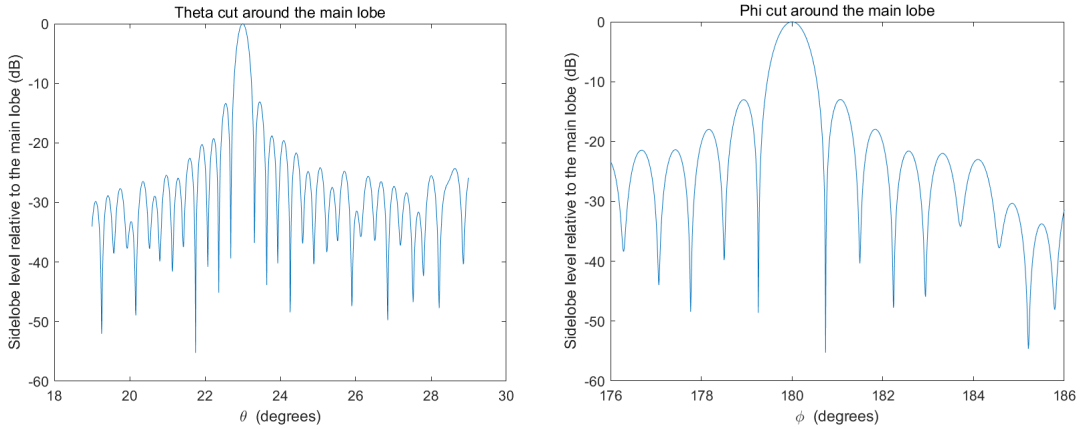
Figure 4.10

The size of the chip is around 305 by 305 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -17.28dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -17.00dB.

Total far-field pattern



a)



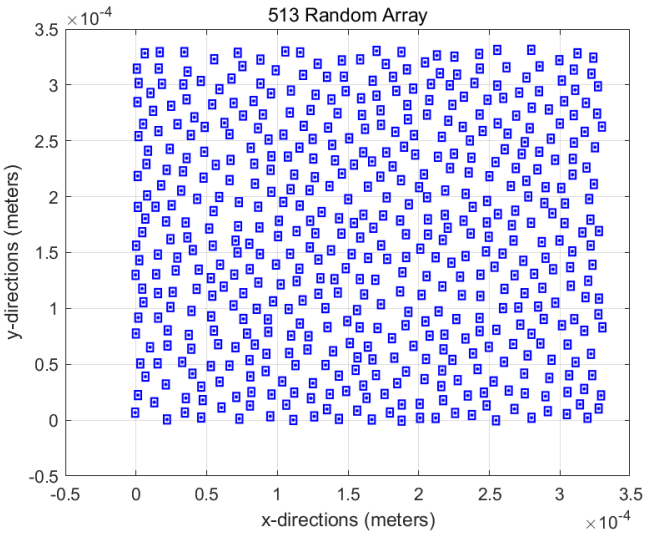
b) *c)*

Figure 4.11

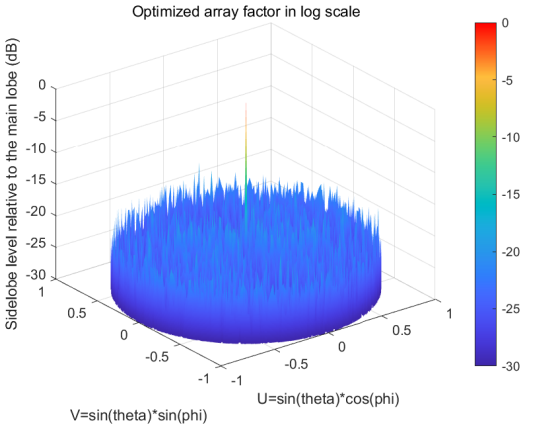
The maximum SLL relative to the main lobe in the total far-field pattern is -21.17dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 4.11

b and c, which is 0.28° in θ angle and 0.65° in ϕ angle.

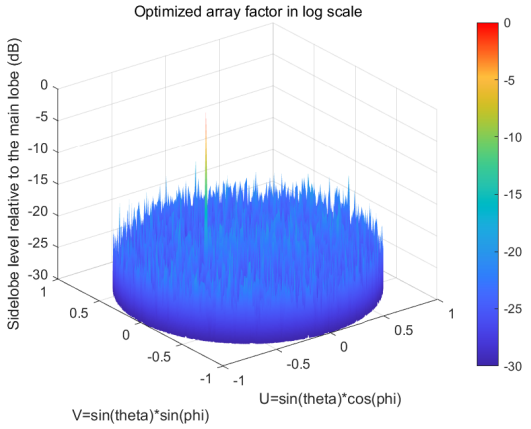
4.3.5 Random Phased Array with 513 Antennas



a)



b)

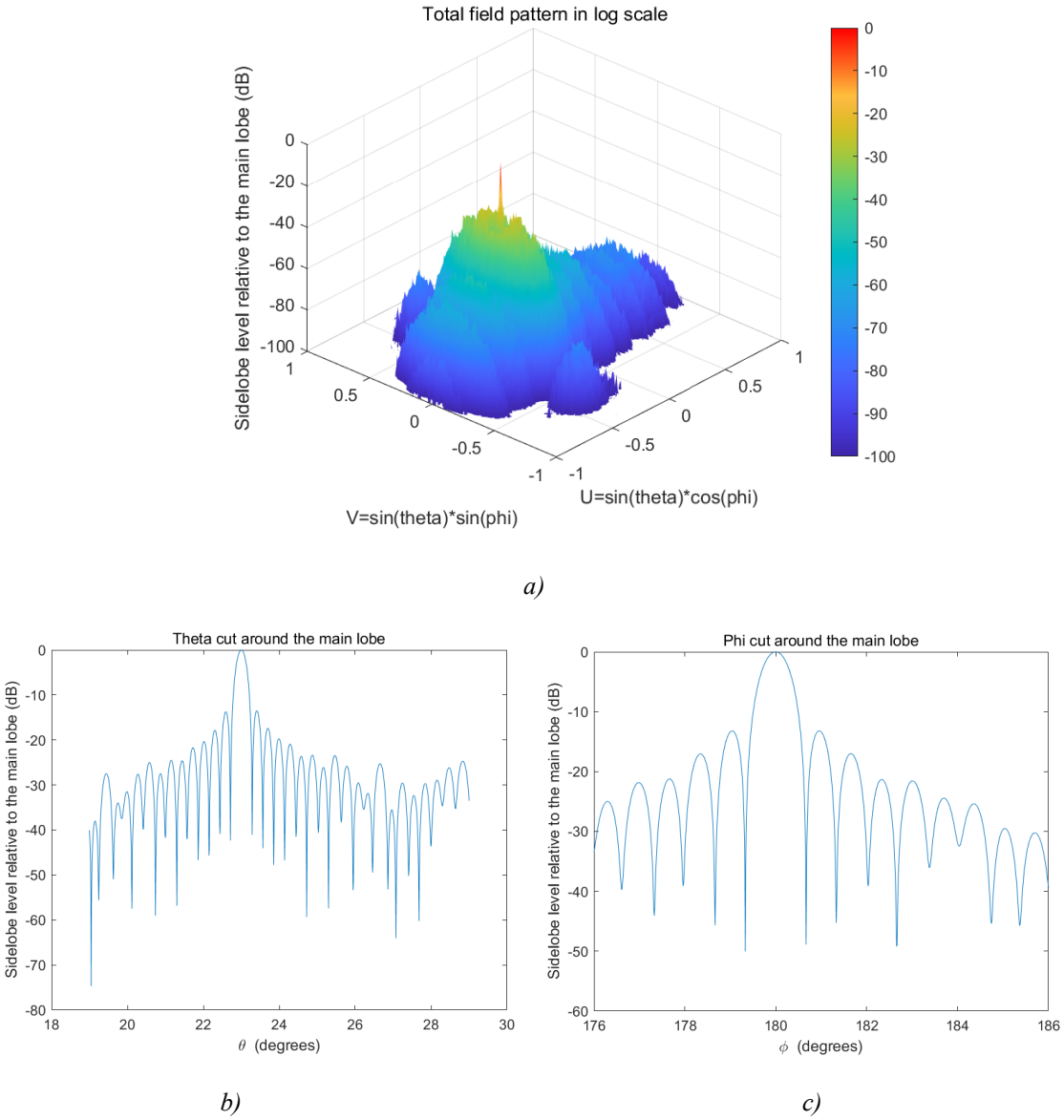


c)

Figure 4.12

The size of the chip is around 343 by 343 microns. The maximum SLL relative to the main lobe in the array factor without beam steering is at -18.58dB. The maximum SLL relative to the main lobe in the array factor with beam steering is -17.64dB.

Total far-field pattern



a)

b)

c)

Figure 4.13

The maximum SLL relative to the main lobe in the total far-field pattern is -22.38dB with the beam steered at 23° in θ angle. The total far-field pattern around the main lobe within 10 degrees is recalculated with the resolution of 0.01 degree in order to obtain the FWHM with higher solution. The FWHM is then calculated from figure 4.13

b and c, which is 0.25° in θ angle and 0.59° in ϕ angle.

4.4 Summary

Table 4.1 recorded several parameters to compare with the circular and the rectangular arrays. Based on table 4.1, the random phased arrays utilize the least chip footprint compared to the circular and the rectangular arrays. The random phased array reaches to similar performance on sidelobe suppression as the circular arrays when over 400 antennas are placed, and when 513 antennas are placed, the random arrays achieved the best SLL suppression compared to the rectangular and circular arrays.

The random and the circular arrays break the geometry symmetry. Hence, the far-field pattern will not have higher power maxima than the rectangular phased arrays. Although a circular phased array has a lower FWHM than a random phased array due to the effective array length [30], the circular phased array utilizes the largest chip size, so this is a trade-off between the beamwidth and the chip size.

These two types of phased arrays are switchable depending on the applications. For example, if a LiDAR project has the following developing goals, where the 3dB beamwidth need to be under 0.2° for both ϕ and θ, and the SLL of the total far-field pattern need to be less than -20dB, and there are no limitations on the chip size. Then a circular array with 513 antennas should be considered. Contrarily, if there are no limitations on the 3dB beamwidth, the chip size must be under 400μm by 400μm. Then, a random array with 513 antennas should be considered.

Table 4.1: Characteristics of the random phased array

Number of antennas	**Chip Size**	**Main lobe FWHM (ϕ)**	**Main lobe FWHM (θ)**	**SLL of total far-field pattern**
Random 126 array	165 x 165 (microns)	0.51°	1.21°	-16.62(dB)
Random 225 array	235 x 235 (microns)	0.37°	0.91°	-17.44(dB)
Random 324 array	260 x 260 (microns)	0.32°	0.75°	-19.40(dB)
Random 423 array	305 x 305 (microns)	0.28°	0.65°	-21.17(dB)
Random 513 array	343 x 343 (microns)	0.25°	0.59°	-22.38(dB)

Chapter 5:

Conclusions

5.1 Comparison of the Phased Array Design

This section summarizes the three different types of phased arrays, namely, the rectangular, circular, and randomly distributed phased arrays. The primary performance metrics of an optical phased array include the far-field sidelobe suppression, chip device footprint, and resolution. Although the rectangular phased array is simplest in design, the detailed analyses conclude that the rectangular design is out-performed by both the circular and random phased arrays due to its poor sidelobe suppression.

The circular phased array generally has the most design complexity because the geometry significantly affects the sidelobe suppression. Thus, the geometry of a circular array must be carefully chosen in the designing phase. Designing a randomly distributed array is more complex than a rectangular array but more straightforward than a circular array since all the antenna elements must be randomly placed in the design space without overlap.

5.2 Characterizing Array Performance

After optimization by implementing the genetic algorithm, the rectangular phased array has the least sidelobe suppression performance and the widest beamwidth in both θ and ϕ angles. On the other hand, the circular phased array has the narrowest beamwidth in both θ and ϕ angles and acceptable performance on sidelobe suppression in the total far-field pattern. With the increasing number of antenna elements, the performance gap between the circular and the random phased array shrinks. However, the randomly distributed phased arrays efficiently utilize the design space on the chip, which effectively decreases the required device footprint. Furthermore, as the number of antennas increases, the FWHM of the main lobe in the far-field pattern will decrease in both of the array designs, promoting a better resolution for scanning the moving objects and surrounding environment [4-5].

We can conclude that a randomly distributed phased array is feasible when over 300 antennas are placed since it decreases the required chip footprint significantly compared to a circular array. With this high number of antenna elements, the trade-off will be the manufacturing complexity, such as the precision in the placement and routing of the antenna elements.

Overall, the circular phased array and the random phased array would be interchangeable solutions depending on applications.

5.3 Future Work

Even though this thesis has demonstrated the possibility of a randomly distributed phased array, there will be coupling and routing issues because the random phased array does not have a certain shape. A possible solution to couple light into the antenna elements is through vertical coupling. The commonly used vertical coupler has a size much larger than what is required by each antenna element. Thus, efficient couplers or coupler arrays need to be developed to realize the random phased array. The feeding system to each antenna element can be designed for circular phased arrays as the ring configuration provides a consistent radiating pattern.

It will be interesting to design and fabricate all three array designs with a similar number of elements (around 500 antenna elements), complete the full layout of the coupling and phase shifter system, and carry out a complete experimental characterization of the three array designs.

References

[1] Yuan, Z.-N., Sun, Z.-B., Kwok, H.-S., Srivastava, A. K. (2021). Fast LIDAR systems based on Ferroelectric Liquid Crystal Dammann grating. Liquid Crystals, 48(10), 1402–1416.

[2] Haupt, R. L., Werner, D. H. (2007). Genetic algorithms in Electromagnetics. IEEE Press.

[3] Chen, Z. (2017). The application of airborne lidar data in the modelling of 3D Urban Landscape Ecology. Cambridge Scholars Publishing.

[4] Muro, S., Yoshida, I., Hashimoto, M. et al. Moving-object detection and tracking by scanning LiDAR mounted on motorcycle based on dynamic background subtraction. Artif Life Robotics 26, 412–422 (2021). https://doi.org/10.1007/s10015-021-00693-z

[5] Devagiri, R., Iyer, N. C., Maralappanavar, S. (2020). Real-time radar and LIDAR sensor fusion for automated driving. Machine Intelligence and Signal Processing, 137–147.

[6] He, J., Dong, T., Xu, Y. (2020). Review of Photonic Integrated Optical phased arrays for Space Optical Communication. IEEE Access, 8, 188284–188298.

[7] Delos, P., Broughton, B., Kraft, J. (2020, May). Phased array antenna patterns—part 1: Linear Array Beam. AnalogDialogue. http://www.oh3ac.fi/ADI_PhasedArrayAntennaPatterns_part1.pdf.

[8] Khajavi, S., Melati, D., Cheben, P., Schmid, J. H., Liu, Q., Xu, D. X.,Ye, W. N. (2021). Compact and highly-efficient broadband surface grating antenna on a silicon platform. Optics Express, 29(5), 7003.

[9] Yin, S. (S., Kim, J. H., Wu, F., Ruffin, P., Luo, C. (F. (2007). Ultra-fast speed, low grating lobe optical beam steering using unequally spaced phased array technique. Optics Communications, 270(1), 41–46.

[10] Fatemi, R., Khachaturian, A., Hajimiri, A. (2019). A nonuniform sparse 2-D large-FOV optical phased array with a low-power PWM drive. IEEE Journal of Solid-State Circuits, 54(5), 1200–1215.

[11] Krivosheev, Y. V., Shishlov, A. V., Denisenko, V. V. (2015). Grating lobe suppression in aperiodic phased array antennas composed of periodic subarrays with large element spacing. IEEE Antennas and Propagation Magazine, 57(1), 76–85.

[12] Bray, M. G., Werner, D. H., Boeringer, D. W., Machuga, D. W. (2002). Optimization of thinned aperiodic linear phased arrays using genetic algorithms to reduce grating lobes during scanning. IEEE Transactions on Antennas and Propagation, 50(12), 1732–1742.

[13] Bataineh, M. H., Ababneh, J. I. (2006). Synthesis of aperiodic linear phased antenna arrays using particle swarm optimization. Electromagnetics, 26(7), 531–541.

[14] Ridwan, M., Abdo, M., Jorswieck, E. (2012). Design of non-uniform antenna arrays using genetic algorithm. Journal of Wireless Networking and Communications, 2(2), 7–10.

[15] Ren, Y., Zhang, L., Zhao, W., Wang, G., Feng, N. N., Wu, W., Sun, X., & Zhang, W.(2021). Compact 2D Serpentine Optical Phased Array. *Applied Optics*, *60*(24), 7158.

[16] Miller, S. A., Chang, Y.-C., Phare, C. T., Shin, M. C., Zadka, M., Roberts, S. P., Stern, B., Ji, X., Mohanty, A., Jimenez Gordillo, O. A., Dave, U. D., & Lipson, M. (2020). Large-scale optical phased array using a low-power multi-pass silicon photonic platform. *Optica*, *7*(1), 3.

[17] Wang, Z., Liao, J., Sun, Y., Han, X., Cao, Y., & Zhao, R. (2020). Chip scale GaAs optical phased arrays for high-speed beam steering. Applied Optics, 59(27), 8310.

[18] Hosseini, A., Kwong, D., Yang Zhao, Yun-Sheng Chen, Crnogorac, F., Pease, R. F. W., & Chen, R. T. (2009). Unequally spaced waveguide arrays for silicon nanomembrane-based efficient large angle optical beam steering. IEEE Journal of Selected Topics in Quantum Electronics, 15(5), 1439–1446.

[19] Van Acoleyen, K., Rogier, H., Baets, R. (2010). Two-dimensional optical phased array antenna on silicon-on-insulator. Optics Express, 18(13), 13655.

[20] Guo, Y., Guo, Y., Li, C., Zhang, H., Zhou, X., Zhang, L. (2021). Integrated Optical phased arrays for beam forming and steering. Applied Sciences, 11(9), 4017. XX

[21] Zhang, D., Zhang, F., Pan, S. (2018). Grating-lobe-suppressed optical phased array with optimized element distribution. Optics Communications, 419, 47–52.

[22] Sanchez, J., Covarrubias-Rosales, D. H., Panduro, M. A. (2009). A synthesis of unequally spaced antenna arrays using Legendre functions. Progress In Electromagnetics Research M, 7, 57–69.

[23] Suarez, S., Leon Fernandez, G., Arrebola, M., Herran Ontanon, L. F., Las-Heras AndrÃ©s, F. (2012). Experimental validation of linear aperiodic array for grating lobe suppression. Progress In Electromagnetics Research C, 26, 193–203.

[24] Noordin, N. H., Zuniga, V., El-Rayis, A. O., Haridas, N., Erdogan, A. T., Arslan, T. (2011). Uniform Circular Arrays for phased array antenna. 2011 Loughborough Antennas Propagation Conference.

[25] Zhang, F., Zhang, D., Pan, S. (2018). Fast and wide-range optical beam steering with ultralow side lobes by applying an optimized multi-circular optical phased array. Applied Optics, 57(18), 4977.

[26] Panduro, M. A., Mendez, A. L., Dominguez, R., Romero, G. (2006). Design of non-uniform circular antenna arrays for side lobe reduction using the method of genetic algorithms. AEU - International Journal of Electronics and Communications, 60(10), 713–717.

[27] Orfanidis, S. J. (2016). Electromagnetic waves and antennas. Sophocles J. Orfanidis.

[28] Bao, Z.Y. (2019). MATLAB Genetic Algorithm and Its Application in scrim-based array antenna (2nd Edition).(ISBN 13: 978-7121376580; ISBN 10 : 712137658X)

[29] Delos, P., Broughton, B., Kraft, J. (2020, December 24). Pattern of phased array antenna part 2: Grating lobe and beam squint. AnalogDialogue. https://ee-paper.com/pattern-of-phased-array-antenna-part-2-grating-lobe-and-beam-squint/.

[30] Mailloux, R. J. (2018). *Phased array antenna handbook*. Artech House.

[31] Liu, Q., Smy, T., Atieh, A., Cheben, P., & Ye, W. N. (2021). *Circular optical phased array with large steering range and high resolution* [Unpublished manuscript]. Department of Electrical Engineering, Carleton University.

[32] Sayyah, K., Efimov, O., Patterson, P., Schaffner, J., White, C., Seurin, J.-F., Xu, G., Miglo, A. (2015). Two-dimensional pseudo-random optical phased array based on tandem optical injection locking of vertical cavity surface emitting lasers. Optics Express, 23(15), 19405.

[33] Tersin, V. V., Bulkin, V. V. (2021). Analysis of the directional pattern of a non-equidistant antenna array with a random arrangement of radiating elements in three-dimensional space. Journal of Physics: Conference Series, 1991(1), 012029.

Appendix A

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Main function performs optimization of the 10 by 10 rectangular phased
% array
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clc;
clear;
close all;
warning off;
addpath 'func\'
addpath 'data\'

global Nx;
global Ny;
global lamdba;
global gen;
global dx;
global dy;

Nx     = 10;
Ny     = 10;
lamdba  = 1550e-9;
dx     = 12e-6;
dy     = 12e-6;

MAXGEN = 200;
NIND   = 30;  % population size
Nums   = 2*Nx*Ny;
Chrom  = crtbp(NIND,Nums*10);


Areas = [];
for i = 1:Nums
    Areas = [Areas,[-2e-6;2e-6]];
end

FieldD = [rep([10],[1,Nums]);Areas;rep([0;0;0;0],[1,Nums])];

gen   = 0;
for a=1:1:NIND
    a
```

```
    Xx          = [rand(1,Nums)];%
    [epls]    = func_obj(Xx);
    E           = epls;
    Js(a,1)      = E;
end

Objv  = (Js+eps);
gen   = 0;

while gen <= MAXGEN;
      gen
      Pe0 = 0.995; % Crossover rate
      pe1 = 0.0065; % Mutation rate

      FitnV=ranking(Objv);
      Selch=select('sus',Chrom,FitnV);
      Selch=recombin('xovsp', Selch,Pe0);
      Selch=mut( Selch,pe1);
      phen1=bs2rv(Selch,FieldD);

      for a=1:1:NIND
          Xx         = phen1(a,:);


          [epls]   = func_obj(Xx);
          E          = epls;
          JJ(a,1)     = E;
      end

      min(JJ)

      if min(JJ)<=-10
          break
      end

      Objvsel=(JJ);
      [Chrom,Objv]=reins(Chrom,Selch,1,1,Objv,Objvsel);
      gen=gen+1;
      idx0=[1:NIND];

      Value(gen) = mean(JJ);
end

figure;
```

```
plot(Value(2:end),'linewidth',2);
grid on
legend('Average fitness');
xlabel('Number of iterations')
ylabel('Sidelobe measurement (dB)')
[VV,II] = min(JJ);
Xbest = [phen1(idx0(II),:)];
```

Appendix B

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function measures the maximum sidelobe of the phased array
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [fitness] = func_obj(Xx);

global Nx;
global Ny;
global xm;
global yn;
global dx;
global dy;

tmps1    = reshape(Xx(1:Nx*Ny),[Nx,Ny]);
tmps2    = reshape(Xx(1+Nx*Ny:end),[Nx,Ny]);

xm    = tmps1;
yn    = tmps2;


for i = 1:Nx
    for j = 1:Ny
         xm(i,j) = xm(i,j)+(i-1)*dx;
         yn(i,j) = yn(i,j)+(j-1)*dy;

    end
end

xPos = reshape(xm,[1 Nx*Ny]);
yPos = reshape(yn,[1 Nx*Ny]);
zPos = zeros(1,Nx*Ny);
elementWeights = ones(1,Nx*Ny);
c = 3e8;
f = c/1550e-9;
phiSteerAngle = 0;
thetaSteerAngle = 0;
thetaScanAngles = linspace(0,90,91);
phiScanAngles = linspace(0,360,361);

[AF] = arrayFactor(xPos, yPos, zPos, elementWeights, f,
c,thetaScanAngles,phiScanAngles,thetaSteerAngle,phiSteerAngle);
AF = abs(AF);
maxp = max(max(AF));
```

```
lg = 20*log10(AF/maxp);

idx = ceil(find(lg==max(lg)));
temp = lg;
temp(idx) = -150;
fitness = max(max(temp));
```

Appendix C

```
function [AF, u, v, w] = arrayFactor(xPos, yPos, zPos, elementWeights, f,
c, thetaScanAngles, phiScanAngles, thetaSteerAngle, phiSteerAngle,
pertStDev)
% arrayFactor - Calculate array factor of 1D, 2D or 3D array
%
%This matlab function calculates the array factor of a 1D, 2D or 3D array
based
%on the position of the elements/sensors and the weight associated with
%each sensor. If no angle is given as input, the scanning angle is theta
%from -90 to 90, and phi from 0 to 360 degrees with 1 degree resolution
%
%[AF, u, v, w] = arrayFactor(xPos, yPos, zPos, elementWeights, f, c,
thetaScanAngles, phiScanAngles, thetaSteerAngle, phiSteerAngle)
%
%IN
%xPos           - 1xP vector of x-positions
%yPos           - 1xP vector of y-positions
%zPos           - 1xP vector of z-positions
%elementWeights  - 1xP vector of element weights
%f              - Wave frequency
%c              - Speed of light
%thetaScanAngles - 1xM vector or NxM matrix of theta scanning angles in
degrees (optional)
%phiScanAngles   - 1XN vector or NxM matrix of phi scanning angles in
degrees (optional)
%thetaSteerAngle - Theta steering angle in degrees (optional)
%phiSteerAngle   - Phi steering angle in degrees (optional)
%
%OUT
%AF             - Calculated array factor
%u              - NxM matrix of u coordinates in UV space
[sin(theta)*cos(phi)]
%v              - NxM matrix of v coordinates in UV space
[sin(theta)*sin(phi)]
%w              - NxM matrix of w coordinates in UV space [cos(theta)]
%

if ~isvector(xPos)
    error('X-positions of array elements must be a 1xP vector where P is
number of elements')
end
if ~isvector(yPos)
```

```
    error('Y-positions of array elements must be a 1xP vector where P is
number of elements')
end
if ~isvector(elementWeights)
    error('Weighting of array elements must be a 1xP vector where P is
number of elements')
end
if ~isscalar(f)
    error('The input frequency must be a single value')
end
% theta is the elevation and is the normal incidence angle from -90 to 90
if ~exist('thetaScanAngles', 'var')
    thetaScanAngles = -pi/2:pi/180:pi/2;
else
    thetaScanAngles = thetaScanAngles*pi/180;
end
% phi is the azimuth, and is the angle in the XY-plane from 0 to 360
if ~exist('phiScanAngles', 'var')
    phiScanAngles = 0:pi/180:2*pi;
else
    phiScanAngles = phiScanAngles*pi/180;
end
% theta, phi steering angles
if ~exist('thetaSteerAngle', 'var')
    thetaSteerAngle = 0;
else
    thetaSteerAngle = thetaSteerAngle*pi/180;
end
if ~exist('phiSteerAngle', 'var')
    phiSteerAngle = 0;
else
    phiSteerAngle = phiSteerAngle*pi/180;
end
if ~exist('pertStDev', 'var')
    pertStDev = ones(1,length(xPos));
elseif ~isvector(pertStDev)
    error('Pertubation must be a 1xP vector where P is number of elements')
end
% Wavenumber
k = 2*pi*f/c;
% Number of elements/sensors in the array
P = length(xPos);
% Calculating wave vector in spherical coordinates
if isvector(thetaScanAngles)
```

```
    % Size of vectors containing theta and phi angles
    M = length(thetaScanAngles);
    N = length(phiScanAngles);

    % Calculate UV coordinates
    u = sin(thetaScanAngles)'*cos(phiScanAngles);
    v = sin(thetaScanAngles)'*sin(phiScanAngles);
    w = repmat(cos(thetaScanAngles)', 1, N);

    % Apply steering
    us = u - sin(thetaSteerAngle)*cos(phiSteerAngle);
    vs = v - sin(thetaSteerAngle)*sin(phiSteerAngle);
    ws = w - cos(thetaSteerAngle);
else
    % Size of matrix containing theta and phi angles
    [M, N] = size(thetaScanAngles);

    % Calculate UV coordinates
    u = sin(thetaScanAngles).*cos(phiScanAngles);
    v = sin(thetaScanAngles).*sin(phiScanAngles);
    w = cos(thetaScanAngles);

    % Apply steering
    us = u - sin(thetaSteerAngle).*cos(phiSteerAngle);
    vs = v - sin(thetaSteerAngle).*sin(phiSteerAngle);
    ws = w - cos(thetaSteerAngle);
end
% Calculate array factor
uu = bsxfun(@times, us, reshape(xPos, 1, 1, P));
vv = bsxfun(@times, vs, reshape(yPos, 1, 1, P));
ww = bsxfun(@times, ws, reshape(zPos, 1, 1, P));
gain = repmat(reshape(elementWeights, 1, 1, P), M, N);
perturbation = repmat(reshape(pertStDev, 1, 1, P), M, N);
AF = sum(gain.*exp(1j*k*(uu + vv + ww)).*perturbation, 3);
% Normalising
AF = abs(AF)./max(max(abs(AF)));
%
%                N
%AF(theta, phi) = sum [ g_n * exp{jk(u*x_n + v*y_n + w*z_n)} ]
%               n=1
%
%u =
```

```
%|sin(theta_0)*cos(phi_0) sin(theta_0)*cos(phi_1) ..
sin(theta_0)*cos(phi_N)|
%|sin(theta_1)*cos(phi_0) sin(theta_1)*cos(phi_1) ..
sin(theta_1)*cos(phi_N)|
%|    .                       .                    .             |
%|sin(theta_M)*cos(phi_0) sin(theta_M)*cos(phi_1) ..
sin(theta_M)*cos(phi_N)|
%v =
%|sin(theta_0)*sin(phi_0) sin(theta_0)*sin(phi_1) ..
sin(theta_0)*sin(phi_N)|
%|sin(theta_1)*sin(phi_0) sin(theta_1)*sin(phi_1) ..
sin(theta_1)*sin(phi_N)|
%|    .                       .                    .             |
%|sin(theta_M)*sin(phi_0) sin(theta_M)*sin(phi_1) ..
sin(theta_M)*sin(phi_N)|
%w =
%|cos(theta_0) cos(theta_0) .. cos(theta_0)|
%|cos(theta_1) cos(theta_1) .. cos(theta_1)|
%|    .           .              .     |
%|cos(theta_N) cos(theta_N) .. cos(theta_N)|
%uu =
%   --------
%  /       /|
% / xPos  / |
%---------  | M (length theta)
%|       |  |
%|   u   |  /
%|       | / P (# elements)
%---------/
%   N (length phi)
%g =
%   --------
%  / g_P   /|
% /       / |
%---------  | M
%|g1   g1|  |
%|   g1  |  /
%|g1   g1| / P (# elements)
%---------/
%   N
```

Appendix D

```
% CRTBP.m - Create an initial population
%
% This function creates a binary population of given size and structure.
%
% Syntax: [Chrom Lind BaseV] = crtbp(Nind, Lind, Base)
%
% Input Parameters:
%
%      Nind    - Either a scalar containing the number of individuals
%            in the new population or a row vector of length two
%                containing the number of individuals and their length.
%
%      Lind    - A scalar containing the length of the individual
%            chromosomes.
%
%      Base    - A scalar containing the base of the chromosome
%                elements or a row vector containing the base(s)
%            of the loci of the chromosomes.
%
% Output Parameters:
%
%      Chrom   - A matrix containing the random valued chromosomes
%                row wise.
%
%      Lind    - A scalar containing the length of the chromosome.
%
%      BaseV   - A row vector containing the base of the
%            chromosome loci.
%
% Author: Andrew Chipperfield
% Date: 19-Jan-94
%
% Tested under MATLAB v6 by Alex Shenfield (20-Jan-03)

function [Chrom, Lind, BaseV] = crtbp(Nind, Lind, Base)
nargs = nargin ;

% Check parameter consistency

if nargs >= 1, [mN, nN] = size(Nind) ; end
if nargs >= 2, [mL, nL] = size(Lind) ; end
```

```
if nargs == 3, [mB, nB] = size(Base) ; end

if nN == 2
   if (nargs == 1)
      Lind = Nind(2) ; Nind = Nind(1) ; BaseV = crtbase(Lind) ;
   elseif (nargs == 2 & nL == 1)
      BaseV = crtbase(Nind(2),Lind) ; Lind = Nind(2) ; Nind = Nind(1) ;
   elseif (nargs == 2 & nL > 1)
      if Lind ~= length(Lind), error('Lind and Base disagree'); end
      BaseV = Lind ; Lind = Nind(2) ; Nind = Nind(1) ;
   end
elseif nN == 1
   if nargs == 2
      if nL == 1, BaseV = crtbase(Lind) ;
      else, BaseV = Lind ; Lind = nL ; end
   elseif nargs == 3
      if nB == 1, BaseV = crtbase(Lind,Base) ;
      elseif nB ~= Lind, error('Lind and Base disagree') ;
      else BaseV = Base ; end
   end
else
   error('Input parameters inconsistent') ;
end

% Create a structure of random chromosomes in row wise order, dimensions
% Nind by Lind. The base of each chromosomes loci is given by the value
% of the corresponding element of the row vector base.

Chrom = floor(rand(Nind,Lind).*BaseV(ones(Nind,1),:)) ;


% End of file
```

Appendix E

```
% RANKING.M     (RANK-based fitness assignment)
%
% This function performs ranking of individuals.
%
% Syntax:  FitnV = ranking(ObjV, RFun, SUBPOP)
%
% This function ranks individuals represented by their associated
% cost, to be *minimized*, and returns a column vector FitnV
% containing the corresponding individual fitnesses. For multiple
% subpopulations the ranking is performed separately for each
% subpopulation.
%
% Input parameters:
%    ObjV      - Column vector containing the objective values of the
%                individuals in the current population (cost values).
%    RFun      - (optional) If RFun is a scalar in [1, 2] linear ranking is
%                assumed and the scalar indicates the selective pressure.
%                If RFun is a 2 element vector:
%                RFun(1): SP - scalar indicating the selective pressure
%                RFun(2): RM - ranking method
%                        RM = 0: linear ranking
%                        RM = 1: non-linear ranking
%                If RFun is a vector with length(Rfun) > 2 it contains
%                the fitness to be assigned to each rank. It should have
%                the same length as ObjV. Usually RFun is monotonously
%                increasing.
%                If RFun is omitted or NaN, linear ranking
%                and a selective pressure of 2 are assumed.
%    SUBPOP    - (optional) Number of subpopulations
%                if omitted or NaN, 1 subpopulation is assumed
%
% Output parameters:
%    FitnV     - Column vector containing the fitness values of the
%                individuals in the current population.
%
%
% Author:     Hartmut Pohlheim (Carlos Fonseca)
% History:    01.03.94     non-linear ranking
%            10.03.94     multiple populations
%            21.01.03     updated for MATLAB v6 by Alex Shenfield

function FitnV = ranking(ObjV, RFun, SUBPOP)
```

```
% Identify the vector size (Nind)
   [Nind,~] = size(ObjV);

   if nargin < 2, RFun = []; end
   if nargin > 1, if isnan(RFun), RFun = []; end, end
   if prod(size(RFun)) == 2,
      if RFun(2) == 1, NonLin = 1;
      elseif RFun(2) == 0, NonLin = 0;
      else error('Parameter for ranking method must be 0 or 1'); end
      RFun = RFun(1);
      if isnan(RFun), RFun = 2; end
   elseif prod(size(RFun)) > 2,
      if prod(size(RFun)) ~= Nind, error('ObjV and RFun disagree'); end
   elseif prod(size(RFun)) < 2, NonLin = 0;
   end

   if nargin < 3, SUBPOP = 1; end
   if nargin > 2,
      if isempty(SUBPOP), SUBPOP = 1;
      elseif isnan(SUBPOP), SUBPOP = 1;
      elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
   end

   if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('ObjV and SUBPOP disagree');
end
   Nind = Nind/SUBPOP;  % Compute number of individuals per subpopulation

% Check ranking function and use default values if necessary
   if isempty(RFun),
      % linear ranking with selective pressure 2
         RFun = 2*[0:Nind-1]'/(Nind-1);
   elseif prod(size(RFun)) == 1
      if NonLin == 1,
         % non-linear ranking
         if RFun(1) < 1, error('Selective pressure must be greater than 1');
         elseif RFun(1) > Nind-2, error('Selective pressure too big'); end
         Root1 = roots([RFun(1)-Nind [RFun(1)*ones(1,Nind-1)]]);
         RFun = (abs(Root1(1)) * ones(Nind,1)) .^ [(0:Nind-1)'];
         RFun = RFun / sum(RFun) * Nind;
      else
         % linear ranking with SP between 1 and 2
         if (RFun(1) < 1 | RFun(1) > 2),
```

```
            error('Selective pressure for linear ranking must be between 1
and 2');
         end
         RFun = 2-RFun + 2*(RFun-1)*[0:Nind-1]'/(Nind-1);
      end
   end;

   FitnV = [];

% loop over all subpopulations
for irun = 1:SUBPOP,
   % Copy objective values of actual subpopulation
      ObjVSub = ObjV((irun-1)*Nind+1:irun*Nind);
   % Sort does not handle NaN values as required. So, find those...
      NaNix = isnan(ObjVSub);
      Validix = find(~NaNix);
   % ... and sort only numeric values (smaller is better).
      [~,ix] = sort(-ObjVSub(Validix));

   % Now build indexing vector assuming NaN are worse than numbers,
   % (including Inf!)...
      ix = [find(NaNix) ; Validix(ix)];
   % ... and obtain a sorted version of ObjV
      Sorted = ObjVSub(ix);

   % Assign fitness according to RFun.
      i = 1;
      FitnVSub = zeros(Nind,1);
      for j = [find(Sorted(1:Nind-1) ~= Sorted(2:Nind)); Nind]',
         FitnVSub(i:j) = sum(RFun(i:j)) * ones(j-i+1,1) / (j-i+1);
         i =j+1;
      end

   % Finally, return unsorted vector.
      [~,uix] = sort(ix);
      FitnVSub = FitnVSub(uix);

   % Add FitnVSub to FitnV
      FitnV = [FitnV; FitnVSub];
end

% End of function
```

Appendix F

```
% SUS.M         (Stochastic Universal Sampling)
%
% This function performs selection with STOCHASTIC UNIVERSAL SAMPLING.
%
% Syntax:  NewChrIx = sus(FitnV, Nsel)
%
% Input parameters:
%    FitnV     - Column vector containing the fitness values of the
%                individuals in the population.
%    Nsel      - number of individuals to be selected
%
% Output parameters:
%    NewChrIx  - column vector containing the indexes of the selected
%                individuals relative to the original population, shuffled.
%                The new population, ready for mating, can be obtained
%                by calculating OldChrom(NewChrIx,:).
%
% Author:     Hartmut Pohlheim (Carlos Fonseca)
% History:    12.12.93     file created
%             22.02.94     clean up, comments
%             22.01.03     tested under MATLAB v6 by Alex Shenfield

function NewChrIx = sus(FitnV,Nsel);

% Identify the population size (Nind)
   [Nind,ans] = size(FitnV);

% Perform stochastic universal sampling
   cumfit = cumsum(FitnV);
   trials = cumfit(Nind) / Nsel * (rand + (0:Nsel-1)');
   Mf = cumfit(:, ones(1, Nsel));
   Mt = trials(:, ones(1, Nind))';
   [NewChrIx, ans] = find(Mt < Mf & [ zeros(1, Nsel); Mf(1:Nind-1, :) ] <=
Mt);

% Shuffle new population
   [ans, shuf] = sort(rand(Nsel, 1));
   NewChrIx = NewChrIx(shuf);


% End of function
```

Appendix G

```
% XOVSP.M       (CROSSOVer Single-Point)
%
% This function performs single-point crossover between pairs of
% individuals and returns the current generation after mating.
%
% Syntax:  NewChrom = xovsp(OldChrom, XOVR)
%
% Input parameters:
%    OldChrom  - Matrix containing the chromosomes of the old
%                population. Each line corresponds to one individual
%                (in any form, not necessarily real values).
%    XOVR      - Probability of recombination occurring between pairs
%                of individuals.
%
% Output parameter:
%    NewChrom  - Matrix containing the chromosomes of the population
%                after mating, ready to be mutated and/or evaluated,
%                in the same format as OldChrom.
%
%  Author:    Hartmut Pohlheim
%  History:   28.03.94     file created
%             22.01.03     tested under MATLAB v6 by Alex Shenfield

function NewChrom = xovsp(OldChrom, XOVR);

if nargin < 2, XOVR = NaN; end

% call low level function with appropriate parameters
   NewChrom = xovmp(OldChrom, XOVR, 1, 0);


% End of function
```

Lower level function

```
% XOVMP.m              Multi-point crossover
%
%       Syntax: NewChrom =  xovmp(OldChrom, Px, Npt, Rs)
%
%       This function takes a matrix OldChrom containing the binary
%       representation of the individuals in the current population,
%       applies crossover to consecutive pairs of individuals with
%       probability Px and returns the resulting population.
```

```
%
%       Npt indicates how many crossover points to use (1 or 2, zero
%       indicates shuffle crossover).
%       Rs indicates whether or not to force the production of
%       offspring different from their parents.
%
%
% Author: Carlos Fonseca,  Updated: Andrew Chipperfield
% Date: 28/09/93,      Date: 27-Jan-94
%
% tested under MATLAB v6 by Alex Shenfield (22-Jan-03)

function NewChrom = xovmp(OldChrom, Px, Npt, Rs);

% Identify the population size (Nind) and the chromosome length (Lind)
[Nind,Lind] = size(OldChrom);

if Lind < 2, NewChrom = OldChrom; return; end

if nargin < 4, Rs = 0; end
if nargin < 3, Npt = 0; Rs = 0; end
if nargin < 2, Px = 0.7; Npt = 0; Rs = 0; end
if isnan(Px), Px = 0.7; end
if isnan(Npt), Npt = 0; end
if isnan(Rs), Rs = 0; end
if isempty(Px), Px = 0.7; end
if isempty(Npt), Npt = 0; end
if isempty(Rs), Rs = 0; end

Xops = floor(Nind/2);
DoCross = rand(Xops,1) < Px;
odd = 1:2:Nind-1;
even = 2:2:Nind;

% Compute the effective length of each chromosome pair
Mask = ~Rs | (OldChrom(odd, :) ~= OldChrom(even, :));
Mask = cumsum(Mask')';

% Compute cross sites for each pair of individuals, according to their
% effective length and Px (two equal cross sites mean no crossover)
xsites(:, 1) = Mask(:, Lind);
if Npt >= 2,
       xsites(:, 1) = ceil(xsites(:, 1) .* rand(Xops, 1));
end
```

```
xsites(:,2) = rem(xsites + ceil((Mask(:, Lind)-1) .* rand(Xops, 1)) ...
                            .* DoCross - 1 , Mask(:, Lind) )+1;

% Express cross sites in terms of a 0-1 mask
Mask = (xsites(:,ones(1,Lind)) < Mask) == ...
                      (xsites(:,2*ones(1,Lind)) < Mask);

if ~Npt,
        shuff = rand(Lind,Xops);
        [ans,shuff] = sort(shuff);
        for i=1:Xops
          OldChrom(odd(i),:)=OldChrom(odd(i),shuff(:,i));
          OldChrom(even(i),:)=OldChrom(even(i),shuff(:,i));
        end
end

% Perform crossover
NewChrom(odd,:) = (OldChrom(odd,:).* Mask) + (OldChrom(even,:).*(~Mask));
NewChrom(even,:) = (OldChrom(odd,:).*(~Mask)) + (OldChrom(even,:).*Mask);

% If the number of individuals is odd, the last individual cannot be mated
% but must be included in the new population
if rem(Nind,2),
  NewChrom(Nind,:)=OldChrom(Nind,:);
end

if ~Npt,
        [ans,unshuff] = sort(shuff);
        for i=1:Xops
          NewChrom(odd(i),:)=NewChrom(odd(i),unshuff(:,i));
          NewChrom(even(i),:)=NewChrom(even(i),unshuff(:,i));
        end
end

% end of function
```

Appendix H

```
% MUT.m
%
% This function takes the representation of the current population,
% mutates each element with given probability and returns the resulting
% population.
%
% Syntax:   NewChrom = mut(OldChrom,Pm,BaseV)
%
% Input parameters:
%
%       OldChrom - A matrix containing the chromosomes of the
%              current population. Each row corresponds to
%              an individuals string representation.
%
%       Pm   - Mutation probability (scalar). Default value
%              of Pm = 0.7/Lind, where Lind is the chromosome
%              length is assumed if omitted.
%
%       BaseV    - Optional row vector of the same length as the
%              chromosome structure defining the base of the
%              individual elements of the chromosome. Binary
%              representation is assumed if omitted.
%
% Output parameter:
%
%       NewChrom - A Matrix containing a mutated version of
%              OldChrom.
%
% Author: Andrew Chipperfield
% Date: 25-Jan-94
%
% Tested under MATLAB v6 by Alex Shenfield (21-Jan-03)

function NewChrom = mut(OldChrom,Pm,BaseV)

% get population size (Nind) and chromosome length (Lind)
[Nind, Lind] = size(OldChrom) ;

% check input parameters
if nargin < 2, Pm = 0.7/Lind ; end
if isnan(Pm), Pm = 0.7/Lind; end
```

```
if (nargin < 3), BaseV = crtbase(Lind);  end
if (isnan(BaseV)), BaseV = crtbase(Lind);  end
if (isempty(BaseV)), BaseV = crtbase(Lind);  end

if (nargin == 3) & (Lind ~= length(BaseV))
   error('OldChrom and BaseV are incompatible'), end

% create mutation mask matrix
BaseM = BaseV(ones(Nind,1),:) ;

% perform mutation on chromosome structure
NewChrom = rem(OldChrom+(rand(Nind,Lind)<Pm).*ceil(rand(Nind,Lind).*(BaseM-
1)),BaseM);
```

Appendix I

```
% BS2RV.m - Binary string to real vector
%
% This function decodes binary chromosomes into vectors of reals. The
% chromosomes are seen as the concatenation of binary strings of given
% length, and decoded into real numbers in a specified interval using
% either standard binary or Gray decoding.
%
% Syntax:       Phen = bs2rv(Chrom,FieldD)
%
% Input parameters:
%
%              Chrom    - Matrix containing the chromosomes of the current
%                         population. Each line corresponds to one
%                         individual's concatenated binary string
%                       representation. Leftmost bits are MSb and
%                          rightmost are LSb.
%
%              FieldD   - Matrix describing the length and how to decode
%                       each substring in the chromosome. It has the
%                       following structure:
%
%               [len;       (num)
%                lb;        (num)
%                ub;        (num)
%                code;      (0=binary     | 1=gray)
%                scale;     (0=arithmetic | 1=logarithmic)
%                lbin;      (0=excluded   | 1=included)
%                ubin];     (0=excluded   | 1=included)
%
%              where
%               len   - row vector containing the length of
%                   each substring in Chrom. sum(len)
%                   should equal the individual length.
%               lb,
%               ub    - Lower and upper bounds for each
%                   variable.
%               code  - binary row vector indicating how each
%                   substring is to be decoded.
%               scale - binary row vector indicating where to
%                   use arithmetic and/or logarithmic
%                   scaling.
%               lbin,
```

```
%               ubin  - binary row vectors indicating whether
%                   or not to include each bound in the
%                   representation range
%
% Output parameter:
%
%              Phen     - Real matrix containing the population phenotypes.
%
% Author: Carlos Fonseca,  Updated: Andrew Chipperfield,
% Date: 08/06/93,          Date: 26-Jan-94,
%
% Tested under MATLAB v6 by Alex Shenfield (17-Jan-03)

function Phen = bs2rv(Chrom,FieldD)

% Identify the population size (Nind)
%      and the chromosome length (Lind)
[Nind,Lind] = size(Chrom);

% Identify the number of decision variables (Nvar)
[seven,Nvar] = size(FieldD);

if seven ~= 7
    error('FieldD must have 7 rows.');
end

% Get substring properties
len = FieldD(1,:);
lb = FieldD(2,:);
ub = FieldD(3,:);
code = ~(~FieldD(4,:));
scale = ~(~FieldD(5,:));
lin = ~(~FieldD(6,:));
uin = ~(~FieldD(7,:));

% Check substring properties for consistency
if sum(len) ~= Lind,
    error('Data in FieldD must agree with chromosome length');
end

if ~all(lb(scale).*ub(scale)>0)
    error('Log-scaled variables must not include 0 in their range');
end
```

```
% Decode chromosomes
Phen = zeros(Nind,Nvar);

lf = cumsum(len);
li = cumsum([1 len]);
Prec = .5 .^ len;

logsgn = sign(lb(scale));
lb(scale) = log( abs(lb(scale)) );
ub(scale) = log( abs(ub(scale)) );
delta = ub - lb;

Prec = .5 .^ len;
num = (~lin) .* Prec;
den = (lin + uin - 1) .* Prec;

for i = 1:Nvar,
    idx = li(i):lf(i);
    if code(i) % Gray decoding
        Chrom(:,idx)=rem(cumsum(Chrom(:,idx)')',2);
    end
    Phen(:,i) = Chrom(:,idx) * [ (.5).^(1:len(i))' ];
    Phen(:,i) = lb(i) + delta(i) * (Phen(:,i) + num(i)) ./ (1 - den(i));
end

expand = ones(Nind,1);
if any(scale)
    Phen(:,scale) = logsgn(expand,:) .* exp(Phen(:,scale));
end
```

Appendix J

```
% REINS.M       (RE-INSertion of offspring in population replacing parents)
%
% This function reinserts offspring in the population.
%
% Syntax: [Chrom, ObjVCh] = reins(Chrom, SelCh, SUBPOP, InsOpt, ObjVCh,
ObjVSel)
%
% Input parameters:
%    Chrom     - Matrix containing the individuals (parents) of the current
%                population. Each row corresponds to one individual.
%    SelCh     - Matrix containing the offspring of the current
%                population. Each row corresponds to one individual.
%    SUBPOP    - (optional) Number of subpopulations
%                if omitted or NaN, 1 subpopulation is assumed
%    InsOpt    - (optional) Vector containing the insertion method
parameters
%                ExOpt(1): Select - number indicating kind of insertion
%                          0 - uniform insertion
%                          1 - fitness-based insertion
%                          if omitted or NaN, 0 is assumed
%                ExOpt(2): INSR - Rate of offspring to be inserted per
%                          subpopulation (% of subpopulation)
%                          if omitted or NaN, 1.0 (100%) is assumed
%    ObjVCh    - (optional) Column vector containing the objective values
%                of the individuals (parents - Chrom) in the current
%                population, needed for fitness-based insertion
%                saves recalculation of objective values for population
%    ObjVSel   - (optional) Column vector containing the objective values
%                of the offspring (SelCh) in the current population, needed
for
%                partial insertion of offspring,
%                saves recalculation of objective values for population
%
% Output parameters:
%    Chrom     - Matrix containing the individuals of the current
%                population after reinsertion.
%    ObjVCh    - if ObjVCh and ObjVSel are input parameters, then column
%                vector containing the objective values of the individuals
%                of the current generation after reinsertion.
%
% Author:     Hartmut Pohlheim
% History:    10.03.94     file created
```

```
%            19.03.94     parameter checking improved
%            26.01.03     tested under MATLAB v6 by Alex Shenfield

function [Chrom, ObjVCh] = reins(Chrom, SelCh, SUBPOP, InsOpt, ObjVCh,
ObjVSel);

% Check parameter consistency
   if nargin < 2, error('Not enough input parameter'); end
   if (nargout == 2 & nargin < 6), error('Input parameter missing: ObjVCh
and/or ObjVSel'); end

   [NindP, NvarP] = size(Chrom);
   [NindO, NvarO] = size(SelCh);

   if nargin == 2, SUBPOP = 1; end
   if nargin > 2,
      if isempty(SUBPOP), SUBPOP = 1;
      elseif isnan(SUBPOP), SUBPOP = 1;
      elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
   end

   if (NindP/SUBPOP) ~= fix(NindP/SUBPOP), error('Chrom and SUBPOP
disagree'); end
   if (NindO/SUBPOP) ~= fix(NindO/SUBPOP), error('SelCh and SUBPOP
disagree'); end
   NIND = NindP/SUBPOP;  % Compute number of individuals per subpopulation
   NSEL = NindO/SUBPOP;  % Compute number of offspring per subpopulation

   IsObjVCh = 0; IsObjVSel = 0;
   if nargin > 4,
      [mO, nO] = size(ObjVCh);
      if nO ~= 1, error('ObjVCh must be a column vector'); end
      if NindP ~= mO, error('Chrom and ObjVCh disagree'); end
      IsObjVCh = 1;
   end
   if nargin > 5,
      [mO, nO] = size(ObjVSel);
      if nO ~= 1, error('ObjVSel must be a column vector'); end
      if NindO ~= mO, error('SelCh and ObjVSel disagree'); end
      IsObjVSel = 1;
   end

   if nargin < 4, INSR = 1.0; Select = 0; end
   if nargin >= 4,
```

```
      if isempty(InsOpt), INSR = 1.0; Select = 0;
      elseif isnan(InsOpt), INSR = 1.0; Select = 0;
      else
         INSR = NaN; Select = NaN;
         if (length(InsOpt) > 2), error('Parameter InsOpt too long'); end
         if (length(InsOpt) >= 1), Select = InsOpt(1); end
         if (length(InsOpt) >= 2), INSR = InsOpt(2); end
         if isnan(Select), Select = 0; end
         if isnan(INSR), INSR =1.0; end
      end
   end

   if (INSR < 0 | INSR > 1), error('Parameter for insertion rate must be a
scalar in [0, 1]'); end
   if (INSR < 1 & IsObjVSel ~= 1), error('For selection of offspring
ObjVSel is needed'); end
   if (Select ~= 0 & Select ~= 1), error('Parameter for selection method
must be 0 or 1'); end
   if (Select == 1 & IsObjVCh == 0), error('ObjVCh for fitness-based
exchange needed'); end

   if INSR == 0, return; end
   NIns = min(max(floor(INSR*NSEL+.5),1),NIND);   % Number of offspring to
insert

% perform insertion for each subpopulation
   for irun = 1:SUBPOP,
      % Calculate positions in old subpopulation, where offspring are
inserted
         if Select == 1,    % fitness-based reinsertion
            [Dummy, ChIx] = sort(-ObjVCh((irun-1)*NIND+1:irun*NIND));
         else               % uniform reinsertion
            [Dummy, ChIx] = sort(rand(NIND,1));
         end
         PopIx = ChIx((1:NIns)')+ (irun-1)*NIND;
      % Calculate position of Nins-% best offspring
         if (NIns < NSEL),  % select best offspring
            [Dummy,OffIx] = sort(ObjVSel((irun-1)*NSEL+1:irun*NSEL));
         else
            OffIx = (1:NIns)';
         end
         SelIx = OffIx((1:NIns)')+(irun-1)*NSEL;
      % Insert offspring in subpopulation -> new subpopulation
         Chrom(PopIx,:) = SelCh(SelIx,:);
```

```
        if (IsObjVCh == 1 & IsObjVSel == 1), ObjVCh(PopIx) =
ObjVSel(SelIx); end
   end

% End of function
```