PERFORMANCE OPTIMIZATION OF MILLIMETER WAVE LENSING USING METAMATERIAL CONCEPT

NSF Summer Undergraduate Fellowship in Sensor Technologies
Amber Sallerson (Mathematics) – University of Maryland, Baltimore County
Advisor: Nader Engheta

ABSTRACT

Because of impracticalities resulting from their much larger size, lensing systems in a millimeter wave band cannot be dealt with in the same manner as optical systems. One technique for implementing this lensing is to transform the phase of the incident field, using metamaterials such as frequency-selective surfaces (FSSs). In this project, the FSSs known as the gangbuster surface (GS) is used to achieve the lensing effect.

We consider the GS above a perfectly conducting ground plane. Incident waves reflect totally, and the phase of the reflected waves depends on characteristics such as the type and period of the GS, the lengths of the wires, and the separation between the GS and the ground plane. Any change in the various parameters used to design the lens results in deterioration in the profile field intensity and the diffraction efficiency, among other unwanted effects.

The goal is to optimize the lens for changes in the angle of incidence such that the intensity of the electric field and the resolution at the focal plane are both optimized. To accomplish this optimization, differences in phase distribution for various angles are minimized by using an existing genetic algorithm to modify the lengths of wires on the GS. The incorporation of an objective function into the genetic algorithm returns the optimal length of wire needed to achieve the desired phase distribution. Through the use of these different wire lengths, a GS can be constructed for a lensing system in the millimeter wave band that is stable for differing angles of incidence.

1. INTRODUCTION

The creation of a new lensing system requires not only the physical engineering of such a system, but also optimization of the proposed scheme. The goal of this research is to create a lensing system for the millimeter wave band that is sufficiently small and has a stable performance for differing angles of illumination. Such a system would be useful in remote sensing, image processing, and communication applications.

1.1 Lensing System

Advanced lensing systems are currently implemented mainly as optical and infrared (night vision) systems. These systems deal with wavelengths on the scale of visible light or infrared waves and require a lens of corresponding size. Research today presents a challenge to this area by establishing the need for a lensing system equipped to deal with the millimeter wave band. The millimeter wave band consists of wavelengths that are approximately 4,000 to 10,000 times larger than those of the optical system. This relationship implies that a lens of many times the magnitude of those currently being used would be needed to create an equivalent stable lensing system. Such a lens would be too large for any practical applications.

A metamaterial concept can be used to avoid the size implications of the millimeter wave band. This concept is based on the idea that to focus a field, it is enough to change a phase distribution of the transmitted or reflected field, at a given surface, in accordance with a special law of optical theory. Frequency selective surfaces (FSSs) can change the phase of both the transmitted and reflected fields, depending on their local geometry. Further, to avoid loss of energy one can deal with a totally reflective system. This would mean using a FSS above the perfectly conducting ground plane as an alternative lensing system that will be smaller than the typical system.

1.2 Optimization

During the latter part of the nineteenth century, the biological sciences underwent a revolution when Charles Darwin discovered the processes by which nature selects and optimizes organisms fit for life [1]. These processes together with the basic laws of genetic inheritance and the powerful computational techniques available through computers enable us to apply nature’s optimization through a method called the genetic algorithm (GA) [2].

Genetic algorithms fall under the category of optimization schemes known as robust stochastic search methods [3]. GAs are particularly effective for solving for global maxima in multidimensional and multimodal problems. In their most basic form, GAs are function optimizers that seek extrema of a given objective function. GAs simultaneously process a population of points in the space that is to be optimized and, using stochastic operators, make the transition from one generation of points to the next, making the optimization process less likely to become trapped in a weak local extrema [4]. For these reasons, the GA is commonly applied to electromagnetics and is one of the obvious choices for optimizing the proposed lensing system.

1.3 Problem

An optical system is designed so that with the aperture and the focal point fixed, any other change in the system’s parameters will not affect performance. When dealing with FSSs this is not the case; there is a trade-off between size and stability. Because the FSS allows for a drastically smaller system than a conventional optical system, it has additional characteristics beyond aperture and focal distance - such as angle of incidence - for which a change in the parameter results in a deterioration of the system's performance.

The goal of this project is to produce a lensing system using an FSS that is stable, within a certain range, for differing angles of the incident waves. The approach is to break up the FSS into a finite number of regions. Within each region the length of the wires is constant. The GA is used to determine the optimal length of the wires for each interval, with the overall goal being to maximize intensity and resolution independent of incident angle.

The general code for the GA was written by Dr. David L. Carroll at the University of Illinois and was provided by Dr. Ahmad Hoofar of Villanova University. The program was carried out in FORTRAN.

2. GEOMETRY

The system consists of a ground plane, above which lies the FSS. The ground plane ensures that all waves will reflect completely. The incoming waves are subject to the constraint of the aperture. The waves reflect from the FSS-ground plane onto the focal plane. The focal plane location is determined by the focal distance of the system. The system is depicted in Figure 1.

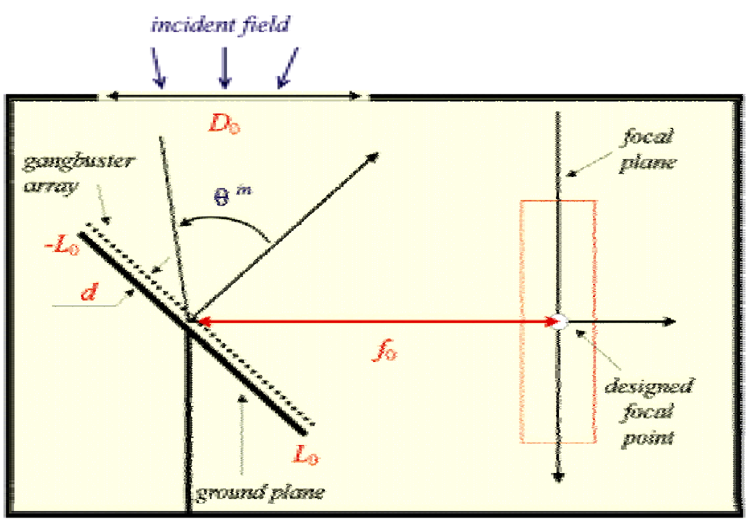


Figure 1: Geometry of lensing system in two dimensions.
Focal distance is represented by f_0.

The GS is the FSS used in this project. GSs are a type of 2-dimensional supercondensed periodic array of thin metal wires of finite length [5]. The wires can vary in length, but the surface for these purposes is designed with a set wire slope and distance between wire centers. Other characteristics inherent to the GS are also set during the creation of the actual surface.

The parameters used include (1) the distances between wire centers, D, which is $D = 0.25$ millimeters (mm); (2) the slope of the wires, n, with $n = 5$; (3) the diameter of the wires, a, with $a = 0.001$ mm; (4) and the separation between the GS and the ground plane, d_0, which is $d_0 = 0.1$ mm. The GS is to be optimally designed and then placed into the system, thus creating a millimeter wave lensing system based on a metamaterial concept. Figure 2 depicts the basic geometry of a GS.

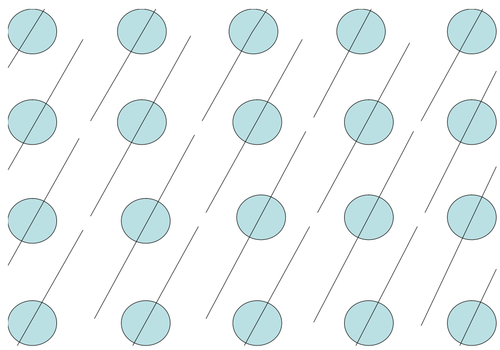


Figure 2: A cross-section of a GS with the circles representing the centers of wires. The phase of the reflected field is dependent upon the wire lengths.

Once in place the gangbuster array does not change, which is why the lengths should be optimized for the system to work regardless of the angle of the incoming wave.

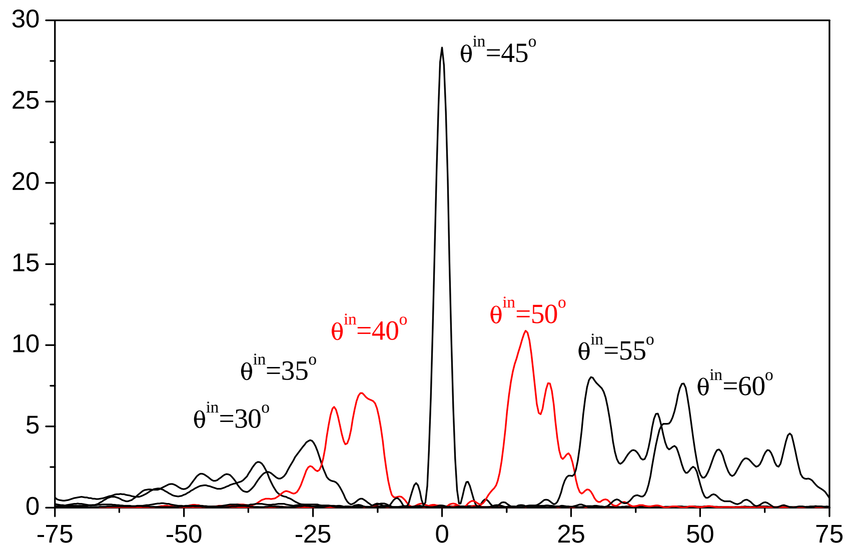
3. METHODS

In this project, the Method of Moment based on the Floquet modes expansion is used to obtain a shunt impedance of the GS with uniform distribution of wire lengths [5]. Knowing the shunt impedance of the GS, one can get a phase of reflection coefficient of the GS above a ground plane by means of a Transmission Line Model Method. Then, taking into account the fact that a lens's design phase does not assume large changes in wire lengths, one can change GS wire lengths according to the phase distribution needed. Knowing the phase of the reflection coefficient of the incident field on the lens surface allows one to obtain the total field on the lens surface, and then to apply the surface equivalence theorem (Huygens's principle) [6]. Therefore, we can obtain any characteristics (in our case, intensity of the electrical field in the focal plane) of the field excited by equivalent currents on the lens surface.

4. OBJECTIVE/DESIGN

The goal of the project is to create a GS and thus a lensing system that works for varying angles of incidence. To determine whether the system works the intensity and resolution of the lensing system are examined.

A system designed for a 45° angle resulted in a high intensity for incoming waves that are at 45°, but comparatively very low intensity for other angles of incidence. Also, the difference between the main peak of the 45° angle and its side lobes is much greater than the difference between other angles' main peaks and side lobes. Ideally, for a single



designed system the goal is to have all incidence angles, within a reasonable range, result in graphs of intensity such that all the main peaks are relatively close to one another in magnitude of intensity and all have relatively small side lobes.

Figure 3: Graph of intensity for a cross section of the focal plane as the result of a system designed for a 45° angle.

The phase of a reflected field is dependent upon the wire lengths, the surface separation, and the period of the GS and upon the amplitude, phase, and angle of incidence of the incident wave. The electrical fields of the reflected waves are dependent upon the phase of the reflected field, and the intensity of the system is related to that electrical field. The link between all the parameters is what accounts for the ability to construct a GS with certain wire lengths in order to create the desired intensity distribution. Thus the relationship between the GS and intensity is utilized. Equations 1 through 3 demonstrate these relationships. (We use the time dependence of $e^{i\omega t}$)

$$R = e^{i\Psi} \tag{1}$$

Equation 1: The length of the wires on the GS results in the reflection coefficient R. Here Ψ is the phase, thus relating wire length and phase.

$$E_x = \iint_{y'z'} \frac{1}{4\pi}\left[-i\frac{\eta_0}{k_0}H_y^{tot}(x',y')\frac{\partial^2 G(x,x',y,y')}{\partial x\partial z} + E_z^{tot}\frac{\partial G(x,x',y,y')}{\partial z}\right]dy'dz' \tag{2.1}$$

$$E_y = \iint_{y'z'} \frac{-i}{4\pi} \frac{\eta_0}{k_0} H_y^{tot}(x', y') \frac{\partial^2 G(x, x', y, y')}{\partial y \partial z} dy'dz' \quad (2.2)$$

$$E_z = \iint_{y'z'} \frac{-i}{4\pi} \left[k_0 \eta_0 H_y^{tot}(x', y') G(x, x', y, y') + \frac{1}{k_0} \eta_0 H_y^{tot}(x', y') \frac{\partial^2 G}{\partial z^2} - i E_z^{tot} \frac{\partial G}{\partial x} \right] dy'dz' \quad (2.3)$$

Equation 2: Electric field components each have separate equations, for the general 3-D case. Derivation of the variables is available in the Appendix.

$$I(y'') = \left| E_{z''}(y'', \Psi) \right|^2 \quad (3)$$

Equation 3: I is intensity, E is electric field, y is position, and Ψ is phase.

5. GENETIC ALGORITHM

The GA is applied in numerous disciplines and thus is in a very general form. To make the GA specific to a problem requires creating an objective function (the function that the GA will optimize) and linking it to the GA code. The objective function used finds the difference between the phases dependent on the wire lengths and the desired phase distribution. The objective function results in the fitness of that generation according to equation 4.

$$\text{Fitness} = \sum_{n=1}^{N} \sum_{m=1}^{M} \left| \Psi_{real}(\alpha_m, x_n, l_n) - \Psi_{optimal}(\alpha_{m,} x_n) \right| \quad (4)$$

The optimal phase was initially the design phase based on each angle. To have the program converge faster the optimal phase was instead calculated to be the average of all the design phases. Thus the average optimal phase is the phase being subtracted from the real phase (phase as the result of the GA) in equation 4. This allows for a more even distribution of phase from the GA's set of wire lengths.

In order to have the GA converge faster, an initial guess for the lengths is read into the program from a file containing the optimal lengths for a design phase based on a 45^0 angle. That set of initial lengths is changed randomly and assigned to each set of parents, where a parent is simply an array holding all of the parameters to be optimized. The fitness of the initial parents is determined and stored via equation (4). The parents then produce children - elements of the parents that have possibly been through processes called mutation, crossover, and creeps. These processes result in changes in the chromosomes of the new generation. Thus making the children not only the result of the original parent's chromosomes, but also the altered chromosomes. The children's fitness is determined and compared to that of the parents. The most fit elements of the two groups replace the old members of parents. The new set of parents is more fit than the last, and the process continues for a specified number of generations.

The most fit parents from all generations after the specified number of runs is the proposed set of wire lengths to be used for the GS and thus the millimeter wave lensing system.

6. RESULTS

Following each run of the GA, the set of optimized wire lengths is examined by viewing the graph of phase versus position. The graph includes the phase distribution resulting from the set of wire lengths that is output, as well as the optimal phase, for the range of angles used. A run's success is determined by comparing the difference between the optimal phase for a given angle and the phase that resulted from the given wire lengths. Figure 4 shows one such graph used for analysis of a particular run of the GA.

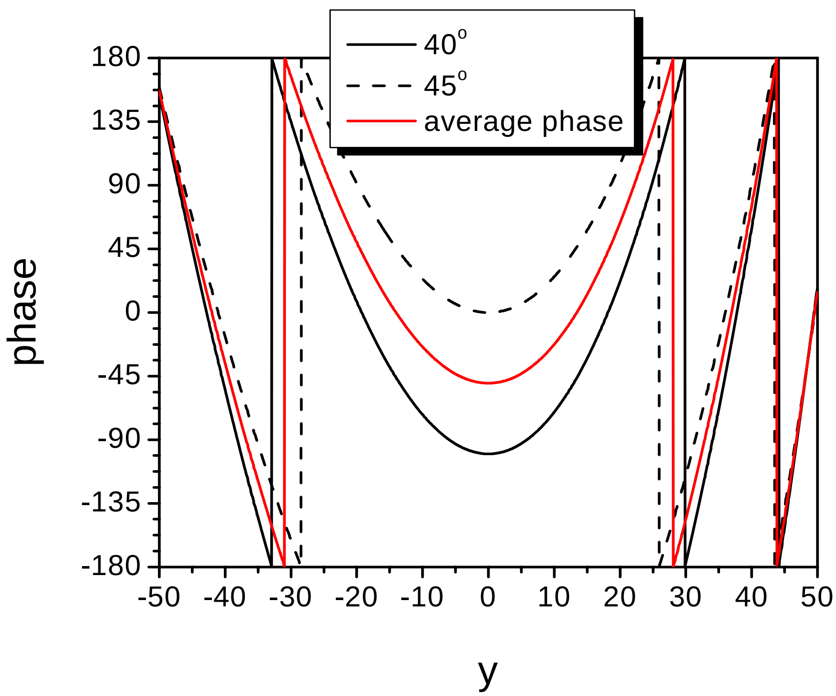


Figure 4: Design phases for 40^0 and 45^0 as well as the average phase, which is used as the input to the GA.

The length of the GS as well as its other set parameters determines how many intervals the surface can be divided into. The maximum number of intervals can be determined by applying equation 5.

$$\text{Maximum number of wires} = \frac{L}{D_0} \tag{5}$$

Equation 5: L is the length of the GS and D_0 is the distance between wires.

$$D_0 = \frac{D}{\sqrt{1+n^2}} \tag{6}$$

Equation 6: D is the distance between wire centers and n is the slope of the wires.

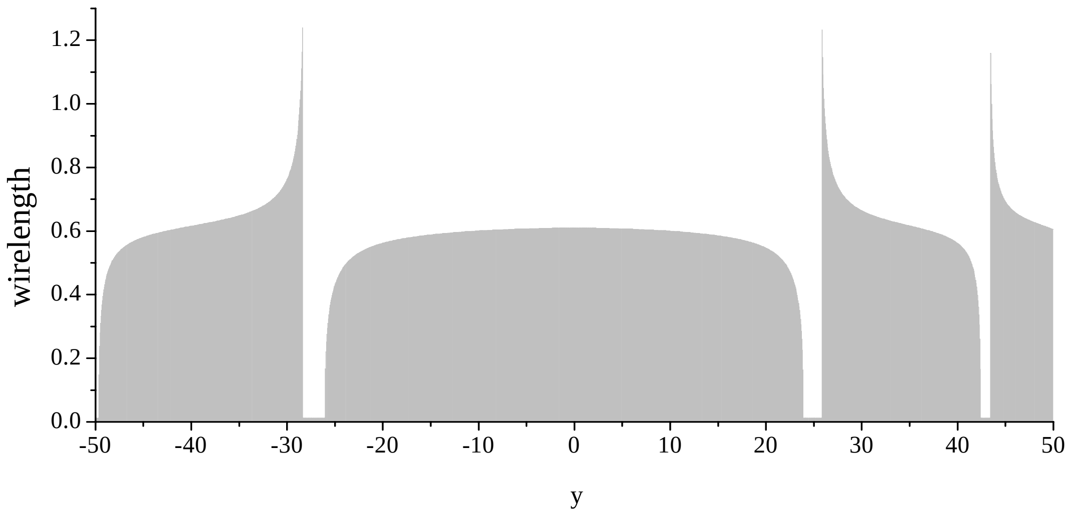
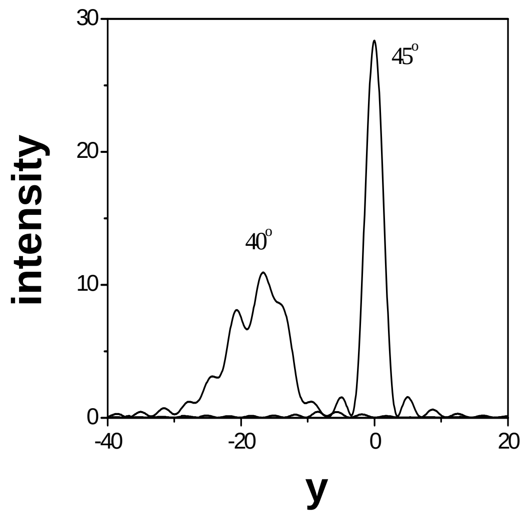


Figure 5: Initial guess for wire lengths used in GA (based on desired average phase)

Using the wire lengths given above by the GA, the phase, the electric field, and the intensity can all be computed. As compared with the graph of intensity found in Figure 6, it is clear that the maximum intensity of the angles is only slightly closer together, but the ratio of the main peaks to their side lobes is larger overall for Figure 7, thus giving better resolution to the reflected image. Of course, the data for the 45° angle has been deteriorated somewhat since the design is no longer optimized for that angle alone. The results below are for angles as far apart as 5°, but theoretically we are approaching a system that should work satisfactorily regardless of the angle of incidence.



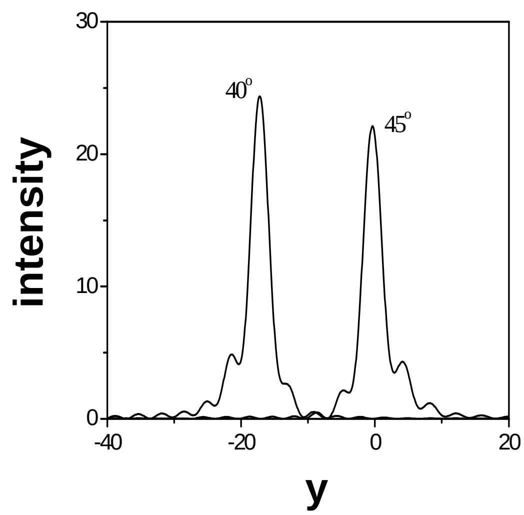


Figure 6: Intensity for a cross section of the focal plane as the result of a system designed for a 45° angle.

Figure 7: Intensity for a cross section of the focal plane using the average phase distribution .

7. RECOMMENDATIONS

Throughout the duration of the project several different objective functions and methods for finding the data used by the objective functions have been tested. The process still does not converge very quickly toward an acceptable answer. Further improvements on the methods used that result in faster convergence would allow for more runs and thus better results.

Although the system in this case is optimized for varying angles of incidence, other parameters can be examined. The general GA is written to optimize a system for numerous parameters at once. Altering the objective function and making some other minor adjustments to the GA would allow for an even more effective GS and lensing system.

All work for this project was done for the 2-dimensional cases. In reality researchers need information based on 3-dimensional models in order to build the GS and lensing system. This adaptation would require changes to the objective function. The code for calculating all the necessary data for a 3-dimensional model already exists and is accessible as Fortran code. The author of the code, Nataliya Blyznyuk, is a postdoctoral student at the University of Pennsylvania working with Dr. Nader Engheta from the Electrical Engineering Department.

8. ACKNOWLEDGMENTS

I owe a great deal to Professor Nader Engheta for participating as an advisor to the SUNFEST Program and for enthusiastically supporting my work. I am also grateful to

Nataliya Blyznyuk, whose help was critical toward the work accomplished during the ten-week internship. Thanks to the National Science Foundation for their support through an NSF-REU grant and to Microsoft Corporation for their financial support.

9. REFERENCES

1. C. Darwin, *The Origin of Species*. Penguin, London, U.K., 1985.

2. D.S. Weile and E. Michielssen, Genetic algorithm optimization applied to electromagnetics: A review, *IEEE Trans. Antennas Propagat.*, 45 (1997) 343.

3. S. Chakravarty, R. Mittra, and N.R. Williams, On the application of the microgenetic algorithm to the design of broad-band microwave absorbers comprising frequency-selective surfaces embedded in multilayered dielectric media, *IEEE Trans. Microwave Theory Tech.*, 49 (2001) 1050.

4. G. Manara, A. Monorchio, and R. Mittra, Frequency selective surface design based on genetic algorithm, Electronic Letters, 35 (1999).

5. C.A. Moses, *A theoretical study of electromagnetic feedforward / feedbackward media and wire media*. PhD thesis, University of Pennsylvania, 1997.

6. C.A. Balanis, *Advanced Engineering Electromagnetics*, Wiley, New York, 1989.

APPENDIX A

$$E_x = \iint_{y'z'} \frac{1}{4\pi}\left[-i\frac{\eta_0}{k_0}H_y^{tot}(x',y')\frac{\partial^2 G(x,x',y,y')}{\partial x\partial z} + E_z^{tot}\frac{\partial G(x,x',y,y')}{\partial z}\right]dy'dz' \quad (2.1)$$

$$E_y = \iint_{y'z'} \frac{-i}{4\pi}\frac{\eta_0}{k_0}H_y^{tot}(x',y')\frac{\partial^2 G(x,x',y,y')}{\partial y\partial z}dy'dz' \quad (2.2)$$

$$E_z = \iint_{y'z'} \frac{-i}{4\pi}\left[k_0\eta_0 H_y^{tot}(x',y')G(x,x',y,y') + \frac{1}{k_0}\eta_0 H_y^{tot}(x',y')\frac{\partial^2 G}{\partial z^2} - iE_z^{tot}\frac{\partial G}{\partial x}\right]dy'dz' \quad (2.3)$$

E_x is the electric field in the x direction, E_y the electric field in the y direction, and E_z is the electric field in the z direction.

$$k_0 = \omega\sqrt{\varepsilon_0\mu_0} \qquad \eta_0 = \sqrt{\frac{\mu_0}{\varepsilon_0}}$$

The value k_0 is the wavenumber, the angular frequency of the system is represented by ω, electric permittivity is ε_0, magnetic permittivity is μ_0, and the free space wave impedance is given by η_0.

$$E_z^{in} = E_0 e^{-ik_0\left(y\sin\vartheta^{in} - x\cos\vartheta^{in}\right)}$$
$$E_z^{refl} = E_0\bar{R}\left(y'\right)e^{-ik_0\left(y\sin\vartheta^{in} + x\cos\vartheta^{in}\right)}$$
$$E^{tot} = E_z^{\ in} + E_z^{\ refl}$$
$$H_y^{in} = -\frac{1}{\eta_0}\cos\vartheta E_0 e^{-ik_0\left(y\sin\vartheta^{in} - x\cos\vartheta^{in}\right)}$$
$$H_y^{refl} = \frac{1}{\eta_0}\bar{R}\left(y'\right)\cos\vartheta E_0 e^{-ik_0\left(y\sin\vartheta^{in} + x\cos\vartheta^{in}\right)}$$
$$H^{tot} = H_y^{\ in} + H_y^{\ refl}$$

Here $\bar{R}$ is the reflection coefficient, x and y are coordinates, and ϑ is the angle of incidence.

$$G(x', y') = -i\pi H_0^{(2)}\left(k|r - r'|\right)$$
$$r = \sqrt{(x - x')^2 + (y - y')^2}$$

G is known as Green's Function and is dependent upon x' and y', which are the coordinates of the source point. The coordinates of the observation point are represented here by x and y.

APPENDIX B

FORTRAN code for the objective function used in the GA.

```
subroutine func(j,funcval)
use arrays
implicit none
integer j,n
real*8 fitness_sum, funcval
            real*8 parent(nparmax,indmax)
real*8 iparent(nchrmax,indmax)

common / ga3 / parent,iparent
desl_w=parent(:,j)! GA length of wires on the
gangbuster for jth individuals
do n=1, Malv
call phases(n)! this subroutine returns phases for given
wirelengths
enddo
funcval=fitness_sum()! returns fitness
return
end
!------------------------------
 real*8 function fitness_sum()
use arrays
implicit none
integer n,m,nparanew
real*8
res,sum_phase,varm(nparmax,Mal),vars(nparmax,Mal
)
common /nparanew/ nparanew
res=0d0
varm=psi_real - psi_opt
vars=psi_real + psi_opt
do n=1,Malv
do m=1, nparanew
res=res+(DABS(varm(m,n))/DABS(vars(m,n)))**2d0
            enddo
enddo
fitness_sum=1d0/res
return
 end
!-------------------------
subroutine phases(j)
use arrays
implicit none
real*8 DQDVAL
logical check
integer n,j
real*8Xdata(Nlma),Fdata(Nlma)
psi_real(:,j)=0d0
check=.true.
do n=1,Nlmax
Xdata(n)=dfloat(n)*Lstep
Fdata(n)=G_ph(n,j)
enddo
do n=1, nparmax
psi_real(n,j)=DQDVAL(desl_w(n),Nlmax,Xdata,Fdat
a,Check)
enddo
END
```

```
MODULE gangbuster
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
! number of angles of incidence
integer, parameter :: Mal=2
integer, save :: Malv=2
! array of alphas
real*8 alpha(Mal)
! values of angles (should be Mal numbers)
data alpha /40D0,45D0/
! psi optimization
integer, parameter :: N_psi_step_p=1000
integer, save :: N_psi_step=1000
real(8), save :: diff
real(8), save :: psi_step
! incident field parameters
! polarization of incident field
!'1' means that electric field has the only component
along wires (phi-polarization)
!'2' means that electric field has two components,
along wires and perpendicular
! to wires (theta-polarization)
! right now codes uses phi-polarization only (=1)
integer, save ::polarization =1
! wavelength of incident field
real(8), save ::lambda0=1.8d0
!~~~~~~~~~~~~~~~~~~~~~~~
!Gangbuster parameters

real(8), save ::gD = 0.25D0! lattice spacing for square
gangbuster
real(8), save ::gn = 5d0! number of gangbuster
real(8), save ::gag = 1d-3! wire element radius
real(8), save ::Lstep=0d0! step along a wire
integer, parameter ::Nlma=100! number of steps along
wire
integer, save ::Nlmax=100! number of steps along
wire
real(8), save ::d= 0.1D0! spacing between the lower
gangbuster and the ground plane
real(8), save ::Lmax! maximum allowed wirelength
real(8), save ::poss_shift=0.1D0! possible shift
allowed for initial guess of wirelength (multiplying
factor)

! MOM and TL model
integer, save ::PQ = 8! number of current modes
integer, save ::KN = 80! number of Floquet modes
real(8), save ::E0! amplitude of incident field for
gangbuster
real(8), save ::etaTL_Z_U! etaTL for upper plane
real(8), save ::etaTL_Y_L! etaTL for lower plane
real(8), save ::kTL! TL wavenumber
real(8), save ::Z0=376.99111843077512D0! free-
space wave impedance
real(8), save ::Ez0=1d0! amplitude of E_z component
(parallel polarization)
```

```
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
! lens
! focal plane design angle
real(8), save ::theta0=45d0 * 0.01745329251994D0

! L0 is the actual lens radius
real(8), save ::L0=50D0

! focus
! projections of normalized focal distance in (y,x)
coordinate system
! center is in the center of lens, y-axis is along lens

real(8), save ::fx0=144.4213562373D0
real(8), save ::fy0=144.4213562373D0

integer, parameter ::foxl=0! number of steps along
focal plane in x direction
integer, parameter ::foyl=500! number of steps along
focal plane in y direction
integer, save ::focxl=0! the same as fox
integer, save ::focyl=500! the same as foy
integer, parameter ::foxu=0! number of steps along
focal plane in x direction
integer, parameter ::foyu=500! number of steps along
focal plane in y direction
integer, save ::focxu=0! the same as fox
integer, save ::focyu=500! the same as foy
real(8), save ::dx1=0.1D0! step in focal plane
real(8), save ::dy1=0.1D0! step in focal plane

END MODULE gangbuster
!~~~~~~~~~~~~~~~~~~~~~~~~~
MODULE arrays
use gangbuster
use params
public

real(8), save :: rad = 0.01745329251994D0
!multiplying factor for degree-radian transformation
save
real(8) in_wires(nparmax)! array of GA wirelengths
(initial guess
real(8) desl_w(nparmax)! array of GA wirelengths
real(8) coord(nparmax)! array of coordinates of wires
real(8) psi_opt(nparmax,Mal)! 2D array of optimal
phase of GS
real(8) psi_real(nparmax,Mal)! 2D array of real phase
of GS
real(8) G_ph(Nlma,Mal)! phase of reflection
coefficient versus wirelengths
! of the uniform gangbuster
real(8) G_ph42(Nlma)! phase of reflection coefficient
versus wirelengths
! of the uniform gangbuster
        real(8) aver_psi_opt(nparmax)
real(8) ksi(0:N_psi_step_p)
!real(8) aver_psi_opt(nparmax)
complex(8) y_z_arr(-foxl:foxu,-foyl:foyu)! Ez field
integrands along z-axis at the lens
complex(8) Ezu(-foxl:foxu,-foyl:foyu)! contains
intensities of the field in the focal space
complex(8) Ezl(-foxl:foxu,-foyl:foyu)! contains
intensities of the field in the focal space
complex(8) Ez_arr(nparmax)

END MODULE arrays
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MODULE ipi
public
real(8), save ::pi = 3.14159265358979323846264 3D0
complex(8), save :: i = ( 0d0, 1d0 )

END MODULE ipi
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MODULE maths
IMPLICIT NONE
real(8), private :: x,y,a

CONTAINS
!-------------------------------------
real(8) function radianphase(z)
complex(8) z
x=dreal(z)
y=dimag(z)
radianphase=DATAN2(y,x)
end function radianphase
!----------------------------------
real(8) function sinp(n,b)
USE ipi
integer n
real(8) b
a=dfloat(2*n-1)*pi/2d0
sinp=sinc(a-b)+sinc(a+b)
end function sinp
!--------------------------------
real(8) function sinq(n,b)
USE ipi
integer n
real(8) b
a=dfloat(n)*pi
 sinq=sinc(a-b)-sinc(a+b)
end function sinq
!---
real(8) function sinc(b)
USE ipi
real(8) b
if (b.eq.0d0) then
sinc=1d0
else
sinc=DSIN(b)/b
endif
end function sinc
!---
complex(8) function sqrtc(z)
USE ipi
complex(8) z
sqrtc=dsqrt(cdabs(z))*cdexp(i*(datan2(dreal(z),-
dimag(z))-pi/2d0)/2d0)
end function sqrtc
!------------------------------------------
END MODULE maths
! ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
MODULE coordinates
```

```
public
real(8), save :: si,co, galp

CONTAINS
!-------------------------------
real(8) function coord_y(m,n)
use gangbuster
implicit none
integer m,n
real(8) tau,zeta

tau=dfloat(n)*gD
zeta=dfloat(m)*gD
coord_y=tau*co-zeta*si

end function coord_y
!------------------------------
real(8) function coord_z(m,n)
use gangbuster
implicit none
integer m,n
real(8) tau,zeta

tau=dfloat(n)*gD
zeta=dfloat(m)*gD
coord_z=tau*si+zeta*co

end function coord_z
!-----------------------------
END MODULE coordinates

MODULE Green
! contains derivatives of the Green function
public
complex(8), save :: G
complex(8), save :: d_dxG

END MODULE Green

INTENSITY
!~~~~~~~~~~~~~~~~~~~~~~~~
subroutine intensity
!~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
integer k,nparanew
common /nparanew/ nparanew
Call read_data_arr
open(unit=11111, file='actual phases.dat')
do k=1, nparanew
write(11111,'(4F15.8)') coord(k), in_wires(k),
psi_real(k,1),psi_real(k,2)
enddo
close(unit=11111)

Call loops ! finding field characteristic for diff.
lambdas and angles of incidence
 End
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
!~~~subroutine read_data_arr
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
 implicit none
 integer k,nparanew
real*8 xx
common /nparanew/ nparanew
open(unit=11111, file='our results.dat')
do k=1, nparanew
read(11111,'(7F15.8)')
xx,xx,xx,xx,xx,psi_real(k,1),psi_real(k,2)
enddo
close(unit=11111)
psi_real=psi_real/57.2957795131D0
end
!~~~~~~~~~~~~~~~~~~~
SUBROUTINE Loops
!~~~~~~~~~~~~~~~~~~~
!loops to vary theta and lambda
!~~~~~~~~~~~~~~~~~~~

Use arrays
implicit none
integer plane
real(8) lambda
common /plane/ plane
Common /lambda/ lambda
!______________________________________________
____________
lambda=lambda0
!______________________________________________
____________
! first angle
plane=1
call wavenumber(alpha(plane))
CALL field
!______________________________________________
____________
! second angle
plane=2
call wavenumber(alpha(plane))
CALL field
!______________________________________________
____________
CALL print_results
 END
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
SUBROUTINE field
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
!returns actual phase values at the points of
!centers of wires at the lens
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
integer m,n,plane
real(8) x1,y1,x,y
common /coord/ y,x
common /plane/ plane

!~~~ writing 2D picture of Intensity ~~~
select case (plane)
 case (1)
Ezu=(0.0D0, 0.0D0)
 case (2)
Ezl=(0.0D0, 0.0D0)
 end select
do n=-focyl,focyu
```

```
do m=-focxl,focxu
x1=dfloat(m)*dx1
y1=dfloat(n)*dy1
x=fx0+x1*dcos(theta0)-y1*dsin(theta0)
y=fy0+x1*dsin(theta0)+y1*dcos(theta0)

Call Get_Intensity(m,n)
select case (plane)
case (1)
Ezu(m,n)=y_z_arr(m,n)
print*,'Intensity of Ez=',y1,cdabs(Ezu(m,n))**2d0
case (2)
Ezl(m,n)=y_z_arr(m,n)
print*,'Intensity of Ez=',y1,cdabs(Ezl(m,n))**2d0
end select
enddo
 enddo

!~~~ end of writing 2D picture of Intensity ~~~
END
!~~~~~~~~~~~~~~~~~~~~~~~~~
SUBROUTINE print_results
!~~~~~~~~~~~~~~~~~~~~~~~~
!returns actual phase values at the points of
!centers of wires at the lens
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
use rad_deg
use maths
implicit none
integer m,n
real(8) lambda,theta_y,theta_z,x1,y1,y
open (UNIT=210, FILE='Intens_1.dat')
open (UNIT=211, FILE='Intens_2.dat')
 do n=-focyl,focyu
do m=-focxl,focxu
x1=dfloat(m)*dx1
y1=dfloat(n)*dy1
write(210,'(2F15.5,4F24.14)')
y1,cdabs(Ezu(m,n))**2d0,cdabs(Ezl(m,n))**2d0
write(211,'(2F15.5,4F24.14)')
y1,x1,Ezl(m,n),radianphase(Ezl(m,n)),cdabs(Ezl(m,n))
**2d0
enddo
enddo
close(unit=210)
close(unit=211)
END
!~~~SUBROUTINE Get_Intensity(m,n)
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!computes E field components at the certain point of
space
!given by common block /coord/
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
implicit none
integer m,n

CALL E_arrays
CALL Int_y_array(m,n)
END
!~~~~~~~~~~~~~~~~~~~~~~~~~
SUBROUTINE Int_y_array(m,n)
!~~~~~~~~~~~~~~~~~~~~~~~~~
!returns the array* with Z integrand values at the lens
!~~~~~~~~~~~~~~~~~~~~~~~~~
Use arrays
integer m
real*8 ER,FLAG
complex*16 RES
complex*16YapproxZ
externalYapproxZ

CALL INTEGLAW(YapproxZ,-L0,L0,0D0,1.D-
4,ER,NN,FLAG,res)
y_z_arr(m,n)=res
END
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
COMPLEX(8) FUNCTION YapproxZ(x)
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
!returns the integrand value at points m,n
!~~~~~~~~~~~~~~~~~~~~~~~~~~
use IMSLF90
use ipi
use arrays
real(8) a11,b11
real*8 DQDVAL,x
logical check
integer n,nparanew
real*8Xdata(nparmax),Fdata1(nparmax),Fdata2(npar
max)
external DQDVAL
common /nparanew/ nparanew
do n=1,nparanew
Fdata1(n)=DREAL(Ez_arr(n))
Fdata2(n)=DIMAG(Ez_arr(n))
enddo
Xdata=coord
check=.true.
a11=DQDVAL(x,nparmax,Xdata,Fdata1,Check)
b11=DQDVAL(x,nparmax,Xdata,Fdata2,Check)
YapproxZ=a11+i*b11
END
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
SUBROUTINE E_arrays
!~~~!returns the array** with integrand values at the
lens
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Use arrays
implicit none
integer n, plane,nparanew
complex(8) Exc,Eyc,Ezc
common /nparanew/ nparanew
Ez_arr=(0d0,0d0)
 do n=1,nparanew
Call Ez_field(n,Ezc)
Ez_arr(n)=Ezc
enddo
END
!~~~~~~~~~~~~~~~~~~~~~~~~~
SUBROUTINE Ez_field(n,Ezc)
!~~~!returns the integrands values at points m,n
!~~~~~~~~~~~~~~~~~~~~~~~~~~
Use arrays
Use ipi
```

```
Use Green
implicit none
integer n
real*8 y,x
real*8 k0,sx,sy,sz
real*8 y1,x1
complex(8) Ezc
complex(8) Az,Fy
 common /coord/ y,x
common /wavenumbers/ k0,sx,sy,sz
y1=coord(n)
x1=0d0
Call totalfields(n,y1,x1,Fy,Az)
Call Green_Function (y1,x1)
Ezc=-i*k0*G*Az - Fy*d_dxG
END
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Subroutine totalfields(n,y1,x1,Fy,Az)
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Use ipi
Use arrays
implicit none
integer n,m, plane
real*8 y1,x1,nx,ny
real*8 k0,sx,sy,sz
complex*16 inc_exp
complex*16 R, Hinc, Hscat, Etot, Htot, Fy, Az
common /wavenumbers/ k0,sx,sy,sz
common /plane/ plane
R=cdexp(i*psi_real(n,plane))
nx=-dcos(alpha(plane))
ny=dsin(alpha(plane))
inc_exp=cdexp(-i*k0*(ny*y1+nx*x1))

Etot= Ez0*inc_exp*(1d0+R)
Hinc=-nx*Ez0
 Hscat=nx*Ez0*R
Htot= (Hscat +Hinc )*inc_exp
Fy=Etot/(4d0*pi)
Az=Htot/(4d0*pi)
END
!*********************************************
******************
SUBROUTINE Green_Function (y1,x1)
use ipi
use Green
implicit none
real(8) y1, x1, dx, dy, R
real(8) y, x
real(8) k0,sx,sy,sz
complex(8) H_2
common /coord/ y,x
common /wavenumbers/ k0,sx,sy,sz
dx=x-x1
dy=y-y1
R=dsqrt(dx*dx+dy*dy)
G=-i*pi*H_2(0,k0*R)
d_dxG=i*pi*k0*dx*H_2(1,k0*R)/R
END

 complex*16 function H_2(n,x)
use ipi
```

```
 implicit none
 integer n
 real*8 x,DBSY0,DBSY1,DBSJ0,DBSJ1,a1,a2
external DBSY0,DBSY1,DBSJ0,DBSJ1
if (n.eq.0)then
a1=DBSJ0(x)
a2=DBSY0(x)
else
a1=DBSJ1(x)
a2=DBSY1(x)
endif
H_2=a1-i*a2
 END
BEST GUESS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!
subroutine best_guess
Use DFPORT
use arrays
implicit none
real*8 parent(nparmax,indmax)
real*8 iparent(nchrmax,indmax)
 real*8
parmax(nparmax),parmin(nparmax),pardel(nparmax)
real*8 max_diff,ranval,allowed_diff
integer c,cr,cm,i,j,k
integer npopsiz,nowrite,nposibl(nparmax)
integer nparam,nchrome
common / ga1 / npopsiz,nowrite
common / ga2 / nparam,nchrome
common / ga3 / parent,iparent
common / ga6 / parmax,parmin,pardel,nposibl
DO j = 1, npopsiz
DO i = 1,nparam
max_diff = parmax(i) - in_wires(i)!finds distance from
upper bound
allowed_diff=poss_shift*in_wires(i) !finds max
allowed variance of wirelength changing
if (allowed_diff.lt.max_diff) max_diff=allowed_diff
CALL SYSTEM_CLOCK(c,cr,cm)!calls system clock
to generate seed
!seeds random number generator
ranval=DRANDM(C)!produces a number,ranval (0<=
ranval <1)

parent(i,j) = DABS(in_wires(i) + ranval*max_diff)call
code(j,i,parent,iparent)ENDDO
ENDDO
open(unit=211111, file='best guess.dat')
do k=1, nparam
write(211111,'(2F15.8)') coord(k), parent(k,1)
enddo
close(unit=211111)
END
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!
subroutine optimize_phase
use arrays
implicit none
integer n, nparanew
common /nparanew/ nparanew
```

```
call move_to_zero
call bounds
call average_optimum
call put_into_180_bounds
open (unit=1, file='opt_opt_phase.dat')
Do n=1,nparanew
write(1,'(4F20.8)') coord(n),psi_opt(n,1), psi_opt(n,2),
aver_psi_opt(n)
enddo
close (unit=1)
end
!~~~
subroutine move_to_zero
!~~~
use arrays
implicit none
integer n, nparanew
real*8 shift
common /nparanew/ nparanew
n=nparanew/2
shift=psi_opt(n,2)
psi_opt=psi_opt-shift
open (unit=1, file='first_opt_phase.dat')
Do n=1,nparanew
write(1,'(3F20.8)') coord(n),psi_opt(n,1), psi_opt(n,2)
enddo
close (unit=1)
End
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
!optimizing intial phase diiference of optimal phase
designs
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
subroutine bounds
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
implicit none
integer nparanew
real*8 delta_psi, first_point_45, first_point_other
common /nparanew/ nparanew
first_point_45 = psi_opt(1,2)
first_point_other = psi_opt(1,1)
delta_psi = first_point_45 - first_point_other
psi_opt(:,1) = psi_opt(:,1) + delta_psi
diff=psi_opt(nparanew,2)-psi_opt(nparanew,1)
psi_step=diff/dfloat(N_psi_step)
end
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
subroutine average_optimum
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
implicit none
integer nparanew, m,n, num(1)
real*8 delta_psi(nparmax),aver_psi(nparmax),
psi_45(nparmax), psi_ot(nparmax)
real*8 ksi_to_be_sum(nparmax)
common /nparanew/ nparanew
psi_45=psi_opt(:,2)
psi_ot=psi_opt(:,1)
do m=0, N_psi_step
psi_ot=psi_opt(:,1)+dfloat(m)*psi_step
aver_psi=(psi_ot+psi_45)/2d0
delta_psi= psi_ot - aver_psi
do n=1, nparanew
ksi_to_be_sum(n)=(delta_psi(n)/aver_psi(n))**2d0
enddo
ksi(m)=sum(ksi_to_be_sum)
enddo
num=minloc(ksi)
psi_opt(:,1)=psi_opt(:,1)+dfloat(num(1))*psi_step
aver_psi_opt=(psi_opt(:,1)+psi_45)/2d0
End
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
subroutine put_into_180_bounds
!~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
use arrays
implicit none
integer nparanew,n
real*8 bounds180
common /nparanew/ nparanew
        do n=1, nparanew
psi_opt(n,1)=bounds180(psi_opt(n,1))
psi_opt(n,2)=bounds180(psi_opt(n,2))
aver_psi_opt(n)=bounds180(aver_psi_opt(n))
enddo
end
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
real*8 function bounds180(ph)
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
use ipi
implicit none
real*8 phase0,ph
phase0=ph-2d0*pi*aint((ph+pi)/(2d0*pi))
if (phase0.le.(-pi)) phase0=phase0+2d0*pi
if (phase0.ge.pi) phase0=phase0-2d0*pi
bounds180=phase0
end
!~~~~~~~~~~~~~~~~~~~~~~~~~~~
```