FRACTAL ANTENNA ARRAYS

NSF/AMP Summer Undergraduate Research
Advisors: Dr. Dwight L. Jaggard, Aaron D. Jaggard, Omar Manuar

ABSTRACT

This research involves the overlaps of antenna theory, fractal geometry, and numerical calculations. Its goal is to investigate how random or periodic antenna array geometry can be improved through the use of fractals. Using a program we developed in Matlab, we plotted the radiation of linear/planar antenna arrays. We were able to correctly analyze radiation of three two-dimensional arrays. Another program whose purpose was to generate fractals was used to allow us to characterize the radiation properties of periodic and random arrays.

1. INTRODUCTION

1.1 History

Despite immense advances in communication technology, the ideal antenna has yet to be found. Antenna arrays are used for imaging and communications. A typical array consists of antenna elements placed on an x–y plane in either a periodic or random distribution. These two schemes of organizing antenna arrays prove to have very different radiation properties and, depending on the intended use of the array, each property has its positive and negative aspects. Periodic arrays have very low side lobe levels, but to achieve this, a very large number of elements need to be used. In comparison, random arrays have higher side lobes, but do not need as many elements. Random arrays also have the advantage that they are more robust in the sense that if one element were to fail, the antenna would most likely continue performing as before.

Since the 1980s, fractals have been investigated in physics and engineering. Fractals attempt to bridge the gap between random and periodic configurations. The unique attributes that fractals maintain of both being ordered in their specific arrangement and being random in their construction make them ideal for exploring the radiation properties that result when the ordered and disordered are combined.

Our project aims to present a clearer understanding of fractals, antenna arrays, and how fractals play a part in antenna engineering. We used Matlab as our main tool for generating fractals and calculating the array factor for linear and planar arrays. Using basic array theory, we were able to interpret the resulting "array factor plots," and begin to characterize some specific properties associated with fractal generated arrays. As the long-run goal is to find a fractal array whose radiation is ideal (as explained further below) continuing research will encompass characterizing more fractal traits, and applying them to new fractal arrays and analyzing the resulting array factor plots.

1.2 Fractal Basics

Although fractals are mainly discussed in mathematical forums, they exist in all parts of nature. For example Mandelbrot [1] discusses the basics of fractal theory as applied to the characteristics of a coastline(see Figure 1.). The length of a coastline depends on the size of the measuring yardstick. As the yardstick we use to measure every turn and detail decreases in length, the coastline perimeter increases exponentially. As the view of a coastline is brought closer, we discover that within the coastline there lie miniature bays and peninsulas. As we examine the coastline on a rescaled map, we discover that each of the bays and peninsulas contain sub-bays and sub-peninsulas. There is a self-similar trait observed as we look at the coastline at various resolutions. The number of microscopic structures begin to approach infinity. In fact, because of the immense number of irregularities, the physical length of a coastline is virtually infinite.

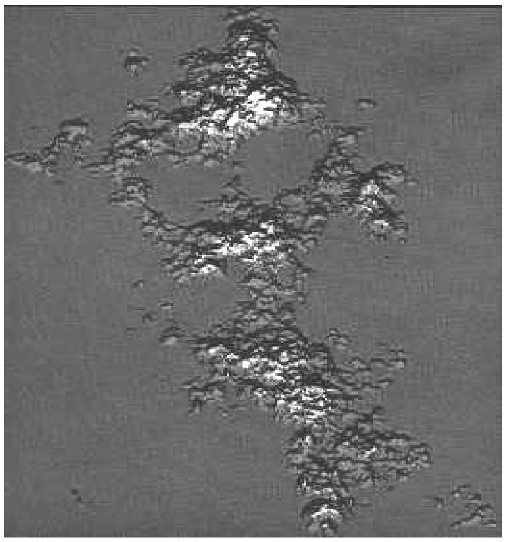


Figure 1: Fractal-generated coastline.

Self similarity (seen in the coast example above) is defined by structures that look the same at variable magnifications. This recurring self-similarity is one of the many attributes of many fractals. Much like the coastline described above, any small part in a self-similar fractal is going to look exactly like the fractal as a whole. Further illustration can be seen in Figure 2, which shows various stages of the well known Sierpinski gasket.

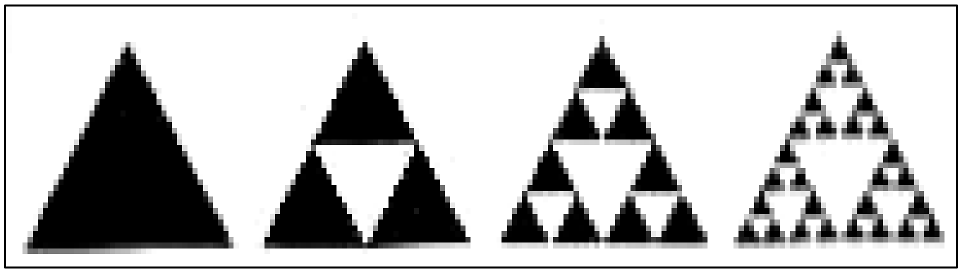


Figure 2: Sierpinski gasket.

The Sierpinski gasket is formed by the same inverted triangle redrawn multiple times at different scales. Every time a new “empty” triangle is formed, the next step of growth places the inverted triangle on all available spots, therefore creating more triangles to fill in the stage of growth.

Another type of figure with a slightly different system of construction uses a generator/initiator relationship. This construction begins by placing an "initiator," which will be the base format for the figure. The initiator is then divided into a collection of lines upon which the generator(s) will be placed. Figure 3 shows the initiator and its first stage of growth where the lines are replaced, or added to, by one of the two generators.

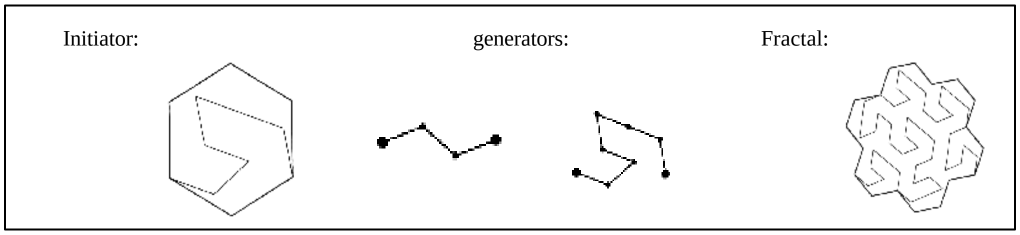


Figure 3: Initiator/generator fractal.

Once the generators replace the lines belonging to the initiator, the generator may repeat "n" number of time, or a different generator may begin growth upon the one already in place. As the fractal grows, it can be noted that the orientation of some figures are arbitrarily chosen.

Another form of fractal is formed by tremas, a term coined by Mandelbrot and derived from the Latin word meaning "hole" (related to "termite"). In some cases, tremas behave like gaps within the fractal. For first-order tremas, the "mass" (of the fractal) is taken out of the figure but redistributed to the outer thirds. An example of a trema-generated fractal is depicted in Figure 4.

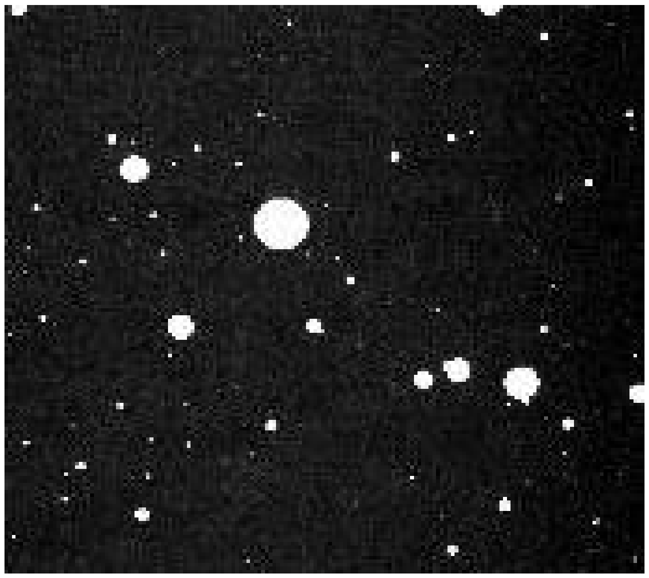


Figure 4: Trema-generated fractal. Tremas (in this case) that were randomly removed were white circular discs.

Although the fractals discussed would theoretically grow to infinity, because of human limitations in fabrication and display, the fractals seen earlier are all “pre-fractals, they are not infinitely grown. References to fractals throughout this research are thus actually to prefractals.

2. PROBLEM STATEMENT AND SOLUTION

2. 1 Linear Arrays

In constructing linear and planar arrays the radiation properties of distinct patterns must be analyzed in order to optimize the array for certain uses. The first example of antenna arrays we looked at were linear arrays upon which we placed “n” number of elements. We used Matlab to calculate the constructive and destructive interference in terms of the array factor (AF), that characterizes the radiated field.

$$AF = \sum_{n=0}^{N-1} I_o e^{-i2\pi N\frac{d}{\lambda}\cos\theta + \alpha} \qquad (1)$$

Here, N is the number of elements in the X-axis, I_N is the charge per element, d/lambda is the ratio of the distance between each element in wavelengths, and alpha is a phase shift which changes the direction of the main beam. To simplify calculations, we kept the charge equal for all elements. We also chose to keep alpha equal to zero due to the effects that phase has on the main beam. Matlab used a “while” loop function to calculate the sum of the array factor for n elements a distance dB apart with uniform voltage. As the elements were placed within 1 wavelength, there was constructive interference such that there was always one main lobe with smaller side lobes as more elements were added. Figure 5 shows the main lobe with high side lobes, which were normalized by taking the 20*$\log_{10}$ of the array factor. This scaling allowed for better analysis of the side lobes and their characteristics.

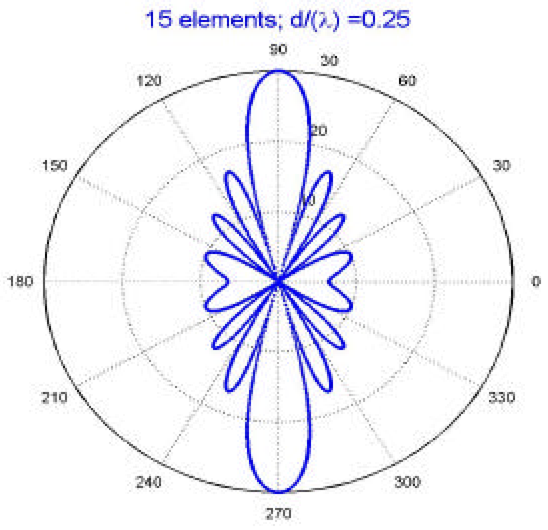


Figure 5: Main lobe with high side beams. 15 elements were placed an equal distance dB apart.

2.2 Qualities of an Optimized Array

An optimized antenna would have no side lobes. The benefit of having only one beam is apparent when we consider airports. As seen in Figure 6A, if the same array in Figure 5 were used in an airport, the side lobes would easily cause air traffic control to confuse a large airplane at the height of the side lobes with a small plane at the peak of the main beam. Another characteristic of an optimized beam involves a thin single main beam. Figure 6B shows the catastrophe that could happen if a thick main beam were to confuse two airplanes as one large plane.

$$AF = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} e^{-i\,2\partial \times M(\frac{d}{l})\cos \grave{e}\, \sin j} \; e^{-i2\partial \times N(\frac{d}{l})\cos \grave{e}\, \cos j} \qquad (2)$$

2.3 Planar Arrays

Two-dimensional antenna arrays allow for more flexibility and variety in placements of array elements. As discussed earlier, two main ways in which elements have been placed on planar arrays are to place them on a grid or to randomly scattering them about a certain area. Although both still cause side lobes to exist, both arrangements have their advantages. Once again, Matlab used a while loop function to calculate the sum for the array factor.

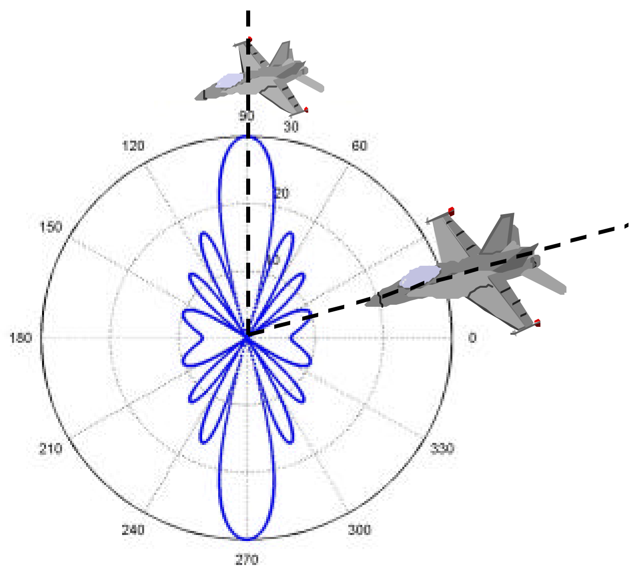


Figure 6A: Array Factor with high side lobes.

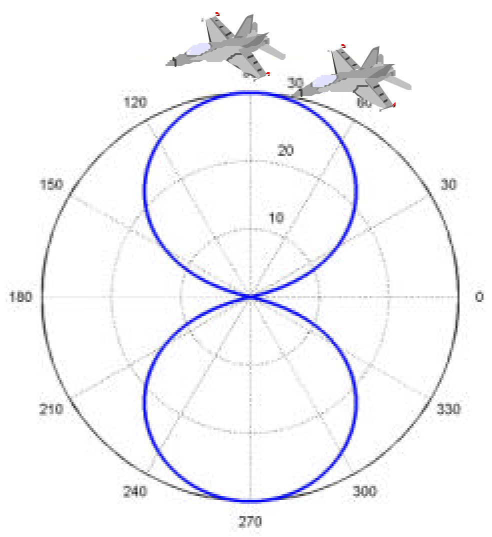


Figure 6B: Array Factor with thick main beam.

To simplify operations we let $f_x = \frac{x}{l\ r}$ and $f_y = \frac{y}{\boldsymbol{l}\ r}$ so that (AF) becomes:

$$AF = \sum_{n=0}^{N-1} e^{-i2p\,(f_x x + f_y y)} \qquad (3)$$

Figure 7 illustrates the problem geometry where we examine the array factor projected on the Z-axis. Here, R is the observer's distance from the origin. Matlab also used "while" loops to generate the points on a 2x2 matrix which was then calculated to plot the array factor on a three dimensional plot.

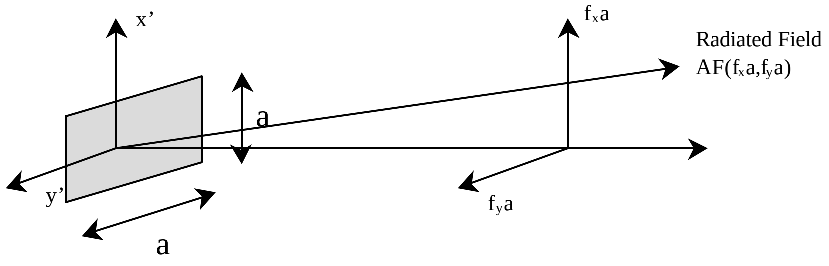
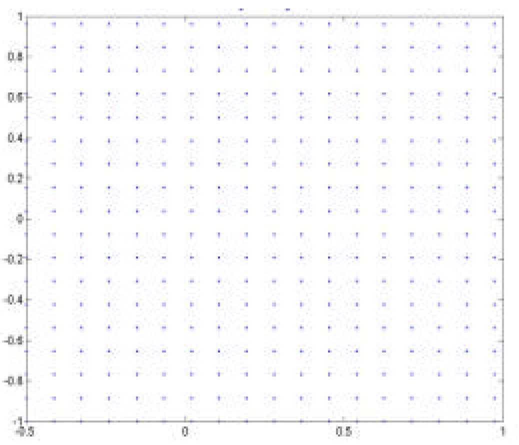


Figure 7: The geometry of the problem, AF is measured in the far-zone.

2.3.1 Periodic Arrays

Planar arrays whose elements are organized in a grid have tendencies to produce main beams and side lobes of the same height. In Figure 8A, 324 elements have been placed on a rectangular grid of 1.5x2 square units. These dimensions are chosen for scaling purposes, as our fractal array(discussed later) uses this size window. Figure 8B shows the array factor in a gray-scaled colormap where black is the lowest point and white is the highest point.



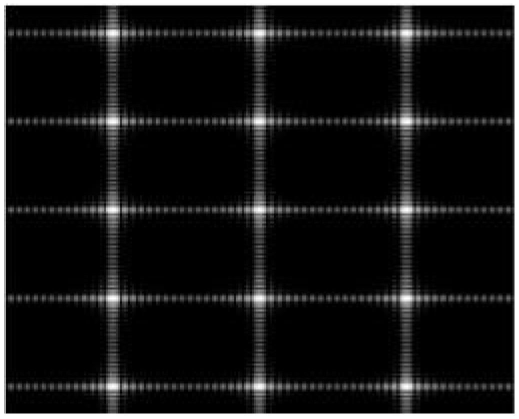


Figure 8A: 324 elements of with equal x distances apart and equal ydistances.

Figure 8B: Radiated field where white highest point, and black is lowest.

2.3.2 Random Arrays

Random planar arrays have radiation characteristics that may be more desirable. We plotted 324 elements on the same rectangular area of 1.5x2 square units. The points were generated using Matlab's random number generator, which produces a set of numbers between 0 and 1(see figure 9A). Certain manipulations were performed to change the points' orientations to fill the given rectangular area. Figure 9B shows how side lobes are generally lower than the ones seen in the first set of ordered points. There is also a 180-degree symmetry, which is more apparent about the main beam.

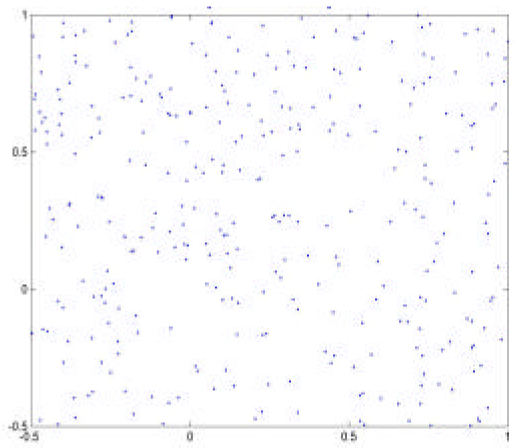


Figure 9A: 324 randomly placed elements.

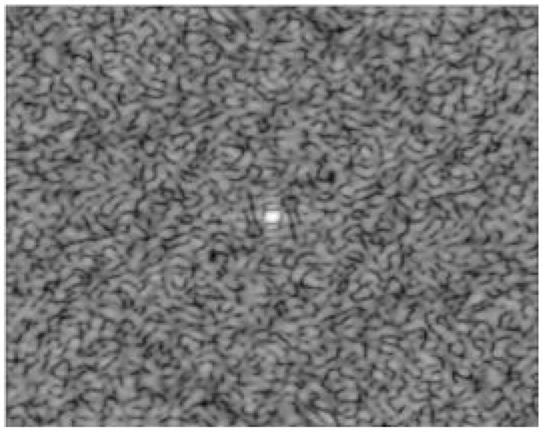


Figure 9B: Radiated field where white is highest point, and black is lowest.

As the goal of our work is to characterize an optimized antenna array, we propose that the radiation properties of both the ordered fractals and the random fractals have redeeming qualities. Yet, there continues to be a gap between the ordered and the disordered. Fractals attempt to bridge that gap where they are constructed in an organized pattern, yet they also attain the qualities of random points depending on their construction and orientation of points.

3. RESULTS AND CONCLUSIONS

3.1 Fractal Generation

To first generate the fractals, we used an Iterated Function System(IFS) in the Matlab script that used probability to randomly place points within initial matrix boundaries. These initial matrix boundaries are specific to the Sierpinski gasket. Using Matlab's random number generator to create the fractal aids in filling the ordered vs. disordered gap and effectively combining the two opposing radiation properties. The Matlab script independently chose to plot the Sierpinski gasket between - 0.5 and 1 on the x-axis, and between - 1 and 1 on the y-axis. For this reason, we adjusted the other two Matlab scripts to place the antenna elements within the same axes. For the sake of comparison and scaling, we also input the gasket to comprise only 324 points. Figure 10 shows the organization of the points and the characteristic radiation properties of this particular organization.

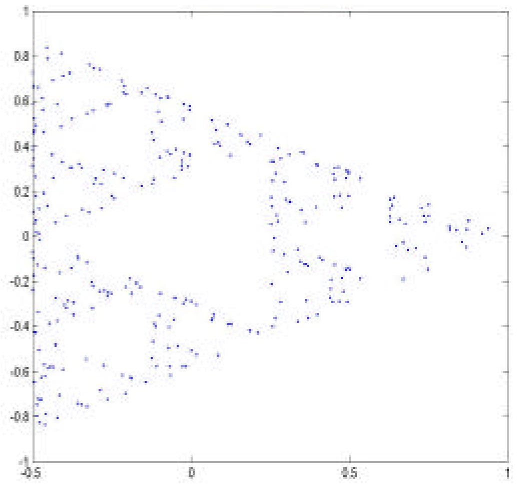


Figure 10: Random-point-generated Sierpinski gasket.

Through many runs, the organization shown in Figure 10 appears to be the average case. The program sometimes had the tendency to plot most of the points around the perimeter. Figure 10 has a good balance in terms of how well distributed the points are on the fractal. Another property of the fractals generated by this program is that various stages of growth in the Sierpinski gasket are apparent. It is possible to begin to see third and fourth stages of growth, though the first stage is not fully completed.

3.2 Fractal-Array Radiation Characteristic

Figure 11 displays the radiated field of the fractal shown above. Array theory explains the lines that can be seen extending from the middle point. For every “line” formed by the antenna elements, the radiation will have higher side lobes on a line perpendicular to the respective lines formed. We observed this property earlier when we plotted the array factor for the points that were in a grid(Figure 8B). Because the elements formed lines in both the x and y axes, there were lines of higher side lobes that overlapped, producing the grid-like organization of side lobes with equal heights. This is also why the random arrays are so effective in lowering the side lobes. Because the random placing of the elements on the array does not allow for “lines” to be formed, no perpendicular side lobe line is formed.

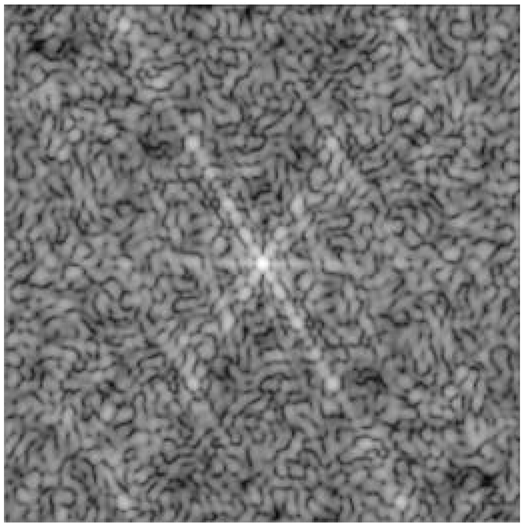


Figure 11: Radiation from random-point-generated Sierpinski gasket.

3.3 Optimized Array Comparison

3.3.1 Main Beam Comparison

The qualities of an optimized antenna array deals mainly with the quality of the main beam, and the side lobe levels (variation in N and M). As seen in the periodic array, the main beam was good in that there was no interference from smaller side lobes coming from the side. This main beam degradation can be seen in the random array where the main beam no longer maintains its point-like appearance on our graphs. The main beam characteristics of the periodic array were imitated by the fractal array.

3.3.2 Side Lobe Comparison

Side lobe levels for periodic elements are higher yet farther apart because of large amounts of elements. The random array achieves lower side lobes with significantly less elements. Like the random array, the fractal had lower side lobes at lower stages of growth. Figures12A and 12B compares the side lobe levels of the random arrays with those of fractal array.

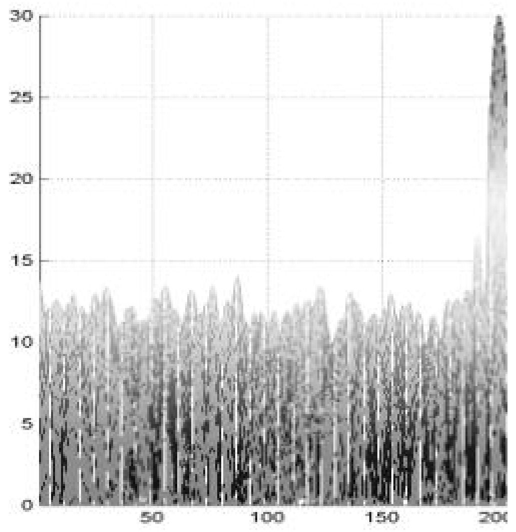


Figure 12A: Side view of array factor plot for random array.

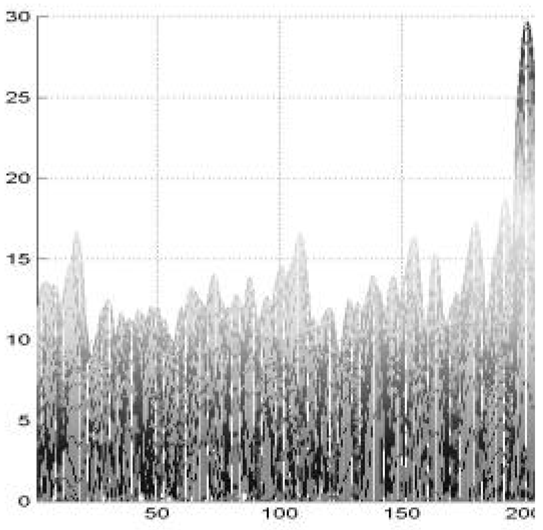


Figure 12B: Side view of array factor plot for fractal array.

It can be observed that the random array performed better than the fractal array. According to previous work, there is an intersection point where the number of elements in a fractal array and random array cause for the one to be more effective as the other. As the array holds more elements, the random arrays perform better. Yet before this intersection point we have described, certain fractal arrays perform much better. Aside form the side lobe levels, the fractal did have overall a better main beam, regardless of the number of points. A draw back for random arrays with many elements at that, as the number of elements increases, main beam degradation is quite significant. The main beam completely abandons its point-like form.

4. RECOMENDATIONS

Because of time-interests, we were only able to compare the three arrays using 324 points. Continuing work must involve testing the fractals with a larger selection of points to which should confirm the effect that the number of elements have on random and fractal arrays. Further work must continue to be done. This project characterized the radiation properties of only fractal, while there is an infinite number of fractals that can be generated and analyzed.

5. ACKNOWLEDGMENTS

I would like to thank the National Science Foundation's AMP and SUNFEST programs for all their resources and support. I would especially like to thank Dr. Jaggard for his inspiration and guidance. I would also like to thank Aaron Jaggard and Omar Manuar for their help. I would like to thank everyone mentioned for allowing me to partake of this opportunity in undergraduate research.

6. REFERENCES

1. Mandelbrot, B.B . *The Fractal Geometry of Nature*. W. H. Freeman and Company, New York. 1983.
2. Weeks. W. L. *Antenna Engineering*. McGraw Hill. London. 1968.
3. Jaggard, D.L. *Fractals in Engineering*. From *Theory to Industrial Applications.* Springer.
4. Jaggard, D.L., Jaggard A.D., *Cantor Ring Arrays*. In *Microwave and Optical Technology Letters*. Vol. 19 No. 2 October 5 1998
5. Jaggard, D.L., Jaggard A.D., Frangos V. P., *Fractal Electrodynamics Surfaces and Superlattices*. Jan. 1999
6. Jaggard, D.L., Jaggard A.D., *Cantor Ring Diffractals*. *In Optics Communications*. Elsevier. October 1998.
7. Ulaby, Fawwaz. *Fundamentals of Applied Electromagnetics*. Prentice Hall, New Jersey. 1999.
8. Lee, S.W. *Antenna Handbook, Theory, Applications, & Design.* Reinhold Company. New York. 1998.
9. Stutzman, W.L. *Antenna Theory & Design*. Wert company. Pennsylvania. 1995.
10. Kirk, Donald. *Contemporary Linear Systems: Using Matlab 4.0*. PWS publishing. Massachusetts. 1996

7. APPENDIXES

Appendix A (Matlab script to calculate the array factor for a linear array)

```
function r2pattern(n,d)
% program that calculates the radiation pattern with the
values input by the user
% n is the number of points
% d is the space between each
a = 0;
x = 0:0.001:(2*pi);
m = 0;
r = 0;
A = 0;
while m < n;
   r = exp(i*m*2*pi*d*cos (x) + a);
   A = (r) + A;
   m = m + 1;
end
f = abs(A/n);
F = 20*log10(f);
q=1;
l = 30;
while q <= 6284;
   if F(q) > -l;
      F(q)= F(q)+l;
   else F(q) = 0;
   end;
   q=q+1;
end
polar(x,(F),'.');
title([num2str(n),' elements; d/(\lambda)
=',num2str(d),],'Color','blue','FontSize',16);
```

Appendix B (Matlab script to calculate the array factor for an ordered planar array)

```
%Array Factor for points on rectangular plane from x[-.5 1]
and y[-1 1].

numberofpoints = 325;
x = -.5:(1.5)/(sqrt(numberofpoints)):1;
y = -1:(2)/sqrt(numberofpoints):1;
x = x';
y = y';
s = 1;
n = 0;
R = 0;
Q = -size(x,1);
W = size(x,1);
A = 0;
P(size(x,1),2) = 0;
while Q < numberofpoints
   R = R + 1;
   A = A + 1;
   n = 0;
   while n < W;
      P(s,1) = x(R);
      P(s,2) = y(n+1);
      n = n + 1;
      s = s + 1;
   end
   Q = Q + size(x,1);
end


figure(3);
[fx,fy] = meshgrid(-20:.1:20, -20:.1:20);
m = 0;
A = 0;
N = size(P,1);
s = 1;
while m < N;
   a = exp(-i*2*pi*(fx*P(m+1,1)+fy*P(m+1,2)));
   plot(P(m+1,1),P(m+1,2),'.');
   hold on;
   A = A + a;
   m = m + 1;
   s = s + 1;
end
title(['[',num2str(P(1)),'
',num2str(P(N+1)),']'],'Color','blue','FontSize',14);
hold off
```

```
f = abs(A/N);

o=1;
 while o <= (size(f,1)*size(f,2))
   f(o) = f(o) + .00001;
   o=o+1;
end


F = 20*log10(f);

q=1;
l = 40;
 while q <= (size(F,1)*size(F,2))
   if F(q) > -l;
      F(q)= F(q)+l;
   else F(q) = 0;
   end;
   q=q+1;
end

figure(1);
pcolor(F)
shading interp
colormap(jet)
axis xy

title(['[',num2str(P(1)),'
',num2str(P(N+1)),']'],'Color','blue','FontSize',14);


figure(2);
mesh(F)
shading interp
colormap(jet)
axis xy
view([90 0])
```

Appendix C (Matlab script to calculate the array factor our Sierpinski planar array)

```
% Array factor formed by the Sierpinski gasket

t = exp(2*pi*(0:2)/3*i);
p3 = [real(t); imag(t)];
x = 4*rand(2,1)-2;
last = x;
N = 324;
for k = 2:N
   t = rand(1,1);
   if t <= 1/3;
      p = p3(:,1);
   elseif t <= 2/3
      p = p3(:,2);
   else
      p = p3(:,3);
   end
   last = 0.5*(last+p);
   x = [x last];
end

X = x';
[fx,fy] = meshgrid(-20:.1:20, -20:0.1:20);
A = 0;
N = size(X,1);
m = 0;
s = 1;
figure(3);
while m < N-10
   a = exp(-i*2*pi*(fx*X(s+10,1)+fy*X(s+10,2)));
   figure(3);
   plot(X(s+10,1),X(s+10,2),'.');
   hold on;
   A = A + a;
   s = s + 1;
   m = m + 1;
end
hold off
title(['[',num2str(X(1,1)),num2str(X(1,2)),'],
',num2str(N),'cycles'],'Color','blue','FontSize',14);

f = abs(A/N);

o=1;
 while o <= (size(f,1)*size(f,2));
   f(o) = f(o) + .00001;
   o=o+1;
```

```
end

F = 20*log10(f);

q=1;
l = 30;
 while q <= (size(F,1)*size(F,2));
   if F(q) > -l;
      F(q)= F(q)+l;
   else F(q) = 0;
   end;
   q=q+1;
end

figure(1);
pcolor(F)
shading interp
colormap(jet)
axis xy

title(['[',num2str(P(1)),'
',num2str(P(N+1)),']'],'Color','blue','FontSize',14);

figure(4);
mesh(F)
shading interp
colormap(jet)
axis xy
xlim([1 401])
ylim([1 401])
zlim([0 30])
view([90 0])
```

Appendix D (Matlab script to calculate the array factor our random planar array)

```
%Array factor from 324 randomly chosen points

N = 324;
P = rand(N,2);
P = 1.5*P;
P = P - 0.5;

figure(1);
[fx,fy] = meshgrid(-20:.1:20, -20:.1:20);
m = 0;
A = 0;
s = 1;
while m < N;
   a = exp(-i*2*pi*(fx*P(s,1)+fy*P(s,2)));
   plot(P(s,1),P(s,2),'.');
   hold on;
   A = A + a;
   m = m + 1;
   s = s + 1;
end
title(['[',num2str(P(1)),'
',num2str(P(N+1)),']'],'Color','blue','FontSize',14);
hold off

f = abs(A/N);

o=1;
 while o <= (size(f,1)*size(f,2))
   f(o) = f(o) + .00001;
   o=o+1;
end

F = 20*log10(f);

q=1;
l = 30;
 while q <= (size(F,1)*size(F,2))
   if F(q) > -l;
      F(q)= F(q)+l;
    else F(q) = 0;
   end;
   q=q+1;
end

%figure(2);
%pcolor(F)
```

```
%shading interp
%colormap(jet)
%axis xy
%xlim([1 401])
%ylim([1 401])

title(['[',num2str(P(1)),'
',num2str(P(N+1)),']'],'Color','blue','FontSize',14);

figure(2);
mesh(F)
shading interp
colormap(jet)
axis xy
xlim([1 401])
ylim([1 401])
zlim([0 30])
view([90 0])

figure(4);
mesh(F)
shading interp
colormap(jet)
axis xy
xlim([1 401])
ylim([1 401])
zlim([0 30])
```