Understanding the Hansen-Woodyard End-Fire Array and Modern Optimization Solutions

EECE: 7275 Antennas and Radiation

Kristtiya V. Guerra

Abstract:

This paper compares the Hansen-Woodyard (HW) end-fire array with an ordinary end-fire array. The Hansen-Woodyard is an optimized end-fire array that was designed to provide higher directivity without losing characteristics from an Ordinary end-fire array. This paper analyzes the design differences and outputs of the two antenna array designs in Matlab. It also tries to further improve the antenna design using gradient descent. While the Hansen-Woodyard array has a higher directivity, it also has a backlobe and a wider beamwidth for the first side lobe. The Hansen-Woodyard is not the most optimized design out there, but it is a better alternative in comparison to the ordinary end-fire in the case where the design requires a large number of elements. The paper also attempts to further optimize the Hansen-Woodyard array through the use of the gradient-descent algorithm. The implementation attempted by the study was able to reduce the backlobe but does not retain key characteristics of an end-fire array.

Background

An antenna array is a system of antennas that work simultaneously and behave as a single device. The geometric organization of antennas opens the door for an infinite number of possibilities for pattern solutions that can cater to different system needs. Like any solution, there will exist costs depending on the approach, such as an increased number of minor lobes, lower directivity, and more. The transmitting antenna elements in the array will have constructive and destructive properties that allow for higher directivity in desired locations and lower gain in undesired directions. The Hansen-Woodyard end-fire array is an arrangement that aims to increase the directivity of the ordinary end-fire array.

The Array Factor

The array factor is a model used for linear antenna arrays. The formula for the normalized array factor is given by:

$$(AF)_n \approx \frac{1}{N} \frac{\sin(\frac{N}{2}\Psi)}{\sin(\frac{1}{2})\Psi} \tag{1}$$

Under the assumption that every element in the array is identical, the gain equation for a dipole can be multiplied by the array factor to produce a pattern for the array. The array factor allows the designer to modify the positioning, amplitude, and phase of the antennas in the array through the factor Ψ, where:

$$\Psi = kd\cos\theta + \beta \tag{2}$$

This gain can be normalized to give us the antenna pattern for the single element and can be multiplied by the array factor to give us the pattern for the whole array.

Ordinary End-Fire Array

An ordinary end-fire array consists of identical antennas that are equally spaced such that the main lopes are pointed at $\theta = 0$ and $\theta = 180$. In order to point the array towards that range, the phase shift, β, should be set to $-kd$ and kd respectively.

Spacing of Elements

An ordinary end-fire array will point in a certain direction based off of β in the the array factor, which is influenced by the distance between the elements in the array. As seen in the plot shown in Figure 1, the spacing is what determines the overall directivity of the system. Starting at a spacing of $\lambda/2$, we can see that the directivity is bidirectional, pointing at 0 and 180 degrees. Decreasing the distance between the elements, the array factor points in the desired direction of 180 degrees. The cost, however, is the directivity has now decreased, meaning the main lobe is wider and opens the system to receiving more disruption from outside noise.

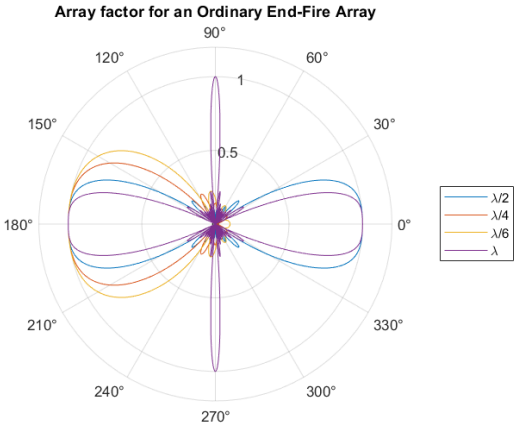


Figure 1: *Polar plot of array factor variations based on changes in the distance between elements.*

We can also see that bringing the elements even closer does not incentivize the designer, the directivity is decreased, and the minor lobes are wider.

The Hansen-Woodyard End-Fire Array

The Hansen-Woodyard end-fire array is an antenna geometry that results in a pattern whose main lobe is directed at $\theta = 0$ or 180 (just as an ordinary end-fire array) however, with an improved directivity. This is done by introducing a progressive phase shift and positioning the elements closely.

The system parameters of a Hansen-Woodyard antenna array are highly dependent on elements in the array. The cost of using a Hansen-Woodyard array is that while there is an increase in directivity, there is a loss in gain. This is evident in Figure 6, as the number of elements increases, the beamwidth of the major beam decreases but the gain decreases. Hansen and Woodyard suggest that a 1.48 times increase in current will bring the gain of the array to be equivalent to that of an ordinary end-fire array.

Progressive Phase Shift

The Hansen-Woodyard end-fire array introduces a progressive phase shift that is dependent on the number of elements in the array.

$$\beta = -(kd\cos(\theta) + \frac{\pi}{N}) \quad \text{for} \quad \theta = 0 \tag{3}$$

$$\beta = (kd\cos(\theta) + \frac{\pi}{N}) \quad \text{for} \quad \theta = 180 \tag{4}$$

with N being the number of elements in the array.

Spacing of Elements

The distance between the elements in the array is dependent on the number of elements:

$$d = (\frac{N-1}{N})\frac{\lambda}{4} \tag{5}$$

Figure 2 shows the array factor produced for various N-element arrays when using the ideal element distance calculated using 5

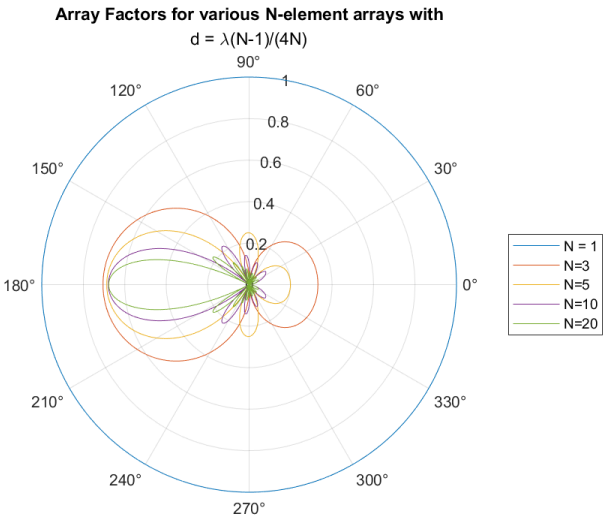


Figure 2: *Array Factor of for N-element Hansen-Woodyard Arrays using the proposed distances between the elements shown in Equation 5.*

When using Equation 5 to approximate desired distances between elements, we see that for the smaller values of N (1,3 and 6), the models have minor backlobes, while the large N elements lack a backlobe which brings us to the proposed approximation for large N element arrays.

For large N arrays, the distance between elements approximates to $\lambda/4$. Since the Hansen-Woodyard end-fire specifications are designed for large arrays, Equation 5 is not necessary to use, but it exists. We can see the approximation in action by seeing how the array factor pattern changes as a result of an increase in the number of elements in the array.

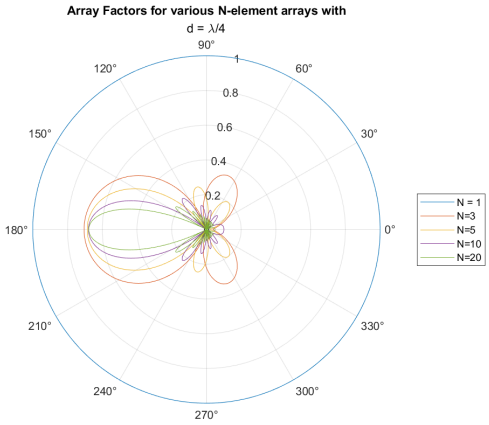


Figure 3: *Array Factor for N-element Hansen-Woodyard Arrays using the $\lambda/4$ approximation for large number N.*

Figure 3 shows us that this approximation does not work for any N. For small N = 1, 3, and 5, the array factors lack the signature opposite facing backlobe present in a typical Hansen-Woodyard end-fire array. This makes it apparent that this approximation only applies to large N values.

In addition to varying the distance between the elements, the phase between the antennas is adjusted depending on the number of elements. Looking at Figure 4, the two models are nearly identical except for the lack of the backlobe. Although the lack of backlobe may seem desirable, looking closer we can see that the pattern for the array that uses Equation 5 has a wider major lobe so there is the trade-off, and is up to the designer.

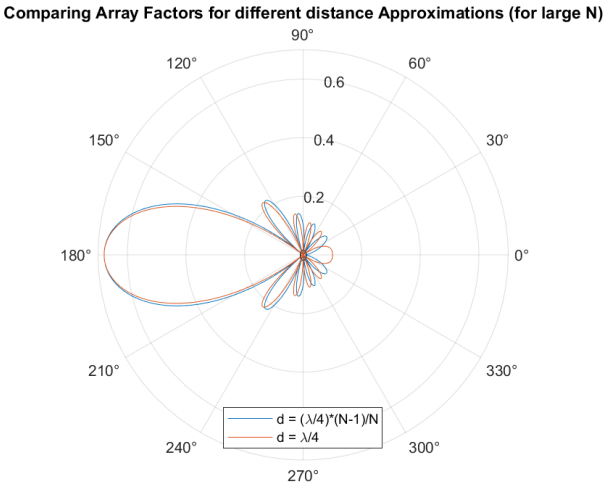


Figure 4: *Comparison of the two proposed approximations for the optimal distance between elements in a large N-element array.*

Larger distances between the antenna will result in a significantly larger backlobe. Distances smaller than $\frac{\lambda}{4}$ remove the minor backlobe but also decreases the directivity of the antenna system. This is evident with Figure 5.

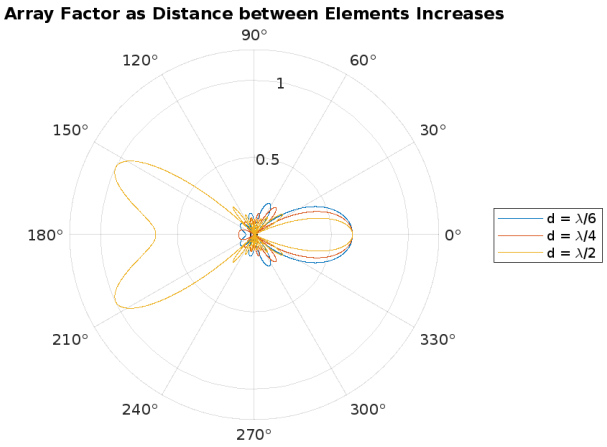


Figure 5: *Polar plot that shows how the distance between the antenna in an array affects the array factor in an end-fire antenna array.*

Further Optimization of the Hansen Woodyard Array

Thanks to modern computers, optimization can be approached through many trials following existing optimization algorithms. For example, Koziel and Ogurtsov propose a gradient-based optimization approach that reduces the gain of the first side lobe. [5] The drawback of their approach is remaining lobes have increased gain.

Gradient-Based Optimization

Gradient descent is an iterative algorithm that can be used to find the inputs that lead to the minimum of a function. Gradient descent finds the minimum of a function by adjusting selected parameters. When applied properly, it can be used to optimize an antenna design. In our case, it can be used to try to reduce values such as beamwidth or gain of select lobes.
The basic formula that defines this algorithm is:

$$x_{n+1} = x_n - \gamma \nabla f(x_n) \tag{6}$$

Where x_n is the current input, γ is the learning rate, and x_{n+1} is the next input value. The learning rate is a value that is selected by the user and is the equivalent to a step size for how far the subsequent guesses (x_{n+1}) should be. The gradient is a vector operation that finds the range of change of an oftentimes multi-variable function. The gradient operator is represented by the *del* symbol, ∇. For example, if we wanted to find the gradient of a three-dimensional function in a rectangular coordinate system would be:

$$\nabla f(x, y, x) = \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y} + \frac{\partial f}{\partial z}$$

This algorithm uses the gradient such that with each iteration the sequence will eventually converge to a center point in the gradient where there is no change.

Methods

For this paper, the following assumptions and specifications will be used for consistency throughout the simulations:

- Each element in the arrays will be half-wave dipoles
- The formula for gain of a half-wave dipole defined in Electromagnetic waves [2].

$$G(\theta, \phi) = 1.64 \frac{\cos^2(\frac{\pi}{2}\cos\theta)}{\sin^2\theta} \tag{7}$$

- Frequency: 2 GHz
- Speed of Light, c : $3 \times 10^8 m/s$

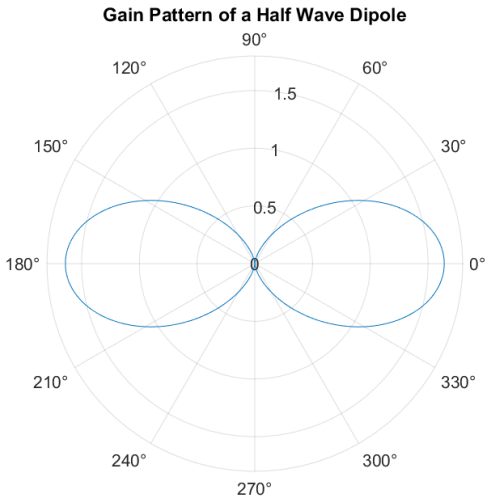


Figure 6: *Polar plot of the gain of halfwave dipole antenna.*

Using Matlab, the Hansen-Woodyard end-fire array and the ordinary end-fire array will be analyzed and compared through plots.

A gradient-descent-based implementation will then be applied to the Hansen-Woodyard end-fire array in an attempt to optimize the design. The goal was to reduce the magnitude of the backlobe by adjusting the distance between elements and the progressive phase shift. What complicates the process is the number of dimensions that are being used. For that reason, Matlab was used as it handles multivariable calculus and matrix math well and has a lot of built-in functions that make the implementation of the algorithm simple in comparison to Python. Equation 6 can be directly implemented into code.

```
x_n(i+1,1) = x_n(i,1) - gamma*grad_x(x_n
    (i,1),x_n(i,2)); %distance
x_n(i+1,2) = x_n(i,2) - gamma*grad_y(x_n
    (i,1),x_n(i,2)); %phase shift
x_n(i+1,3) = AF_backlobe(x_n(i+1,1),x_n(
    i+1,2)); % Backlobe
```

Figure 7: *Two variable implementation of the gradient descent algorithm.*

The GitHub repository containing figures, relevant paper, and codes can be found at: https://github.com/Kristtiya/HansenWoodyardEndFireArray

Results:

Ordinary vs Hansen Woodyard

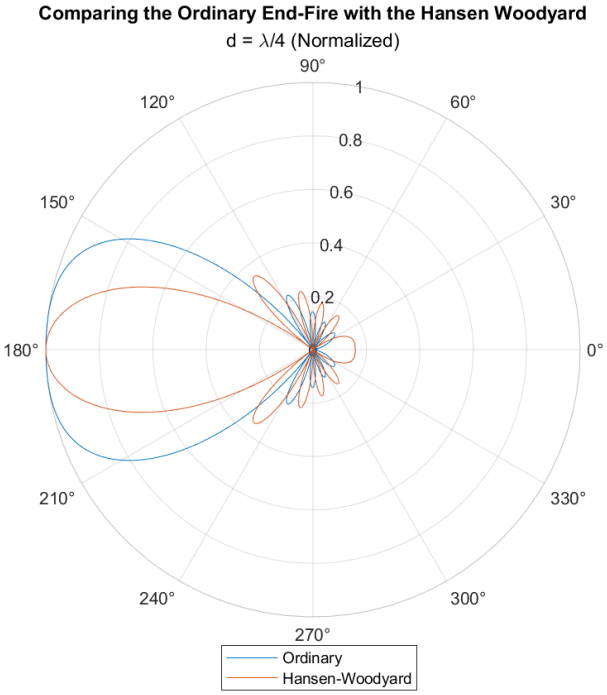


Figure 8: *Comparison of normalized array pattern for a Hansen-Woodyard array and Ordinary end-fire array*

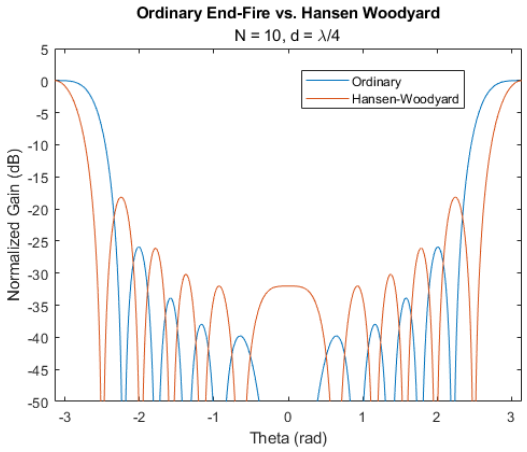


Figure 9: *Beamwidth of the Hansen-Woodyard end-fire Array compared to the beamwidth of the Ordinary end-fire array.*

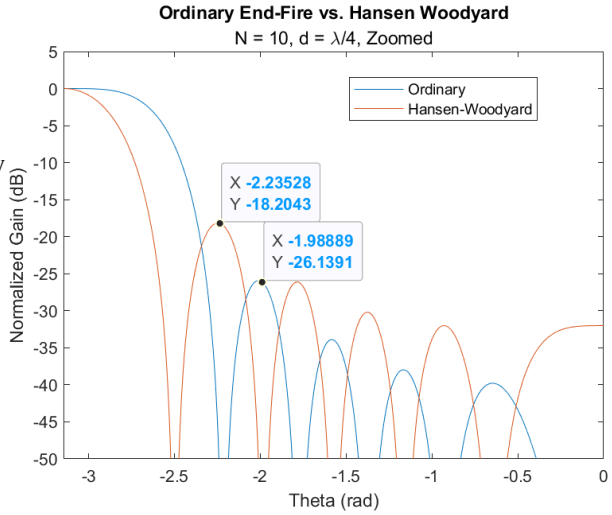


Figure 10: *Zoomed section of the beamwidth comparison of the Hansen-WOodyard array and the ordinary end-fire array.*

Optimization of the Hansen-Woodyard Array

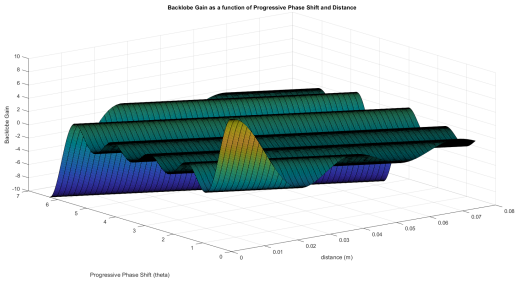


Figure 11: *Mesh grid of the backlobe for an antenna with a progressive phase shift.*

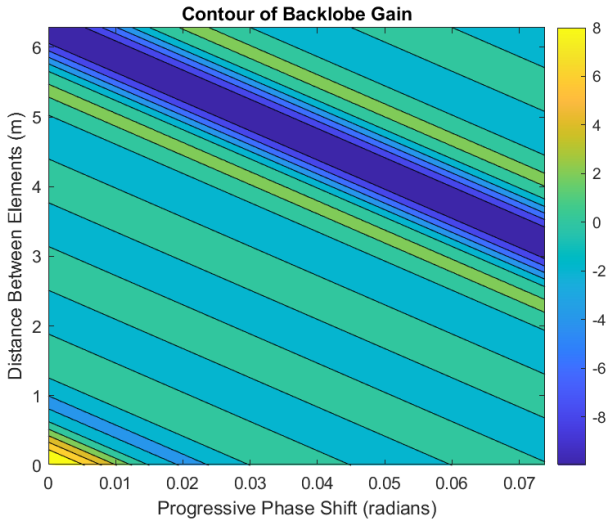


Figure 12: *Contour of the backlobe as a function of distance between elements and adjustments to progressive phase shift.*

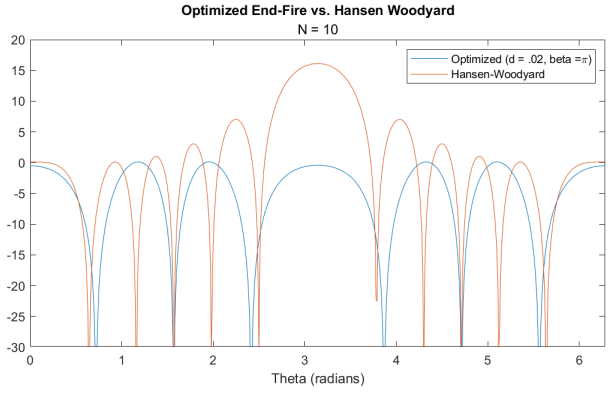


Figure 13: *Comparison between optimized array and Hansen-Woodyard Array. (Optimized parameters: d=0.02m, phase shift = π)*

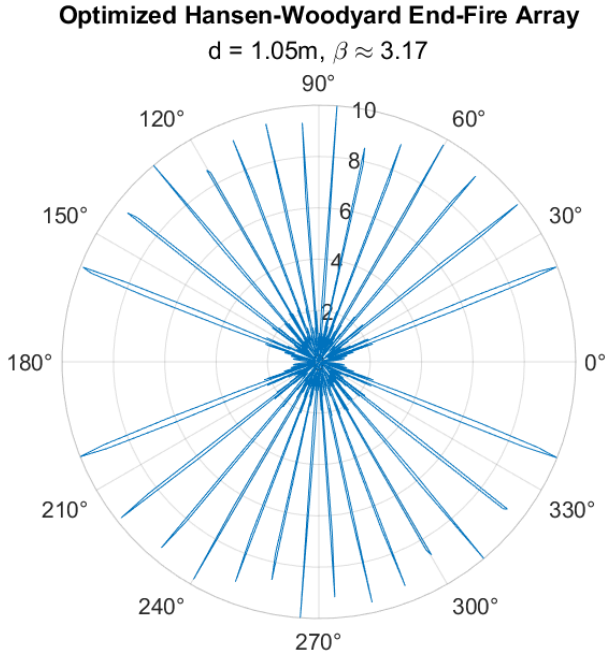


Figure 14: *Array pattern of design parameters proposed by the algorithm. (distance = 1.05m, phase shift = 3.17 radians).*

Discussion:

By overlaying the plots, such as in Figure 8, of the Hansen-Woodyard array and the Ordinary array the two designs can be compared to at a surface level. Figure 9, we can see the Hansen-Woodyard end-fire array has a higher directivity than the ordinary end-fire. In the 10-element array scenario, we can see that the Hansen-Woodyard array's beamwidth was 32 degrees less than that of the ordinary end-fire. This is a significant improvement.

The two costs to the design of the Hansen-Woodyard end-fire design are visible in Figure 10, where we can see that the first side lobe has about an 8dB greater gain and a wider beamwidth. There is also a backlobe that is not present in the ordinary end-fire design, just as discussed in the background.

Knowing these absolute costs of using the Hansen-Woodyard end-fire design, the gradient descent algorithm could be used to try to minimize the cost of such a design. Figure 11 is a mesh grid that shows the backlobe of the array as the progressive phase shit and distance between elements were adjusted. Figure 12 is a flattened version of this plot that gives a better visual. The goal of the gradient descent is to find the minimum, but there is a line of minimums. Initially, my learning rate step size was too small such that the output parameters reduced the backlobe by a bit but also decreased the front lobe to match the backlobe.

Another problem that arose due to the system not being constrained enough is the increase in side lobes and loss of end-fire characteristics. This is seen with Figure 14, where the backlobe is minimized to 0, but there now lacks a main lobe and the side lobes radiate to maximum magnitude. After trial and error, it was decided that it was no longer efficient to keep guessing what the ideal learning rate should be.

Conclusion:

The Hansen-Woodyard array is a modified, optimization solution of the ordinary end-fire array. The design specifications are simple enough such that the main considerations needed are the amount of elements that will be used. The specifications are designed for infinitely large arrays, but based on the simulations, even just 10 elements is enough to experience the higher directivity of a Hansen-Woodyard array compared to the Ordinary end-fire.

The cost of using the Hansen-Woodyard design is that the first side lobes are larger than that of an ordinary end-fire using the same number of elements and distance between elements. In addition, it also has a

backlobe. It is up to the designer if these cons are worth it when implementing the design.
The Hansen-Woodyard specifications do not give a perfect solution. There is still room to better optimize the array while following the process used to find the original Hansen-Woodyard specification. This is possible with modern computers and the gradient descent optimization algorithm. This algorithm can be used to target and minimize certain characteristics of a system that can be modeled mathematically. The attempt made in this paper, while unsuccessfully at finding a better design than the Hansen-Woodyard, shows the possibility of the gradient descent algorithm in finding optimal designs. It was able to successfully reduce the magnitude of the backlobe of an array.

References:

1. C. A. Balanis,"Arrays: Linear, Planar, and Circular," in Antenna Theory Analysis and Design, Ed., Hoboken, New Jersey, United States: John Wiley & Sons, Inc, 2016, pp. 285-318.

2. D. Staelin, A. Morgenthaler and J. Kong, *Electromagnetic Waves*. Englewood Cliffs, N.J. : Prentice Hall, 1994

3. Visser, H.J. (2005). The Linear End-fire Array Antenna. In Array and Phased Array Antenna Basics, H.J. Visser (Ed.). https://doi.org/10.1002/0470871199.ch6

4. W. W. Hansen and J. R. Woodyard, "A New Principle in Directional Antenna Design," in Proceedings of the Institute of Radio Engineers, vol. 26, no. 3, pp. 333-345, March 1938, doi: 10.1109/JRPROC.1938.228128.

5. S. Koziel and S. Ogurtsov, "End-fire array enhancement with gradient-based numerical optimization," 2012 International Conference on Electromagnetics in Advanced Applications, Cape Town, South Africa, 2012, pp. 292-295

6. I. Ahmadianfar, O. Bozorg-Haddad, and X. Chu, "Gradient-based optimizer: A new metaheuristic optimization algorithm," Information Sciences, vol. 540, pp. 131–159, Nov. 2020, doi: https://doi.org/10.1016/j.ins.2020.06.037.

Appendix:

The following pages will contain pages of the Matlab code used to compare the ordinary end-fire array and the Hansen-Woodyard end-fire array. In addition, the attempt at gradient descent will be included.

Understanding the Hansen-Woodyard End Fire Array

An analysis of the Hansen-Woodyard End-Fire Array in comparison to an ordinary array.

Kristtiya Guerra

```
clear;
```

Antenna Arrays

Array Factor

The array factor is a formula that is used to represent a N-element array. The model uses the assumption that each element in the array are identical nodes.

$$AF_N = \frac{1}{N}\frac{sin(\psi N/2)}{sin(\frac{\psi}{2})}$$

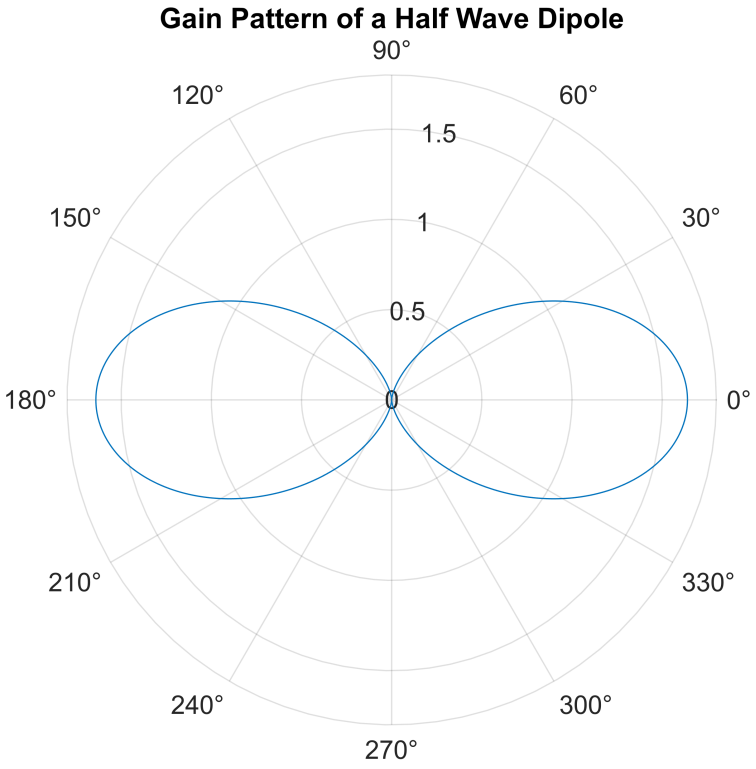
The product of the array factor with an antenna model of the designer's choice (for example, the gain of a dipole) is an N-element array.

Parameters:

```
f = 2*10^9;                 % Frequency
c = 3*10^8;                 % Speed of Light
lambda = c/f;               % Wavelenght
k = 2*pi/lambda;            % Wave Number
theta = 0:.001:2*pi;      % Angle (theta)
```

Antenna Models

```
%G = (1.67.*sin(theta).^3)
theta_rotate = theta+pi./2;
G = 1.64.*(cos(cos(theta_rotate).*pi./2).^2)./(sin(theta_rotate).^2); %Half-Wave
Dipole
%G = (3./2).*sin(theta).^2;                              %
figure
polarplot(theta, G)
hold on
title('Gain Pattern of a Half Wave Dipole');
hold off
```



Ordinary End-Fire Arrays

An ordinary end-fire array is characterized by an N-element array of equally spaced antennas. Each antenna are identical and of equally magnitude. The ordinary end-fire array can only be modified by the number of elements in the array and the distance between each element. The spacing determines if the antenna will point in both directions, or in only one direction.

```
N = 10;                                                 % Number of elements in the
array
d = [lambda/2;lambda/4;lambda/6;lambda];      % Distance between each antenna
element
k = 2.*pi./lambda;                                      % Wave number
```

The phase shift for an ordinary end fire array is equal to $\beta = -kd$ for $\theta = 0$ and $\beta = kd°$ for $\theta = 180°$. It is not influenced by the number of elements.

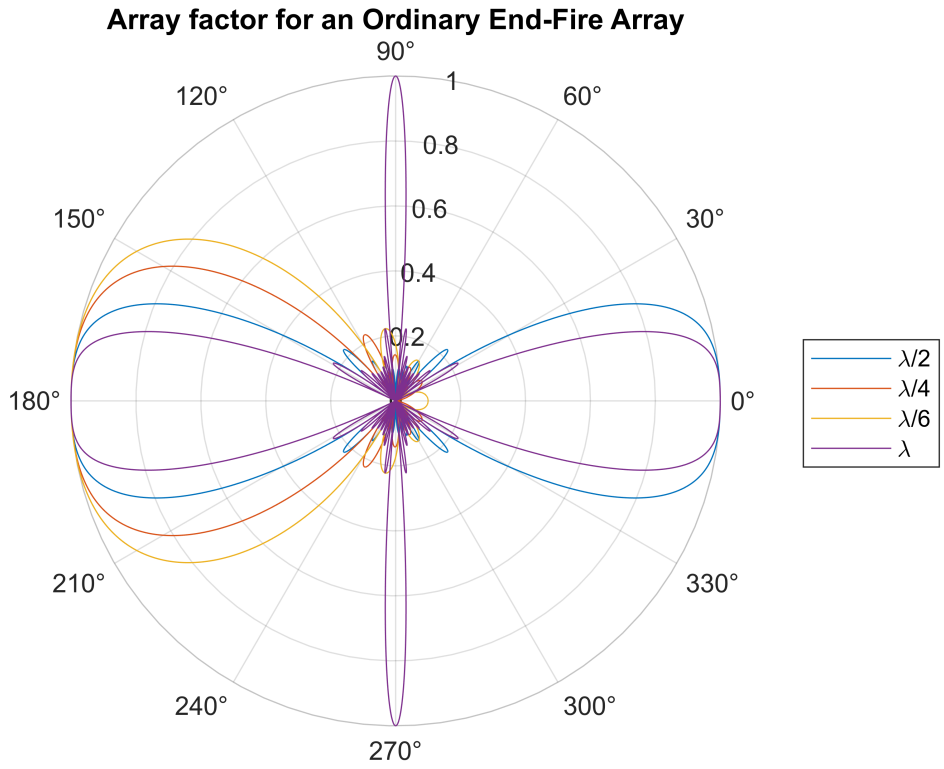
```
beta = k.*d;                                            % Phase Shift
psi = (k*d.*cos(theta) + beta);
AF_n_ordinary = abs((sin(psi.*N./2) ./ sin(.5.*psi) ));
AF_n_ordinary_norm = (AF_n_ordinary./N);      % Array Factor
```

You can visibly see in this plot how the distance between antenna elements affects the overall pattern of the array factor. For the antenna to have major lobes pointing at both ends, the space between the elements should be $d = \lambda/2, \lambda, n\lambda$ where $n = 1, 2, 3, \ldots$.

However, the further the distance between the elements, the more high intensity lobes will appear.

Ideally, a proper ordinary end-fire array should have spacing at most $\lambda/2$

```
figure
polarplot(theta,AF_n_ordinary_norm);
hold on
title('Array factor for an Ordinary End-Fire Array');
legend('\lambda/2','\lambda/4','\lambda/6','\lambda');
hold off
```



Hanson-Woodyard End-Fire Array

The Hansen-Woodyard Array is a modified form of the Ordinary End-fire array. It features a progressive phase shift that is influenced by the number of elements.

$\beta = -(kd + \frac{2.92}{N})$ for $\theta = 0°$

$\beta = +(kd + \frac{2.92}{N})$ for $\theta = 180°$

The distance between elements is also influenced by the number of elements - the smaller amount of elements, the closer together. The higher number, the closer to $d \approx \lambda/4$.

$d = (\frac{N-1}{N})\frac{\lambda}{4} \approx \frac{\lambda}{4}$

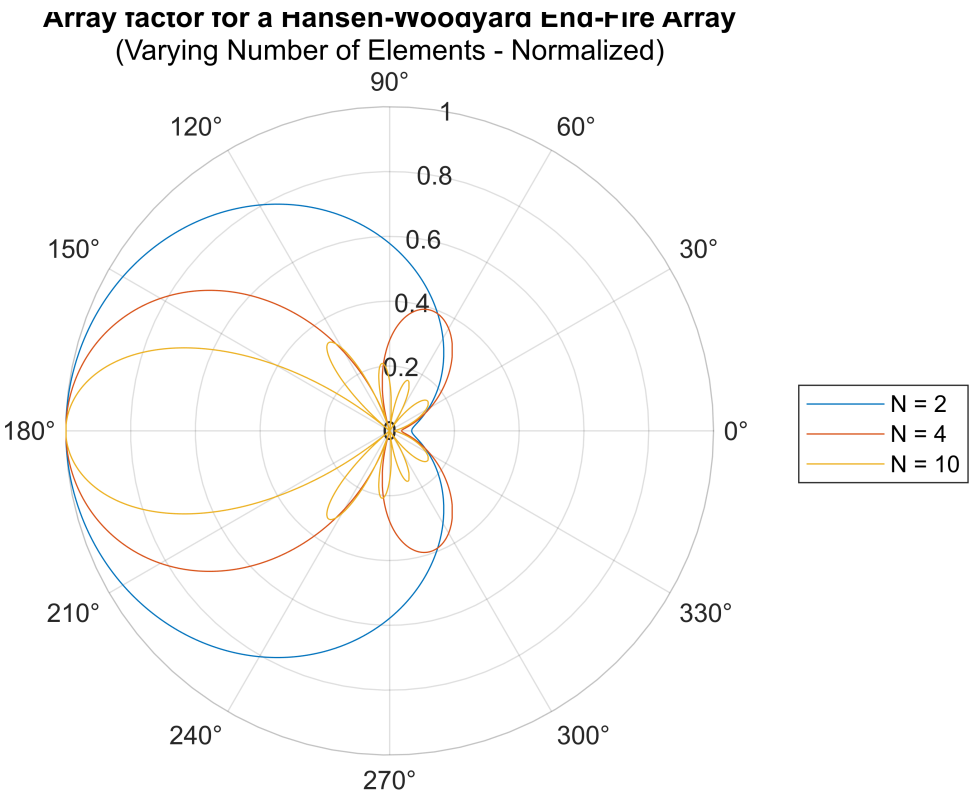
This design is useful for large N. We can see this with the by using the model for small N.

Parameters:

```
f = 2*10^9;                 % Frequency
c = 3*10^8;                 % Speed of Light
lambda = c/f;               % Wavelenght
k = 2*pi/lambda;            % Wave Number
theta = 0:.00001:2*pi;      % Angle (theta)

N = [2;4;10];
d = ((N-1)./N).*(lambda./4);        % Distance between each antenna element
beta = k.*d + 2.94./N;              % Phase Shift
psi = k.*d.*cos(theta) + beta;      % Relative Phase
AF_N = abs((sin(psi.*N./2) ./ sin(.5.*psi) ));
Max = [max(AF_N(1,:)); max(AF_N(2,:)); max(AF_N(3,:))];
AF_HW = abs(AF_N)./Max;
```

The directivity is low. The main beams are so wide that an ordinary end-fire array would be preferred over the Hansen-Woodyard.

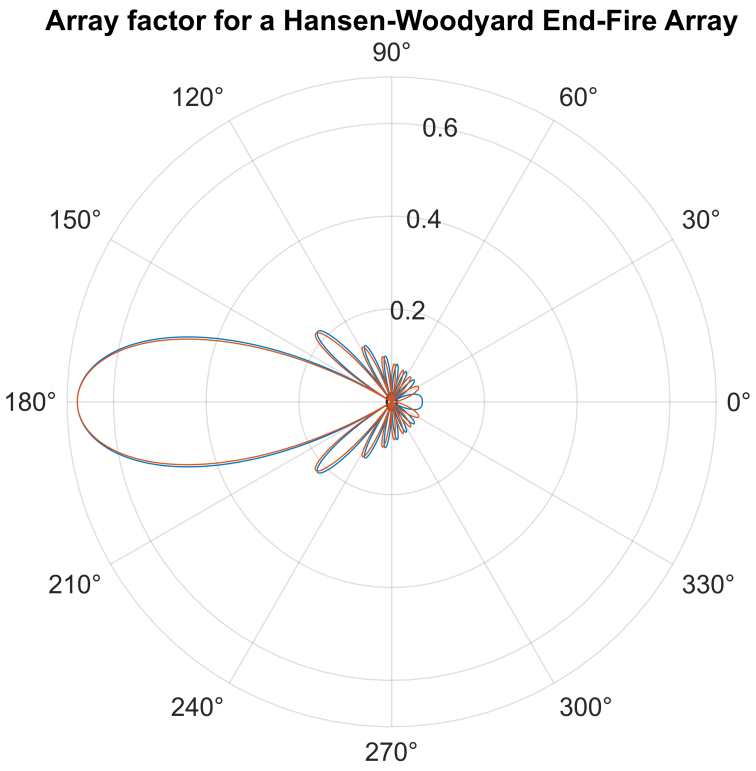
```
figure(3)
polarplot(theta,AF_HW);
hold on
title('Array factor for a Hansen-Woodyard End-Fire Array','(Varying Number of
Elements - Normalized)')
legend('N = 2','N = 4','N = 10');
hold off
```



For this analysis, I found that making $N \geq 10$ allows for a much fairer comparison between the two array designs, so onward I will be using 10-element arrays.

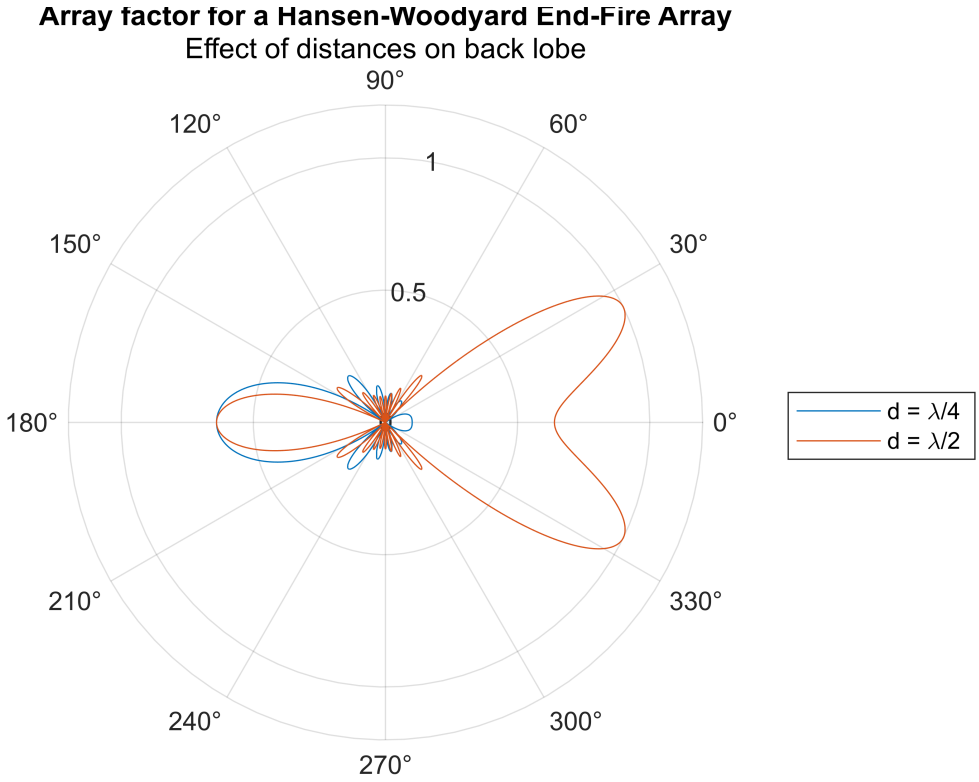
Comparing the distance formula and the proposed $\lambda/4$ between each element.

```
N = 15;
d = [((N-1)./N).*(lambda./4);lambda/4];                    % Distance between each
antenna element
beta = k.*d + 2.94./N;                              % Phase Shift
psi = k*d.*cos(theta) + beta;
AF_num = sin(psi.*N./2);
AF_den = sin(.5.*psi)*N;
AF_N = abs(AF_num./AF_den);

figure(4)
polarplot(theta,AF_N);
hold on
title('Array factor for a Hansen-Woodyard End-Fire Array')
hold off
```



Effect of the distance between elements and the backlobe

```
N = 10;
d = [lambda./4;lambda/2];                    % Distance between each antenna element
beta = (k.*d + pi./N);                          % Phase Shift
psi = k*d.*cos(theta) + beta;
AF_num = sin(psi.*N./2);
AF_den = sin(.5.*psi);
AF_N = abs(AF_num./AF_den);
AF_N_norm = AF_N.*(1/N);

figure(5)
polarplot(theta,AF_N_norm);
hold on
title('Array factor for a Hansen-Woodyard End-Fire Array','Effect of distances on
back lobe')
legend('d = \lambda/4','d = \lambda/2');
hold off
```



Comparing the Hansen-Woodyard End-fire array with the Ordinary End-fire array

```
theta = -pi:.00001:pi;
N = 10;
d = lambda/4;                    % Distance between each antenna element
beta_HW = (k.*d + pi./N);        % Phase Shift - HW
beta = k.*d;                     % Phase Shift - Ordinary
psi_ordinary = k*d.*cos(theta) + beta;
psi_HW = k*d.*cos(theta) + beta_HW
```

```
psi_HW = 1×628319
    0.3142    0.3142    0.3142    0.3142    0.3142    0.3142    0.3142    0.3142 ···
```

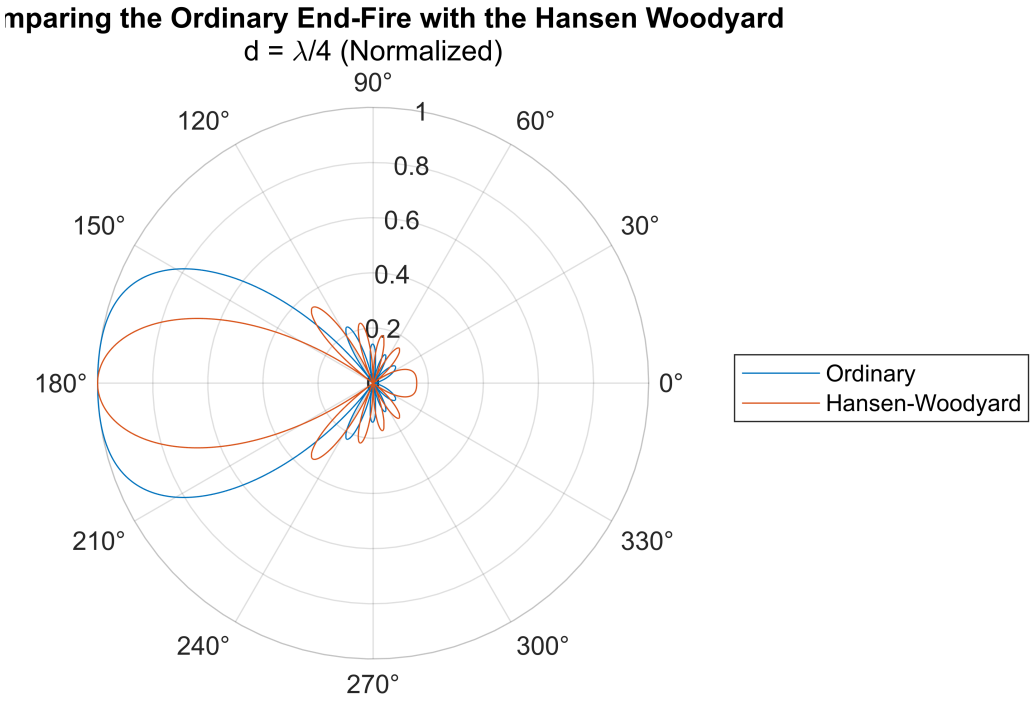
```
psi = [psi_ordinary; psi_HW];
AF_num = sin(psi.*N./2);
AF_den = sin(.5.*psi);
AF_N = abs(AF_num./AF_den);
AF_N_norm = AF_N(1,:).*(1/N);
AF_HW_Norm = AF_N(2,:)./(max(AF_N(2,:)))
```

```
AF_HW_Norm = 1×628319
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000 ···
```

```
Null_first_index = find(AF_HW_Norm<.05,1);
Null_first_index(1);
MainLobeBW = rad2deg(2*pi + 2*(theta(Null_first_index)))
```

```
MainLobeBW = 71.3734
```

```
polarplot(theta,AF_N_norm);
hold on
polarplot(theta,AF_HW_Norm);
title('Comparing the Ordinary End-Fire with the Hansen Woodyard','d = \lambda/4
(Normalized)')
legend('Ordinary','Hansen-Woodyard');
hold off
```

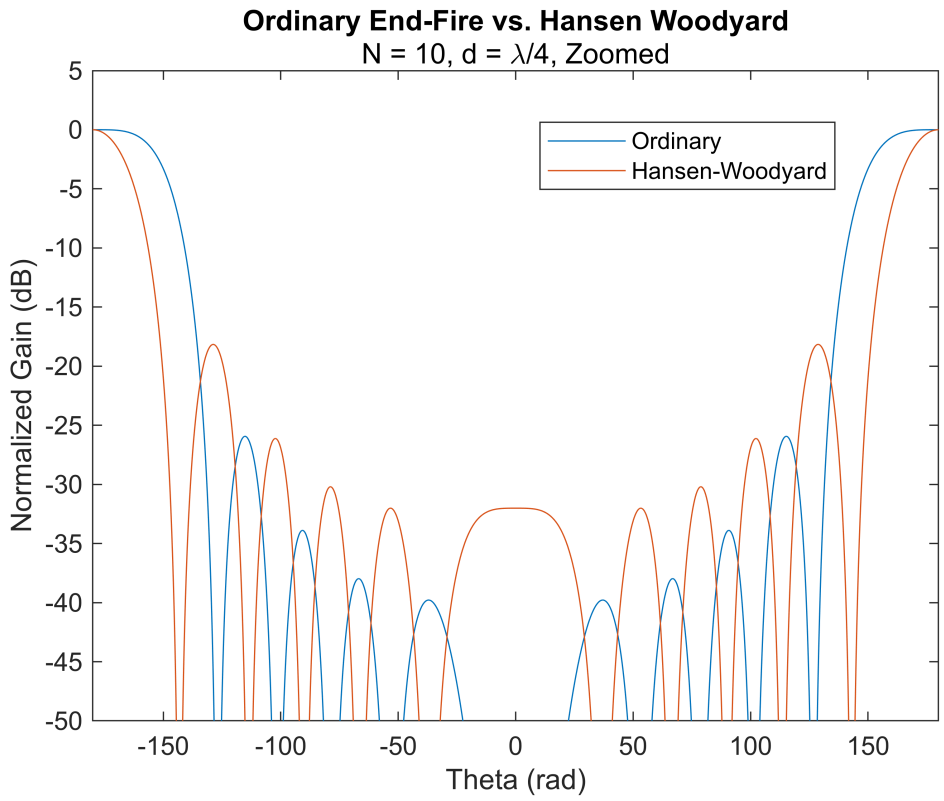


What I find odd is using the given formula for the nromalized array factor, I do not get magnitude of 1. It goes against the concept of normalization.

Radiation Intensity of the Antenna Arrays

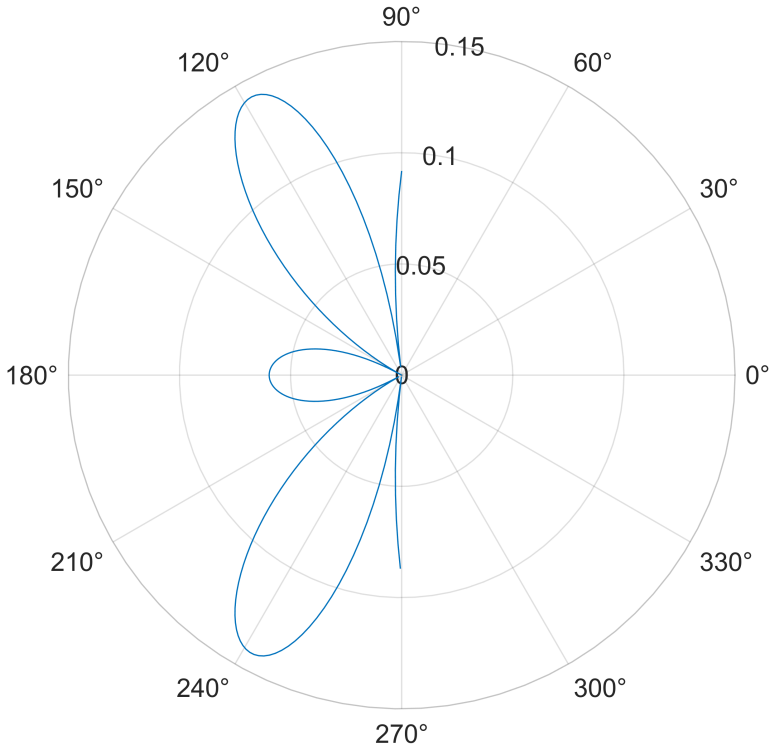
```
figure
plot(rad2deg(theta),mag2db(AF_N_norm.^2))
hold on
plot(rad2deg(theta), mag2db((AF_HW_Norm).^2))
xlim([-180 180])
ylim([-50 5])
title('Ordinary End-Fire vs. Hansen Woodyard','N = 10, d = \lambda/4')
legend('Ordinary','Hansen-Woodyard');
legend("Position", [0.55,0.8,0.25,0.0])
```

```
xlabel('Theta (rad)');
ylabel('Normalized Gain (dB)')
hold off
```

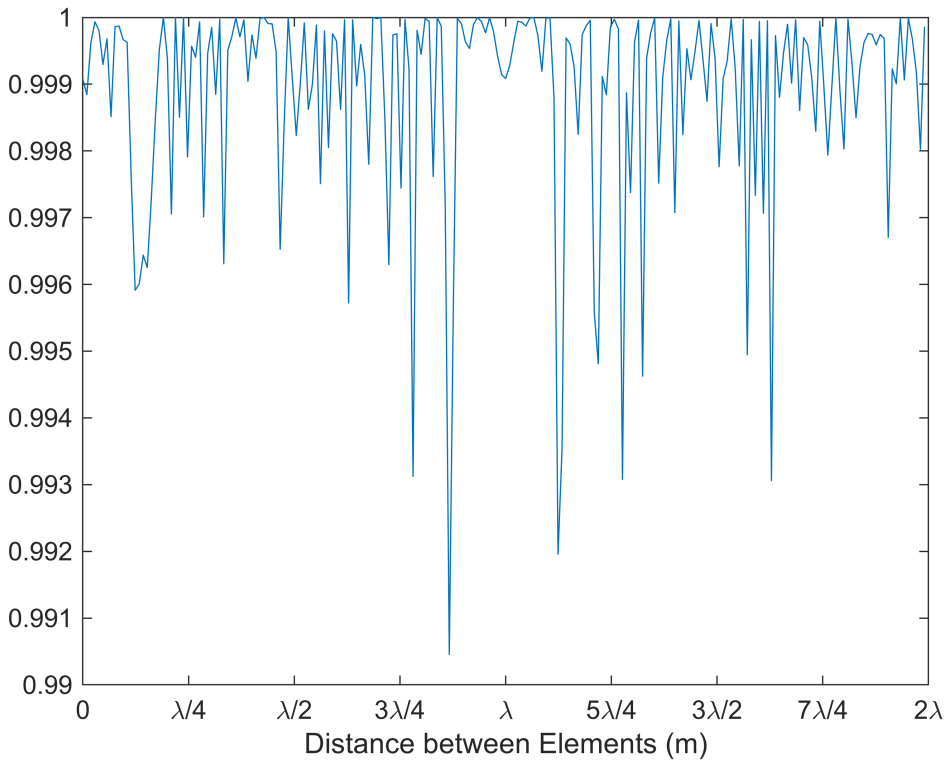


Effect of distance between each element and the back lobe

```
theta = pi/2:lambda/10:3*pi/2;      % Angle (theta)
N = 10;
a = size(theta,2);
% d = [2*lambda;lambda;lambda/2;lambda/4;lambda/6];                  % Distance
between each antenna element
d = 0:2*lambda/a:2*lambda;
d = d(1:end-1).';
beta = -(k.*d + pi./N);                          % Phase Shift
psi = k.*d.*cos(theta) + beta;
AF_num = sin(psi.*N./2);
AF_den = sin(psi./2);
AF_N = abs(AF_num./(AF_den.*N));
peak = max(AF_N);

figure
polarplot(theta, AF_N(15,:))
```




```
figure
plot(d,peak)
hold on
xlabel('Distance between Elements (m)')
set(gca,'Xtick',0:lambda/4:2*lambda,'XTickLabel',{'0','\lambda/4','\lambda/
2','3\lambda/4','\lambda','5\lambda/4','3\lambda/2','7\lambda/4','2\lambda'});
hold off
```

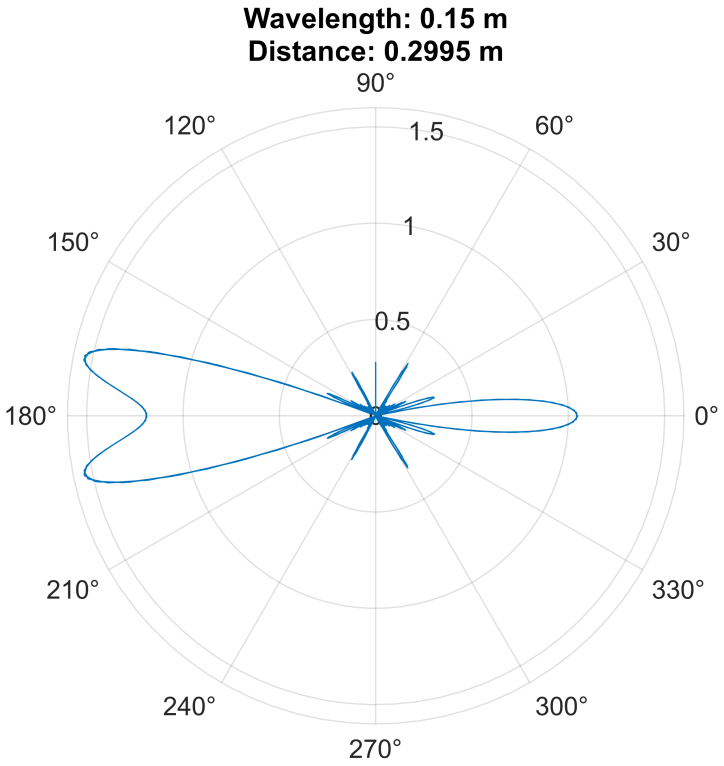



```
theta = pi/2:lambda/5:3*2*pi;      % Angle (theta)
N = 10;
a = size(theta,2);
G = 1.64.*(cos(cos(theta+pi/2).*pi./2).^2)./(sin(theta+pi/2).^2); %Half-Wave Dipole

% d = [2*lambda;lambda;lambda/2;lambda/4;lambda/6];                  % Distance
between each antenna element
d = 0:2*lambda/a:2*lambda;
d = d(1:end-1).';
beta = -(k.*d + pi./N);                          % Phase Shift
psi = k.*d.*cos(theta) + beta;
AF_num = sin(psi.*N./2);
AF_den = sin(.5.*psi);
AF_N = abs(AF_num./AF_den)./N;
Pattern = G.*AF_N;
peak = max(Pattern);

N = [1,2,3,4,5]
```

```
N = 1×5
     1     2     3     4     5
```

```
dipole_pos = d.*N;
Y = ones(1,5);

for i=1:size(theta,2)
```

```
    polarplot(theta,Pattern(i,:))
        h = title({sprintf('Wavelength: %.2f m',lambda);sprintf('Distance: %.4f
m',d(i))}); % title is 13 chars in both cases
    drawnow
    pause(.1)
end
```



Optimizing the ordinary end-fire array

We want to maximize the directivity of our antenna, which means we want to minimize the beamwidth of our main lobe. Decreasing the half power beamwidth (HPBW)

```
lambda = 0.15;
N = 10;
learning_rate = 0.1
k = 2.* pi/lambda
d = .05
input_d = lambda/10:lambda/100:.05;
```

The HPBW approximation for an ordinary endfire is applicable when $\frac{\pi d}{\lambda} << 1$. This means that the distance between our elements are a limited range.

```
Qualify = pi.*input_d./lambda
plot(input_d,Qualify)
```

```
hold on
yline(1,'r')
xlim([.015,.075])
hold off

HPBW = 2.*acos(1 - (1.39.* lambda./(pi.*N.*input_d)))
plot(input_d, rad2deg(HPBW))
hold on
xlabel('Distance between elements (m)')
ylabel('HPBW (degrees)');
title('Approximated Half Power Beamwidth (HPBW)','as distance between elements is
varied');
hold off
```

When using a simple approach as this, the smallest we can get our beamwidth, if we are being generous with what counts as what is much less than 1, is around 60 degrees.

```
beta = k.*d;                                              % Phase Shift
psi = (k*d.*cos(theta) + beta);
AF_n_ordinary = abs((sin(psi.*N./2) ./ sin(.5.*psi) ));
AF_n_ordinary_norm = (AF_n_ordinary./N);      % Array Factor
figure(3)
polarplot(theta,AF_n_ordinary_norm);
hold on
title('Array factor for an Ordinary End-Fire Array','d = 0.05 m');
hold off
```

```
syms d_change
HPBW_sym = 2.*acos(1 - (1.39.* lambda./(pi.*N.*d_change)))
HPBW_p1 = diff(HPBW_sym)
HPBW = subs(HPBW_p1,d_change,input_d)
```

```
clear;
f = 2*10^9;                 % Frequency
N =10
```

```
N = 10
```

```
c = 3*10^8;                 % Speed of Light
lambda = c/f;               % Wavelenght
k = 2*pi/lambda;            % Wave Number
theta = 0:.01:2*pi;      % Angle (theta)
mx = lambda/2
```

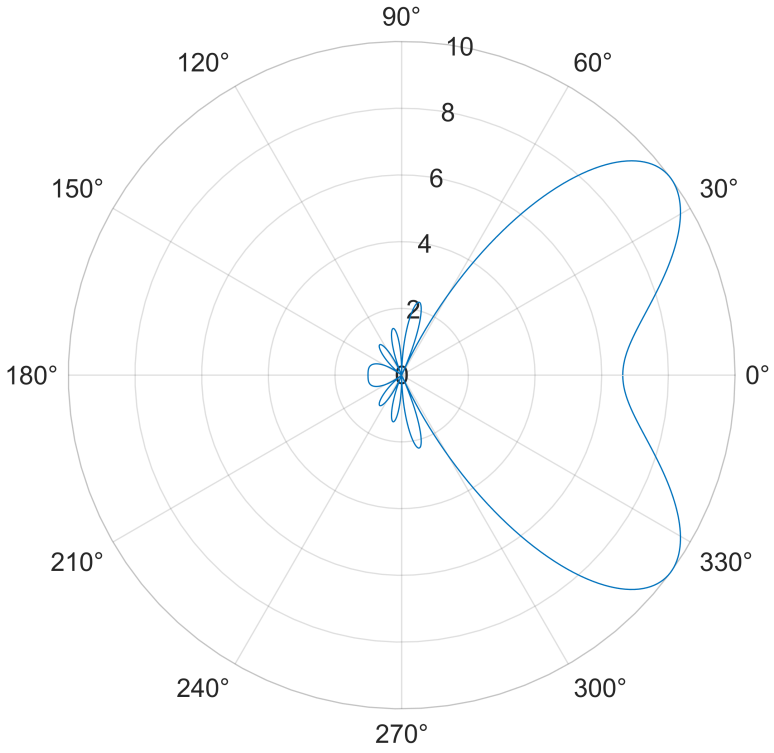
```
mx = 0.0750
```

```
% x =  0:mx/length(theta):mx;
% x = x(1:end-1).';
```

```
x = lambda/4
```

```
x = 0.0375
```

```
y =  0:2*pi/length(theta):2*pi;
y = y(1:end-1).';
AF = sin((k.*x.*cos(theta) + y).*N./2)./ sin(.5.*(k.*x.*cos(theta) + +y));

for i=1:size(theta,2)
    polarplot(theta,abs(AF(i,:)))
    drawnow
end
```




```
AF = sin((k.*x.*cos(0) + y).*N./2)./ sin(.5.*(k.*x.*cos(0 ) + +y));

plot(y,AF)
```

```
%% Define the main function
% Defining the main function that we would like to optimize. For the
% Hansen-Woodyard array, we should focus on changing just the distance
% between elements and determining the ideal progressive phase shift.
% This means we want to focus on changing the value of d and beta.
% My goal is to try to minimize the backlobe of the main lobe.

%% Parameters:
clear;
lambda = 0.15;      %(m)
k = 2*pi/lambda;    % Wave Number
N = 10;
theta = 0:.1:2*pi;

AF = @(x,y) sin(N.*(k.*x.*cos(0) + k.*x +y)./2)./ sin(.5.*(k.*x.*cos(0)+ k.*x +y));
AF_backlobe = @(x,y) sin(N.*(k.*x.*cos(0) + y)./2)./ sin(.5.*(k.*x.*cos(0) +y));

syms AF_Sym x_sym y_sym
AF_Sym = sin(N.*(k.*x_sym.*cos(0) + y_sym)./2)./ sin(.5.*(k.*x_sym.*cos(0) +y_sym));

mx = lambda/4;
x =  lambda/8:mx/length(theta):mx;
x = x(1:end-1).';
y = 0:.01:2*pi;
[X,Y] = meshgrid(x,y);

Z = AF_backlobe(X,Y);

% Gradient Descent (find minima)
    % gen equation: z2 = z1 - alpha*gradient(f(x,y)), alpha is learning rate

% Setup Equation (partial derivatives -- do analytically or using MATLAB)
Df_x = diff(AF_Sym,x_sym);                  % symbolic expression
Df_y = diff(AF_Sym,y_sym);

grad_x = matlabFunction(Df_x);      % symbolic --> anonymous function
grad_y = matlabFunction(Df_y);

% Initial Conditions
x_n(1,1) = lambda/4;                        % initial x position
x_n(1,2) = pi;                          % initial y position
x_n(1,3) = AF_backlobe(x_n(1,1),x_n(1,2));    % initial z position
gamma = 0.001;                           % learning rate


% Algorithm
err_x = 5; err_y = 5;
i = 1;
```

```
while err_x > 0.01 || err_y > 0.01
    % Calculate next x,y,z (subtract partial derivs)
    x_n(i+1,1) = x_n(i,1) - gamma*grad_x(x_n(i,1),x_n(i,2)); %x
    x_n(i+1,2) = x_n(i,2) - gamma*grad_y(x_n(i,1),x_n(i,2)); %y
    x_n(i+1,3) = AF_backlobe(x_n(i+1,1),x_n(i+1,2)); %z (from x,y)

    % Update difference between consecutive points
    err_x = abs(x_n(i+1,1) - x_n(i,1));
    err_y = abs(x_n(i+1,2) - x_n(i,2));

    % increment i (move to next row)
    i = i + 1;

    % break loop if going too long
    if i > 200
        break
    end
end

%
surf(X,Y,Z),
% hold on
% xlabel('distance (m)'),ylabel('Progressive Phase Shift (theta)'),zlabel('Backlobe ↙
Gain')
% title('Backlobe Gain as a function of Progressive Phase Shift and Distance');
% scatter(x_n(:,1),x_n(:,2),'r','filled')
% hold off
% Output
fprintf("Reached target in %d iterations\n",length(x_n))
fprintf("Final pt is (x=%1.2f,y=%1.2f,z=%1.2f)\n",x_n(end,1),x_n(end,2),x_n(end,3))

%
% axis off
% set(gca,'color','none')
% set(gcf,'color','none')

%% Display Figures:

figure
contourf(X,Y,Z)
hold on
title('Contour of Backlobe Gain')
xlabel('Progressive Phase Shift (radians)')
ylabel('Distance Between Elements (m)')
scatter(x_n(1,:),x_n(2,:))
colorbar;
hold off
```