Linear Processing of an Antenna Array

EEL 6935 - SPRING.2018 Hector Lopez 1-13-2018

Description

Consider the most trivial of all linear array processors that has beamformer weight values `w_i = 1 for every i = 1,2...M` assume that a unit-value signal `(m(t)=1)` on a carrier frequency `fc` is arriving at your antenna array with a direction of arrival `theta`. Assume inter-element spacing equal to half the carrier wavelength, `d=lambda_c/2`.

Plot the array output `|y|^2` in plain scale and in dB log-scale as a function of `theta` for `M=3,5,10` (this is known as a power pattern or power beam pattern).

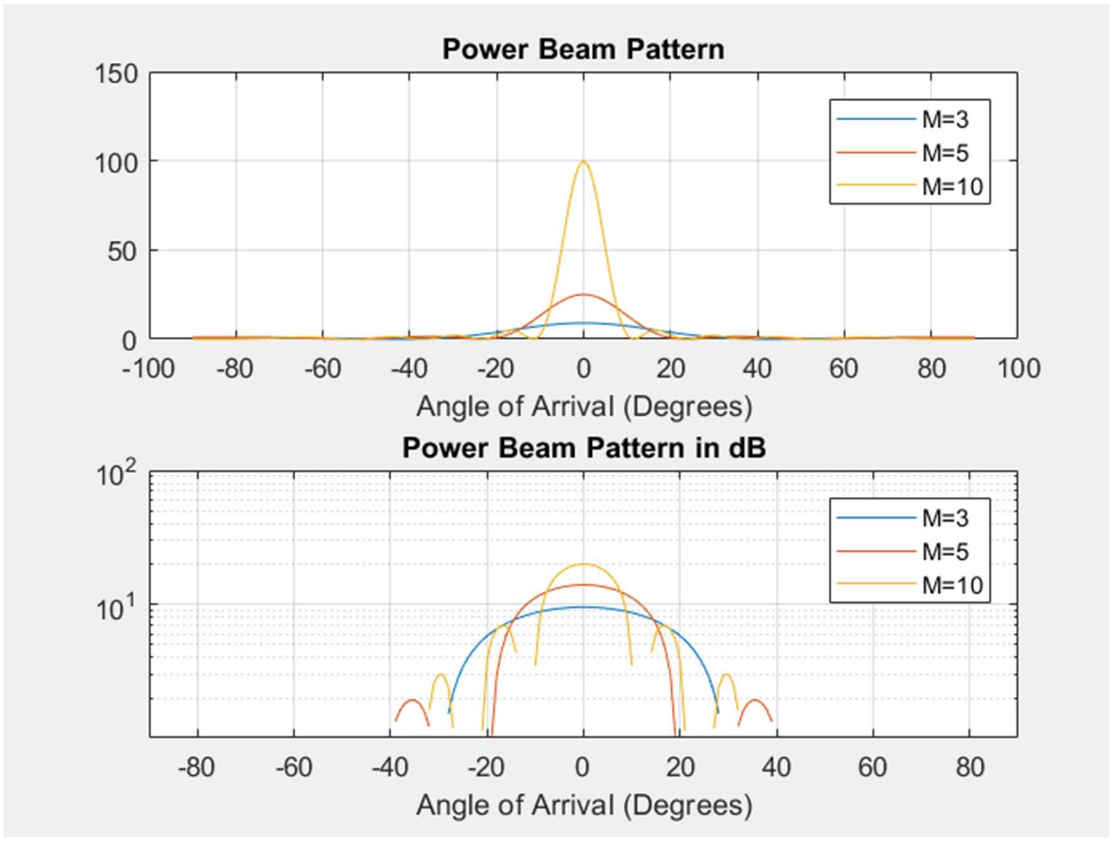
What do you observe? Discuss your findings. Would an array with fixed unit weights like that (plain adder) be useful in some way?

Observation

1. The greater the number of antenna elements in the array means that the output signal will have a proportional amount of energy. This makes sense because the arriving signals are summed together to create the output signal.
2. The angle of arrival, theta, is at its peak when it is at 0 degrees. It would make sense that the most power would come if all the signals where the angle of arrival is dead on to all the antenna elements at once. The signal should be weakest one directly at +/-90 degrees and hitting either the first or last antenna exclusively.
3. The scaling of power in the signal based on number of elements increases in a nonlinear way. From 5 to 10 antenna elements increases the output signal by orders of magnitude. Doubling from 10 to 20 may have the same exponential increase in the output signals power. I would assume that by increasing the power in the signal the input signals are amplified. The amplification can allow for an easier removal of any additive noises and allow for more information to be transmitted reliably.

Would an array with fixed unit weights be useful?

This type of configuration may be useful in testing linear array elements. Confirming the overall response has not diminished could verify that the antennas are performing optimally. Another use would be to map the surrounding area around the array. In a perfect world all the arriving signals would be un-impeded but in the real world there could be terrain or obstacles that would change the strength of each arriving signal at different angles. A unit weight vector could be useful in detecting these obstacles. And by adjusting the weights these a "clean" response could be achieved to mimic the graphs published.



```
%% Linear Antenna Array Processor
% Hector Lopez EEL6935 SPRING2018
%%
theta=-90:90;
figure
Y3=processor(3,1,theta,false);
Y5=processor(5,1,theta,false);
Y10=processor(10,1,theta,false);
subplot(2,1,1);
plot(theta,Y3,theta,Y5,theta,Y10);
grid on;
title('Power Beam Pattern');
legend('M=3','M=5','M=10');
xlabel('Angle of Arrival (Degrees)');

Y3=processor(3,1,theta,true);
Y5=processor(5,1,theta,true);
Y10=processor(10,1,theta,true);
subplot(2,1,2);
semilogy(theta,Y3,theta,Y5,theta,Y10);
grid on;
axis([-90 90 0 100]);
title('Power Beam Pattern in dB');
legend('M=3','M=5','M=10');
xlabel('Angle of Arrival (Degrees)');
```

```matlab
%% Processor(M,T)
% M=number of antenna elements in linear array
% T= maximum time window, max samples of time, T=1; t0.
% theta= 1d vector, angles(degree) of arrival for incoming signal
% dB=method of outputting the power beam pattern
function Y_theta = processor(M,T,theta,dB)
%carrier frequency
fc=2*10^9;
%speed of light
c=3*10^8;
%carrier wavelength
lambda_c =c/fc;
%optimum nyquist element spacing
d=lambda_c/2;
dd=0:1:M-1;
D=dd*d;

%array response vector
%theta is a vector that will effect evey input x
%arriving unit signal for each sample of T
m=ones(1,T);
%create unit vector array of weights
a = ones(1,M);
w=ctranspose(complex(a,0));

P=1:size(theta);
p=0;

for th = theta
    y=1:T;
    p=p+1;
    for t = 1:T
        S=exp(-1i*2*pi*(D/lambda_c)*sind(th));
        x=m(t)*exp(1i*2*pi*fc*t)*transpose(S);
        xw=x.*w;
        y(t)=sum(xw);
    end
    %Y=trapz(y);
    Y=y;
    if(dB==true)
        i=10*log10(abs(Y)^2);
        if(i>1)
            P(p)=i;
        else
            P(p)=0;
        end
    else
        P(p)=abs(Y)^2;
    end
end
Y_theta=P;
end
```