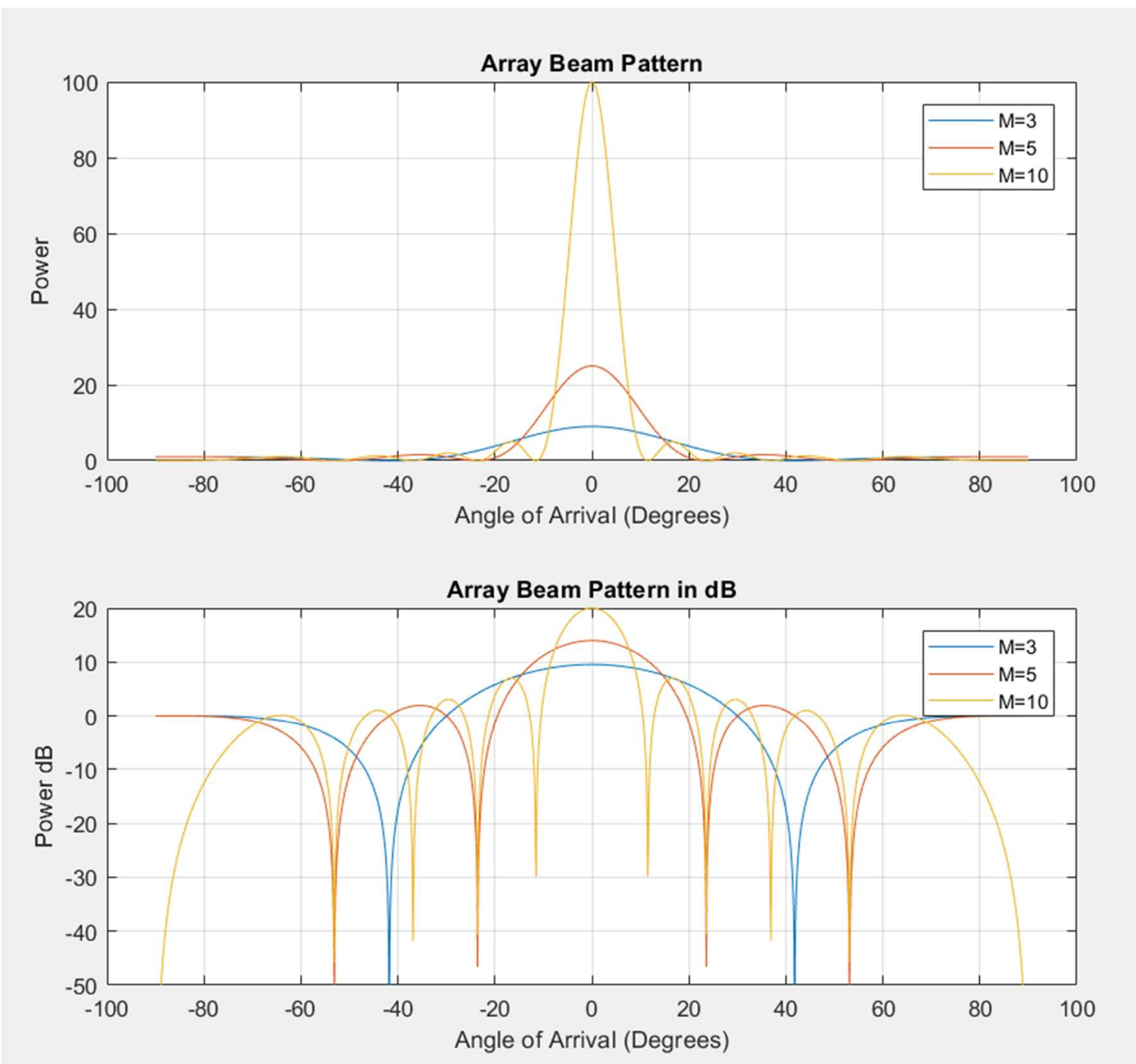
Linear Processing of an Antenna Array

EEL 6935 - SPRING.2018 Hector Lopez 1-16-2018

Problem 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).

Results



Observations

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 powerful output occurs when all the arriving signals are directly perpendicular to the linear antenna array. The output signal should be weakest at an angle of arrival of +/-90 degrees because that's when the arriving signal is only coming hitting either the first or last antenna exclusively.
3. Nulls; The more antennas the more nulls across the array there are. Even at an optimum Nyquist distance from each antenna on the array there exists angles of arrival that would not be detected by the array. The more antennas the less those lobes filter out signals.
4. By having many antennas tuning can be used to change the filter configuration of the array.
5. An amplified power signal would allow for greater details being observed. If the arriving signal is the same signal at all the antennas, then greater antenna elements could be helpful in "cleaning" the signal and cross referencing it to reduce noise.

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.

MATLAB Source Code

```
%% Linear Antenna Array Processor
% Hector Lopez EEL6935 SPRING2018
% Assignment 1
% Description:
%  Given a linear array of antenna elements of sizes
%  3,5,and 10, plot the array beam pattern output y(theta)
%  for arriving angle, theta ; -90 to 90 degrees. Plot
%  a dB and linear scale.
%%

theta=-90:0.1:90;

Y_M3=processor(3,theta);
Y_M5=processor(5,theta);
Y_M10=processor(10,theta);

figure
close all;
subplot(2,1,1);
plot(theta,((abs(Y_M3).^2)), ...
     theta,((abs(Y_M5).^2)), ...
     theta,((abs(Y_M10).^2)));
```

```
grid on;
title('Array Beam Pattern');
legend('M=3','M=5','M=10');
xlabel('Angle of Arrival (Degrees)');
ylabel('Power');

subplot(2,1,2);
plot(theta,10*log10((abs(Y_M3).^2)), ...
    theta,10*log10((abs(Y_M5).^2)), ...
    theta,10*log10((abs(Y_M10).^2)));
ylim=[-50 20];
set(gca,'YLim',[-50 20])
grid on;
title('Array Beam Pattern in dB');
legend('M=3','M=5','M=10');
xlabel('Angle of Arrival (Degrees)');
ylabel('Power dB');


%% Processor(M,T)
% M=number of antenna elements in linear array
% theta= 1d vector, angles(degree) of arrival for incoming signal
function Y = processor(M,theta)
      %carrier frequency
      fc=2*10^9;
      %speed of light
      c=3*10^8;
      %carrier wavelength
      lambda_c =c/fc;
      %optimum nyquist element spacing
      D=0:1:M-1;
      D=D*(lambda_c/2);

      %wieghts
      W=ones(M,1);
      %arriving signal
      m0=1;
      %time
      t=0;

      Y=1:size(theta);
      k=0;
      for th = theta
          k=k+1;
          S=exp(-1i*pi*(D/lambda_c)*sind(th));
          X=m0*exp(1i*2*pi*fc*t)*transpose(S);
          Y(k)=sum(W'*X);
      end
end
```