

This problem set is due Wednesday June 3.

Array Design: We have noticed that tapering the aperture illumination will reduce sidelobes at a cost of also reducing gain and increasing the beamwidth. This tradeoff also occurs with array antennas, except with an array we have more control over the aperture illumination. Thus we can consider adjusting the gain of each element in order to optimize the sidelobe level. This is exactly the same as the design of lowpass digital filters and we can use the tools of digital filter design to do the job. In Matlab the tool for finding the element gains that give equal sidelobes is called “firpm”. The tool for plotting the transfer function (beam shape) is “freqz”.

The filter design script “firpm” requires $N =$ the number of elements minus one. We will be designing an 8 element array because it is the biggest we can simulate with SuperNEC. So $N=7$. It requires two (or more) bands and the gain in each. We will specify a pass band of zero width and a stop band. The code does an equal ripple approximation. If we specify the pass band has zero width, it will put all the zeros in the stop band. This is good for antennas (generally). However it will not enforce unity gain in the main lobe, but we can enforce it by setting the sum of the element gains to unity.

Try: $b = \text{firpm}(7,[0 \ 0 \ .40 \ 1],[1 \ 1 \ 0 \ 0])$. This will give you a b vector of length 8. Correct the gain to unity by setting $b = b/\text{sum}(b)$. Display the transfer function (beam pattern) using $\text{freqz}(b,1,512,'whole')$. The pattern will have equal sidelobes and 7 zeros. The sidelobes will be about 35 dB below the main beam. Changing the stop band edge from 0.40 will change the sidelobe level. Increasing it will push the zeroes closer together on the unit circle, and thus lower the sidelobes. You can see this by plotting the zeros with “ $\text{plot}(\text{roots}(b))$.” Compare the roots of the uniform weight $b=\text{ones}(8,1)/8$ with those for $b = b = \text{firpm}(7,[0 \ 0 \ .40 \ 1],[1 \ 1 \ 0 \ 0])$.

1. Find the weight vectors b needed to create a 8 element arrays with sidelobe levels of, -20dB, -30dB, -40dB, and -50dB. Overplot the bandpasses for the 4 arrays using $\text{freqz}(b, 1, 512, 'whole')$. Add to that the uniformly illuminated array for which $b=[1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]/8$. Notice that we have not specified the element separation. That does not affect the sidelobe levels with respect to the main beam, it affects only the grating lobe spacing and thus the beamwidth and the gain. You will see the tradeoff between beamwidth and sidelobe levels directly.

2. Write a Matlab script to return the gain of a linear array of isotropic radiators $G(\cos\theta)$ given three vectors: the weight \mathbf{b} ; the element position vector \mathbf{p}/λ ; and the angle vector $\mathbf{c}\cos\theta$. Normalize the gain by integrating over the visible region as you did in assignment 2. Test it on a uniformly excited array of 8 elements with uniform spacing $d/\lambda = 1$. The “core” of this script can be written very efficiently in matlab, make sure you understand it because it gives you an idea of the power of the matrix/vector notation.

```
function gain = array(b,p,cs)
%program to calculate array gain
%inputs are element weights b, positions p, and cos(theta) all row vectors
%the cos(theta) vector must be equally spaced and cover (-1 to +1)
e= b*exp(j*2*pi*p'*cs); %note col*row = matrix, so the exp(...) is a matrix
g=e.*conj(e); %e = row* matrix = row can be complex and g=|e|^2
gain=2*g/(sum(g)*(cs(2)-cs(1))); %cs(2)-cs(1) = step in cos(theta)
end
```

For each array find the maximum element spacing d/λ such that the grating lobe does not exceed the sidelobe level. Overplot $G(\cos\theta)$ for $-1 < \cos\theta < 1$ using the optimal $d/\lambda = 0.8$ for each of the equal sidelobe arrays and also for the uniform array for comparison. Tabulate the tradeoff between sidelobe level, gain, and -3dB beamwidth.

3. It is interesting to see how close to the ideal behavior we can achieve with a real array including the effects of mutual impedances. For this we can simulate the arrays that you designed in part 2 above using SuperNEC. You can use the translate feature of the SuperNEC input (which will also duplicate), but you would have to edit the voltage on each element separately. You will find the assembly `linarray.m` on the classweb site that does this for you. To make it available through the GUI, put `linarray.m` in your directory `snec/matlab/input/assembly/antennas/`. It will ask you for a voltage vector. Test it with `[1 1 1 1 1 1 1]`. Adjust the spacing to `[0 0 .8]` which will stack the elements in z . Edit the simulation settings to add a 3-D radiation pattern on 2 degree inc. Simulate at the single default frequency and plot the gain. Make a 2-D plot by taking a phi slice of the 3-D plot at the peak of the element pattern, so the sidelobes are clearly visible.

The gain will be 3 dB higher than the Matlab calculation for isotropic radiators. Why?

Simulate the equal-sidelobe arrays designed above using perpendicular dipoles and compare their patterns with the uniformly weighted array. Overplot their gain patterns and that for the uniform array for comparison. Are the sidelobes in fact equal, as they are in Matlab? If not, why not?

Tabulate the peak gain, beamwidth, and first sidelobe level for comparison with the theoretical values.