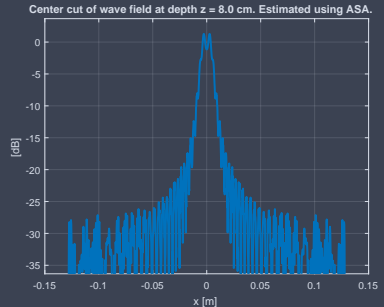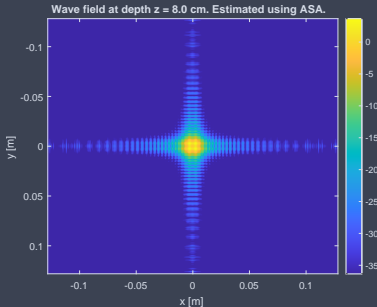# Project I: Apertures

Simon Andreas Bjørn
March 8, 2023

» **Unfocused ASA Simulation**

We want to run the ASA.m and inspect the results. Running the ASA.m code yields the following to plots


* Left plot is the wave field $U(x, y, z = z_0)$, of a uniform square point source with $D = 15\lambda$
* Right plot is the pressure along the x-axis at $y = 0, z = z_0$.
* Gibbs effect $\rightarrow$ Square source gives frequency discontinuety.
* Ripples in the x- and yaxis

The wave field has propagated a minimal amount from the true source at $z = 0$.

» Unfocused ASA Simulation 2

We want to taper the propagator, and propagate the wave field for some distance $z = 0.4$. Modifying the `ASA.m` script with the additions written in the assignment text,
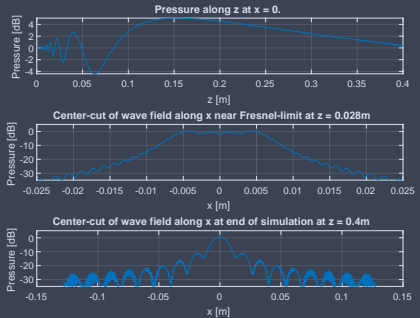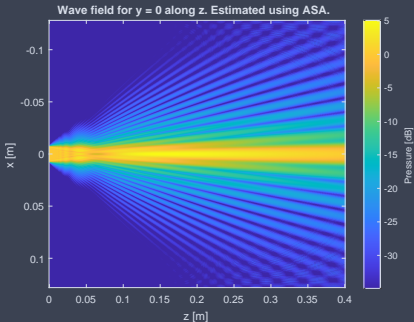

For the near field, I chose the Fresnel-limit at $d_F = \frac{D^2}{4\lambda} \approx 0.028m$. We see a clear difference between the wave field at the Fresnel-limit and in the far field.

* Taper has reduced the high frequencies of the Gibbs effect.

» **Array Pattern - Grating lobes**

Given an array of $M = 24$ elements, we want to find the beampattern of the array with different spacings of the elements. Here I use the `beampattern.m` code.

```
1  DXs = [P.lambda/4   P.lambda/2 ...
2         P.lambda   2*P.lambda];
3
4  ks = linspace(-1,1,N);
5  kx = 2*pi/P.lambda*ks;
6  weights = ones(M, 1);
7
8  for dx_idx = 1:numel(DXs)
9      dx = DXs(dx_idx);
10     D = M*dx;
11     xpos = linspace(-D/2, D/2, M);
12
13     W = beampattern(xpos, kx, weights);
14 end
```
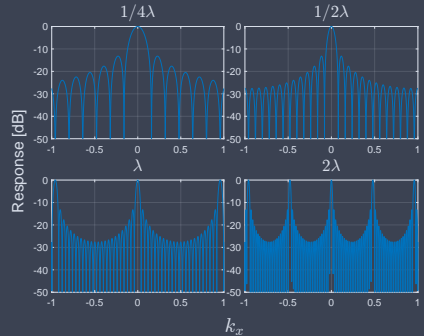

We see that for element distance $D \leq \frac{\lambda}{2}$ we don't see any grating lobes, while for $D \geq \frac{\lambda}{2}$ we see the grating lobes appear at the end of the array pattern.

» **Element weighting and spacing**

We now want to apply a temper to the array, using a Kaiser windo and see how different windows impact the array pattern.
Applying the window is pretty streight forward

1. Create a window of given length and shape

2. Normalize window so sum of elements are 1

3. Apply temper to beampattern

```
1 beta = betas(beta_idx);
2 win = kaiser(M, beta);
3 norm_win = win / sum(win(:));
4 W = beampattern(xpos, kx, norm_win);
```

Using analyzeBP.m we log the mainlobe width at -3dB and -6dB, as well as the max height of the sidelobes.

» **Element weighting and spacing 2**

We also want to find the white noise gain of the array. Since we use a normalized kaiser window, we assume that $W^H a(\omega, \tau) = 1$ which gives the gain $GW(\theta) = \|\vec{w}\|^{-2}$. Plotting the results from our logged values we have the following

```
1  win = kaiser(M, beta); norm_win = win / sum(win(:));
2  W = beampattern(xpos, kx, norm_win);
3  analyze = analyzeBP(ks, W); GW = 1 / norm(norm_win)^2;
```
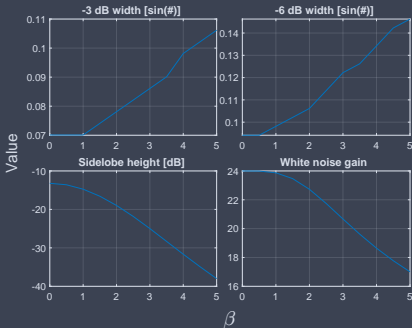

Looking at the plot to the left, we see that increasing beta

* Increases width of mainlobe

* Suppresses the height of the sidelobes

* Decreases the white noise gain

Conclusion: there is a trade-off when using the temper. It supresses the sidelobes to some degree at the cost of angular resolution.

» **Irregular Linear Array**

Now we have a irregular linear array with unity weights which we compare to the previous.[1]

* Plotting the array pattern gives the plot on the left

* Lower maximum sidelobe than ULA with same spacing

* Aperture width is 0.3mm smaller than the ULA

* With the current spacing corresponding to $d = \frac{1}{2}\lambda$, we see no grating lobes in either array.

Array pattern of irregular linear array
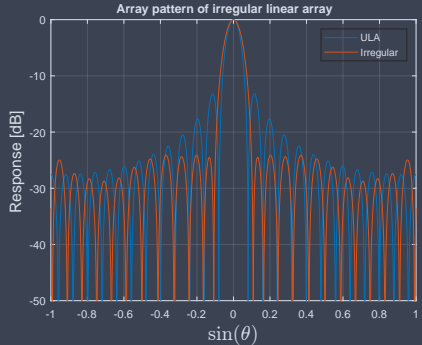
[1] Matlab code from the assignment is used to generate the element positions.

» **Irregular Linear Array 2**

Comparing the new irregular array to ULA with respect to the beampattern:

	-3dB Width [k]	-6dB Width [k]	Sidelobe height [dB]	White Noise Gain
ULA	0.070	0.096	-13.211	24
Irregular	0.086	0.117	-24.111	24

From the table above, we see that

* The mainlobe is a bit wider while its

* Sidelobes are lower.

* Both arrays have similar white noise gain.

Since both arrays use normalized unit weights, we have that the white noise gain is

$$GW = \|\vec{w}\|^{-2} = \frac{1}{\left(\sqrt{\sum^M \left(\frac{1}{M}\right)^2}\right)^2} = \frac{1}{M \cdot \frac{1}{M^2}} = M$$
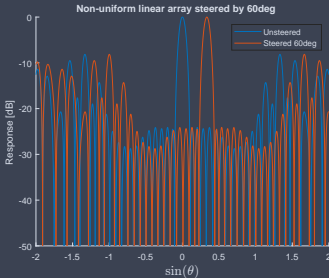
where M is the number of elements. Thus, the measured white noise gain is as expected, as we have $M = 24$ elements.
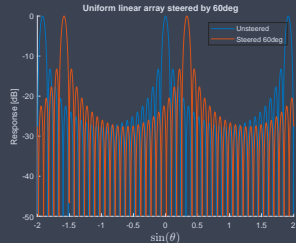
» Steering Irregular Array

Now we want to apply steering to the irregular array and compare it to the ULA. We want to get find the beampattern of the array with steering, which we apply by $W = (k_x - k_x^\circ)$.

Non-uniform linear array steered by 60deg

* From the plot on the left, we can see the effect of steering the array by 60 degrees. However, compared to steering the ULA, we see that no grating lobes appear.

Uniform linear array steered by 60deg

```
1  ks = linspace(-2, 2, N);
2  kx = 2*pi/P.lambda*ks; k0 = pi/(3*d);
3  W1 = beampattern(ElPos, kx - k0,
        norm_win);
```

We now want to steer the array different amounts and log the width of the mainlobe as a function of the steering angle.

-3dB Width of mainlobe with steering
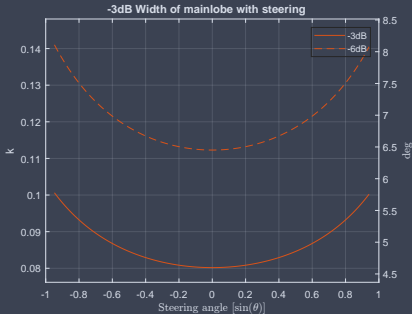
* -3dB and -6dB follows the same shape
* Increasing the steering angle widen the main lobe

» **Array thinning**

We assume an array with $M = 101$ elements with spacing $d = \frac{\lambda}{2}$ We want to analyse the array pattern when a selection of random elements are responsive.

	-3dB Width	-6dB Width	Mean Sidelobe	Max Sidelobe
25 random elements	0.0148±0.0017	0.0214±0.0026	-7.5813	0.9823
50 random elements	0.0149±0.0016	0.0219±0.0013	-11.4279	1.2940
75 random elements	0.0147±0.0015	0.0220±0.0002	-13.1505	1.4858
Dense array (101)	0.0140	0.0220	-13.3277	- N/A -
Minimum Beamwidth	0.0180	0.0260	-12.0399	- N/A -
Minimum Sidelobe	0.0180	0.0260	-12.0399	- N/A -

* Increasing number of elements narrows mainlobe width

* Increasing number of elements suppress sidelobe levels

* Dense array performs the best of all arrays analyzed

* Both optimized arrays perform identically (code bug)

* The optimized arrays supress sidelobes to the etter than 50 elements
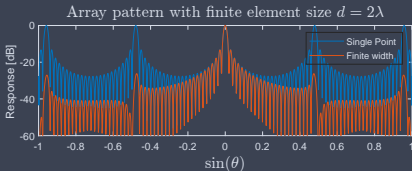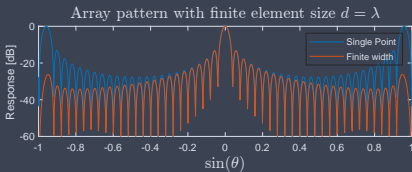
» Element directivity

We want to again consider the regular array from before, with $d = \lambda$ and $d = 2\lambda$, but this time assume element has finite width d.

We use the formula for the combined array pattern

$$W_{tot}\left(\vec{k}\right) = W_a\left(\vec{k}\right) \cdot W_e\left(\vec{k}\right)$$

where W_e is the element response given by $W_e\left(\vec{k}\right) = \frac{\sin(dk/2)}{k/2}$.

Array pattern with finite element size $d = \lambda$

Array pattern with finite element size $d = 2\lambda$

* Grating lobes appear at same places as with array with points

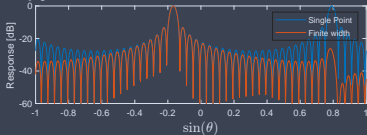* Grating lobes are suppressed when using finite element sizes

```
1  We = sin(kx*d/2) ./ (kx / 2);
2  Wa = beampattern(xpos, kx, weights);
3  Wtot = We.'.*Wa;
```
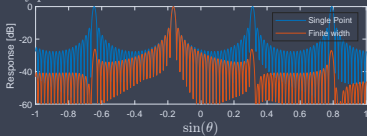
» **Element directivity but with a twist**

Now we want to implement steering to the array.
We apply steering to the elements and the array the same way as previously

Array pattern with finite element size $d = \lambda$ steered to $\theta = -30°$

Array pattern with finite element size $d = 2\lambda$ steered to $\theta = -30°$

* For $d = \lambda$ a small skew is observed in the first grating lobes.

* The wrapped grating lobes are smaller than for a similar array with point sensors.

```
1  We = sin((kx-k0)*d/2) ./ ((kx-k0) / 2);
2  Wa = beampattern(xpos, kx-k0, weights);
3  Wtot = We.'.*Wa;
```