Project III

MIMO Pulse-echo imageing

Simon Andreas Bjørn

May 4, 2023

» 1 - Pulse compression

We want to implement pulse compression (match filtering).
The implementation of pulse compression looks like this

```
# Match filter       |--Perform Match Filter--||-----Get the positive lag------|
match_filter(x, y) = conv(x, reverse(conj(y)))[length(y)÷2+1:end-(length(y)÷2)]
```

We can find the theoretical time-resolution in seconds for the sequence after pulse compression by using the bandwidth B which is given by

$$\delta\tau = \frac{1}{B}$$

```
δτ = 1/B |> toUnit(u"ms")
@show δτ; # → δτ = 0.1 ms
```

So the theoretical time-resolution in seconds are $0.0001s$ or $0.1ms$

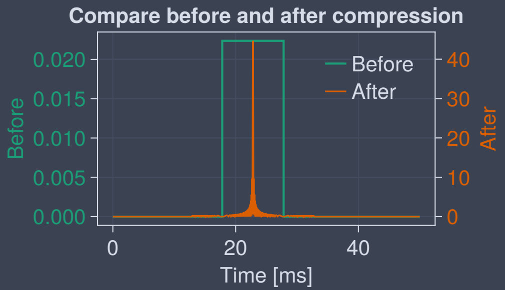
» 1 - Test Pulse compression

We want to test the pulse compression on a channel from the `tdma_data` and plot before and after filtering. First, we create the transmitted LFM pulses as defined in the assignment.

```
α = B/Tₚ
S_up   = (t -> @. exp(1im*2π*((fc - B/2)*t + α*t^2/2)))(0s:1/fs:Tₚ);
S_down = (t -> @. exp(1im*2π*((fc + B/2)*t - α*t^2/2)))(0s:1/fs:Tₚ);
```

Then, we apply `match_filter()` on the data

```
raw_signal = tdma_data[:,1,1]
match_signal = match_filter(raw_signal, S_up)
```



» 1 - Pulse compression measured performance

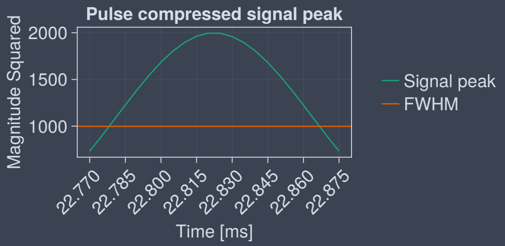
Measure the FWHM of the pulse compression around the peak to find performance.

```
# Assumes only 1 peak at interval x
fwhm(x) = (count(a -> a >= 0.5maximum(x), x)+2) / fs
```

Evaluating `fwhm(x)` on `match_signal` we find the measured time-resolution

```
P_match = @. abs(match_signal)^2
τₚ = fwhm(P_match) |> toUnit(u"ms")
@show τₚ; # → τₚ = 0.1 ms
```

We see that this matches exactly with the theoretical resolution.



» 2 - Virtual Arrays

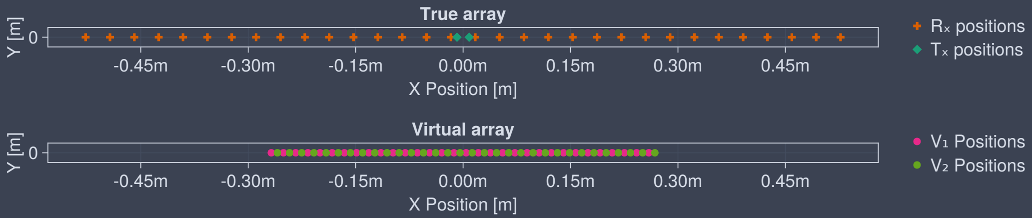
We want to construct a virtual array of our setup. We know that each virtual sensor is defined as

$$V_x = \frac{(R_x + T_x)}{2}$$

Mapping the receiver positions on each transmit, we get 2 groups of virtual sensors.

```
v_pos1 = map(x->(x+tₓ_pos[1]) / 2, rₓ_pos)
v_pos2 = map(x->(x+tₓ_pos[2]) / 2, rₓ_pos)
v_pos = sort([v_pos1; v_pos2])
```

Plotting these virtual arrays we get the following plot



» 2 - Theoretical resolution estimation

We now want to estimate the theoretical resolution in both axial and lateral direction.

* Theoretical lateral resolution in radians

```
λ = c/fc |> toUnit(m)
L₁ = v_pos1[end] - v_pos1[1]
Lₐ = v_pos[end] - v_pos[1]

@show δβ₁ = λ/2L₁; # → δβ₁ = 0.0323
@show δβₐ = λ/2Lₐ; # → δβₐ = 0.0317
```

* Approximate lateral resolutin at 4m distance

```
@show δβ₁_4m = δβ₁*4m; # → δβ₁_4m = 0.129 m
@show δβₐ_4m = δβₐ*4m; # → δβₐ_4m = 0.127 m
```

* Axial pulse-echo resolution

```
δr = c/2B |> toUnit(m);
@show δr; # → δr = 0.017 m
```

» 3 - Delay-And-Sum

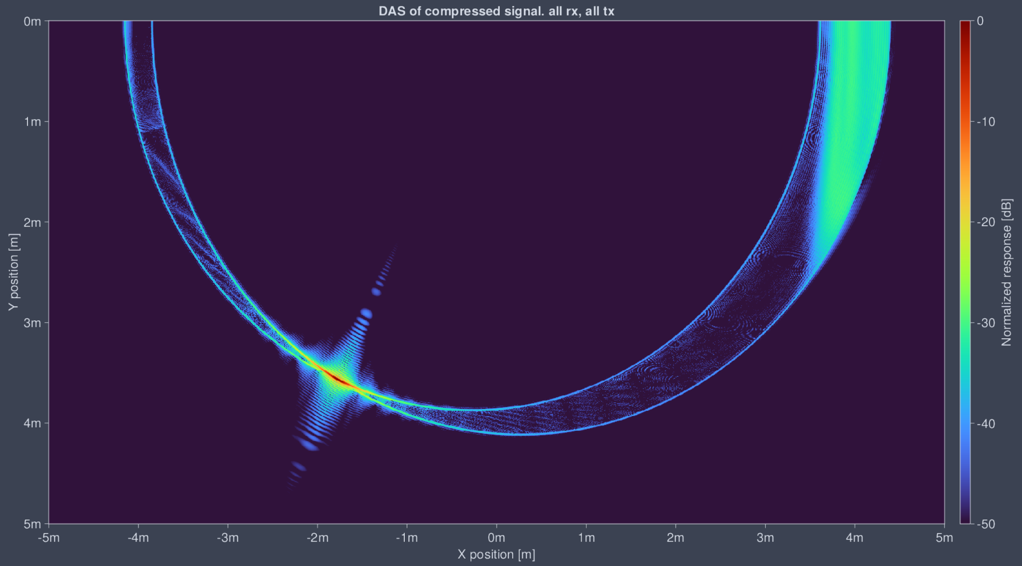
Implementation of DAS

```julia
function DAS(grid, channel_data, tx, rx)
    result = zeros(ComplexF64, size(grid))
    for k in eachindex(tx)     # For each transmit
        for j in eachindex(rx) # For each receiver
            # For each pixel
            for i in CartesianIndices(grid)
                # Get Delay
                rᵣ = norm([rx[j], 0m] .- grid[i])
                rₜ = norm([tx[k], 0m] .- grid[i])
                τ = (rₜ+rᵣ)/c
                # Get index from delay
                τ_idx = round(Int, τ * fs)
                # Sum over delayed signal
                result[i] += channel_data[τ_idx,j,k]
            end
        end
    end
    result
end;
```

Running DAS on the tdma data. Choosing a resolution smaller than $\delta\beta$ and δr such that reflector has some size larger than a signle pixel. Choosing $\delta x = \delta\beta/4$ and $\delta y = \delta r/2$

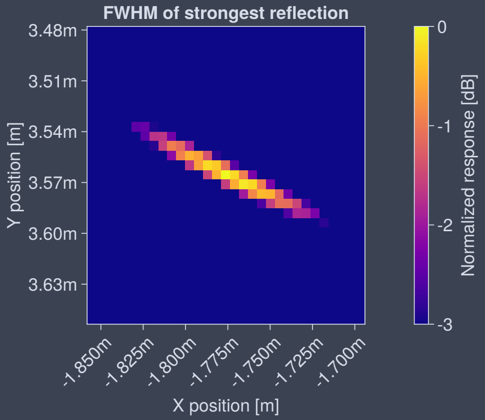
```julia
X = (-5:δβₐ/4:5)m
Y =  0m:δr/2:5m
grid = [ [i,j] for i=X, j=Y ]
mfdata = mapslices(channel -> match_filter(
             channel, S_up), tdma_data, dims=(1,))
result = DAS(grid, mfdata, tₓ_pos, rₓ_pos);
```

Plotting the results yields image on next slide.


DAS of compressed signal. all rx, all tx
0m
1m
2m
3m
4m
5m
Y position [m]
-5m
-4m
-3m
-2m
-1m
0m
1m
2m
3m
4m
5m
X position [m]
0
-10
-20
-30
-40
-50
Normalized response [dB]

» 3. Measuring resolution of DAS image

We now want to locate the position of the recletor. We look at a reagion around the maximum response at $0dB$ and limit the range to $[-3dB, 0dB]$ and get the following.



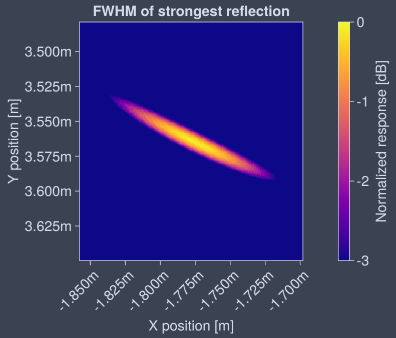
* Strongest reflector at

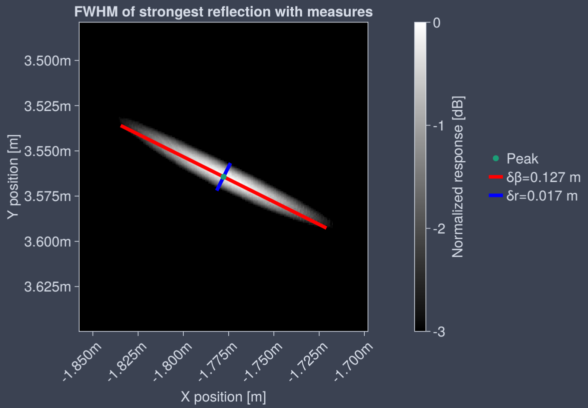
```
R = argmax(db_result)
@show (xs[R[1]], ys[R[2]]);
# → (-1.777m, 3.565m)
```

» 3. Supersampling around reflector

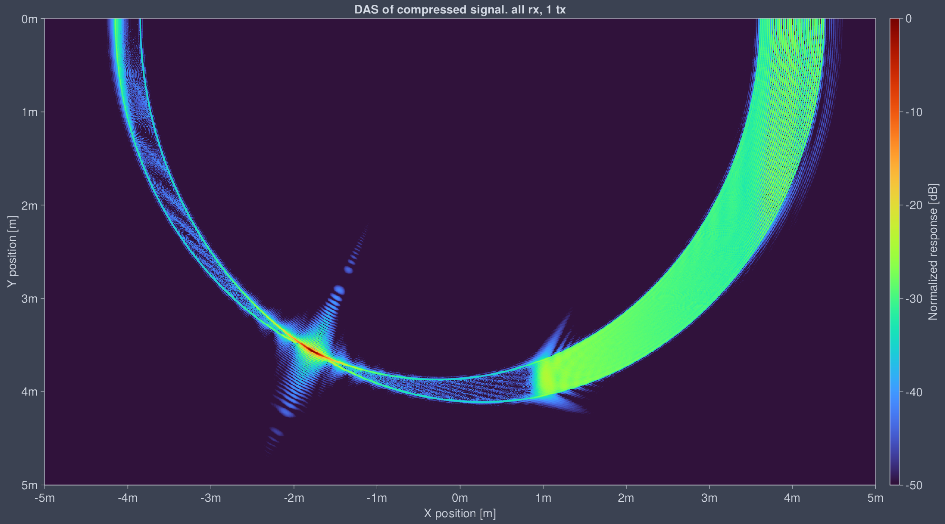
Because we know where the reflector is, we can beamform a much higher reslution grid around it to better measure it.

Drawing axial and lateral line segments equal to the theoretical resolution of both gives





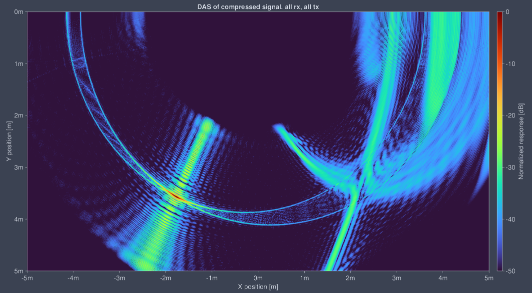
» 4. Single transmit DAS



» 5. Delay-and-sum on CDMA data

Similar to TDMA, but all the data is in 1 recording, and the two transmits use different chirps. Pulse compress the signal with up and down chirps to get 2 transmit signals, 1 for each chirp.

```
mfdata_cdma = Array{ComplexF64,3}(undef, Nₜ, Nᵣₓ, Nₜₓ)
mfdata_cdma[:,:,1] .= mapslices(channel -> match_filter(channel, S_down), cdma_data, dims=(1,))
mfdata_cdma[:,:,2] .= mapslices(channel -> match_filter(channel, S_up),   cdma_data, dims=(1,))
```



* We see there is a lot of cross talk compared to previous