Module 5: Sonar

a)

- What is the theoretical angular resolution of the system at the center frequency?

To find the theoretical angular resolution we can use the following formula:

$$\delta\beta = \frac{\lambda}{L}$$

Where λ is the wavelength and L is the total array length Given the number of elements N_h with length D, the angular resolution becomes:

$$\delta\beta = \frac{\lambda}{N_h \cdot d} = \frac{0.0149m}{32 * 0.0375m}$$
$$\delta\beta = \underline{\underline{0.0124}}$$

- What is the angular field of view of the system at the center frequency?

The field view (the beamwidth of the main lobe) is given by:

$$\beta \approx \frac{\lambda}{D} = \frac{0.0149m}{0.0375m} = \underline{\underline{0.3973}} \tag{1}$$

(2)

- What is the range (depth) resolution?

  Given the assumed speed of sound in water c and transmit bandwidth B, the range resolution is given by:

$$\delta r = \frac{c}{2B} = \frac{1540 \text{ m/s}}{2 \cdot 30,000 \text{ s}^{-1}}$$
$$= 0.0257\text{m} = \underline{\underline{25.67\text{mm}}}$$

b)

- Use Eq. (1) to generate a synthetic replica of the transmitted signal. Code must be presented.

To generate a replica of the transmitted signal, we consider the specified transmit equation:

$$s_{Tx}(t) = \begin{cases} \exp(j2\pi\alpha t^2/2) & -T_p/2 \leq t \leq T_p/2 \\ 0 & |t| \geq T_p/2 \end{cases}$$

We first need to identify the parameters of the equation, and the number of samples to use in the array. We initially only construct the non-zero part of the signal (though it's later padded for the FFT). The number of transmit samples must be coherent with the time resolution used for the receive channel data. Otherwise, the convolution pulse compression implementation will give incorrect results. We can use the sampling frequency in the USTB object to determine the number of transmit samples:

```
alpha = (bw)/t_p;
n_transmit_samples = floor(t_p*channel_data.sampling_frequency);
t_transmit = linspace(-t_p/2, t_p/2, n_transmit_samples)';
s_Tx = exp(j*2*pi*alpha*t_transmit.^2/2);
```

- Implement a pulse compression algorithm in MATLAB.

For the implementation of pulse compression, we use the Fourier Transform formulation:

$$s_m(\tau) = \mathcal{FT}^{-1}\left\{\mathcal{FT}\left\{s_{Rx}(t)\right\}\mathcal{FT}\left\{s_{Tx}(t)\right\}^*\right\}$$

The empty result buffer is created, with twice the length of the original signal. This is necessary, since the convolution operation (done here through the FFT) doubles the length as the two signals "pass through" each other:

```
match_filtered_data = zeros(2*n_receive_samples-1, channel_data.N_elements);
```

Then, for each array element, we perform the pulse compression using the fast fourier transform. The received channel data is extracted, and the FFT is performed on the receive and transmit data. The number of samples `n` is set equal to the result length, which pads the data. Finally, we take the inverse fourier transform of the transformed receive data multiplied by the complex conjugate of the transformed transmit data. We also make sure to only store the first `n_receive_samples` elements of the result into the USTB object:

```
for elem = 1:channel_data.N_elements
    s_Rx = channel_data.data(:, elem);
    % Do FFT compression
    S_Rx = fft(s_Rx, 2 * n_receive_samples - 1);
    S_Tx = fft(s_Tx, 2 * n_receive_samples - 1);
    s_m = ifft(S_Rx .* conj(S_Tx));
    match_filtered_data(:, elem) = s_m;
end
channel_data_compressed.data = match_filtered_data(1:n_receive_samples, :);
```

- Describe the difference between the raw data and the pulse compressed data. Explain why there is a difference.

From the figure, we can see that the matched filter (pulse compression) significantly reduces the noise in the signal. This is possible because we know the exact (theoretical) transmitted pulse, which is used by the matched filter. Conceptually, the filter tries to identify (or "match") the transmit signal in the received signal by performing a convolution. This way we can filter out parts of the receive signal that are not related to the transmitted signal. The result is a signal with maximum Signal-to-Noise Ratio (given additive white gaussian noise).

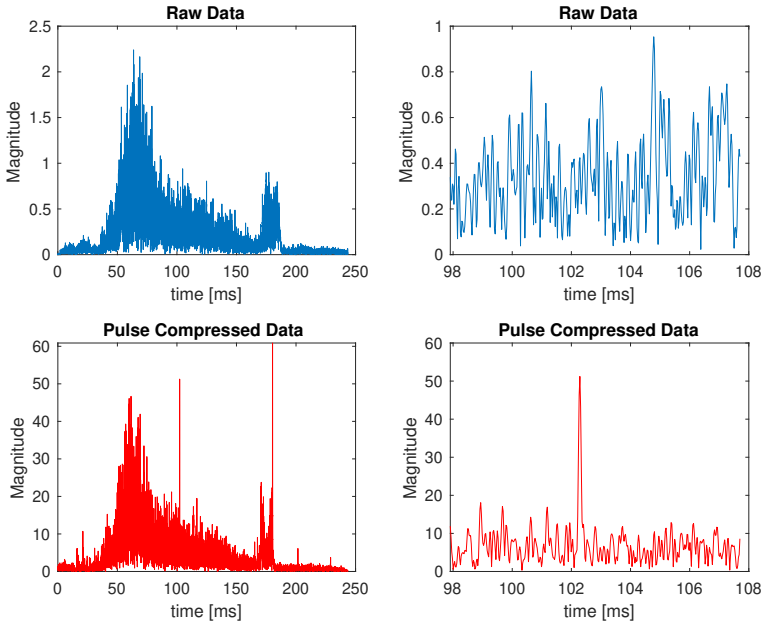


Figure 1: The result of pulse compression on a single channel

c)

- Define the image scan

Here, we define a simple sector scan using the parameters found in a):

```
az_res = 0.0124;
range_res = 0.02566;
scan = uff.sector_scan();
scan.depth_axis = [0:range_res:160];
scan.azimuth_axis = [-0.2:0.0124:0.2];
```

- Run the processing object on both channel data objects.

We set up a delay-and-sum pipeline in USTB and perform beamforming without receive or transmit apodization:

```
    az_res = 0.0124;
    range_res = 0.02566;
    scan = uff.sector_scan();
    scan.depth_axis = [0:range_res:160];
    scan.azimuth_axis = [-0.2:0.0124:0.2];
```

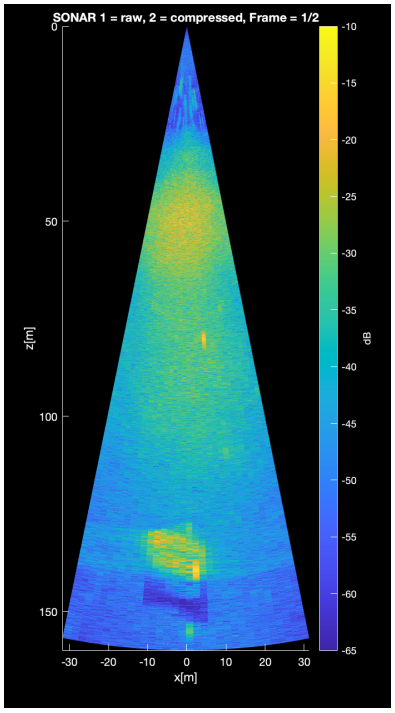
- Run the processing object on both channel data objects

We build and run a midprocess USTB pipeline and simply swap the channel data for the raw/compressed signals:

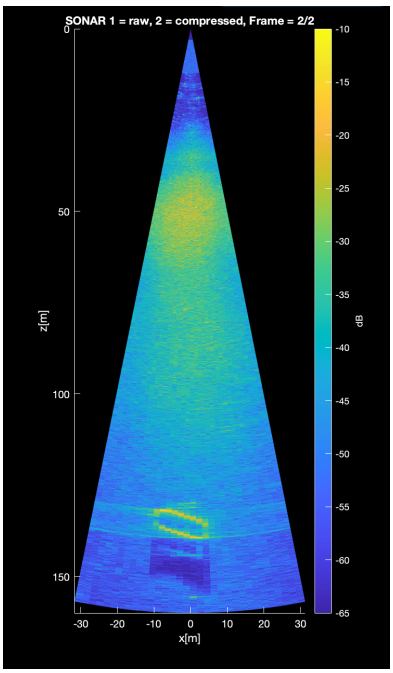
```
mid = midprocess.das();
mid.scan = scan;
mid.transmit_apodization.window=uff.window.none;
mid.receive_apodization.window=uff.window.none;
mid.code = 'matlab';
mid.channel_data = channel_data;
% beamform raw data
b_data = mid.go();
% beamform pulse compressed data
mid.channel_data = channel_data_compressed;
b_data_compressed = mid.go();
```

- Display the results

We first show the final result of beamforming on the raw and pulse compressed channel data:



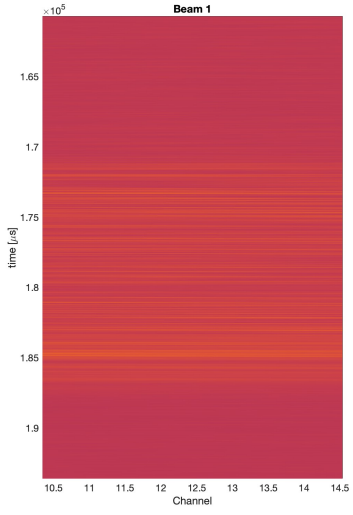
(a) Raw beamformed image



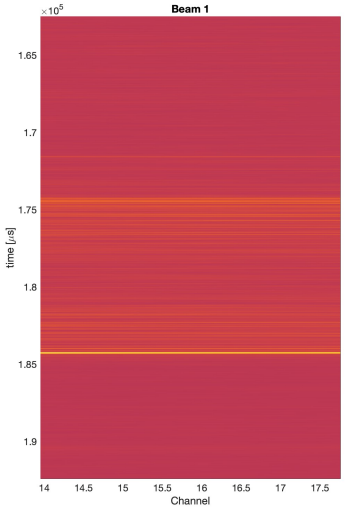
(b) Pulse compressed beamformed image

We see a clear improvement in SNR and contrast, and the contour of the object of interest is much clearer. In general, we also see a reduction in the ambient noise in the image.

To visualize the data before beamforming (raw and compressed), we can zoom into an area of interest. Here we have zoomed in on the object appearing in the scan. From the figure, we can see that the pulse compressed channel data is more distinct, with clearly defined borders:



(a) Raw channel signals



(b) Pulse compressed channel signals

- There is a target at approximately 130 m range. What is it? How large is it in meters?

We can measure the approximate size of the object by looking directly at the plot, since the axes are in meters.

$$L \approx 14.7\text{m}$$

$$D \approx 8.0\text{m}$$

I'm not quite sure what the object is. Maybe it is a boat? :)