# Beamforming

## *Aims:*

The purpose of this exercise is to become familiarized with beamforming, including the use of apodization on the receive aperture, and the use of expanding aperture to control the F-number. The exercise covers the process from channel data to beamformed image. One of the main challenges will be to perform calculations on multidimensional data. It is recommended to always keep track of the dimensions of the matrices using size(). Other useful MATLAB expressions and functions:

```
mult = X.*Z; XsqrRoot = sqrt(X); Xsqr = X.^2;
squeeze(); size(); permute();
```

Attached to this exercise you will find two datasets, simData and invivoData, containing channel data from an ultrasound simulation using single point scatterers, and from the neck of a healthy volunteer. The channel data are acquired from plane wave transmissions, acquisitions in which all elements are fired simultaneously, and an entire ultrasound image may be generated from one transmission. The simulations used the same probe and center frequency as in the recording. Also attached to the exercise are two MATLAB functions, *interpTOF* and *generateApod.*

The data files contain the variables *RFdata, RF_t* and *elPosX*.

- *RFdata*: Array containing channel data. The format of the data is [time, channels]
- *RF_t*: Time axis for the RFdata.
- *elPosX*: Positions along the x-axis for the elements on the transducer. The transducer is positioned on the axis, so that the z-positions are all 0.

## Task 1: Time of Flight calculation

In this first task we will calculate the beamforming delays for each spatial point in the image and transducer element. In a plane wave transmission, it is assumed that the plane wave front travels at the speed of sound c=1540 m/s in the z-direction below the transducers. When it hits a scatterer, the backscattered signal travels radially away from the scatterer until it hits the transducer elements. The total time from transmission until the backscattered signal reaches the elements is referred to as the Time of Flight (TOF).

- Set x=0 and z=0.01 m, and calculate the TOF for each channel.
- Load the dataset *simData*. Plot the RFdata versus time and channel number on logarithmic scale, using for example `imagesc()`. The simulation contains a point scatterer in position x=0, z=0.01 m. Plot the TOF versus channel data in the same plot and verify visually that the TOF matches the channel data.

The following code generates X and Z-coordinates for the beamforming grid, the spatial points from which we want to generate the image, and initializes the Time of Flight matrix:

```
x = linspace( -2e-2, 2e-2, 256); %x-coordinates
z = linspace( 0, 4.5e-2, 512); %z-coordinates
[X, Z] = meshgrid( x, z); %make X and Z matrices

% initialize TOF-matrix
TOF = zeros( length(z), length(x), length(elPosX) );
```

- Fill out the TOF matrix with the total Time of Flight for each z-position, x-position and transducer element.

## Task 2: Apodization

The function `delayedData = interpTOF( RFdata, RF_t, TOF)` interpolates channel data such that for each receive focus point (x,z), the signal from this point is aligned across the different channels. The output is the realigned data [channels, z, x]

In addition to beamforming we need to obtain the envelope of the signal. In this exercise we will use the Hilbert transform to produce the pre-envelope, and obtain the envelope by taking the absolute value.

```
envelope = abs( hilbert( bf_RFdata) );
```

- Use the Time of Flight matrix generated in task 1 to realign the channel data for each point (x,z). Sum the predelayed data across all channels, and calculate the envelope to obtain the beamformed image. Normalize your image such that the strongest pixel has value 0 dB and plot using a dynamic range of 50 dB.

*Apodization* refers to the process of weighting the different channels before summation. Apodization is typically used to suppress sidelobes in the point spread function, on the cost of lateral resolution, but it may also be used to control the size and position of the aperture.

Experiment with different apodization functions, show the result and comment on how they affect the image. What happens when you..

- ..weight the channels using a Hamming window. Use the MATLAB function *Hamming*.
- .. use a Hamming window on the middle 32 elements and weight the rest of the elements with 0.
- .. use a Hamming window on the first 32 elements and weight the rest of the elements with 0.

## Task 3: Expanding aperture

If the aperture is kept constant, the effective F-number on receive will depend on the depth. To compensate for this, it is common to gradually expand the aperture with increasing depth. The function `apod = generateApod(elPosX, xpos, z, fnumber)` takes as input the position of the elements *elPosX*, a single x-position *xpos*, a depth matrix *z*, and the desired F-number *fnumber*, and outputs an apodization matrix for all depths. The format of the output is [channels, z].

- Plot the apodization function versus depth and channel number for `x = 0` and `fnumber = 1`. Beyond what depth is it impossible to maintain a constant F-number?
- Regenerate your image, but now use expanding aperture. Select a suitable F-number. What happens if the F-number becomes very high? Or very low? (Tip! Use a `for` loop over x-positions.)

## Task 4: In vivo data

- Generate an ultrasound image from the in vivo channel data. Select an appropriate F-number and use expanding aperture. Also select appropriate dynamic range and gain and plot your image. Use `axis equal` to ensure that the aspect of the image is correct. The acquisition is taken from the carotid bifurcation, where the carotid artery splits into two arteries: the interna, which supplies the brain, and the externa, which supplies the face.