2 TP2 : sound source localization with a microphones array : beamforming approaches

Note that some questions have to be prepared before the session. They are highlighted with a black bar on the left in the following. Please answer those questions carrefely directly in your Notebook before the session.

You have characterized and analyzed the sound propagation in the previous practical. We will now exploit theses properties to infer one sound source position w.r.t. a linear microphone array made of $N = 8$ omnidirectional MEMS microphones. The system you will be using is the same as before; thus, most of the code you already wrote to acquire signals, plot them, etc. will remain the same.

In all the following, we will work with a sampling frequency $F_s = 20\text{kHz}$, and with a buffer of size $\texttt{BLK} = 2048$.

1. To begin, start the acquisition of the audio system, and capture one audio buffer. Plot the resulting signals as a function of time.

Beamforming consists in applying a filter on each microphone signals and to sum their outputs to form the beamformer signal $y(t)$, as showed in Figure 5. As explained during the course, one

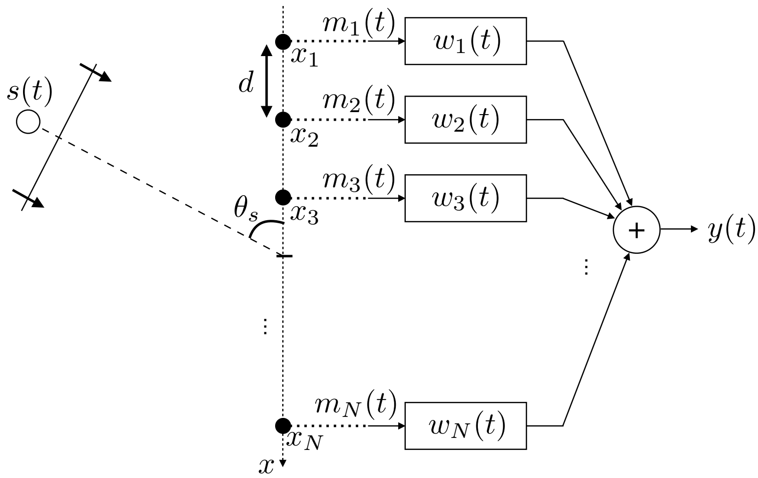

FIGURE 5 – TODO.

will use filters with a frequency response $W_n(f)$ given by

$$W_n(f) = \text{TF}[w_n(t)] = e^{-j2\pi \frac{f}{c} x_n \cos\theta_0},$$

where x_n is the position of the n^{th} microphone along its axis, and θ_0 is the angular direction in which the beamformer is focalized.

2.1 Coding the beamformer filters and analyzing their properties

These first questions have to be prepared **before** the practical session.

2. Write the position x_n as a function of n and interspace d. As a convention, the first microphone number is selected as 0, and the origin of the frame is placed at the center of the array.
3. Propose a function `beam_filter` returning the filter frequency response for one microphone number `mic_nb`. The function prototype must be :

```
def beam_filter(array, freq_vector, theta0=0, mic_nb: int = 0):
    """Compute the filter frequency response of a DSB beamformer for one
                                            microphone
```

```
    Args:
    array (array_server obj): array structure controlling the acquisition
                                      system.
    freq_vector (np.array): frequency vector.
    theta0 (int, optional): focusing angular direction (in degrees). Defaults
                                      to 0.
    mic_nb (int, optional): microphone number. Defaults to 0.

    Returns:
    np.array: the filter frequency response. Shape is (len(freq_vector),).
    """

        # Microphone position x
        x = # TO BE COMPLETED
        # Filter's frequency response
        return # TO BE COMPLETED
```

4. Plot the two frequency responses obtained for two filters associated to two different microphone outputs when $\theta_0 = 0°$ and for frequencies between 0 and 5kHz. Explain the effect of these filters on the signals.
5. Compare again the filters obtained when $\theta_0 = 90°$. Explain the differences.

2.2 Using the filters : coding of the beamforming

Basically, the beamforming algorithm is the following :

(a) acquire an audio frame
(b) compute the corresponding FFT
(c) analyze the FFT to define which frequency(ies) you would like to localize
(d) restrict the FFT to the frequencies of interest
(e) for one given θ_0, for the frequencies selected before, and for each microphone :
   — compute the corresponding filters frequency responses with the `beam_filter` function
   — apply these filters to the microphone outputs
(f) compute the beamformer output associated to the angular polarization θ_0
(g) repeat all these last steps for each θ_0 you want to test
(h) finally, decide of the angular position of the source by detecting for which θ_0 the beamformer output is maximum.

For now, we will try to localize only one frequency. We will select here $F_0 = 1$kHz the frequency of the sinus tone to localize (and emitted by your phone through one of the application listed in the introduction).

6. **Step (a) and (b) :** After acquiring an audio buffer, compute its FFT in an array `M_fft`. Plot the result of this analysis as a function of the frequency when emitting a pure sine tone with a frequency $F_0 = 1$kHz.
7. **Step (c) and (d) :** Among all the frequencies you obtained from the FFT, select the one corresponding to the source frequency. Give its exact value and index k_0 in the frequency array, and collect the corresponding FFT values of each microphone outputs in one vector `M` of length N.
8. **Step (e) :** In a loop among all microphones, compute each filters for the position θ_0 and the frequency value you obtained in the previous step. Apply then these filters to the array `M` defined before.
9. **Step (f) :** Combine then the filters outputs to form the beamformer output $Y_{\theta_0}[k_0]$. $Y_{\theta_0}[k_0]$ *is obviously a complex value which corresponds to the frequency contribution of the source*

to the k_0^{th} *frequency component of the beamformer output when focalized in the direction* θ_0. Compute then the corresponding power $P(\theta_0)$ at k_0 of the beamformer output.

10. For a direction θ_0 of your choice, compute $P(\theta_0)$ for (i) a source emitting from a direction close to θ_0, or (ii) far from it. Compare the two values.
11. **Step (g) :** Repeat now the previous code in a loop for θ_0 values ranging from 0 to 180°. You should then obtain an array P where each value corresponds to the power of the beamformer output at F_0 for each angular polarization. Plot the array P as a function of the angle θ_0.
12. **Step (h) :** Find the θ_0 value corresponding to position of the maximum in P and compare it with the actual (but approximate) position of the sound source.

2.3 Analyzing the beamformer performances

From now on, you can use your own code written in Section 2.2, or use the provided `beamformer` function which exactly reproduces the beamformer algorithm. You might then add `from beamformer import beamformer` in your Notebook before being able to use the `beamformer` function.

```python
def beamformer(buffer, theta, F0, Fs):
    """Compute the energy map from a Delay-And-Sum beamforming

    Args:
        buffer (np.array): audio buffer, of size N x BLK_SIZE
        theta (np.array): array of angular value in degrees listing the
                                                polarization angle of the
                                                beamformer
        F0 (float): source frequency to localize
        Fs (float): sampling frequency
    """
```

13. Plot the energy maps you obtain when using source frequencies $F_0 = 400$Hz, $F_0 = 1$kHz, $F_0 = 2$kHz and $F_0 = 4$kHz emitting from a fixed arbitrary position. Comment and explain carefully the differences between these curves.
14. For a frequency $F_0 = 1$kHz and a source moving aroud the array, plot the estimated position as a function of time. Comment the effectiveness of the approach and its limits.