Speech and Audio Signal Processing

Tutorial 2

Prof. Dr.-Ing. H. Puder
Adaptive Systems for Processing of Speech and Audio Signals
Graduate School Computational Engineering

The first part of the tutorial is about creating a model to generate input signals for a beamformer. We assume to operate in free field conditions throughout the entire tutorial. Since Problem 2 and Problem 3 build up on this model, you are provided with the necessary microphone signals later on if you cannot solve Problem 1. However, you are required to implement the differential beamformer of Problem 2 in order to do the application example in the last task.

Problem 1 Signal Propagation Model

The signal propagation model shall generate input signals for a beamformer with two microphones separated by a distance of d. The setup including the reference coordinate system for all angles is depicted in Figure 1.

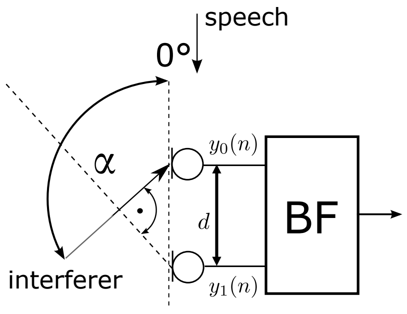


Figure 1: Signal propagation model with a beamformer consisting of two microphones.

a) Due to the microphone distance d and the angle of incidence α the sound propagation delay T_d might result in non-integer multiples of sampling period T_s. Therefore, the delay process cannot be realized solely by inserting zeros at the beginning of the signal in the delayed microphone path. Instead, write a function

```
signalDelayed = floatDelay(signal, Td, N)
```

which returns an input `signal` delayed by any arbitrary, positive, real number `Td` of samples. The integer part of T_d can still be implemented by inserting zero values. For the fractional part, use the provided `thiran`-function to calculate the **b**- and **a**-coefficients of an allpass filter of order `N` and $0 < D < 1$ samples delay. Choose $N = 30$ in your implementation. Note that the delay introduced by the Thiran filter is $D + N$. Therefore, append N zeros at the end of the signal before applying the filter. After filtering, drop N samples in the beginning of the signal to end up with the desired delay. For convenience, you should truncate `signalDelayed` so that it is of same length as the input signal.

MATLAB functions: `filter`, `thiran`

b) Now, write a function

```
[micFront, micRear] = recordingModel(signal, interferer, params)
```

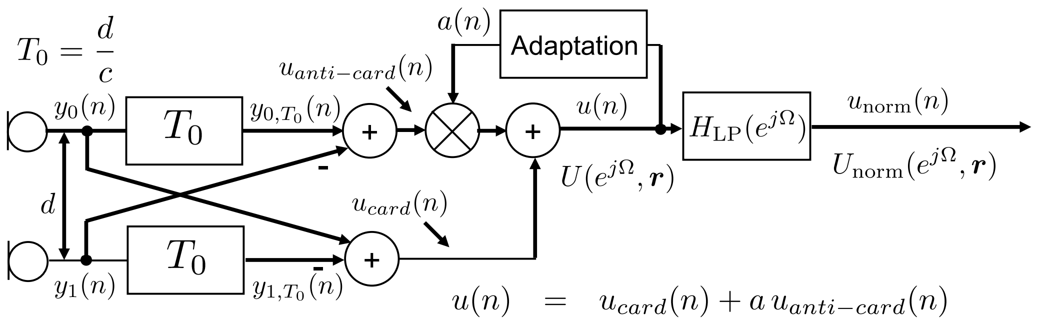
that returns two signals captured by microphones placed d meters apart from each other. The angles of incidence are given by α_s for the useful (speech) signal and α_i for the interferer. The speed of sound is considered to be constant with $c = 343\,\text{m}\,\text{s}^{-1}$. Find the difference in the traveling path Δx from the front to the rear microphone in order to calculate the required delay value.

Because the number of parameters is quite large, put them in a `struct` which you can use as function argument like `params.fs = 24000; params.c = 343; ...` The following parameters will be needed:

f_s	Sample rate (in Hz)
d	Microphone distance (in m)
g_s	Signal gain (linear)
g_i	Interferer gain (linear)
α_s	Angle of incidence of speech (in °)
α_i	Angle of incidence of interferer (in °)

Make sure to handle the cases of which microphone the wave front hits first and apply the resulting difference in propagation delay using the `floatDelay` function from the previous task. The output signals are composed of the sum of the desired signal and the interference, each weighted with its corresponding gain factor.

Problem 2 Differential Beamformer



$$
\begin{aligned}
u(n) &= u_{card}(n) + a\, u_{anti-card}(n) \\
&= y_0(n) - y_{1,T_0}(n) + a\,[-y_1(n) + y_{0,T_0}(n)]
\end{aligned}
$$

Figure 2: Signal flow chart of an adaptive differential beamformer (ADBF) with two microphones.

This second part is about the signal processing in an adaptive differential beamformer (ADBF) with a structure as given in Figure 2. For simplicity, assume the microphone distance to be $d = c/f_\text{s} = 14.29\,\text{mm}$ with the speed of sound $c = 343\,\text{m}\,\text{s}^{-1}$ and $f_\text{s} = 24\,\text{kHz}$ resulting in a single tap delay $T_0 = T_\text{s} = f_\text{s}^{-1}$.

a) Write a function

```
[uNorm, a] = diffBF(y0, y1, adaptFlag, alphaBF)
```

which implements the operations in the provided signal flow chart. It takes the two microphone signals $y_0(n)$ and $y_1(n)$, a flag to toggle the adaptation (`0`: no adaptation, `1`: adaptation active), and an angle α_BF as input arguments. The function should return the output signal $u_\text{norm}(n)$ as well as a vector containing the anti-card weights $a(n)$ for every processed sample n.

Distinguish between two cases which are controlled by the adaptation flag:

- No adaption, fixed weight, i.e., $a(n) = a$. Choose a according to the following formula where the notch direction of the beamformer is a function of α_BF.

$$a = \frac{1 + \cos(\alpha_\text{BF})}{1 - \cos(\alpha_\text{BF})} \tag{1}$$

- With adaptation, where $a(n)$ is determined using an NLMS approach according to Eq. 2

$$a(n+1) = a(n) + \mu \frac{y_1(n) - y_{0,T_0}(n)}{\sigma^2_{\mathrm{ac}}(n)} u(n) \tag{2}$$

  with

$$u_{\mathrm{ac}}(n) = y_{0,T_0}(n) - y_1(n),$$

$$\sigma^2_{ac}(n) = \beta \sigma^2_{ac}(n-1) + (1-\beta)|u_{\mathrm{ac}}(n)|^2.$$

Make sure that the weight does not exceed its restricted domain of $a(n) \in [-1,1]\ \forall\ n$. Use $\beta = 0.99$, $\mu = 10^{-4}$ and an initial value of $\sigma^2_{ac}(0) = 1$. Finally, the low-pass filter H_{LP} needs to be applied. It can be realized using MATLAB's `filter` function and should fulfill the following transfer function:

$$H_{\mathrm{LP}}(z) = \frac{1}{1 - z^{-2}}$$

MATLAB function: `filter`

Problem 3 Application Example

We now want to apply the previously implemented model using the differential beamformer of Problem 2. Load the audio files `speech_24kHz.wav` and `noise_24kHz.wav`. Then, set up a struct with all the necessary input parameters for the recording model. For convenience, you can truncate the longer signal so that they are of same length. In case difficulties arose in Problem 1, use the provided microphone signals instead of simulating them with `recordingModel.m`.

a) Use your `recordingModel`-function to create two microphone signals only containing the interferer, i.e., $g_{\mathrm{s}} = 0$, $g_{\mathrm{i}} = 1$. Assume the interferer to be located at $\alpha_{\mathrm{i}} = 120°$. Now process the resulting signals in the differential beamformer using no adaptation and $\alpha_{\mathrm{BF}} = \alpha_{\mathrm{i}} = 120°$.

   Compare the power spectral densities of one microphone signal and of the beamformer output, respectively, by plotting them in one figure. You can use Welch's method to calculate the PSD estimates.

   MATLAB function: `pwelch`

b) Continue using the microphone signals created in a) and process again without any adaptation. However, this time use $\alpha_{\mathrm{BF}} = 160°$ which means that the beamformer will be now slightly off the original interferer direction. Compare the PSD of the output signal to the case where α_{BF} was chosen appropriately.

c) Create two microphone signals as in a), this time with both, interferer and speech signals as input. Choose the following parameters: $g_{\mathrm{s}} = 1$, $g_{\mathrm{i}} = 1$, $\alpha_{\mathrm{s}} = 0$, $\alpha_{\mathrm{i}} = 120°$.

   Process the signals in the beamformer with adaption inactive and $\alpha_{\mathrm{BF}} = 120°$. Listen to the output of the beamformer and compare it to an unprocessed microphone signal. Also, save the returned values of a for comparison.

d) Process the same microphone signals using the ADBF, i.e., with adaption of $a(n)$. Listen to the output again. Can you notice any differences?

   Lastly, plot a over time, both for the fixed and the unfixed case in one figure and compare the results.