Simulation of a phased array algorithm

Lucas Åkerstedt, Mika Söderström

July 13, 2022

1 Introduction

A proposed beamforming algorithm is tested in an environment to see if the algorithm performs as expected, and also to make improvements. The algorithm works with a corresponding phased array antenna, defined by the user. Furthermore, the data that the beamforing algorithm acts upon is generated by another algorithm that is built on a similar principle, although not the exact same.

2 Generation of the data

2.1 The geometry of the environment

In order to evaluate the performance of the beamforming algorithm, we must first generate an environment that has all the necessary components for the beamforming algorithm to work. This means that we must create an environment with an existing array antenna and sources for the array antenna to sense. Such an environment can be seen in Figure 1.

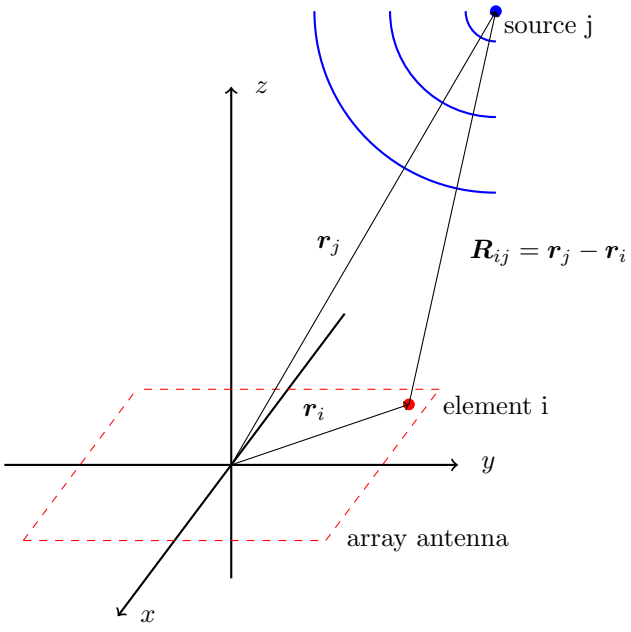

Figure 1: Location of the j:th source relative to the array antenna and its i:th element.

Every source's location is defined by two angles (θ and φ according to a spherical coordinate system) and a distance ρ from the origin. The emitted signal from the sources are defined by a discrete sum of sine waves with frequencies defined by the user. For example, the user could define a source's signal as a sum of two sine waves, one at 440 Hz, and the other at 659 Hz. The wave velocity of the emitted spherical waves can also be determined by the user, however, no relativistic effects are accounted for.

2.2 Generating signals

On each antenna element of the array antenna, there will be a signal generated. The signal generated will be dependent on how many sources there are, the frequencies of the sine waves emitted from the sources, and the distance between the antenna element and the sources. The distance $|\boldsymbol{R}_{ij}|$ between the i:th antenna element

and the j:th source will have an effect on the amplitude and the phase of the generated signal on the i:th antenna element. The propagation of the waves generated by the sources are expected to behave as spherical waves

$$\frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r}-\boldsymbol{r}'|}, \tag{1}$$

where the wave vector k is defined as $k = \omega/c$, where c is the wave velocity and ω is the angular frequency of the wave.

Each source will generate a signal that from its perspective would appear as

$$y_j(t) = \sum_{g=1}^{G} \sin(2\pi f_g t), \tag{2}$$

where the index g relates to the frequencies of the sine waves the complete signal consists of. We can then use this model together with the spherical wave 1 to calculate the generated signals on each antenna element

$$x_i(t) = \sum_{j=1}^{J}\sum_{g=1}^{G} \frac{1}{|\boldsymbol{r}_j - \boldsymbol{r}_i|} \sin(2\pi f_{jg} t - k_g|\boldsymbol{r}_j - \boldsymbol{r}_i|). \tag{3}$$

There is a slight modification that has to be done to this model to work with the proposed beamforming algorithm. The signal has to be a discrete signal rather than a time-continuous one. We can easily create a discrete signal by sampling the time-continous signal

$$x_i[n] = \sum_{j=1}^{J}\sum_{g=1}^{G} \frac{1}{|\boldsymbol{r}_j - \boldsymbol{r}_i|} \sin(2\pi f_{jg} t[n] - k_g|\boldsymbol{r}_j - \boldsymbol{r}_i|), \tag{4}$$

where $t[n]$ is the discrete time, determined by a sampling frequency f_s.

In Figure 2 we can see two cases of when signals are generated with the model of 4.

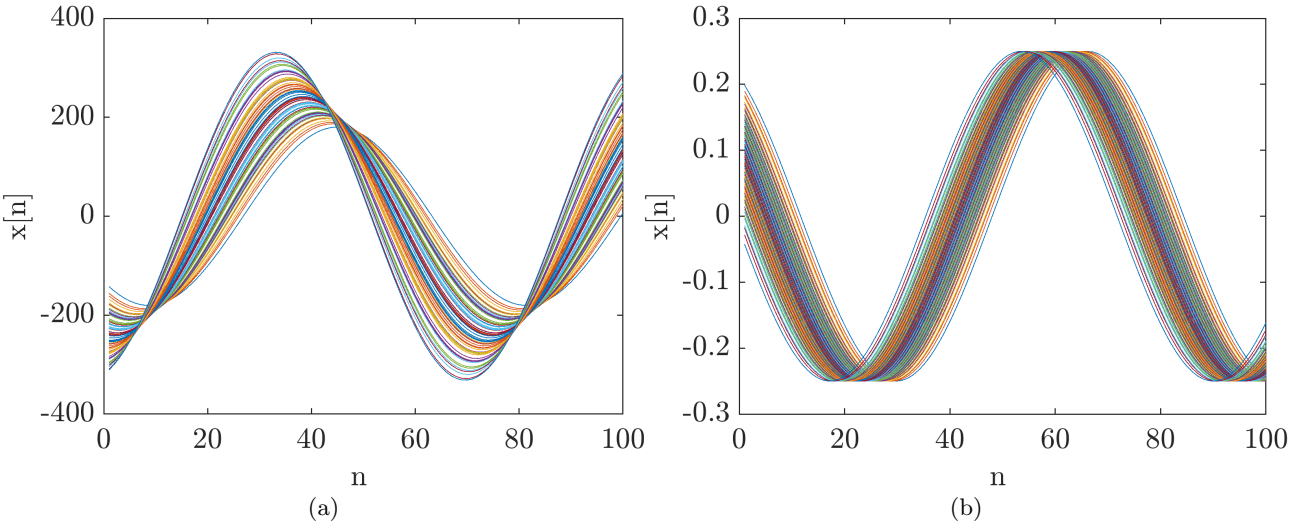

Figure 2: Generated signals on an eight by eight uniform array located in the origin and the xy-plane, where the source is at a distance of 0.2 m from the origin and at an angle of $\theta = \frac{\pi}{4}$ and $\varphi = \frac{\pi}{6}$, in (a). The generated signals on the same array as in (a), but with the source located 200 m from the origin and at an angle of$\theta = \frac{\pi}{4}$ and $\varphi = \frac{\pi}{6}$ in (b).

3 The beamforming algorithm

3.1 Theory

The proposed beamforming algorithm is based upon the far-field approximation of spherical waves. This approximation will coincide well if we have the following conditions met: $kr \gg 1$ and $r \gg r'$. We can thus approximate the spherical wave in 1 as following:

$$|\boldsymbol{r}-\boldsymbol{r}'| = \sqrt{(\boldsymbol{r}-\boldsymbol{r}')\cdot(\boldsymbol{r}-\boldsymbol{r}')} = \sqrt{r^2 + (r')^2 - 2\boldsymbol{r}\cdot\boldsymbol{r}'} = r\sqrt{1 + \frac{(r')^2}{r^2} - 2\frac{\hat{\boldsymbol{r}}\cdot\boldsymbol{r}'}{r}}, \tag{5}$$

expanding the contents under the square root with the small argument Taylor expansion

$$(1+\varepsilon)^p \approx 1 + p\varepsilon + \mathcal{O}\left(\varepsilon^2\right) \tag{6}$$

we obtain

$$|\boldsymbol{r} - \boldsymbol{r}'| \approx r\left(1 - \frac{\hat{\boldsymbol{r}}\cdot\boldsymbol{r}'}{r} + \frac{1}{2}\frac{(r')^2}{r^2}\right) = r - \hat{\boldsymbol{r}}\cdot\boldsymbol{r}' + \mathcal{O}\left(\frac{1}{r}\right), \tag{7}$$

which yields

$$\frac{e^{-jk|\boldsymbol{r}-\boldsymbol{r}'|}}{|\boldsymbol{r}-\boldsymbol{r}'|} \approx \frac{1}{r}e^{-jkr}e^{jk\hat{\boldsymbol{r}}\cdot\boldsymbol{r}'}. \tag{8}$$

For larger distances we can see, from 8, that the signals generated on each antenna element has the same amplitude as well as a phase-shift only dependent on the direction of the incoming signal, the location of the antenna element, and the angular frequency of the wave. This means that we can add element-unique phase-shifts to the generated signals on each element to obtain a constructive interference in a desired direction. In other words, we can listen in a certain direction. For know, however, we can only listen for narrowband signals (single frequencies). To obtain a wide-band solution, we must filter our incoming signals into multiple small bands, and then apply our frequency dependent phase-shifts.

From 8 we can also see that if we want constructive interference in the $\hat{\boldsymbol{r}}$ direction, we need to multiply with a phasor p such that the element-unique signals are in phase for every frequency band. One solution is

$$p = e^{-jk\hat{\boldsymbol{r}}\cdot\boldsymbol{r}'_i}. \tag{9}$$

In time domain, this phase shift is simply $-k\hat{\boldsymbol{r}}\cdot\boldsymbol{r}'_i$. Performing this element-unique phase-shift would grant constructive interference in the desired direction.

To perform a phase-shift in the time domain, we need some kind of filter. Since we are working with discrete signals, we will need a digital filter. To create such a filter, we first start with a trigonometric relation

$$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta. \tag{10}$$

Let $\alpha = 2\pi\nu n$ and $\beta = \phi_0$ and we instead have

$$\sin(2\pi\nu n + \phi_0) = \sin(2\pi\nu n)\cos\phi_0 + \cos(2\pi\nu n)\sin\phi_0. \tag{11}$$

We then rewrite $\cos(2\pi\nu n)$ as $1/(2\pi\nu)\frac{\mathrm{d}}{\mathrm{d}n}\sin(2\pi\nu n)$

$$\sin(2\pi\nu n + \phi_0) = \sin(2\pi\nu n)\cos\phi_0 + \frac{\sin\phi_0}{2\pi\nu}\frac{\mathrm{d}}{\mathrm{d}n}\sin(2\pi\nu n). \tag{12}$$

We can approximate the derivative in 12 with numerical differentiation

$$\sin(2\pi\nu n + \phi_0) = \sin(2\pi\nu n)\cos\phi_0 + \frac{\sin\phi_0}{2\pi\nu}\frac{1}{2}\left[\sin(2\pi\nu(n+1)) - \sin(2\pi\nu(n-1))\right]. \tag{13}$$

If we thus have a discrete sine signal $x[n] = \sin(2\pi\nu n)$ and we want to phase-shift it ϕ_0, we can thus utilize the derived difference equation

$$y[n] = x[n]\cos\phi_0 + \frac{\sin\phi_0}{4\pi\nu}\left(x[n+1] - x[n-1]\right). \tag{14}$$

From here, we would like to further specify the phase shift ϕ_0 to be applied in order to get constructive interference in a desired direction. We do this by calculating the dot product in 9. For this beamforming algorithm, the array (or arrays) are assumed to lie in the xy-plane (although changing this is not a problem). We can thus express our element vector $\boldsymbol{r}'_i$ as

$$\boldsymbol{r}'_i = x_i\hat{\boldsymbol{x}} + y_i\hat{\boldsymbol{y}}. \tag{15}$$

Its worth noticing that we use the same index i on both the x coordinate as well as the y coordinate. This is because in this algorithm, we stack our element vectors into a matrix $\boldsymbol{r}'$. This means that we can still have a 2D array, although we only have one index i.

$$\boldsymbol{r}' = \begin{bmatrix} x_1 & x_2 & \dots & x_M \\ y_1 & y_2 & \dots & y_M \\ 0 & 0 & \dots & 0 \end{bmatrix}, \tag{16}$$

where M indicates the last element in the array.

We now only need to perform the dot product between 15 and $\hat{\boldsymbol{r}}$

$$\boldsymbol{r}_i' \cdot \hat{\boldsymbol{r}} = (x_i\hat{\boldsymbol{x}} + y_i\hat{\boldsymbol{y}}) \cdot (\sin\theta\cos\varphi\hat{\boldsymbol{x}} + \sin\theta\sin\varphi\hat{\boldsymbol{y}} + \cos\theta\hat{\boldsymbol{z}}) = x_i\sin\theta\cos\varphi + y_i\sin\theta\sin\varphi. \tag{17}$$

The frequency dependent phase shift to apply thus becomes

$$k\left(x_i\sin\theta\cos\varphi + y_i\sin\theta\sin\varphi\right) = \phi_i. \tag{18}$$

We have thus derived an element-unique phase shift (18), and the means to apply that phase shift (14).

3.2 Beamforming of recorded data

3.2.1 Basic idea

When performing the beamforming algorithm, there are mainly two ways of operating: Scanning the whole space to determine in what direction there could be a source, or to simply perform the beamforming algorithm in one direction to enhance the signal coming from that direction. In this section, we will explain how the algorithm works in the simplest case; with recorded data.

We start with the signals from the 2D array

$$\begin{bmatrix} x_1^1 & x_1^2 & ... & x_1^M \\ ... & ... & ... & ... \\ x_{n-2}^1 & x_{n-2}^2 & ... & x_{n-2}^M \\ x_{n-1}^1 & x_{n-1}^2 & ... & x_{n-1}^M \\ x_n^1 & x_n^2 & ... & x_n^M \end{bmatrix}, \tag{19}$$

where x_n^i indicates the n:th sample recorded from element i. Every column of 19 thus refer to one discrete signal of one element. We can rewrite our data matrix into a row vector of column vectors (a matrix)

$$\begin{bmatrix} \boldsymbol{x}^1 & \boldsymbol{x}^2 & ... & \boldsymbol{x}^M \end{bmatrix}. \tag{20}$$

In 20, every element ($\boldsymbol{x}^1$, $\boldsymbol{x}^2$ etc.) represents the complete signal recorded from one element. From here, we want to filter every element signal into narrow band signals. If we want to filter our signals into 20 narrowband signals, we need to filter every element signal 20 times. A block diagram can be seen in Figure 3.

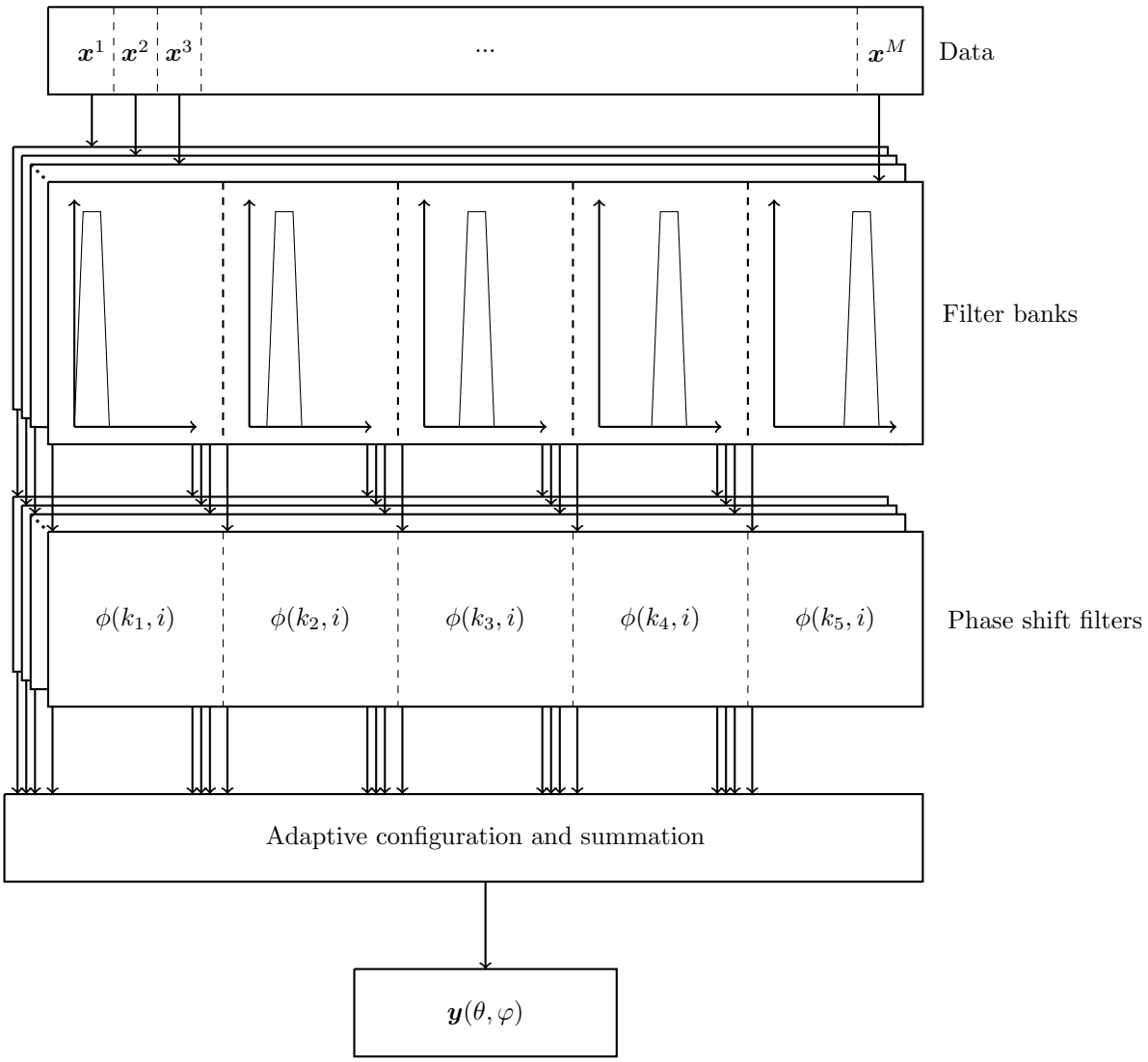

Figure 3: Block diagram of beamforming algorithm acting on recorded data. In this example, the data is filtered in five bands. The output signal is the spatially filtered signal $x[n](\theta, \varphi)$.

The algorithm in Figure 3 will only spatially filter the signal in one direction. This means that the output $\boldsymbol{y}(\theta, \varphi)$ will be dependent on how the phase filters are configured. This algorithm has to run for multiple directions in order to scan the whole space.

3.2.2 Filter banks

Since we need to filter our signals into many small band signals, we will use a filter bank. What this means is that we simply will apply many narrow band-pass filters. We will also only use FIR-filters to avoid resonance (finite impulse response).

The easiest and most convenient way of designing the filter bank is to use the fir1 filter (see matlab's description [1]) and to space out the cut-off frequencies linearly in the desired frequency band.

Applying these digital FIR-filters can be done in time domain

$$y[n] = \frac{1}{a_0} \sum_{i=0}^{P} b_i x[n-i], \tag{21}$$

where $y[n]$ is the output signal, $x[n]$ is the input signal, a_0 is the magnitude of the output signal, b_i is the filter coefficients, and P is the filter order. The filter coefficients can be obtained using Matlab (see fir1 function). When using the fir1 function in Matlab, one has to define the filter order, and the two cut-off frequencies (we want a band-pass filter). What is recommended to do is to define a center frequency for the band-pass filter, and then to define the cut-off frequencies as $\pm$ 1 % of the center frequency.

As can be seen in Equation 21, the output signal is a scalar value, and not a vector. We thus perform the arithmetic on the right hand side of 21 for every n, where the amount of operations depends on how many

samples long our recorded data is. We can also see that there will be problematic to output the first filtered output $y[1]$. When we filter the sampled signal $x[n]$, we rely on older samples. In the very beginning, there may not exist any older sample/samples. We thus has to make the best out of the situation, and use the oldest samples we can get our hands on.

3.2.3 Phase shift filters

The phase shift filters can be realized staright from the theory in 14 and 18. To perform phase shift filtering as the block diagram in Figure 3 suggests, one must however create a phase shift function that can take in the two parameters k and i. These two parameters refer to the frequency of which the signal has been filtered to (k), and which antenna element the signal is coming from (i). Furthermore, the phase filter function must know the direction (θ,φ) to spatially filter the incoming signal.

The phase shifting filter will also need to know the normalized frequency ν, which means that the sampling frequency f_s must be known.

3.2.4 Adaptive configuration

Adaptive configuration is a weighting-technique that enhances the beamforming algorithms precision at lower frequencies. It works by not adding the signal from some antenna elements when the signal has been filtered with a band-pass filter with low cut-off frequencies. How to choose which antenna element to ignore and when is at this stage decided by a simple look-up table. First, if the wavelength of the frequency is too long compared to the distance between the antenna elements, a certain configuration mode will be activated. This configuration mode will tell which element to stay on, and which to be turned off.

These configuration modes are heavily dependent on the geometry of the array antenna/antennas used. For further reading, see [2].

3.2.5 Multiple directions

If we want to perform the beamforming algorithm in Figure 3 for multiple angles to sweep a surface, there are some things to take in mind.

Usually, we want to sweep over a surface in a Cartesian coordinate system. We then have to make some sort of translation when we use the beamforming algorithm, since it only takes in the angle-parameters θ and φ, see Figure 4.

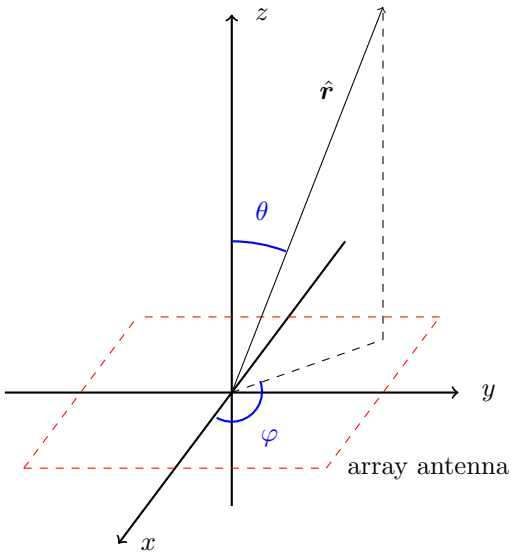

Figure 4: The beamforming listening direction $\hat{\boldsymbol{r}}(\theta,\varphi)$ and its relation to the Cartesian coordinate system.

What we mathematically do here is to project a rectangular cut-out of a sphere onto the xy-plane, see Figure 5.

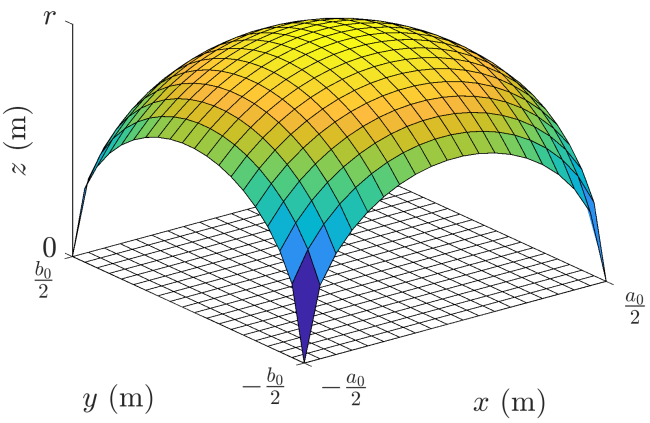

Figure 5: The rectangular cut-out of a sphere projected onto the xy-plane.

We define the rectangle as

$$-\frac{a_0}{2} \leq x \leq \frac{a_0}{2}, \quad -\frac{b_0}{2} \leq x \leq \frac{b_0}{2}. \tag{22}$$

We further need to define a radius to the sphere we cut from. This radius will have some constraints

$$\frac{1}{2}\sqrt{a_0^2 + b_0^2} \leq r. \tag{23}$$

We now have everything we need to translate the Cartesian coordinates to spherical coordinates

$$\theta = \arccos\left(\frac{\sqrt{r^2 - x^2 - y^2}}{r^2}\right), \quad \varphi = \arctan\frac{y}{x}. \tag{24}$$

The translation for φ will need some modification when used in a function since the arctan-function will only yield values in the region $\frac{-\pi}{2}$ to $\frac{\pi}{2}$. Instead when implementing a function, use `atan2(y,x)`.

With the proposed translation, it is possible to use the beamforming algorithm for every coordinate in a grid. Doing so yields a data matrix consisting of many output signals $\boldsymbol{y}$ for every coordinate in this grid

$$\begin{bmatrix} \boldsymbol{y}_{1,1} & \boldsymbol{y}_{1,2} & \cdots & \boldsymbol{y}_{1,B} \\ \boldsymbol{y}_{2,1} & \boldsymbol{y}_{2,2} & \cdots & \boldsymbol{y}_{2,B} \\ \cdots & \cdots & \cdots & \cdots \\ \boldsymbol{y}_{A,1} & \boldsymbol{y}_{A,2} & \cdots & \boldsymbol{y}_{A,B} \end{bmatrix}, \tag{25}$$

where A is the amount of rows in the grid, and B is the amount of columns in the grid. If we for every entry in matrix 25 take the dot product of the entry and itself (see Hadamard product, $\boldsymbol{y}_{1,1} \cdot \boldsymbol{y}_{1,1}$ etc.) we get a new matrix, where every entry would be proportional to the intensity in the direction represented by the entry

$$\boldsymbol{I}^{bf} = \begin{bmatrix} I_{1,1} & I_{1,2} & ... & I_{1,B} \\ I_{2,1} & I_{2,2} & ... & I_{2,B} \\ ... & ... & ... & ... \\ I_{A,1} & I_{A,2} & ... & I_{A,B} \end{bmatrix}. \tag{26}$$

This intensity map $\boldsymbol{I}^{bf}$ is something we can plot to get perception of where in the space there can be sources.

3.2.6 Post processing

It is possible to enhance the intensity map $\boldsymbol{I}^{bf}$ such that the location of the sources is much more clear. To do this, We define the following matrices

$$C^x_{a,b} = (I^{bf}_{a,(b+1)} - I^{bf}_{a,(b-1)})/x_{step}, \quad 1 \leq a \leq A, \quad 1 < b < B, \tag{27}$$

$$C^y_{a,b} = (I^{bf}_{(a+1),b} - I^{bf}_{(a-1),b})/y_{step}, \quad 1 < a < A, \quad 1 \leq b \leq B, \tag{28}$$

$$D_{a,b} = 1/|C^x_{a,b} + C^y_{a,b}|, \quad 1 < a < A, \quad 1 < b < B, \tag{29}$$

$$E_{a,b} = -\left[(C^y_{(a+1),b} - C^y_{(a-1),b})/y_{step} + (C^x_{a,(b+1)} - C^x_{a,(b-1)})/x_{step}\right], \quad 2 < a < A-1, \quad 2 < b < B-1, \tag{30}$$

where a and b are row indices and column indices respectively, x_{step} is the distance between the points in the grid in the x-direction, and y_{step} is the distance between the points in the grid in the y-direction. Now to enhance the intensity map $\boldsymbol{I}^{bf}$ we perform the following operation

$$F_{a,b} = I_{a,b}^{bf} D_{a,b} E_{a,b}, \tag{31}$$

where $\boldsymbol{F}$ is the enhanced intensity map.

3.2.7 Performance

We test the algorithm with the following settings:

Table 1: Source settings

Source	f_{start} (Hz)	f_{end} (Hz)	$N_{\text{sine waves}}$	θ (deg)	φ (deg)	ρ (m)	t_{start} (s)	t_{end} (s)
1	2600	3300	40	25	180	5	0	0.35
2	2700	2900	40	20	0	5	0.15	0.5

Table 2: Array settings

Array	$\boldsymbol{r}_a$	columns	rows	uniform distance (m)
1	(0,0)	8	8	0.02

Table 3: Beamforming settings

f_s (Hz)	t_{start} (s)	t_{end} (s)	a_0 (m)	b_0 (m)	r (m)	x_{step} (m)	y_{step} (m)	adaptive config.
16000	0	0.5	2	2	$\sqrt{2}$	0.067	0.067	ON

This yields the result seen in Figure 6.

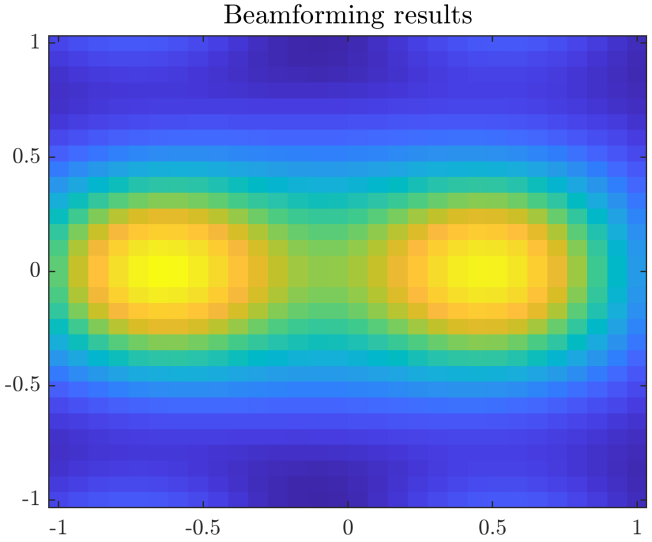

Figure 6: The normalized intensity map generated by the beamforming algorithm with the settings in Table 3.

The normalized enhanced intensity map and the correct locations of the sources can be seen in Figure 7.

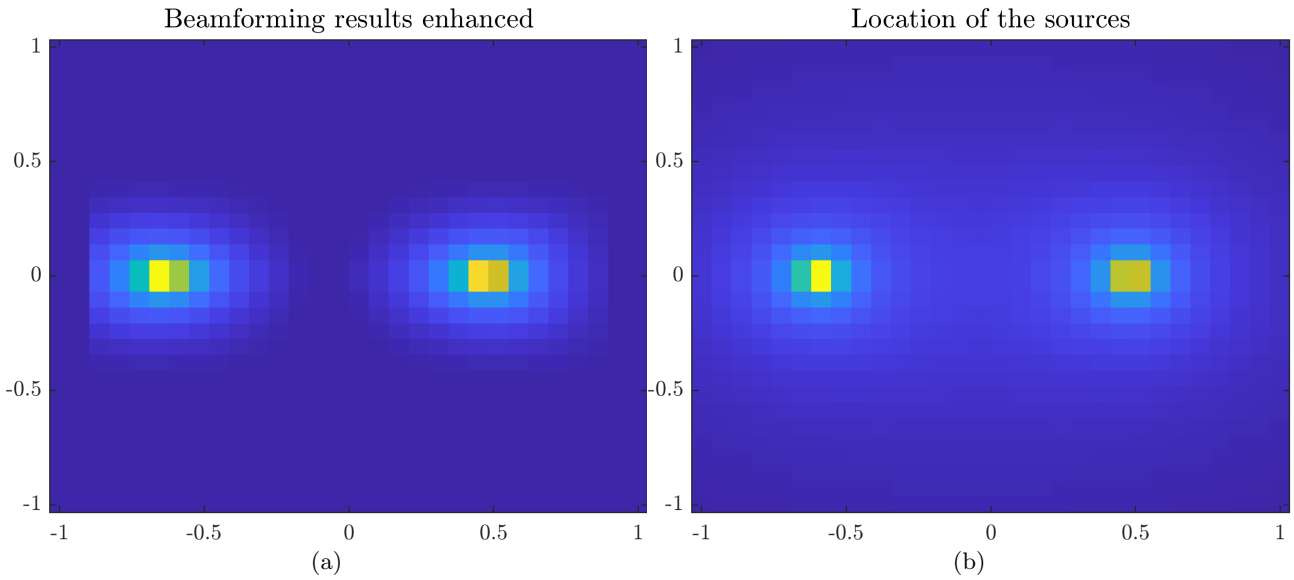

Figure 7: The enhanced intensity map in (a), and the correct location of the sources in (b).

Letting an island-finder function operate on the enhanced intensity map yields two islands with corresponding angles $\theta_1 = 26°$, $\varphi_1 = 180°$, $\theta_2 = 20°$ and $\varphi_2 = 0°$.

Listening in the direction of source 2 we get the result seen in Figure 8 (b).

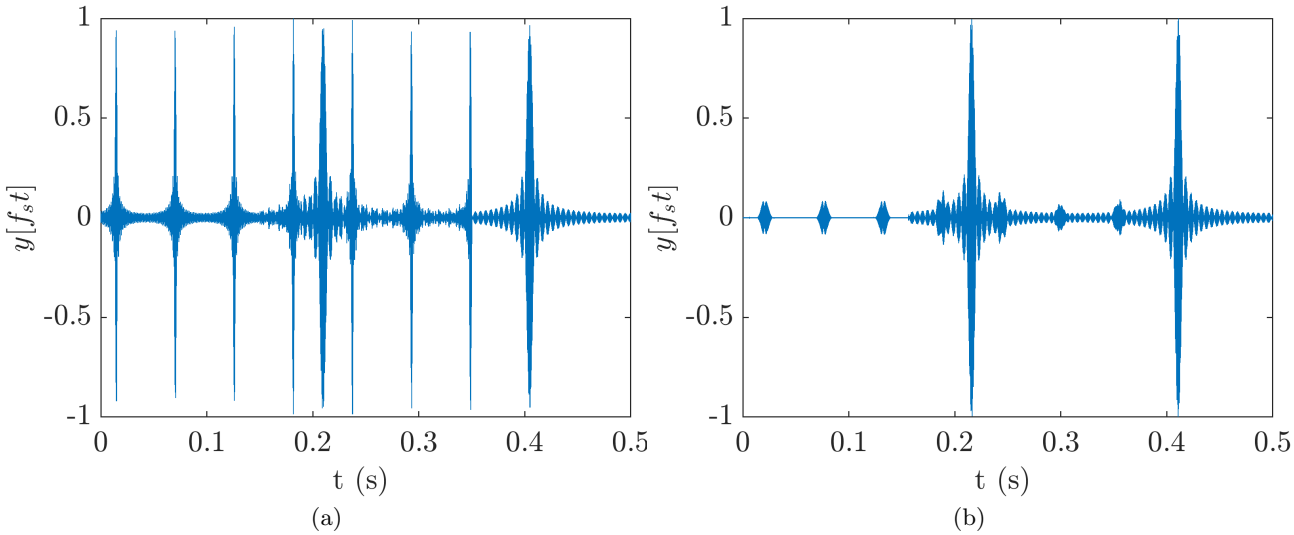

Figure 8: The raw signal received by the first antenna element in (a). The signal processed by the single direction beamforming algorithm in (b).

3.3 Real-time beamforming

The main difference between real-time beamforming and beamforming performed on recorded data is that the real-time beamforming algorithm outputs one single scalar value (one sample) every time it runs. The real-time beamforming algorithm also does not need as much data as the beamforming algorithm performing on recorded data.

The real-time beamforming algorithm will need some recorded samples from the past. How many samples from the past that is needed depends on the filter order of the filter banks. If the filter banks use filters of order 20, the 20 latest samples will have to be stored, for every time it runs.

If we take a look at the difference equation we utilize to perform the phase shift (Equation 14), we see that the output $y[n]$ is dependent on $x[n+1]$. Since we can not see into the future, we will instead have to consider $n+1$ the latest sample and output a sample that corresponds to the second latest samples. Our real-time beamforming algorithm will therefor have a delay of one sample.

3.3.1 Data management

For every time the real-time beamforming algorithm runs, the input data will be very similar, although the structure of the input data will be completely different. If x_n^i indicates the sample recorded from element i at the time $t = t_0$, then our data matrix will look like

$$\begin{bmatrix} x_{n-P}^1 & x_{n-P}^2 & ... & x_{n-P}^M \\ ... & ... & ... & ... \\ x_{n-2}^1 & x_{n-2}^2 & ... & x_{n-2}^M \\ x_{n-1}^1 & x_{n-1}^2 & ... & x_{n-1}^M \\ x_n^1 & x_n^2 & ... & x_n^M \end{bmatrix}, \tag{32}$$

where P is the filter order of the filters used in the filter bank. Then for the next run, we sample all the elements at the time $t = t_0 + T_s$, where T_s is the sample period $(1/f_s)$, we get the data matrix

$$\begin{bmatrix} x_{n-P+1}^1 & x_{n-P+1}^2 & ... & x_{n-P+1}^M \\ ... & ... & ... & ... \\ x_{n-1}^1 & x_{n-1}^2 & ... & x_{n-1}^M \\ x_n^1 & x_n^2 & ... & x_n^M \\ x_{n+1}^1 & x_{n+1}^2 & ... & x_{n+1}^M \end{bmatrix}, \tag{33}$$

where $n + 1$ now corresponds to the latest sample. For every run we thus need to "shift up" the former samples until enough time has past and we don not need those samples any longer.

3.3.2 Real-time filtering

All of the filtering stages (filter banks and phase shift filtering) will only output one scalar value every time the algorithm runs.

3.4 Parallelisation

References

[1] MATLAB, *version 9.11.0 (R2021b)*. Natick, Massachusetts: The MathWorks Inc., 2021.

[2] L. Åkerstedt and M. Söderström, "2d phased arrays," 2022.