QUESTIONS:

The plots below show two example sinc pulses. How will you generate pulses that look like this? Which parameter should be changed? What effect do you think this will have on the final image?

The parameter SincP controls the width of the pulse. Increasing SincP narrows the plot. Specifically, altering the value of SincP in the code will affect the sinc function by modifying the number of zero crossings within the given time range.

For instance, if we increase SincP from 1.0 to 2.0, the sinc function's main lobe will become narrower, resulting in a faster oscillation. Conversely, reducing SincP to, say, 0.5, will widen the pulse, making the oscillations slower.

The provided code generates a sinc pulse waveform, as depicted in the plots. The way it does is:

Sinc Function:
The heart of the waveform generation lies in the wsrc(t) function, which implements the sinc function. The np.sinc() function in Python's NumPy library calculates the sinc function for the given input. The sinc function is defined as $\sin(\pi x)/(\pi x)$.

Time and Waveform Generation:
The code initializes t as time, time as an empty list for storing time values, and wave as an empty list for storing the waveform values. It then generates the waveform over time by iterating through a specified number of samples (Nsamp) and calculating the waveform values using the wsrc(t) function.
For each sample, it appends the time value (t) to the time list and the corresponding waveform value (calculated using wsrc(t)) to the wave list. The t value increments based on the distance per sample and the speed of sound, ensuring a sequence of time values over which the waveform is generated.

Plotting:
After generating the time and waveform values, the code uses Matplotlib to visualize the waveform. It plots the waveform against time using plt.plot(), with labels for the axes, a title for the plot, and grid lines for better visualization.

Does it make sense to reconstruct up toNsamp? What value is more reasonable as an upper limit for the x-axis here?

We need to reconstruct only up to Nsamp/2 as upper limit, because after reflection, if we go beyond that distance, it will be beyond Nsamp and so it wont be recorded in the sampling, as reflection would be Nsamp.

Regarding the choice of the upper limit:
Reconstructing up to Nsamp: Reconstructing up to Nsamp involves considering the entire sampled signal. While this captures the complete temporal information, in certain scenarios, reconstructing the entire signal might contain unnecessary data beyond the required analysis range.
Reconstructing up to Nsamp/2: Limiting the reconstruction to Nsamp/2, especially in applications such as obstacle detection or localization, can be more relevant. Here's why: Given that the round-trip distance traveled by a signal involves reaching an obstacle and then returning to the source, a reconstruction up to Nsamp/2 aligns with this perspective.
Taking into account the time taken for the signal to reach the obstacle and return, the relevant data for analysis falls within this range. Beyond Nsamp/2, the delayed signals might exceed the sampling range, and any information about the return trip would lie outside the sampled data.

The (x, y) coordinates corresponding to the maximum amplitude (yellow colour) is approximately (30, 22). Explain why this is the correct expected position for the given obstacle.

30,22 represents the sample index and mic index, since dist per sample is 0.1, 30 indicates an x position of 3, and the position of mic 22 is around -1, so it gives 3,-1 which is the obstacle coordinates.

The coordinates (30, 22) correspond to the maximum amplitude, represented by the yellow color. These coordinates signify the intersection of the sample index and the microphone index in the analysis.

Therefore, in this context:
The x-coordinate (30) implies a position of 3 units along the x-axis.
The y-coordinate (22) indicates the 22nd microphone in the array or the relative position of the signal received at that index.
Interpreting the coordinates (30, 22) in terms of the physical setup, with a dist per sample of 0.1, the x-coordinate 30 translates to an x-position of 3, while the y-coordinate 22 represents the microphone index.

If the microphone indices are aligned with specific positions or angles, and considering the distance interpretation per sample, the coordinates (30, 22) represent an expected position of an obstacle. For example, assuming the mic indices align with negative y-values and considering the x-position of 3, it might suggest that the obstacle is at position (3, -1).

Therefore, the analysis concludes that the obstacle is expected to be located at coordinates 3 along the x-axis and approximately -1 on the y-axis, based on the specific interpretation of the sample and microphone indices within the context of the setup.

What is the maximum obstacle x- and y- coordinate that you can use and still have an image reconstructed?

For it to be detected the y coordinate must not go beyond the mics and also the x coordinate must be such that the total time of travel of waves does not exceed the sampling duration. Since we take 200 samples with distance per sample of 0.1, the point roughly should not go beyond a circle of radius 0,0 centred at origin.

To determine the maximum obstacle x- and y-coordinates that still allow for a reconstructed image within the sampled data, several constraints need consideration:

Y-coordinate constraints:
The y-coordinate of the obstacle must remain within the range of the available microphone array. If the obstacle's y-coordinate exceeds the maximum or falls below the minimum microphone index, the signal data won't be captured by the microphones, rendering detection impossible along the y-axis.

X-coordinate constraints:
Considering the distance per sample of 0.1 and the total of 200 samples, the maximum distance the signal can travel is the product of these two values. In this case, the sampled area is roughly within a circle of radius 20 units (200 samples * 0.1 distance per sample), centered at the origin (0,0).

What happens if C is different - if C is decreased it looks like the image becomes sharper. Can you explain why intuitively?

If C is decreased, wavelength decreases, and since resolving power is inverse to wavelength, it increases.

When the speed of sound (C) is decreased in the context of signal processing, especially in scenarios like imaging or obstacle detection, it can lead to an apparent improvement in image sharpness. This change can be understood in terms of the relationship between the speed of sound, wavelength, and resolving power.

Effect on Wavelength:
When the speed of sound is decreased, the wavelength of the signal decreases as well. Wavelength is inversely proportional to the speed of sound in a medium. So, as C decreases, the distance covered by one complete cycle of the signal decreases.

Impact on Resolving Power:
The resolving power of a system or sensor, in this case, an array of microphones, is inversely related to the wavelength. Smaller wavelengths generally lead to higher resolving power.
With a decreased wavelength, the system can distinguish between finer details in the received signals. In imaging scenarios, this effectively means better differentiation between different points or features, leading to a sharper image.

What happens if Nmics is increased or decreased?

If Nmics is increased, the range in y axis increases and also, we will get a more precise object location.

Increasing the number of microphones (Nmics) in an array setup has two significant impacts on the imaging or signal processing scenario, specifically in applications like obstacle detection or spatial localization:

Expanded Range in Y-axis:
As the number of microphones increases, the coverage or span along the y-axis expands. Each microphone in the array represents a discrete point or region where the signal is received or captured. Therefore, with more microphones, the coverage along the y-axis widens. This expansion implies a broader range of detection or reception capabilities, allowing for a larger area to be covered along the y-axis in the imaging process.
Increased Precision in Object Location:
More microphones provide higher spatial resolution. The greater number of discrete points at which the signal is received allows for more precise triangulation or localization of the object or obstacle.
Therefore, by increasing the number of microphones in the array setup, we not only extend the range of coverage along the y-axis but also enhance the system's ability to precisely pinpoint and locate objects or sources within that expanded coverage.

Do the experiments with Nmics=[8,32,64] and Nsamp = [50, 100, 200] (all combinations). Attach the resulting images.

Plots and Images obtained:

Here are the plots attached for (64, 100) combination.

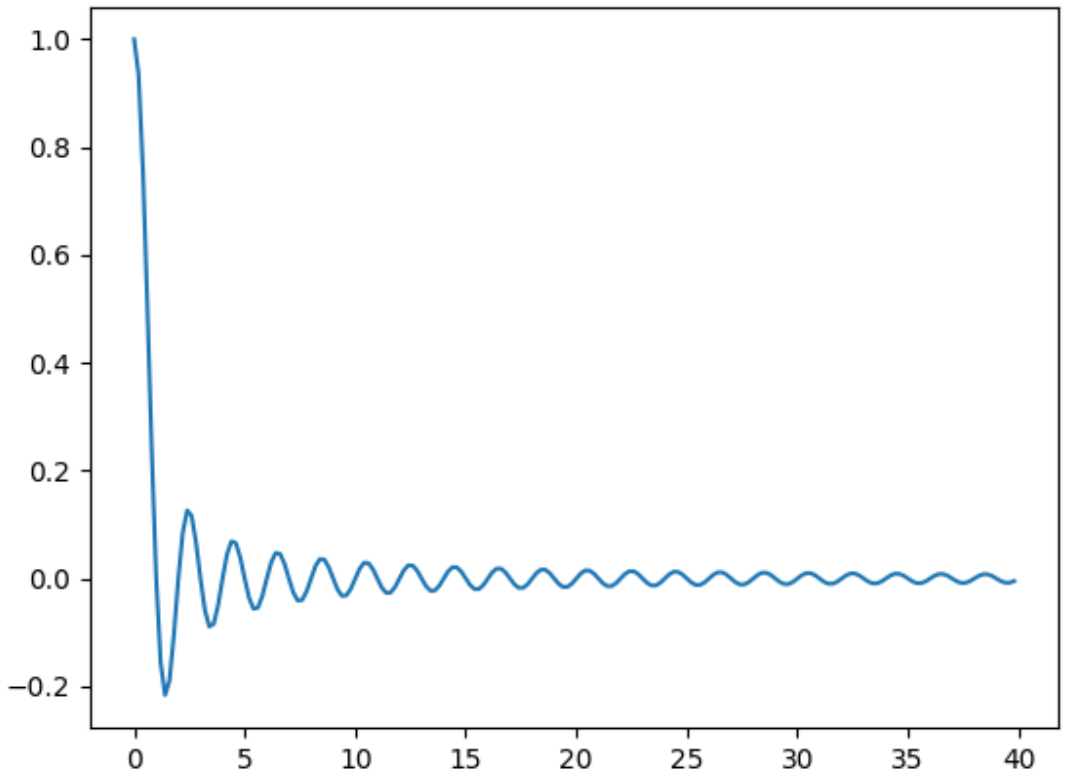


Image 1

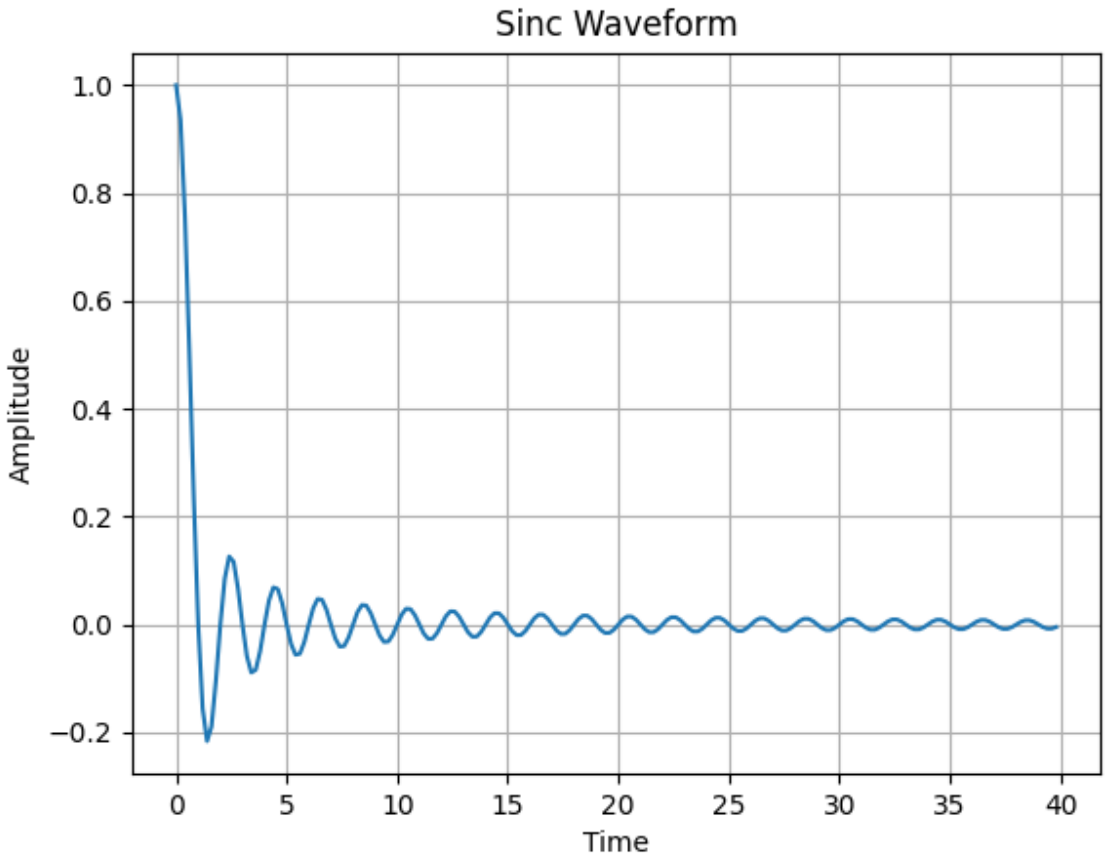


Image 1 (with a grid)

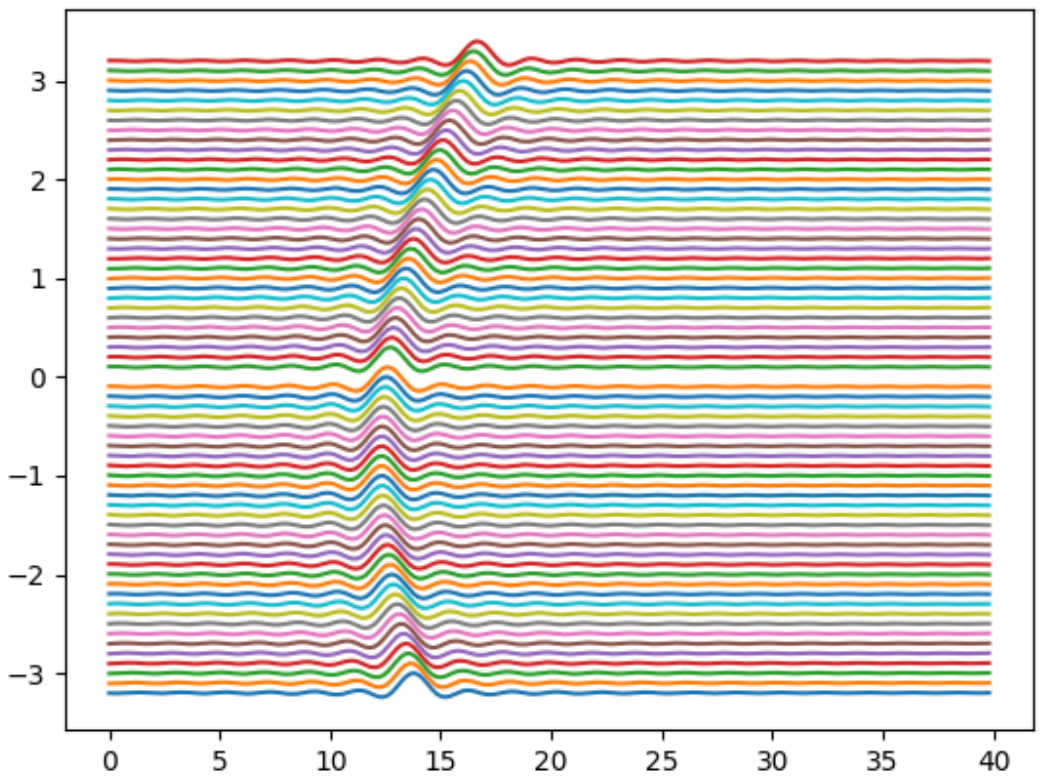


Image 2

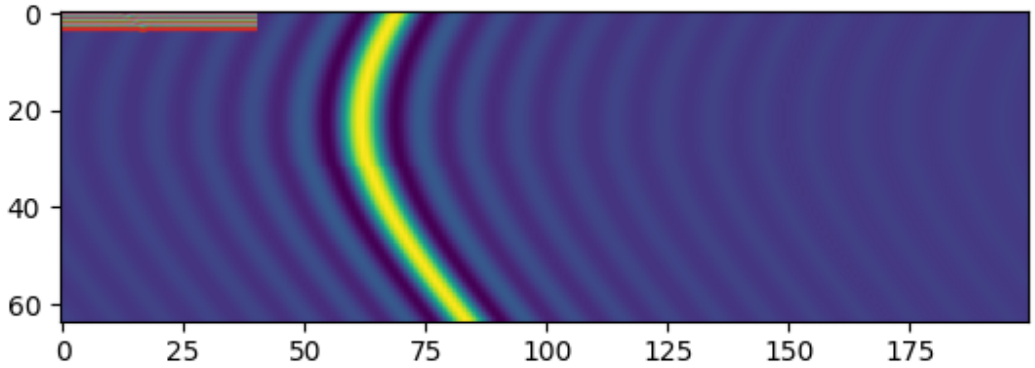


Image 3

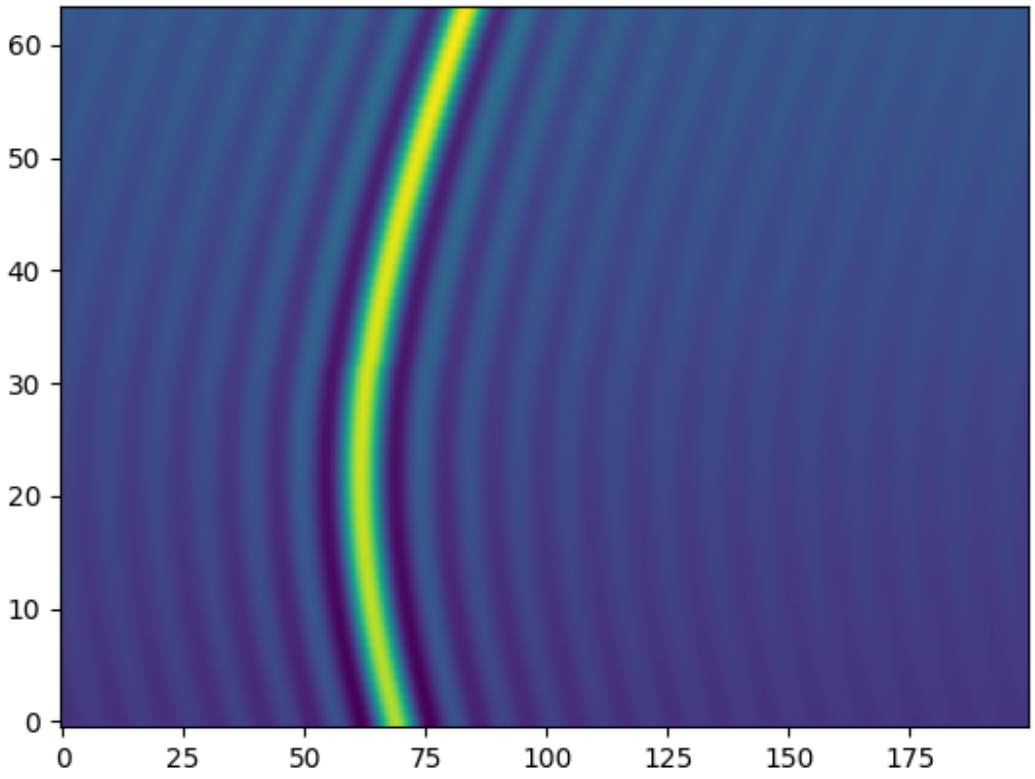


Image 3 (larger)

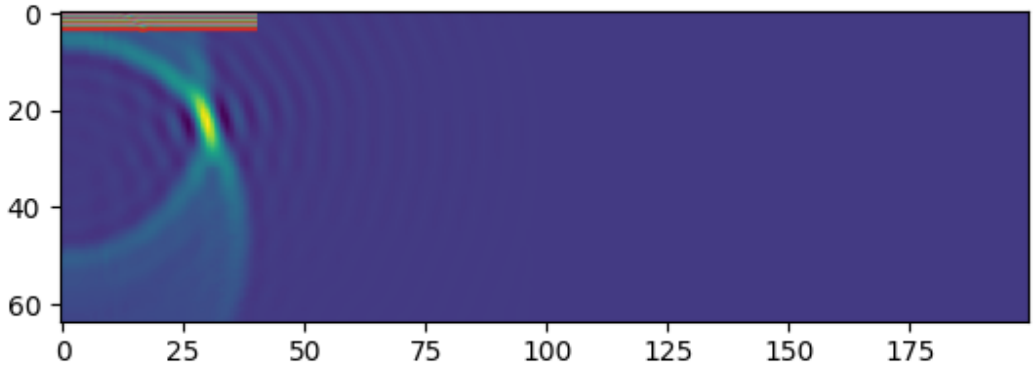


Image 4

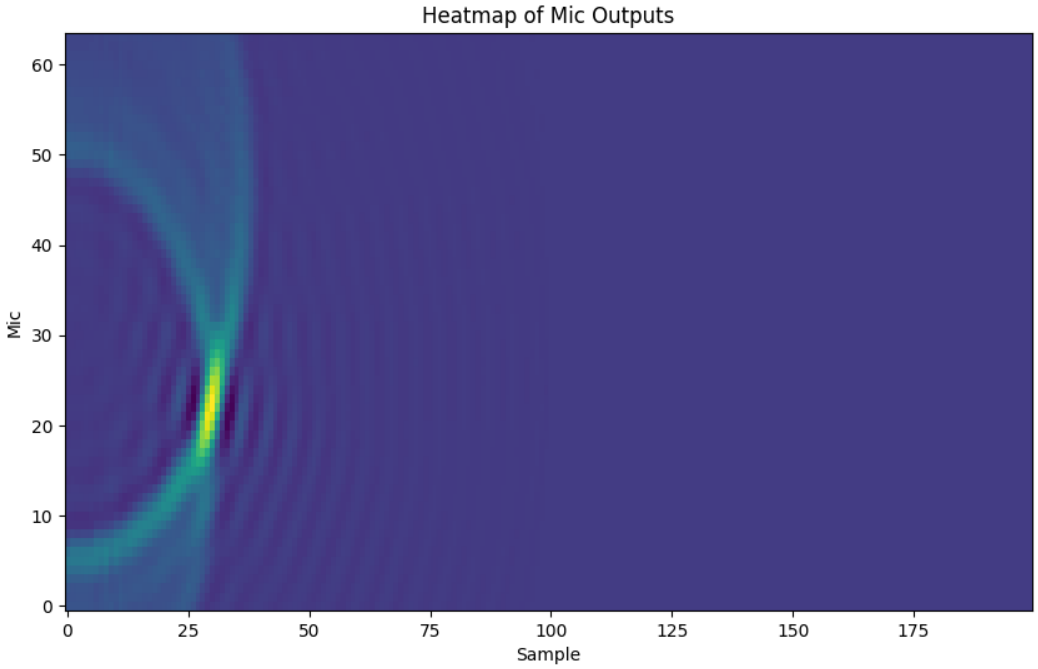


Image 4 (larger)

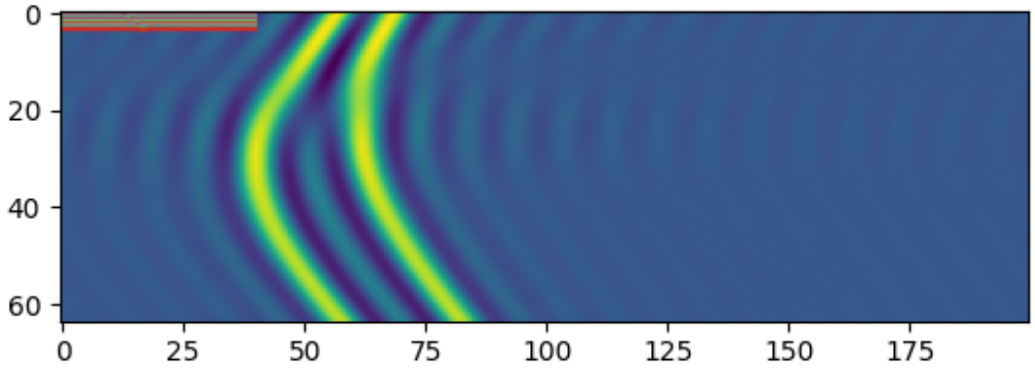


Image 5

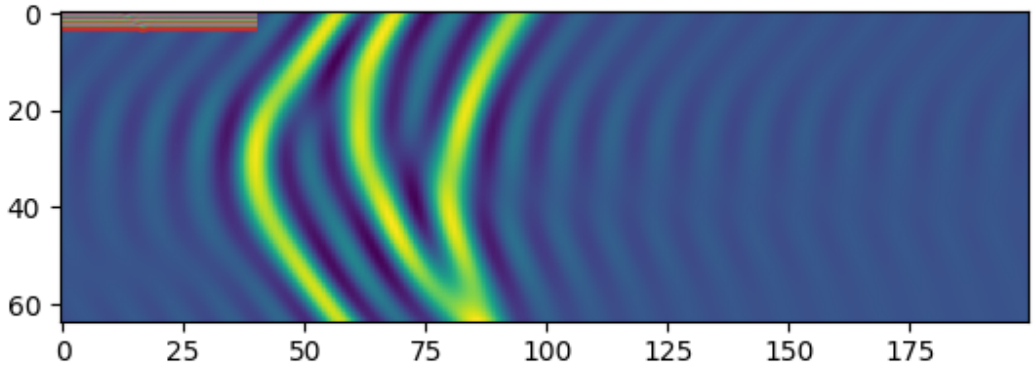
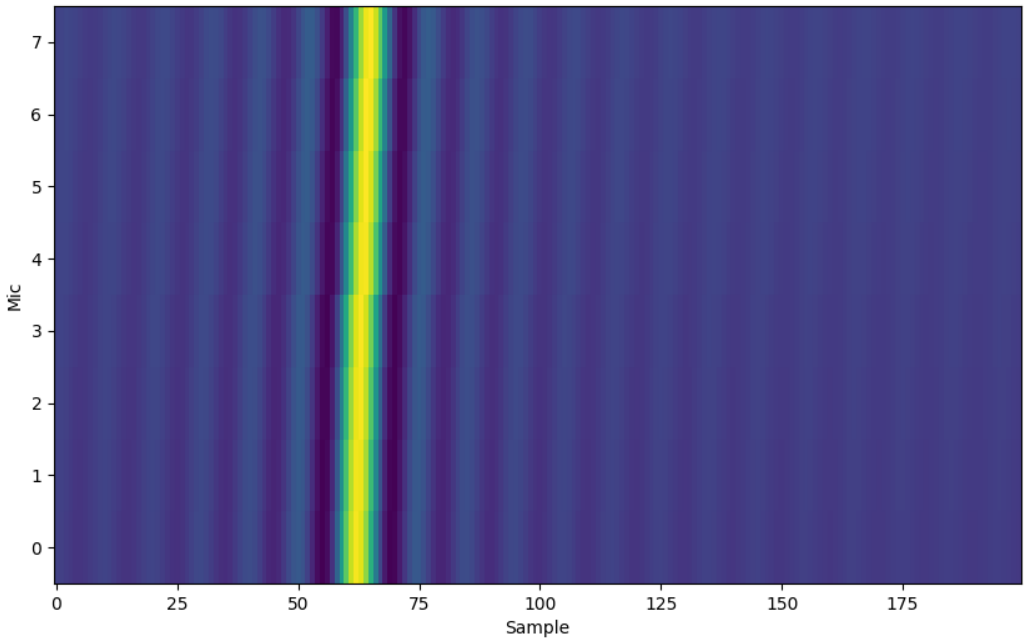
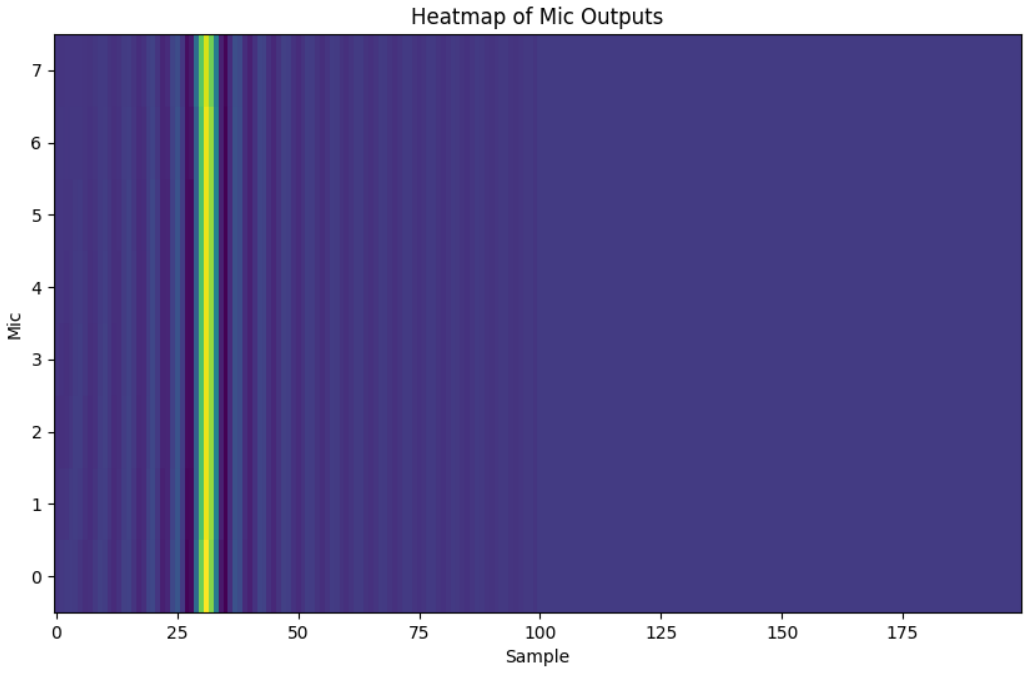


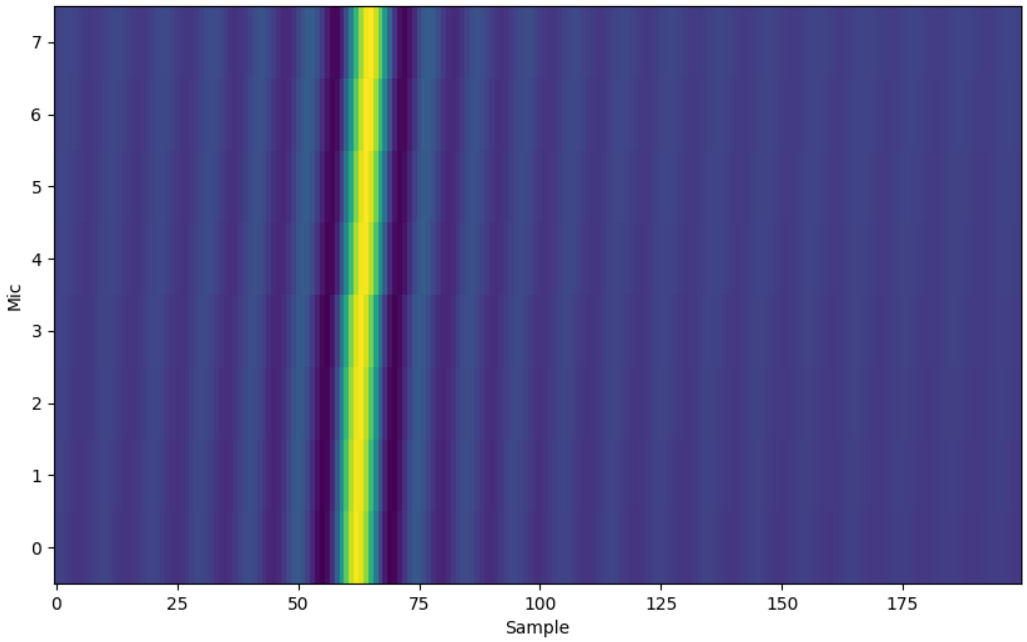
Image 6

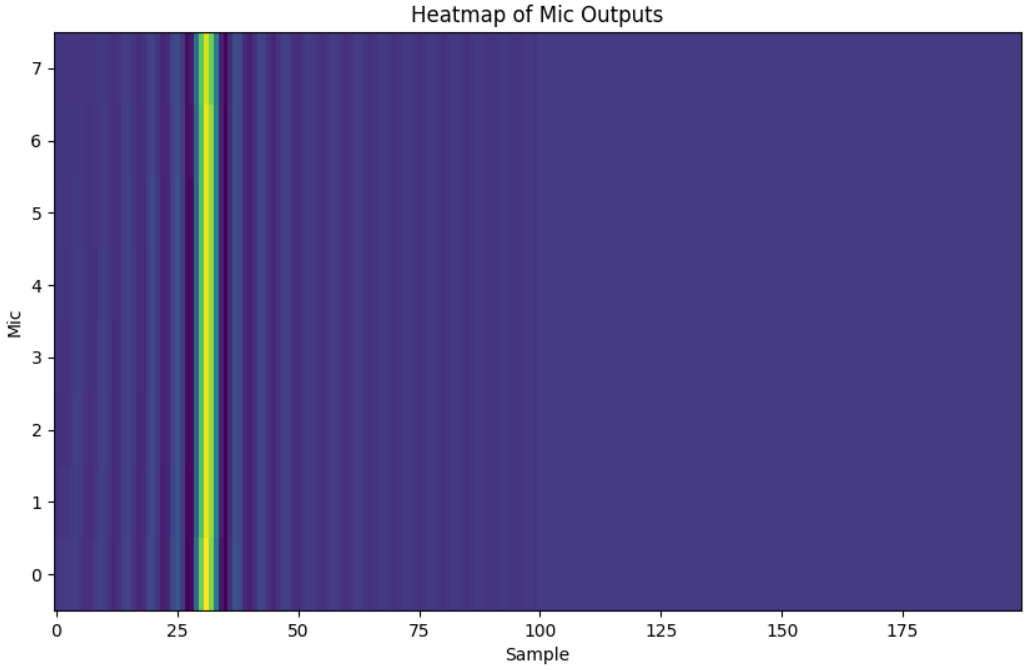
These are the plots for (8, 50).



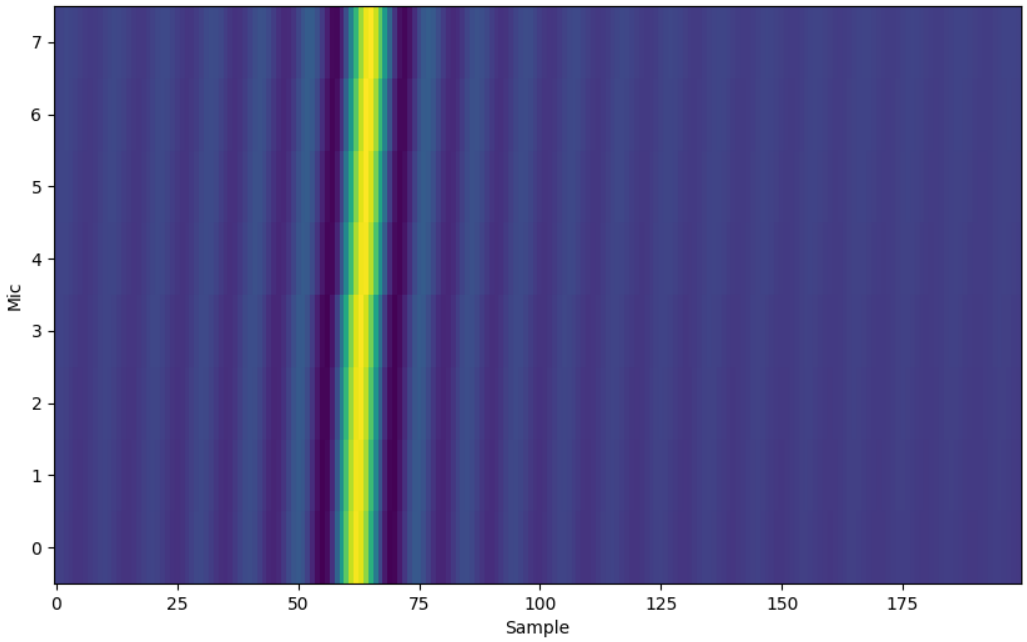


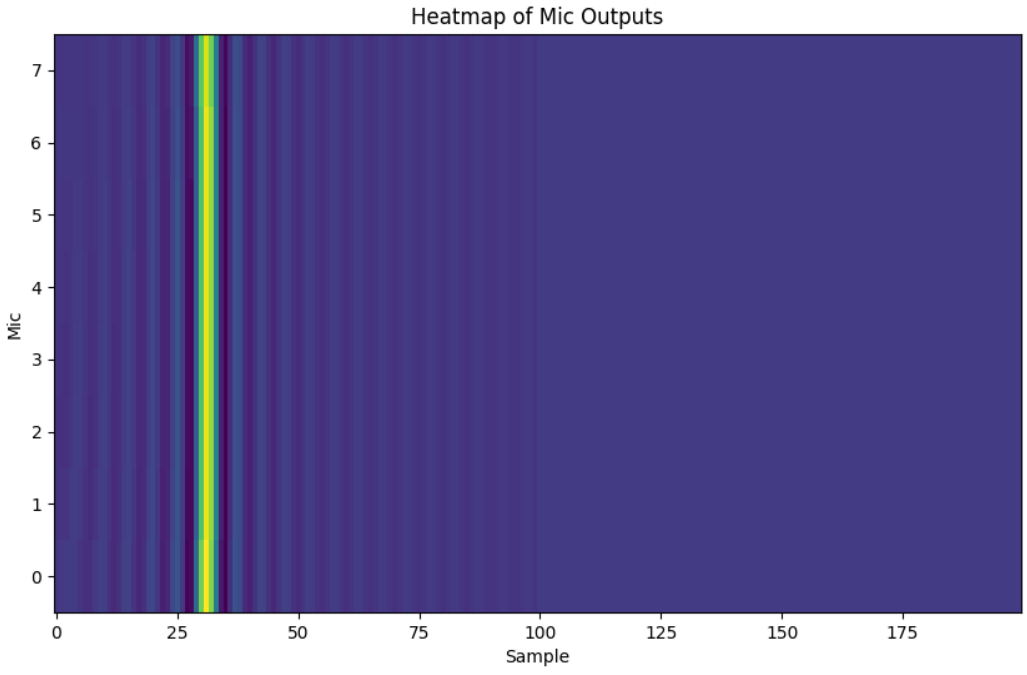
These are the plots for (8, 100).



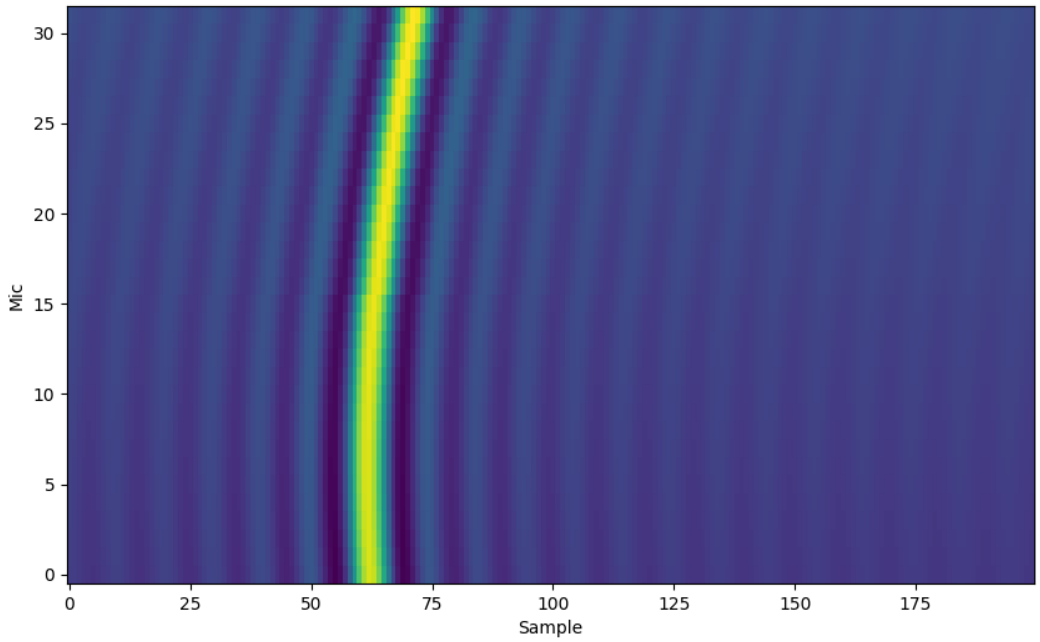


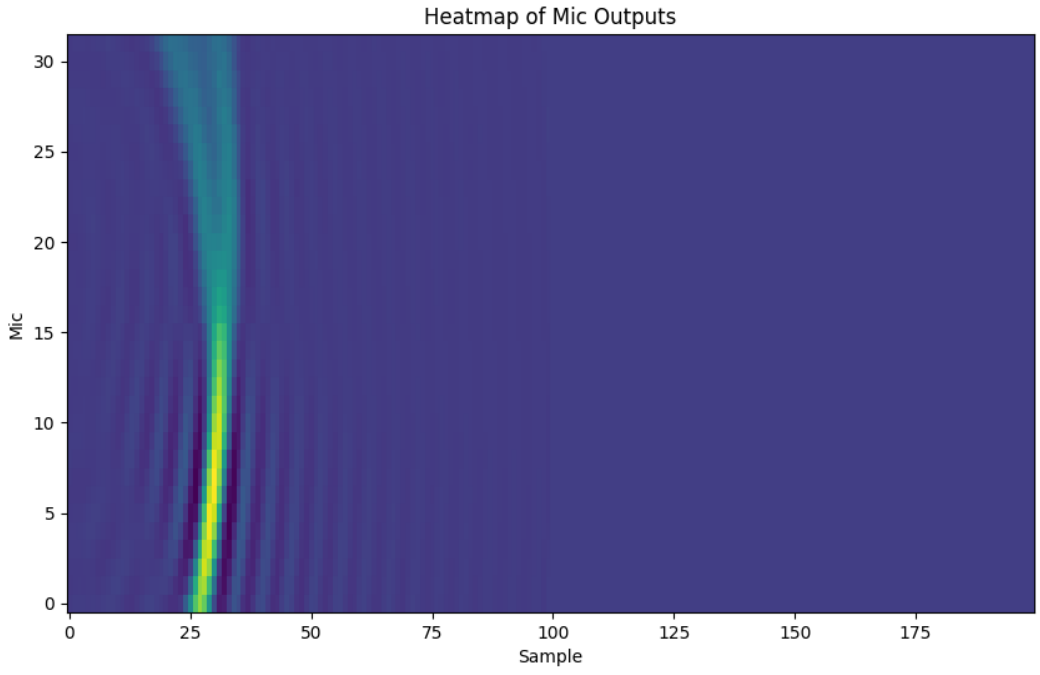
These are the plots for (8, 200).



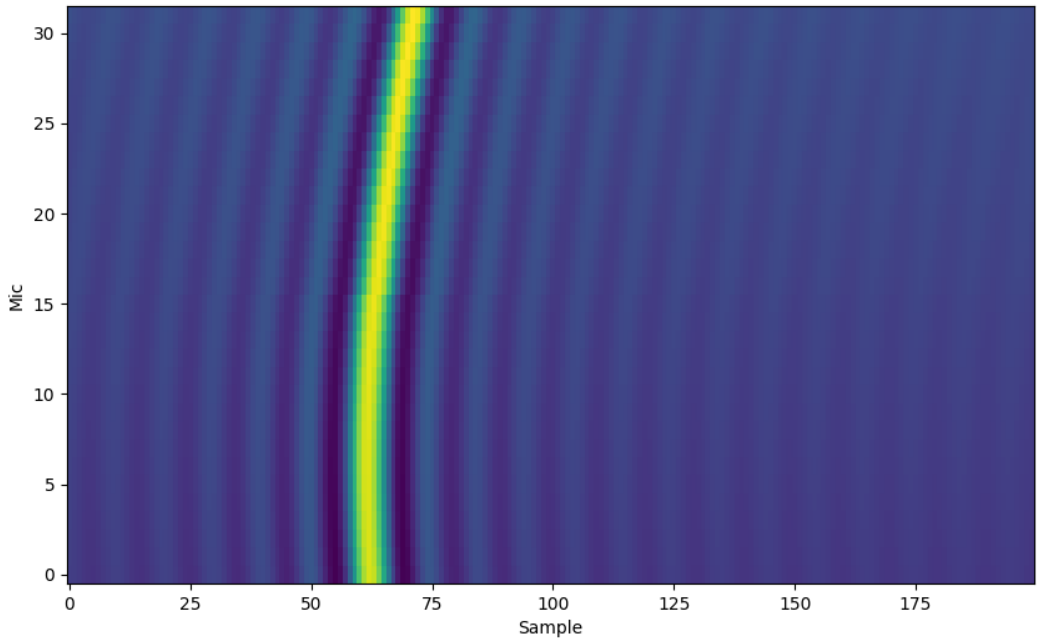


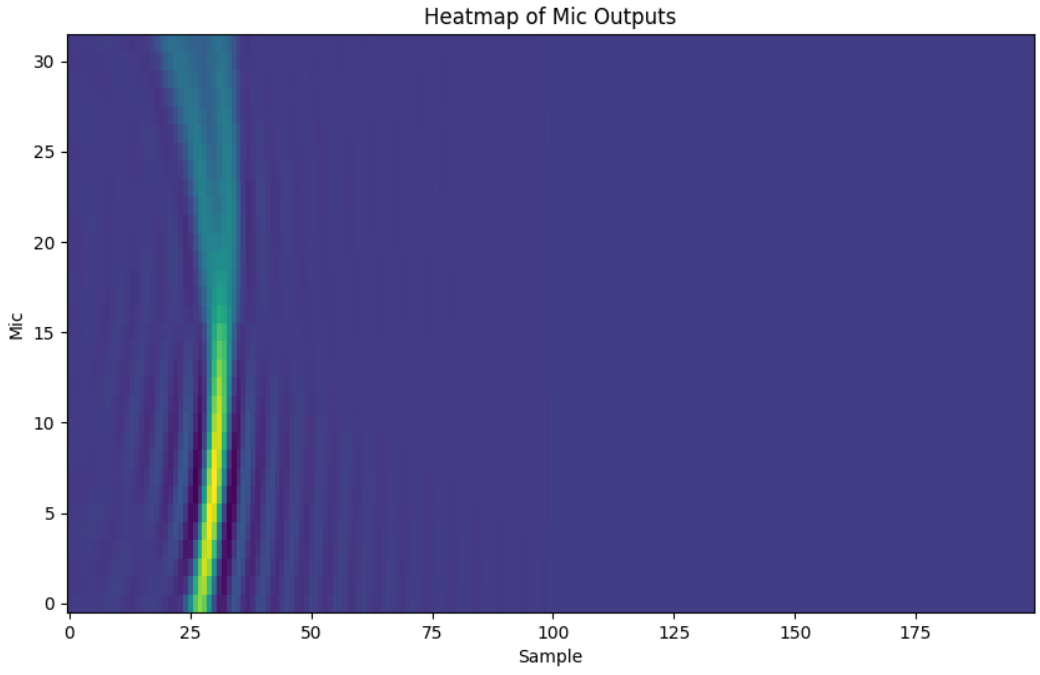
These are the plots for (32, 50).



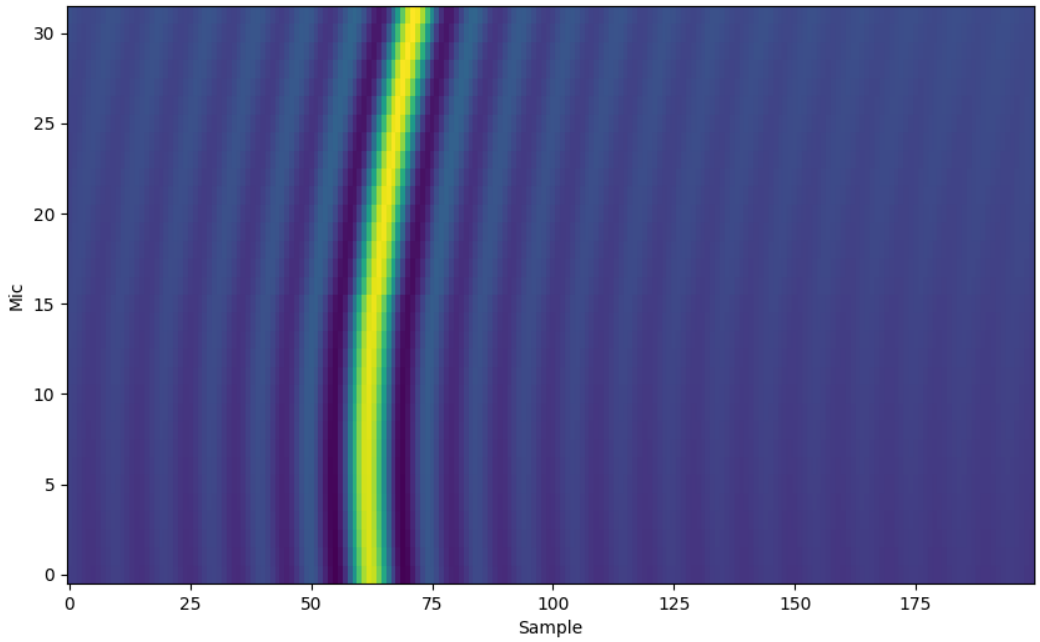


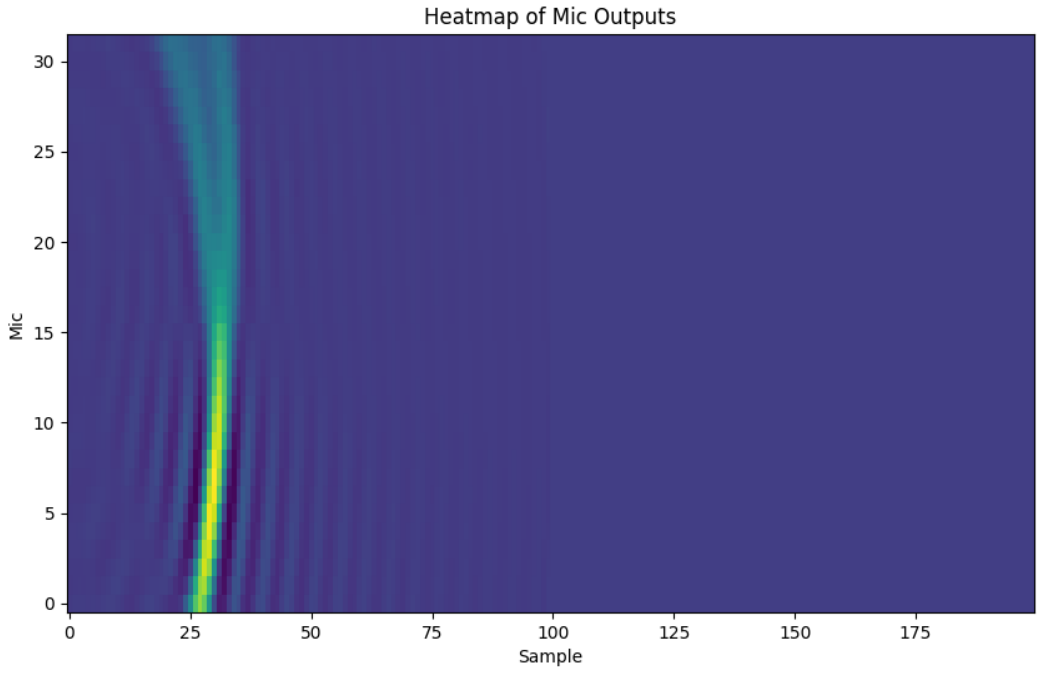
These are the plots for (32, 100).



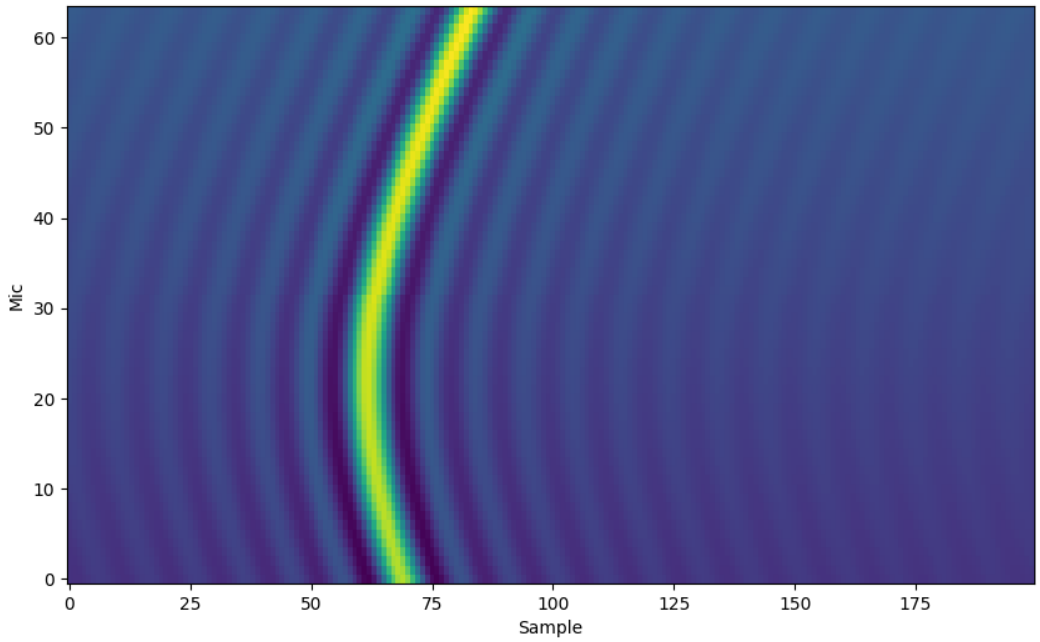


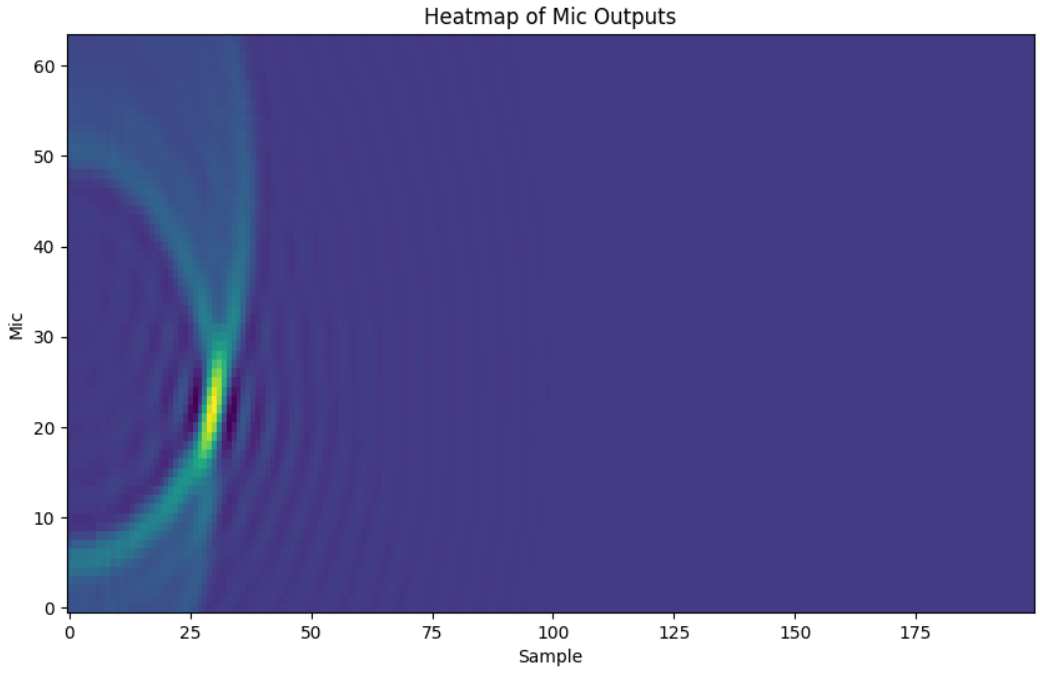
These are the plots for (32, 100).



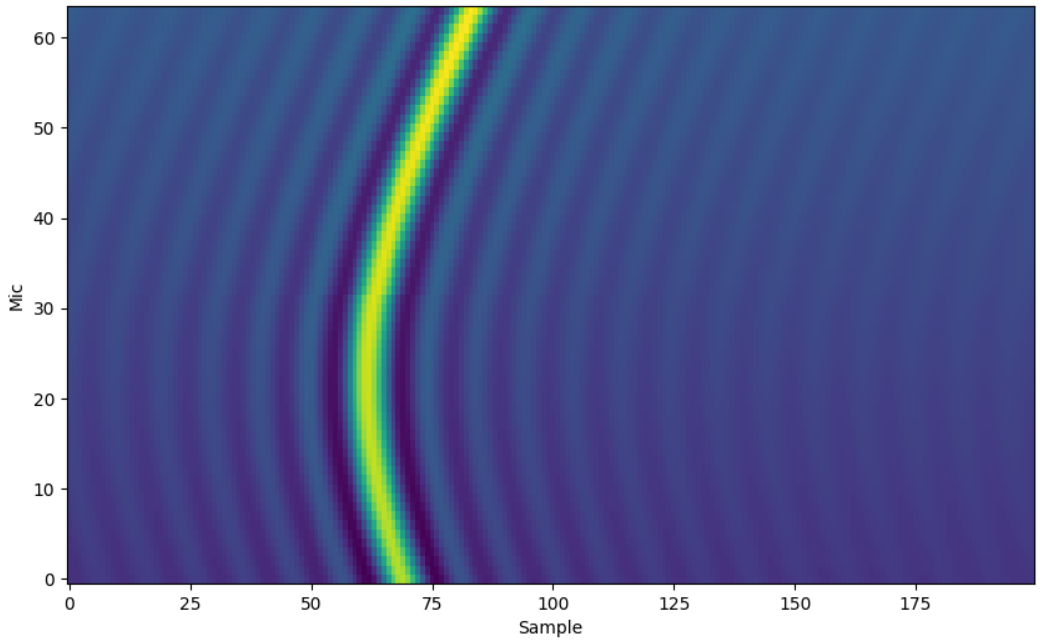


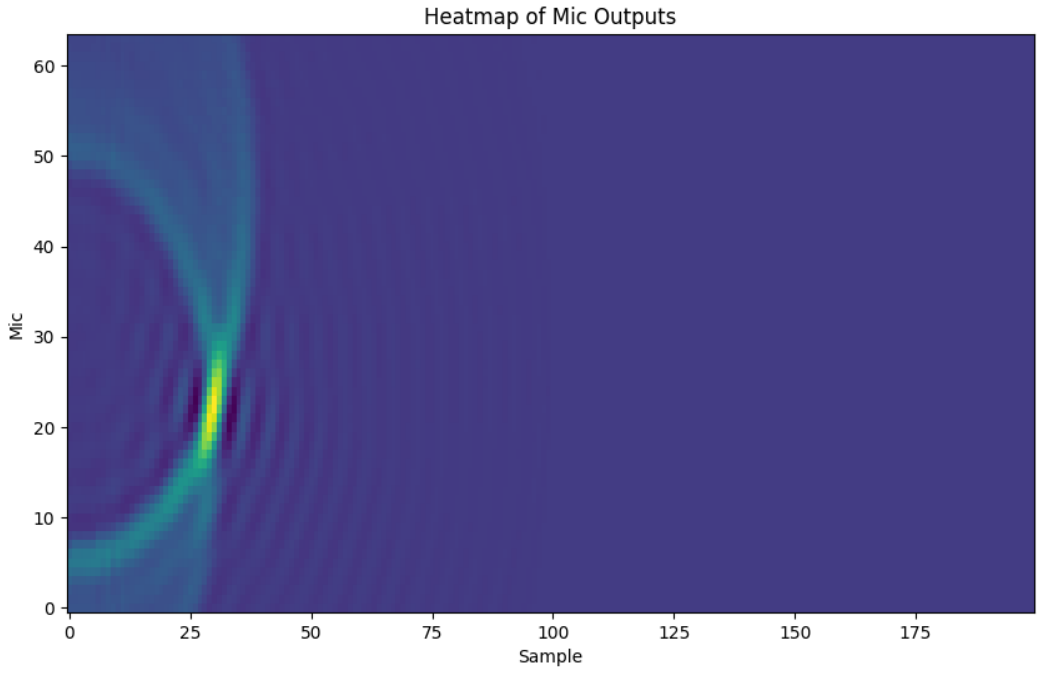
These are the plots for (64, 50).



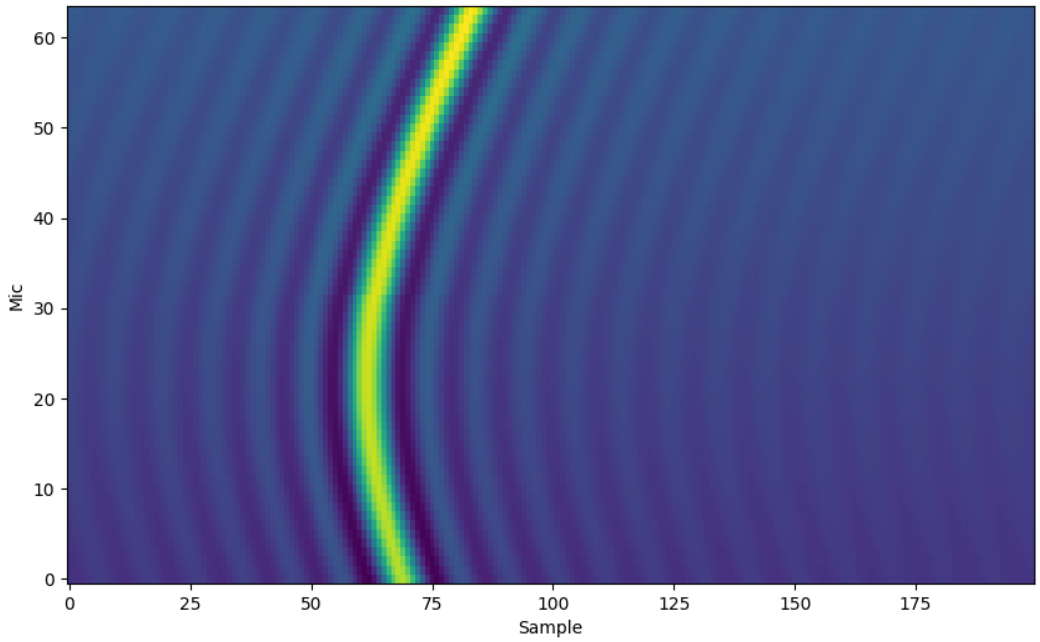


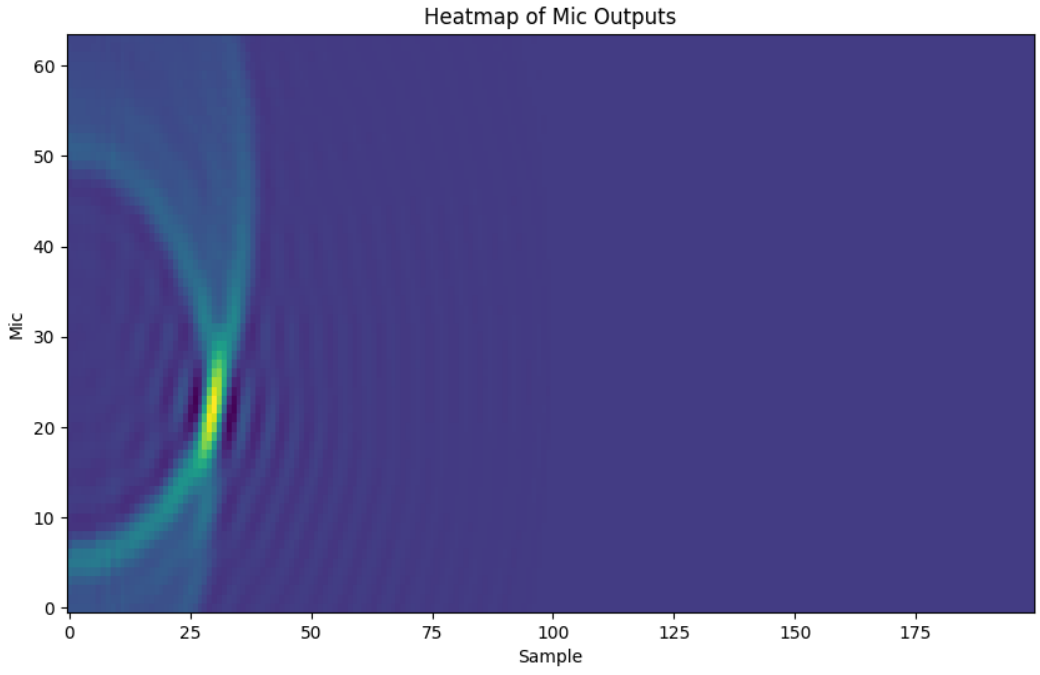
These are the plots for (64,100).





These are the plots for (64, 200).





Preview of the Code:

The code simulates the propagation of an acoustic waveform in a modeled environment involving a source, an obstacle, and a set of microphones. Let's break down the code and functions:

Code Overview:

System Setup:
Nmics, Nsamp, and other parameters define the system:
Nmics: Number of microphones
Nsamp: Number of samples
Waveform Generation:
wsrc(t) Function:
Generates a Sinc waveform given a time variable t.
Parameters:
SincP: Controls the width of the Sinc waveform.
Output:
A Sinc waveform over time.
Microphone Positions:
Calculates the positions of the microphones based on the system parameters.

Signal Propagation and Processing:

dist() Function:
Computes the distance between different points: source to obstacle and obstacle to microphone.
Propagation Calculations:
Calculates the time delays from the source to each microphone via the obstacle.
Microphone Outputs:
Generates and processes the received signals at each microphone.
Displays the received waveforms:
Individually (image-2.png).
Combined as an image (image-3.png).
As a heatmap representation (image-4.png).
Data Visualization:
Loads and displays data from external files (rx2.txt and rx3.txt) as images (image-#5.png and image-6.png).

Functions:

wsrc(t) Function:
Generates a Sinc waveform given a time variable t.
Utilizes the np.sinc() function to generate the waveform.
Modulates the width of the Sinc pulse through the SincP parameter.

dist(src, pt, mic) Function:
Computes the distance between two points.
Utilizes the Euclidean distance formula between src and pt, as well as mic and pt.

Overall Functionality:

Generating the Waveform:
A Sinc waveform is generated and propagated through the modeled environment.

Visualizing Waveform Propagation:
Shows how the waveform reaches each microphone individually and collectively, displayed as an image and a heatmap.

External Data Display:
Loads and displays external data files as images.

The code showcases the simulated propagation of a waveform through a system of microphones in a modeled environment, providing visual representations of the received signals and processed data.