

# RESULTS OF THE TESTING OF 2D ARRAY BEAMFORMER\*

Sara Joiner  
Austin Bratton  
Jeanne Guillory  
Ray Hwong

This work is produced by The Connexions Project and licensed under the  
Creative Commons Attribution License <sup>†</sup>

## Abstract

In this module, we discuss the results of the testing we did of our two-dimensional array delay and sum beamformer.

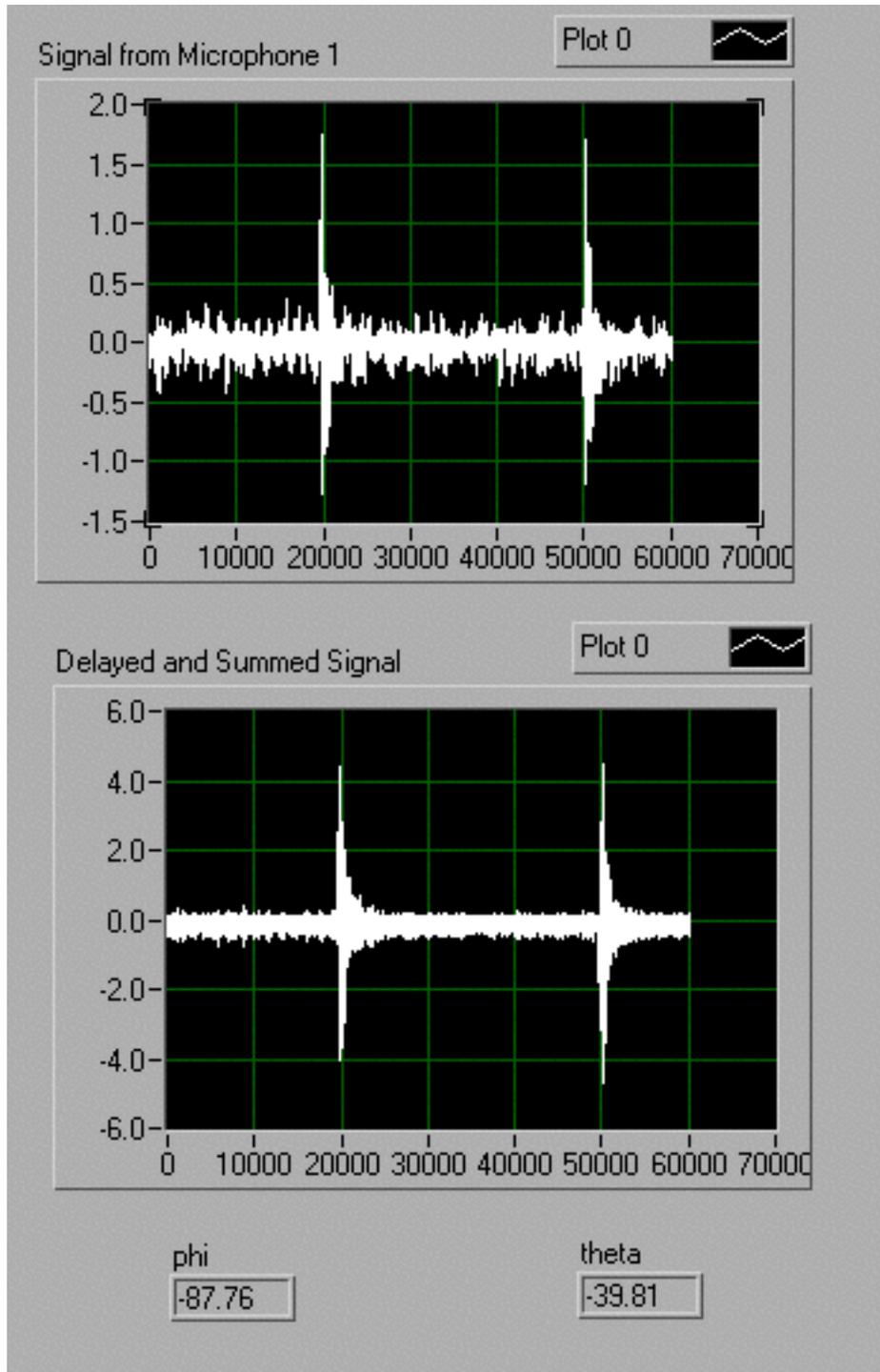
In this section, we discuss the results that we received upon testing our two-dimensional array delay and sum beamformer. The output we designed<sup>1</sup> for our system was relatively simple: the waveform for the delayed and summed signal, the waveform for the first signal (displayed for comparison purposes), and the calculated values for **theta** and **phi**, the two angles used to express where in space our signal could be found.

---

\*Version 1.3: Feb 4, 2006 4:59 pm -0600

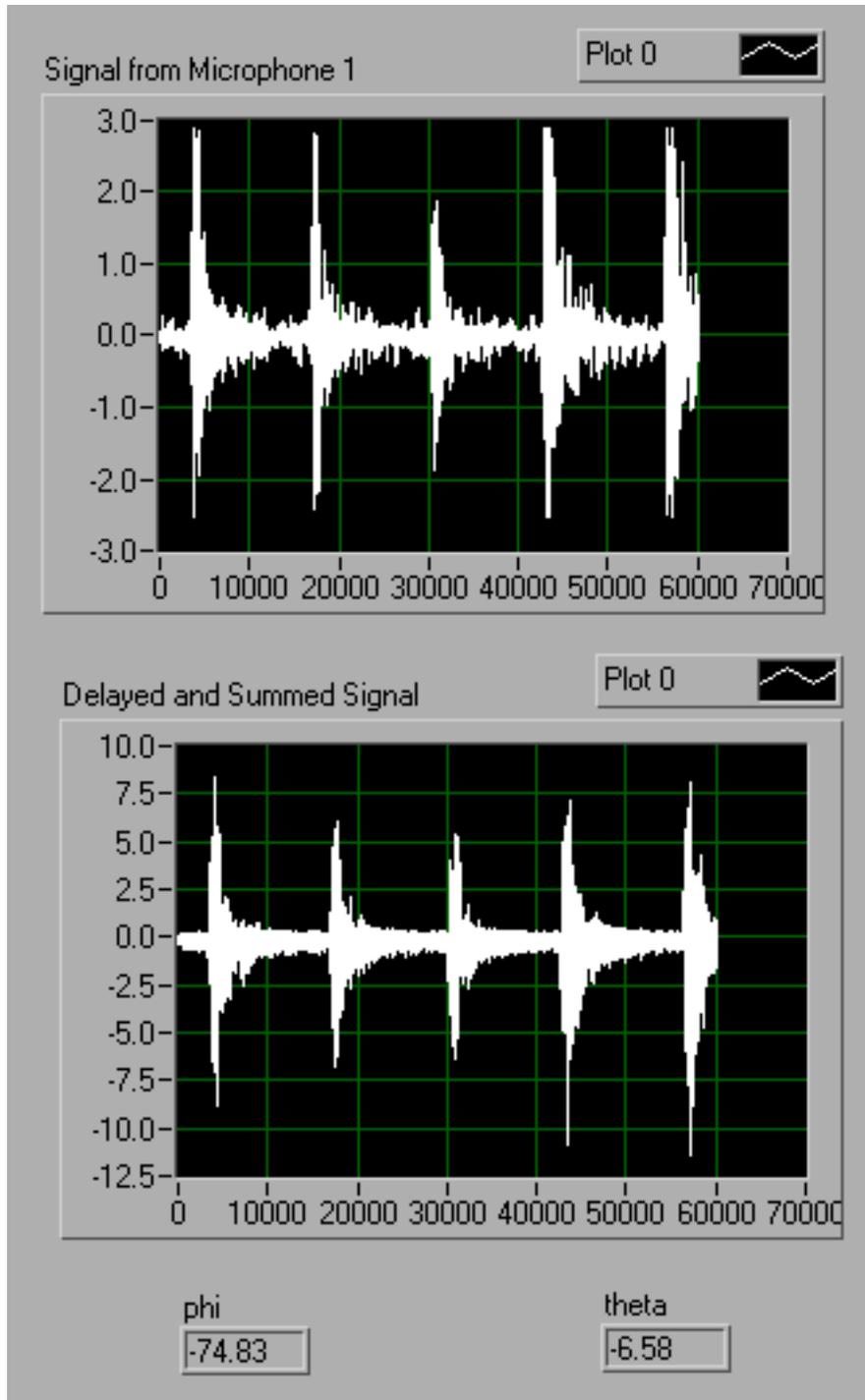
<sup>†</sup><http://creativecommons.org/licenses/by/2.0/>

<sup>1</sup>"Labview Implementation of 2D Array Delay and Sum Beamformer" <<http://cnx.org/content/m13163/latest/>>



**Figure 1:** Our first example of successful output.

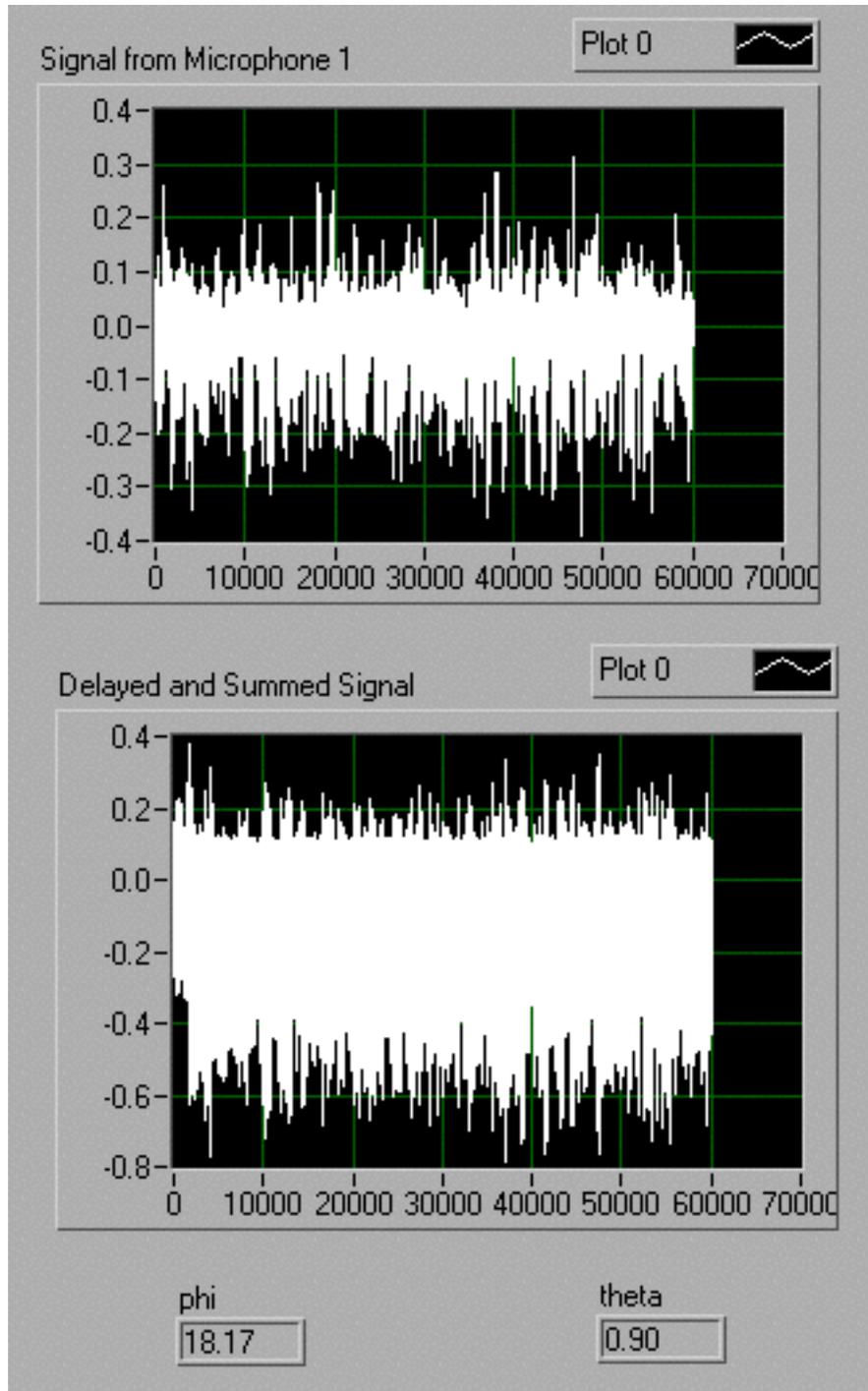
Note how the delayed and summed signal, as it should, bears a striking similarity to the initial signal. However, due to the nature of the delay and sum algorithm, the final result has a great deal higher magnitude, and the signal-to-noise ratio is lower. This is due to the fact that the signal contributes the most to the inner product calculations done to find the proper delays, and thus the signal is matched up properly; the noise, on the other hand, being essentially random, is additive in some places and destructive in others, leading to an overall relative decrease in noise.



**Figure 2:** A similar example, with a somewhat different signal

This signal, as can be seen, was somewhat closer to zero degrees and a bit higher off the xy plane (smaller phi) than the previous example. We noticed rather quickly that, although it is a relatively simple matter to vary theta (as that just involves moving around), it is less simple to gain significant variation in phi: at far field distances, one must get a great deal higher off the ground before the angle of the signal reaching the array changes significantly.

As it is, this particular phi value (at a significantly higher angle than before), was accomplished by coming a great deal nearer to the array – thus bringing in possible complications due to the fact that the signal source could now be reasonably considered as near field. However, even for middling-near field sources, the far field approximation still holds to a certain extent, as can be shown by the accuracy this data maintained.



**Figure 3:** Did we mention we happened to chose the computer nearest to a loud fan?

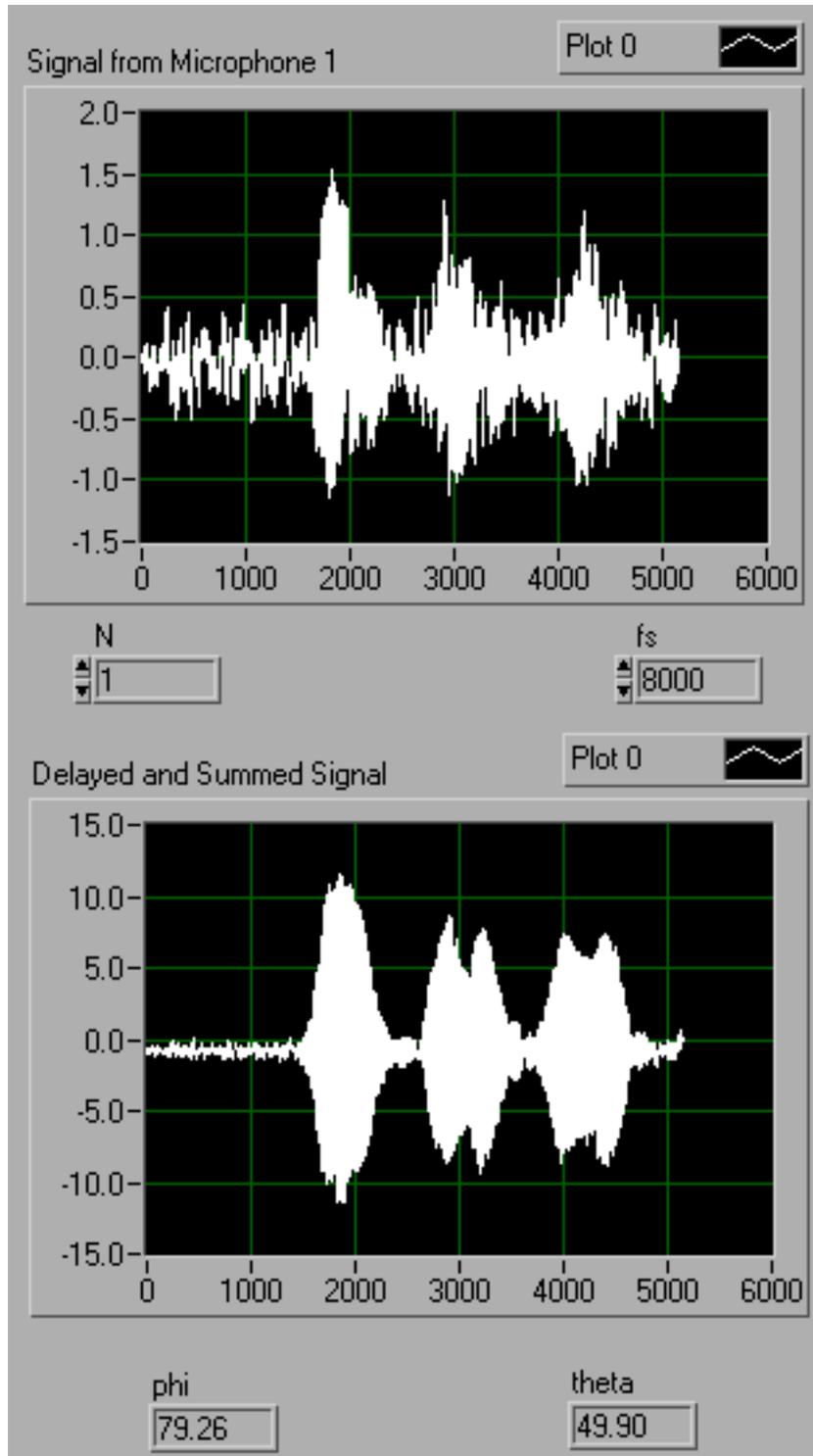
Speaking of noise ... due to the existence of a relatively noisy fan quite near to our work area, the background noise level ended up being somewhat higher than the ideal. However, said fan proved quite useful as yet another test source.

The delayed and summed waveform in this case shows a great deal more "constant" level, if you will, of noise than the individual waveform; thus lending weight to our contention that, since only a small part of the signal is used for matching and since noise is, on the whole, relatively random, the noise is both additive and destructive in such a way as to make the overall level of noise less than one might expect.

Although the more regular signals experience an increase in overall magnitude of at least 2x as a rule, it is interesting to note that, much as the previous paragraph suggests, the noise level here remains more or less the same.

Also of note is, of course, the angle of the noise itself – as we might expect from a fan placed nearly directly overhead, our computations do indeed yield a small theta (given that it was not exactly overhead, but in the approximate zero direction) and a small phi – indicative that, despite the random nature of the noise, our algorithm was still able to find enough regularity to deduce the position of the "signal".

All the previous examples were run at an upsampling rate of 10x and a sampling frequency of 8 kHz. However, both of the previously mentioned variables can be changed at will; we also went through and did a couple of tests at other levels of upsampling, although we generally stayed at or around the same frequency, as that was initially chosen to cut down on the amount of aliasing that would be induced by the analog-to-digital conversion.



**Figure 4:** An example of how our program ran with no upsampling.

As you can see, even with no upsampling, there is still a significant improvement in the signal between the initially input signals and the final, delayed and summed output. The less upsampling there is, the fewer numbers there are that the program must iterate through; thus it should come as no surprise that when run with no upsampling, the program completed far faster than with 10x.

However, it is also worth noting that when the signal is not upsampled, there is a lot less data to work with, and the amount shifted to compensate for delay is more likely to be significantly different from the actual time the signal should be delayed in an analog setting.

Thus, calculations may be prone to slightly more error, and as a result the final product may also not be quite as effective at noise suppression as its upsampled cousins – although as you can see from the figure above, "not quite" still shows a significant change, even if the noise is still noticeably more spiky than the previous results.