Fast Fourier Transforms

Aleksandar Ivanov

November 11, 2020

1 Problem Statement

You're provided with audio samples of calls of the Eurasian eagle-owl (*Bubo bubo*). Calculate the autocorrelation functions and Fourier Transforms of the calls and try to determine which owl is recorded in which sample.

2 Methodology

To analyze the samples we use the `numpy` package of Python3. Speciffically, the function `numpy.fft.ftt` to calculate the Fast Fourier Transform [3] of all the signals and the `numpy.correlate` [2] function to calculate the autocorrelation of the signals.

The `numpy.correlate` function has three modes `full`, `same` and `valid` all of which try to solve particular problems of discretizing the autocorrelation. The default mode `valid` gives a result only when we have total overlap, which in the case of the autocorrelation is only a single point, which makes it unusable. For us, the mode `full` is most usefull because it has the minimal amount of edge effects, while still giving us a meaningful result. This mode calculates the autocorrelation at all points where there is *some* overlap, outputting an array of size $2N - 1$, where N is the size of our signal array. In effect, this mode zero pads the signal [4] for us before calculating the autocorrelation.

The form of the cross-correlation that `numpy.correlate` uses is

$$\phi_k(u, v) = \sum_{n=-N+1}^{N-1} \tilde{u}_{n+k}(\tilde{v}_n)^* \qquad (1)$$

where $\tilde{u}$ is the zero padded verison of the array u and * denotes complex conjugation.

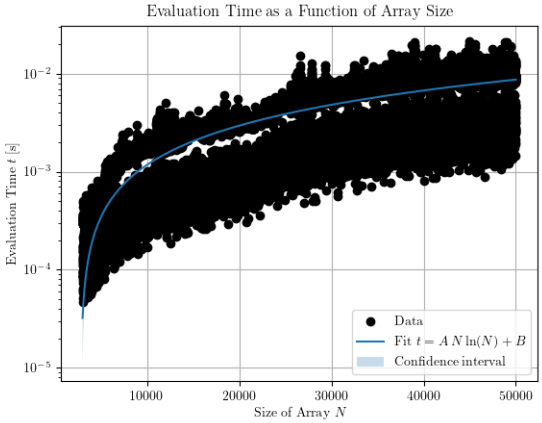

Figure 1: Evaluation time t for `numpy.fft.ftt` as a function of array size N, tested on random arrays.

The evaluation time for an input array of size N for `numpy.fft.ftt` is shown in fig. 1, while the one for `numpy.correlate` is shown in fig. 2. We see that the FFT follows the predicted $N\ln(N)$ asymptotic time complexity. [1]The autocorrelation time complexity is a bit harder to read; all we can say is that it grows faster than a power law.

3 Fourier Spectrum

Plotting the Fourier spectra of the samples we get figs. 3 to 8. In fig. 3, the almost backgroundless signal for the first owl, we can very clearly see peaks at 379 Hz, 759 Hz, 1143 Hz, 1521 Hz, 1892 Hz..., which are clearly the harmonics of the owl's call. Similarly, in fig. 4 we see 334 Hz, 669 Hz, 1005 Hz, 1340 Hz, 1675 Hz..., which are the harmonics of the second owl's call. These are going to help us differentiate the owls in the samples with more back-

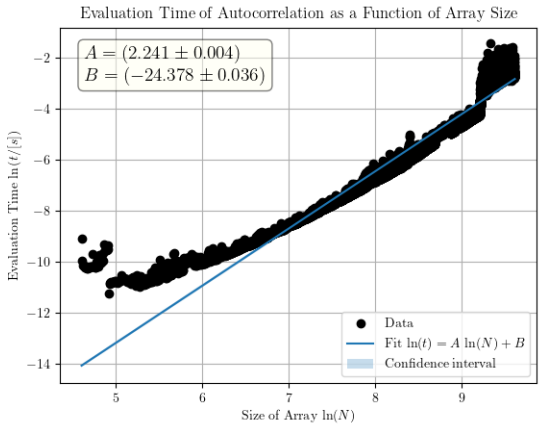

Figure 2: Evaluation time t for `numpy.correlate` as a function of array size N, tested on random arrays.

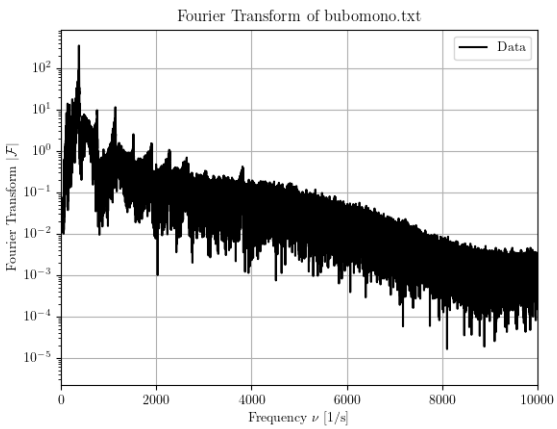

Figure 3: Fourier spectrum of `bubomono.txt` in a log scale.

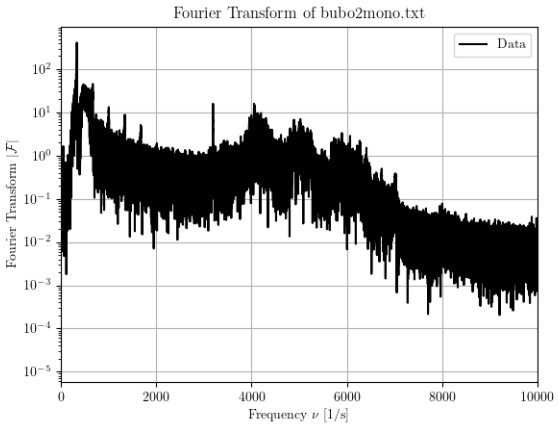

Figure 4: Fourier spectrum of `bubo2mono.txt` in a log scale.

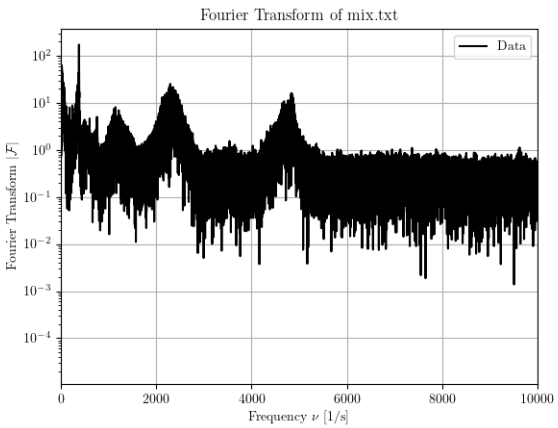

Figure 5: Fourier spectrum of `mix.txt` in a log scale.

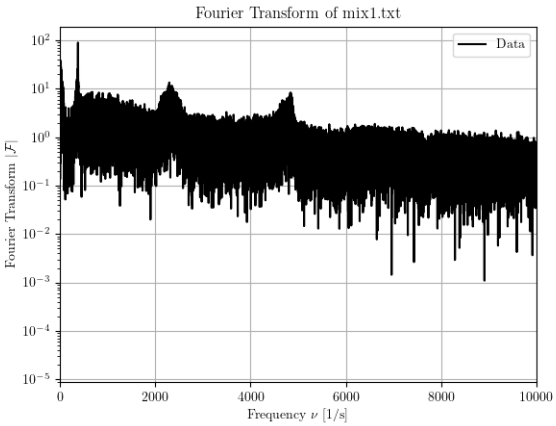

Figure 6: Fourier spectrum of `mix1.txt` in a log scale.

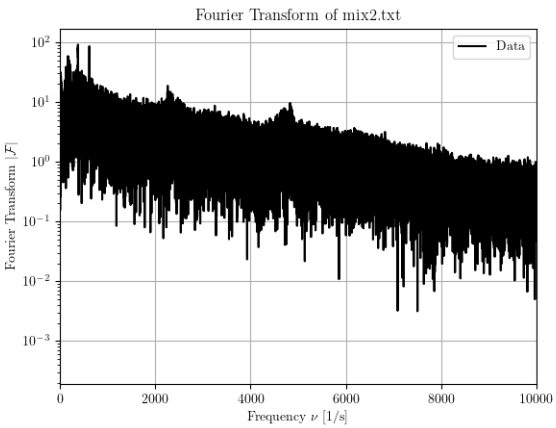

Figure 7: Fourier spectrum of `mix2.txt` in a log scale.

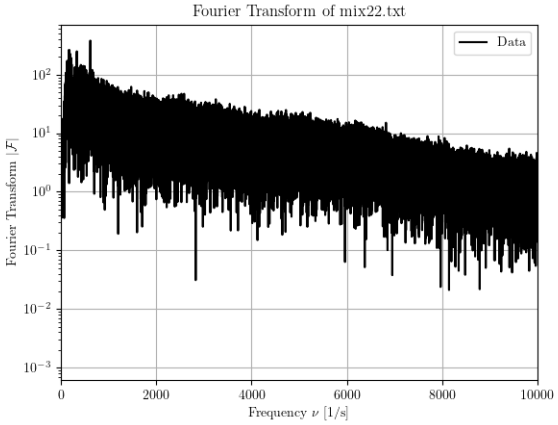

Figure 8: Fourier spectrum of `mix22.txt` in a log scale.

ground noise. Looking at figs. 5 to 8, we immediately see that we can at least differentiate the fundamental mode. Comparing the frequency at those peaks, we get the guesses

$$\begin{aligned} \texttt{mix.txt} &\Rightarrow \text{owl } 1 \\ \texttt{mix1.txt} &\Rightarrow \text{owl } 1 \\ \texttt{mix2.txt} &\Rightarrow \text{owl } 1 \\ \texttt{mix22.txt} &\Rightarrow \text{owl } 2 \end{aligned} \tag{2}$$

The river in the samples also cases a peak in frequency; when it's a small stream we get the humps in figs. 5 and 6 and less prominently in fig. 7, when it's a more vigorous part of the river, the effect becomes just noise. This is something we can hear by listening to the samples.

4 Autocorrelation

The autocorrelation graphs of the different signals are plotted in figs. 9 to 14. The range we have chosen starts at 0, since the autocorrelation of a real function is symmetric. We see that, in general, the autocorrelation is a function that oscillates, since our signal is also oscillatory, and that the amplitude of these oscillations mostly goes down as we go to higher offsets, an effect of the zero padding. As expected, the highest value is always at the offset of 0. The more noise we have, the higher this

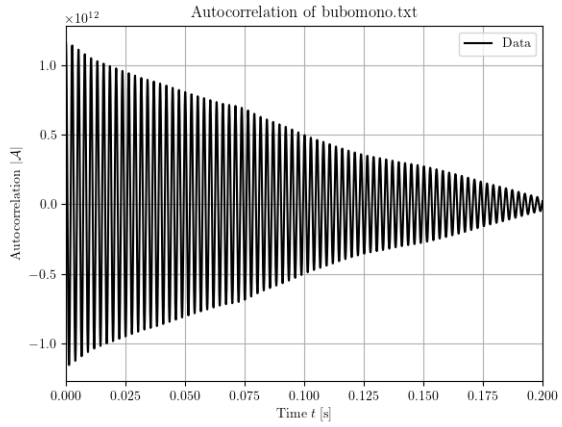

Figure 9: Autocorrelation of `bubomono.txt` signal.

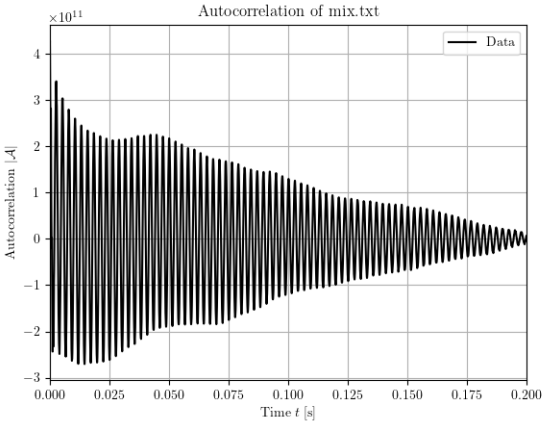

Figure 11: Autocorrelation of `mix.txt` signal.

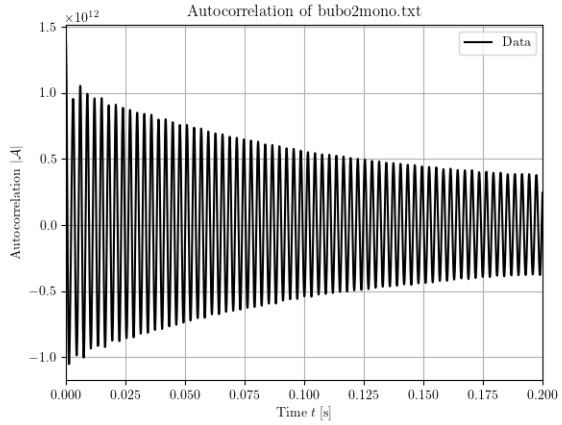

Figure 10: Autocorrelation of `bubo2mono.txt` signal.

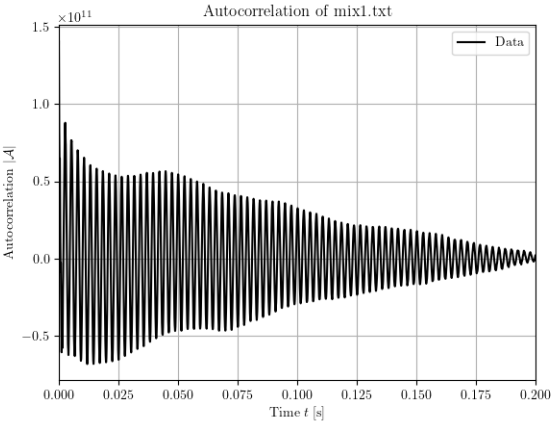

Figure 12: Autocorrelation of `mix1.txt` signal.

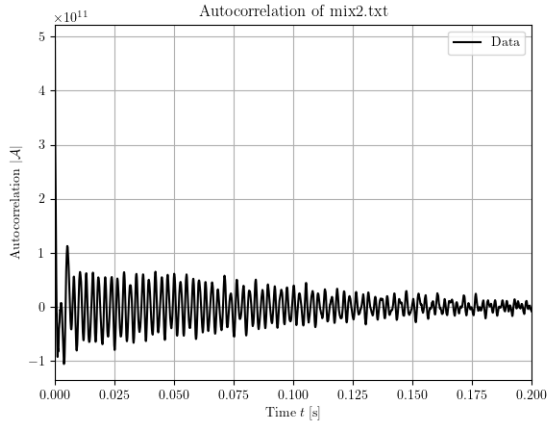

Figure 13: Autocorrelation of `mix2.txt` signal.

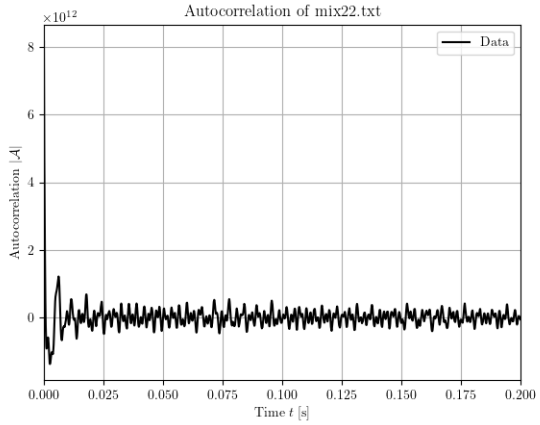

Figure 14: Autocorrelation of `mix22.txt` signal.

peak at 0 will be relative to the rest of the autocorrelation, since the signal is less similar to itself at non-zero offsets. The same effect makes the oscillations less regular.

References

[1] Bevan Baas. *DFT (Discrete Fourier Transform) and FFT (Fast Fourier Transform)*. [Last accessed: 10.11.2020]. 2020. URL: `https://www.ece.ucdavis.edu/~bbaas/281/notes/Handout.fft1.pdf`.

[2] The SciPy community. *numpy.correlate*. [Last accessed: 10.11.2020]. 2008-2020. URL: `https://numpy.org/doc/stable/reference/generated/numpy.correlate.html`.

[3] J. Cooley and John W. Tukey. "An algorithm for the machine calculation of complex Fourier series". In: *Mathematics of Computation* 19 (1965), pp. 297–301.

[4] DSP Illustrations. *Spectral Leakage and Zero-Padding of the Discrete Fourier Transform*. [Last accessed: 10.11.2020]. 2019. URL: `https://dspillustrations.com/pages/posts/misc/spectral-leakage-zero-padding-and-frequency-resolution.html`.