My beetle recognition program uses correlation to classify a .wav file as either one of the specified beetles or as noise. The classifier creates a master key out of the FFTs of the training data, then integrates the correlation signal between the FFT of the test data and each master key to provide a numerical value of how similar the test data is to each of the keys. The program classifies the data as the key that produced the highest correlation value. My program is written in Python, and the non-standard libraries I use are wavio, numpy, and matplotlib. The program must be in the same folder as the folders containing the training data. My program is documented with comments that explain what each piece is doing except for parts that recur multiple times, in which only the first instance of the logic will be documented.

My program starts by creating a master key out of the training data. To do this, I read each sample with wavio then normalize each sample to ensure that they all have the same volume. I then take the FFT of each sample, average the FFTs together, and normalize again. The second normalization is important, because it ensures that the classifier selects the best frequency data fit, as opposed to the loudest key.

Next, my classifier reads and normalizes the test data, then takes the FFT of the test data and normalizes the results. Finally, my classifier correlates the normalized test FFT with each of the keys and sums each correlation signal. The classifier selects the highest value and outputs the classification and the correlation value to the command line.

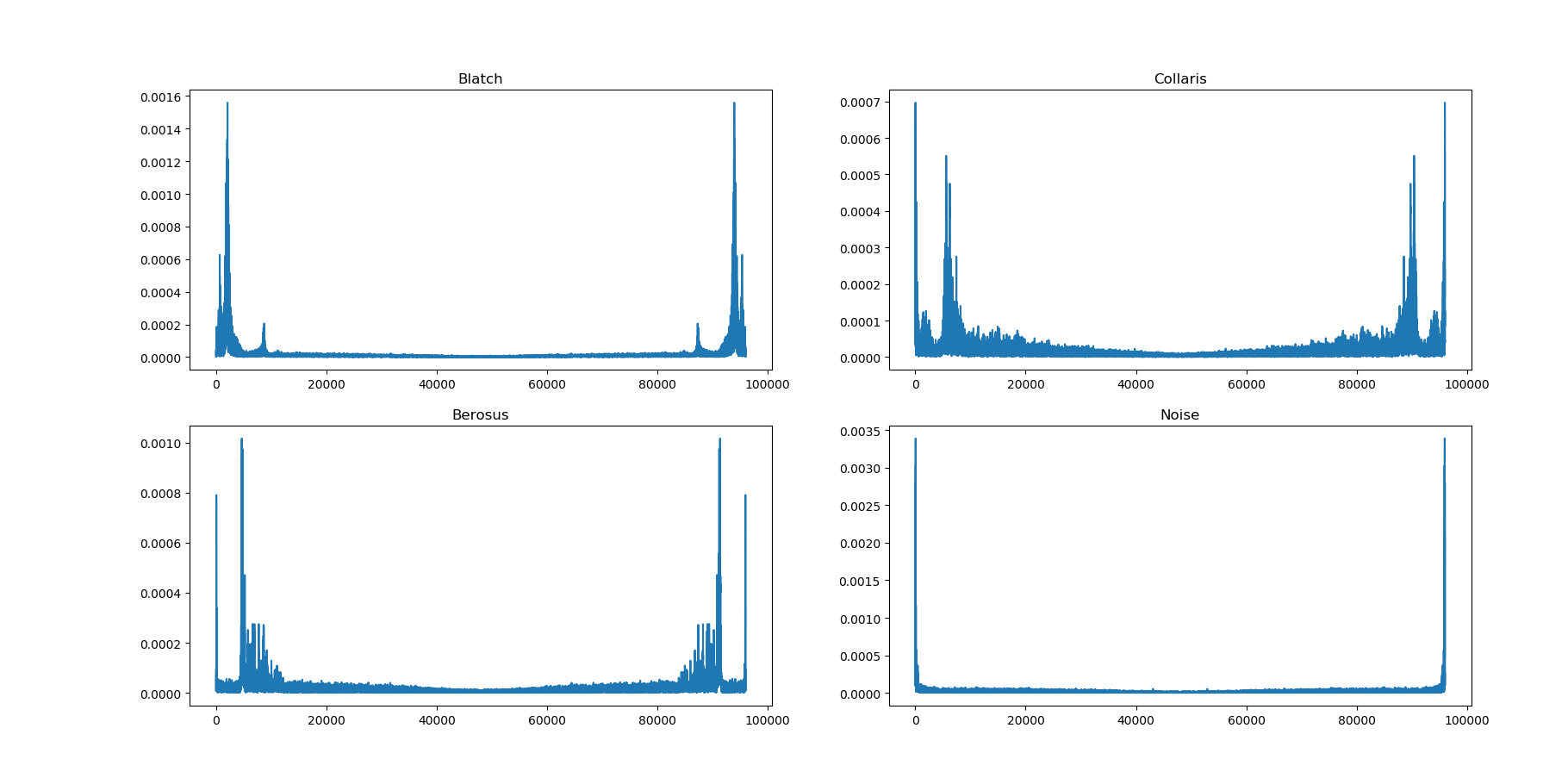


Figure : Normalized FFT data for master keys