Computational Methods to Analyze Echolocation in Eptesicus fuscus

Ryan Himmelwright[1], Laura Kloepper[2], Jason Gaudette[2], and James Simmons[2]

[1] Lafayette Neuroscience Program, Easton PA and [2]Simmons Bat Lab, Brown University, Providence RI

Poster #8

The Big Brown Bat

A bat in the lab (Dori), goes to eat a meal worm during training. Filmed with an infrared lens on the GoPro camera.

The Big Brown Bat uses echolocation To navigate during nocturnal hunting.

*- The **Big Brown Bat (Eptesicus fuscus)**, is a fairly common species in North America. Considered "large" for an American bat, Eptesicus fuscus has an average weight between **14 and 21 grams, with a 12-16 inch wingspan.***

*- Being a **nocturnal hunter**, the Big Brown Bat uses **echolocation** to **navigate** flight and to **capture prey**, which consists of **insects**.*

*- During echolocation, the bat emits many **sound pulses**. Based on the **strength** and **timing** of the **returning echos,** the bat constructs a map of its surroundings.*

Research Background

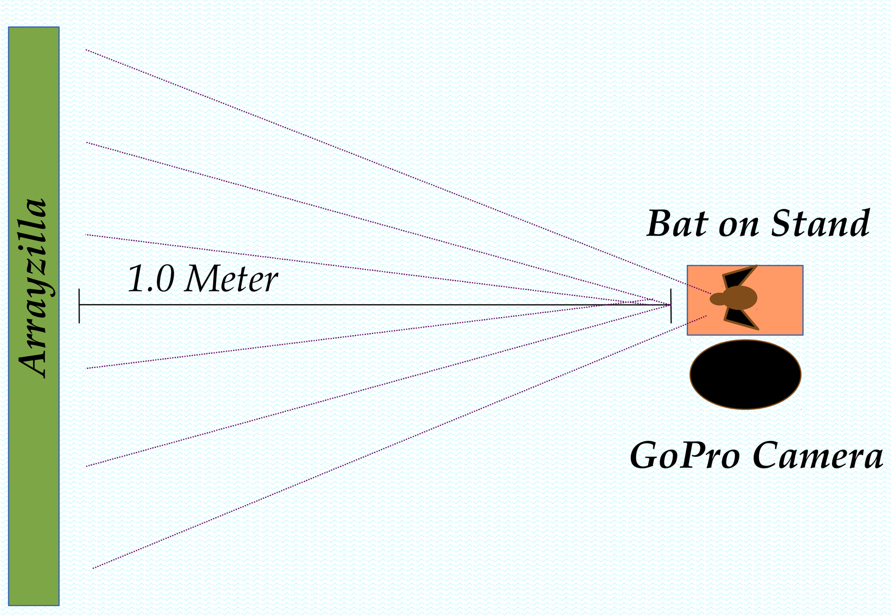
*- Recent research has shown that Bats produce calls in what are known as '**strobe groups**', or groups of a few pulses emitted close together.*

*-This helps show that bat calls are **dynamic signals**, rather than a single sound pulse that is sent and returned.*

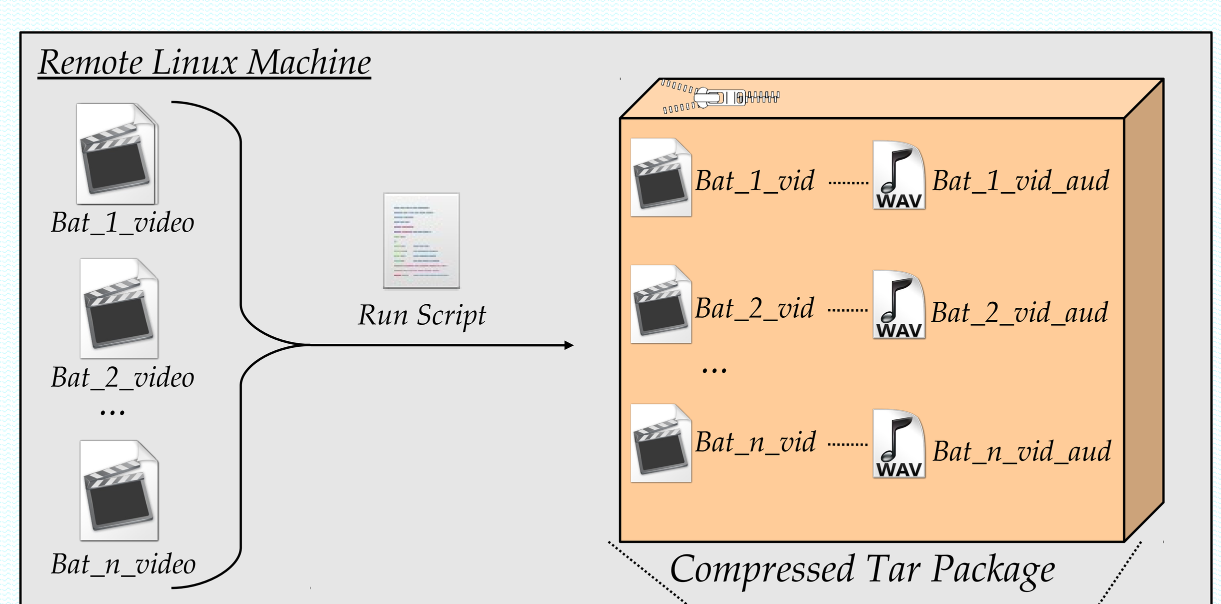
*- Current research is trying to show if, how, and why these strobe groups **vary from bat to bat**.*

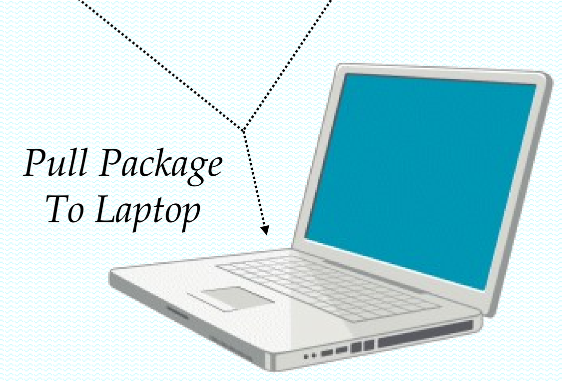
*-Additionally, Simmons lab wants to look at the **3 dimensional sound profiles** of the bat **sound beam across different frequency bands** and correlate these sound profiles to the **physical characteristics** of the bat's **mouth and ears**.*

*-To do this, the lab has constructed a **228 microphone array (arrayzilla)** for **spatial audio recording** of the sound beam, and uses a **GoPro camera** with an infrared filter for **high speed video capture** of the bat as it calls.*



Audio Extraction Scripts





*-In order to compare the **audio data** from **both** the **GoPro video** and a **high quality microphone**, the audio had to first be efficiently and correctly **extracted from the GoPro video file.***

*-**ffmpeg** and **sox** were installed to a **remote Linux machine** that the lab used and a **custom shell script** was written to strip the audio from the GoPro Video.*

*-The shell script automated the process by asking the user to select the file type to convert. When the user confirmed the files, the **script pulled the audio from the GoPro video files and saved it to a .wav file.***

*-The user was then prompted if they would like to **package the sound files for easier transfer**, and if they **wanted to include the original video files** in the package as well.*

*-After the package was created, the user could easily **transfer it to another computer** for analysis.*

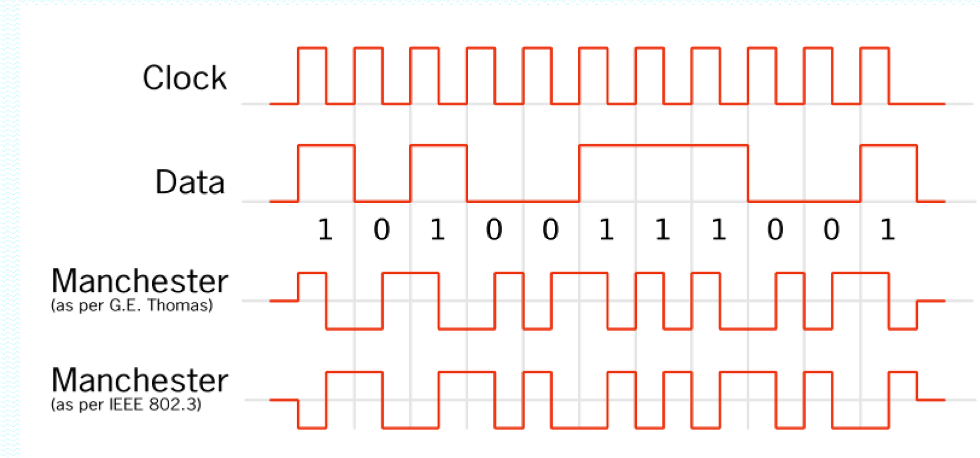
Decoding the Manchester Signal

- In order to synchronize all the different audio recording tools, an encoded Manchester audio signal is played and recorded by the devices. The signal gives **several pairs of 64-bit sequences** that represent increasing **identification numbers and timestamps**.

- A Manchester signal encodes temporal **data in the switches** between the bits. So for the G.E. Thomas Convention, going **from a 1 to 0** bit **represents a 1** and going **from a 0 to 1 represents a 0** in the Manchester signal.

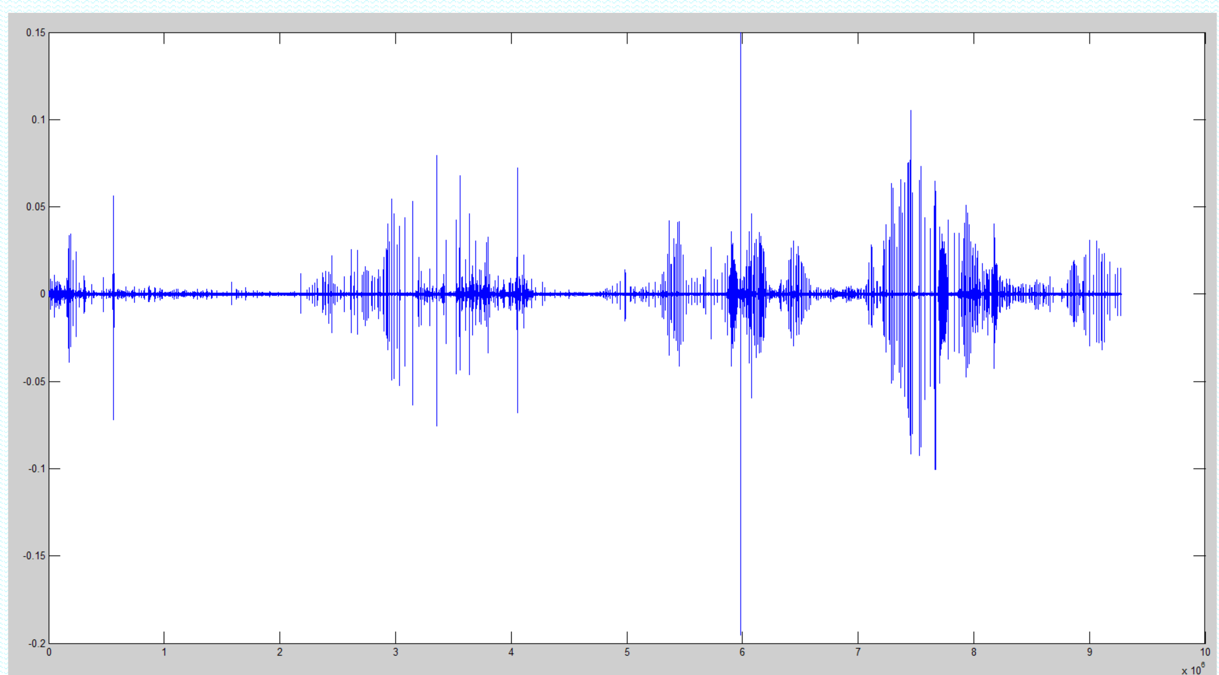
- In order to convert the audio signal into useful integer values, a script in MATLAB was created to **read the audio data** and **convert the Manchester values** to binary, and then **integer values**.

- First the script **found the 64-bit Manchester chunks** by looking at the **difference values of the data points** and applying a **time filter** (the chunks all lasted **4 *ms***). Once the chunks are extracted, the **switches were are** by again looking at the large **jumps in the data**. The switches were used to directly read the Manchester code, that was then **converedto binary**, and finally **integer values**.



A Description of Manchester encoding. The binary data bits are represented with two bits in the Manchester code. The switch between a 0 to 1 or 1 to 0 determines the data value. Every bit has both a 0 and a 1. It is the ordering that determines whether this sequence represents a 1 or 0 in the data.

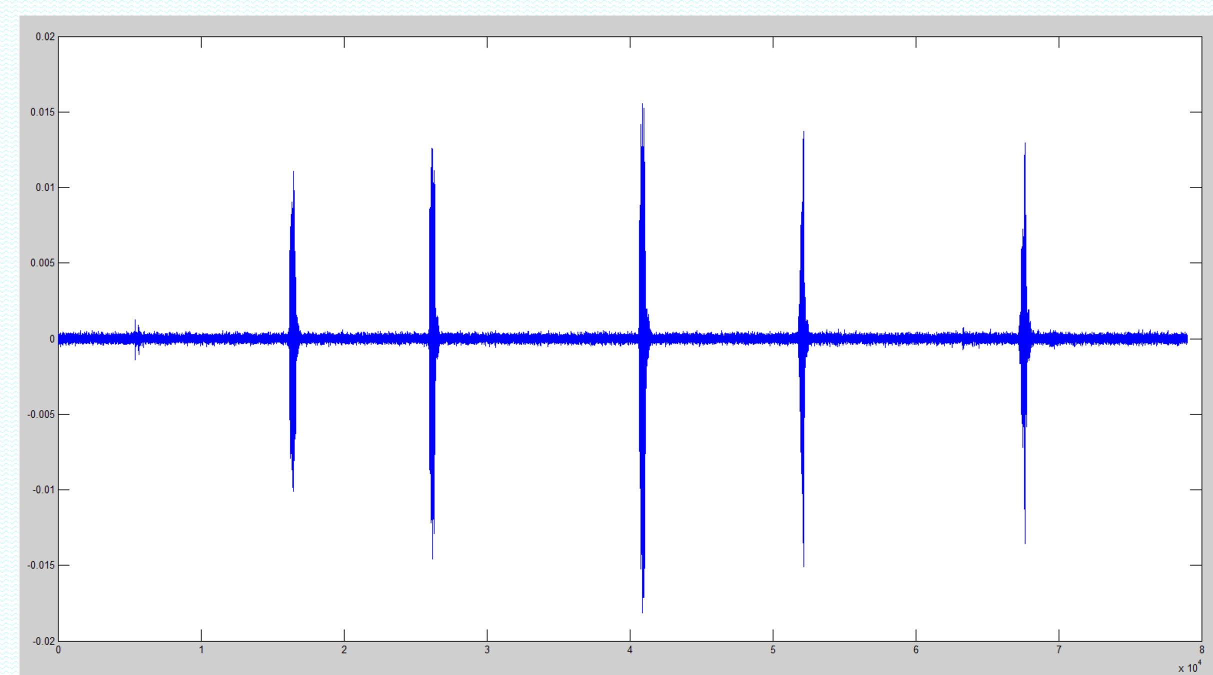
Extracting the Pulses



An example of the plotted bat call audio data.

-Previously, when analyzing the audio data, the bat calls had to be extracted manually. This made it very time consuming to obtain a large number of call samples to use for research.

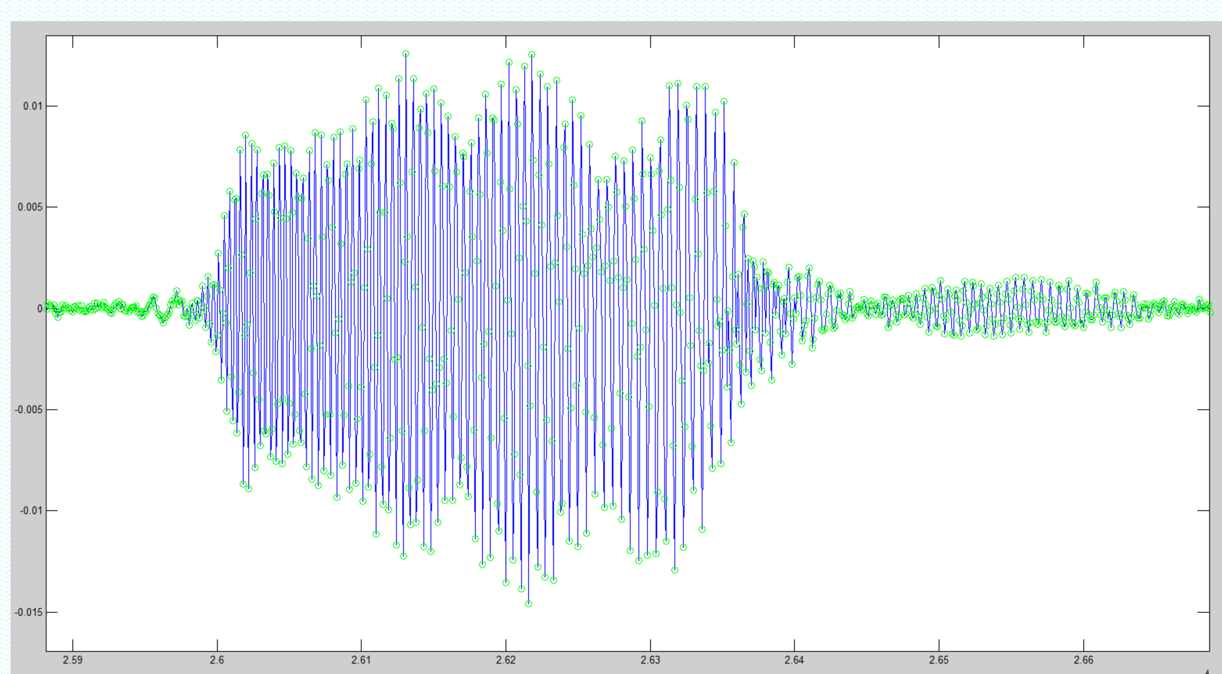
-A MATLAB script was made to help find and extract the bat pulses.



Zoomed in section of several bat calls

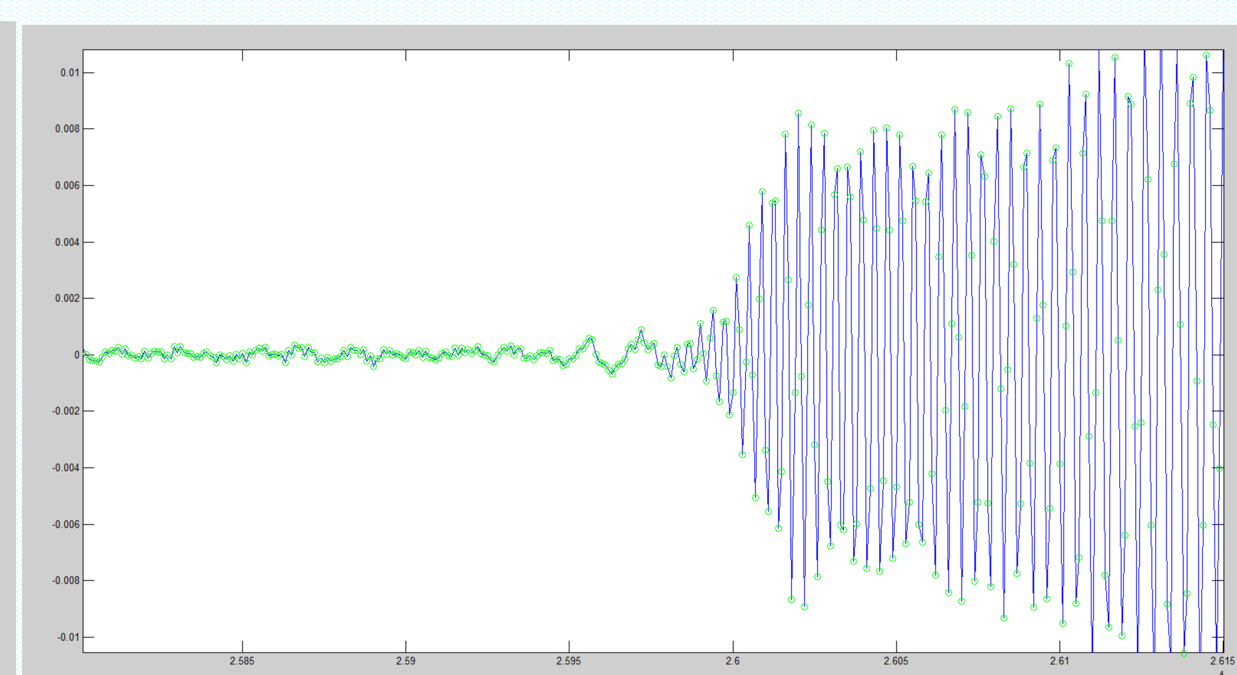
-The bat pulses have a specific profile that was exploited when locating them from the rest of the noise.

-The pulses have a rather consistent duration. This time window was used to determine if a sound was likely to be a bat call.



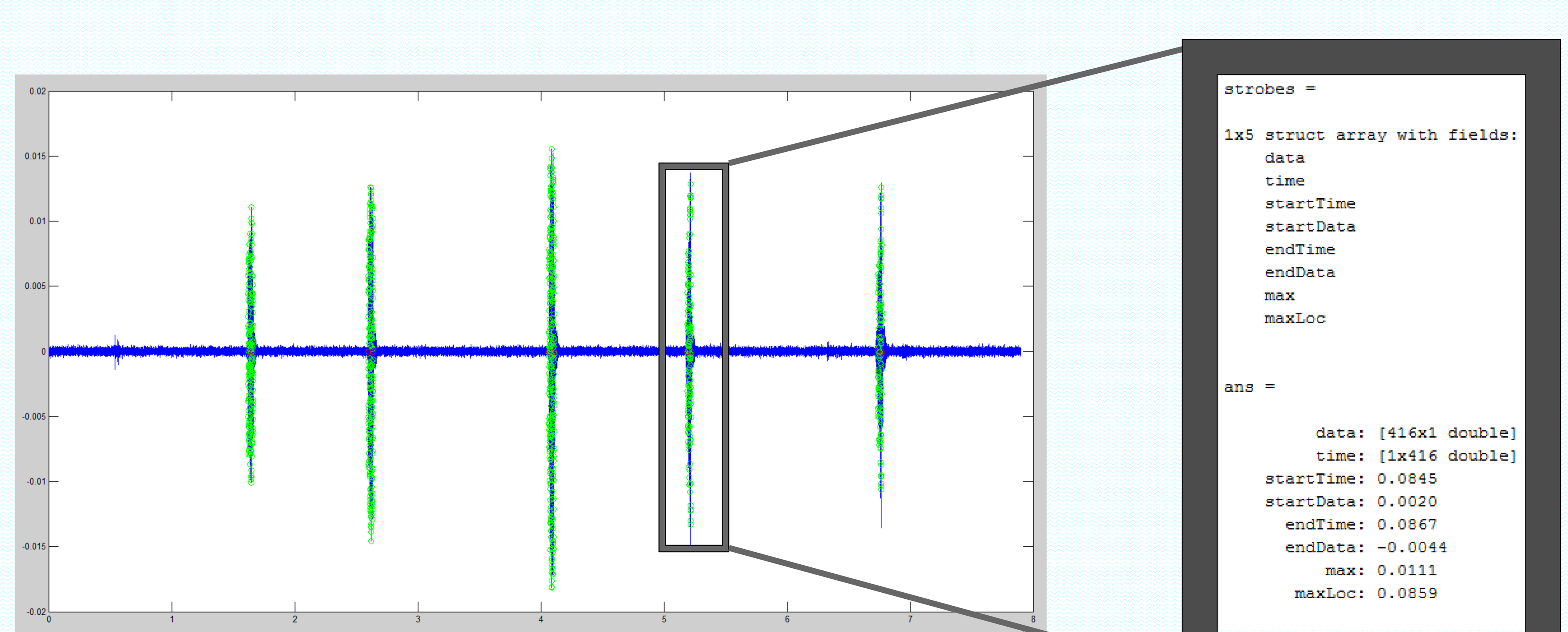
A zoomed in section of a single pulse. The green dots are the locations of the actual data points.

-Due to the audio being recorded at a constant sampling rate, when a call occurs, the difference between the data points during the calls is much larger than the difference between data the points for the background noise.



A zoomed in section of the front of a pulse. The green dots are the locations of the actual data points.

-By looking for this increase in the difference between data points, the sounds above background noise were located. Specific windows were then applied to further specify if the noise was a pulse from a bat call.



The estimated locations of the bat pulses. The green indicates where the script thinks there is a pulse.

The information of each pulse was gathered in a MATLAB structure for easy use in further analysis.

-After the pulses were found, they were added to a **MATLAB structure**, where **each element represented a different pulse**.

-The structure contained an **array of corresponding data values**, an **array of corresponding time values**, the **pulse start time**, the **pulse end time**, the **data start time**, the **data end time**, the **max amplitude** of the pulse, and the **location of the max amplitude** for **each pulse** in the structure.