ICP Signal Percussion Peak Detection

Zachary Whitlock
Written as part of a course project for Signals and Systems,
Oregon Institute of Technology, Winter Term, 2021

***Abstract*—A GNU Octave/Matlab function for the purpose of detecting percussion peaks in Intercranial Pressure signals is designed, developed, and tested in this report. The algorithm uses IIR filters to remove quantization noise and to lowpass filter the signal to it's first fundamental frequency. The algorithm returned lengths of indexes accurate to <3% for stable, nearly time-invariant signals, and lengths of indexes accurate to <20% for less stable, very time-variant signals.**

***Index Terms*—Beat Detection, Percussion Peaks, Matlab, Digital Signal Processing**

I. Introduction

The measurement and analysis of Intracranial Pressure (ICP) data pressure data is often done with simplified algorithms such as mean value and spectral power. The first step is taken in this report to providing another method of analysis by looking at the beat-by-beat changes in the base signal. In this way, differences in aspects such as the distance between (in Y and X) a percussion peak and a dichrotic peak can be measured and characterized with ease. Fig.1 shows the percussion peaks on a raw ICP signal, and Fig.2 shows an example of an analyzed signal with it's percussion peaks labeled. Fig.2 was labeled by a professional or by a better algorithm, the data set was provided.

The objective of this report is to create a Matlab/GNU Octave function designed and developed for the purpose of finding the percussion peaks of ICP data. The function is characterized by the following: fi = PressureDetect(xi, fs, pf);

- xi = Input signal
- fs = Signal sample rate (Hz). Default = 125Hz
- pf = Plot flag: 0=none (default), 1=screen
- fi = Percussion peak (P1) index, samples

To use the function, you feed it an input pressure signal, and optionally specify the sample rate and print flag. If the print flag is set, the function will plot a segment of the data for observation and analysis. The function will analyze the input data and return a set of indexes representing the X coordinates of the peaks, as well as corresponding Y values.

II. Methodology

A. The Dataset

The dataset used for testing and validation was obtained from the class resources and is comprised of approximately 1 hour or 3600 seconds of ICP signals. The signals 'icp1' and 'icp2' are provided in the ICP.mat file. Also included in the ICP.mat file are the indexes of pre-recorded peaks already located. Each signal is largely a clean signal with slight variations and a section of artifacts and signal distortion. Both signals share a sampling frequency of 125Hz. The signal 'icp2' is generally cleaner and yields more reliable results with this algorithm.

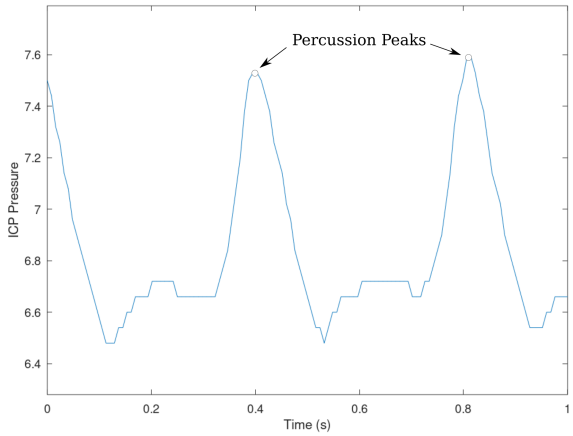

Fig. 1. Percussion peaks on a ICP signal. This is the 'icp1' dataset. In the beginning of this dataset there is no typical ICP pattern. As shown, the pressure has a peak before the tall percussion peak, instead of after it.

B. Algorithm Overview

There were a multitude of steps involved in choosing an algorithm. The end result is similar to how M. Aboy's algorithm works [1]. In designing the algorithm, the signal(s) were examined in a spectogram in order to see what the frequency components would do over time. From there, I settled on a four-step approach. I determined that the first fundamental frequency did not move around significantly, and with some testing I found that low-passing the signals to only include the first major frequency component yielded consistent results. The peaks of the low-passed signal are found with a find-next-max loop and recorded. From there, each lowpass-peak is examined and an area of the original signal is searched centering around the lowpass-peak. The area searched is slightly less than the period of the low-pass filter's cutoff frequency, which makes it impossible for a single lowpass-peak to find multiple percussion peaks in the ICP data. The basic design elements are laid out in Fig.3.

C. Quantization Lowpass

The first step of the journey is to remove the quantization noise created by sampling the signal in hardware. The highest significant frequency is manually determined and a lowpass

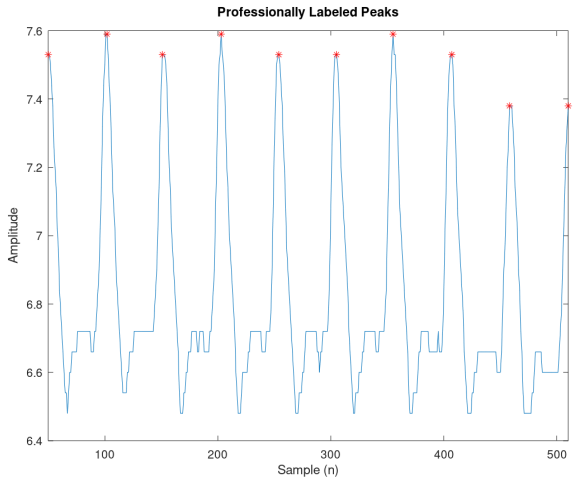

Fig. 2. Professionally labeled Percussion peaks on a raw ICP signal. This graph represents the 'icp1' dataset plotted against the 'dDT1' peak set.

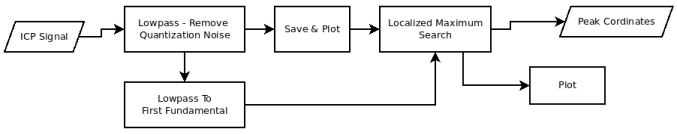

Fig. 3. Basic flow diagram of the PressureDetect function

filter is implemented with a 5th order Butterworth filter. For the ICP signals, the cuttoff frequency is 15Hz, but 10Hz can be used for the ABP signals. Fig.4 Shows the response of the quantization filter used.

D. Fundamental Lowpass

The second lowpass filter used is a 10th order Butterworth filter. The higher order helps to ensure a clean near-sinusoid without secondary peaks which would potentially interfere with the algorithm. Once a near-sinusoid signal is created,

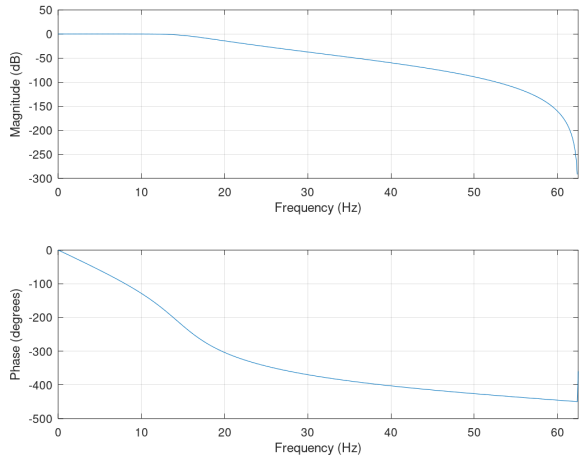

Fig. 4. Frequency response of the quantization filter used

the signal is looped over and the peaks are located. The peak-detection algorithm for the simplified signal works by looking forward from a previously located peak. The algorithm starts about half a period forward of the last peak, ideally starting in an inverse peak or valley. It then scans forward for roughly one period of the signal and locates the highest point in this range. Fig.5 shows this search zone. Fig.6 shows the frequency and phase response of the 10th-order Butterworth filter. It's evident that this second filter has a much sharper response.

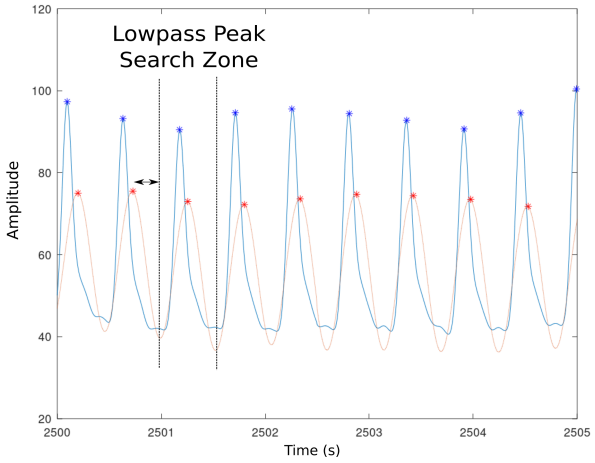

Fig. 5. Search algorithm for the peaks of the low-passed signal applied to the 'abp1' signal provided for the project. The orange signal is low-passed and the blue is the original signal. Red and blue points represent the peaks the algorithm has found

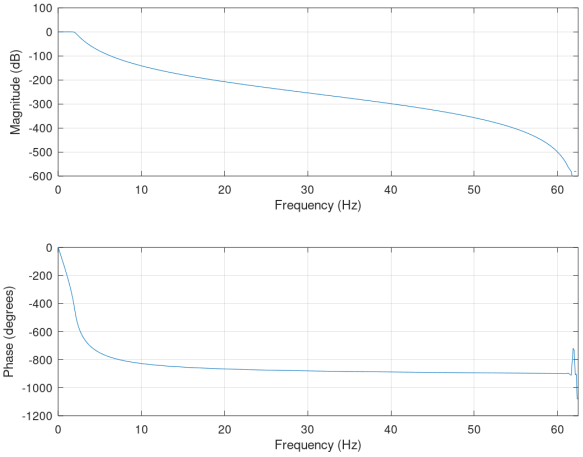

Fig. 6. Frequency and phase response of the 10th-order Butterworth filter used to lowpass the signal to the first significant frequency.

E. Percussion Peak Search

To find the percussion peaks themselves, the peaks of the low-passed signal are used as reference points. A period of time is examined on each side of the low-passed peaks, but

instead of the lowpass signal being examined for maximums the signal from the first filter is used. As such, the peaks are located in the quantization-error-removed signal and recorded. The period of time searched is equal to the period of a 2Hz signal. 2Hz was chosen because it's the average period of the first significant frequency for the icp signals. By using the peaks of the low-passed signal we have a resilient search algorithm that works over a range of variations in the original signal. Fig.7 shows the search range of the algorithm as it goes over the lowpass peaks.

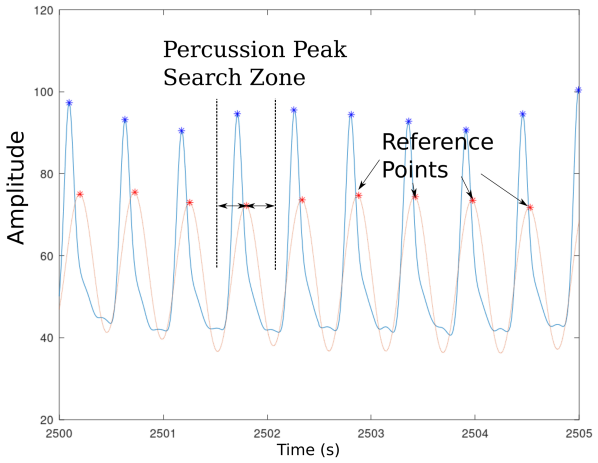

Fig. 7. Search range and reference points for locating the percussion peaks in "icp2"

III. Results

A. Expected Results

A high-performance algorithm would accurately recreate coordinates labeled by an expert or another algorithm. Therefore, the expected results of this algorithm would be to closely recreate the coordinates given by the class Professor. The most complete datasets for the peak values seem to be "dDT1" and "dDT2" for "icp1" and "icp2" respectably. TABLE1 shows all of the available indexes for comparison. I'm making an assumption that the smaller lengths mean higher accuracy and may imply a human indexed them manually.

TABLE I
Peak Indexes Available

Name	Length (N samples)
d1	8635
d2	5249
dDT1	8350
dDT2	5241
dJM1	1431
dJM2	869
dTT1	154
dTT2	87

B. "ICP1" Results

It is difficult to directly compare the index arrays since the function returns a signal which has been filtered and no longer is one-to-one with the original signal. Therefore, I'm visually comparing the results of the function as well as comparing the lengths of the index array returned by the function and the length of the given index arrays.

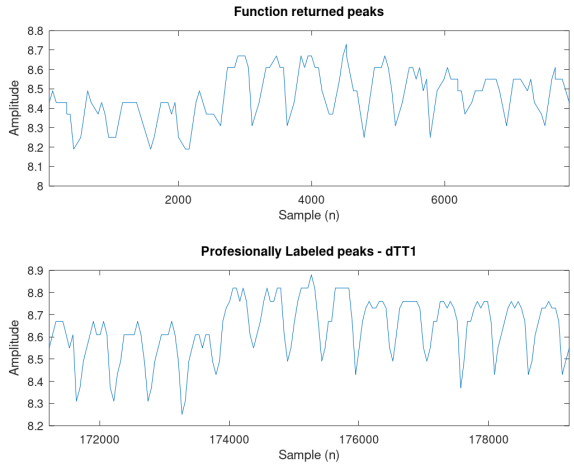

Fig. 8. Comparison of the dTT1 index array versus the result from the function. the dTTx arrays are the shortest and presumably the most accurate. It can be seen that the same general shape is seen for both graphs, but the function does differ in a lot of ways.

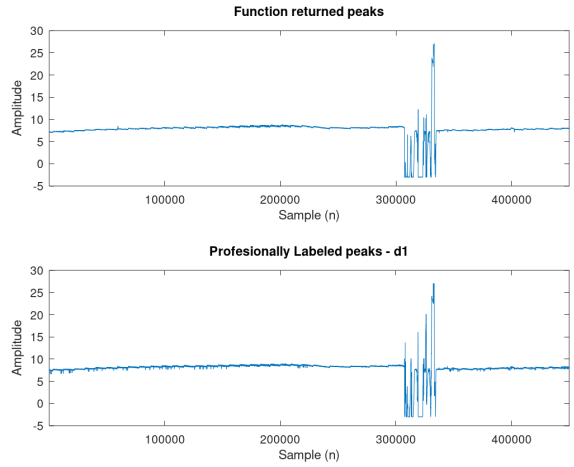

Fig. 9. Comparison of the d1 indexes with the result from the function.

As seen in Fig.8 the results aren't exactly the same as the given index arrays. However, the results of the PressureDetect function closely follows the given index arrays as is evident in Fig.9. TABLE2 shows the error in the array length between the function results and the relevant index array. The cleaner signal "icp2" yielded better error percentages by far. Only the dTT2 error was greater than 1% for the second ICP signal being passed through the function.

TABLE II
PEAK INDEX COMPARISONS

Name	Length (N samples)	Function Result (N sampels)	% Error
d1	8635	7251	19.09%
d2	5249	5266	0.32%
dDT1	8350	7242	15.30%
dDT2	5241	5270	0.55%
dJM1	1431	1228	16.53%
dJM2	869	867	0.23%
dTT1	154	135	14.07%
dTT2	87	85	2.35%

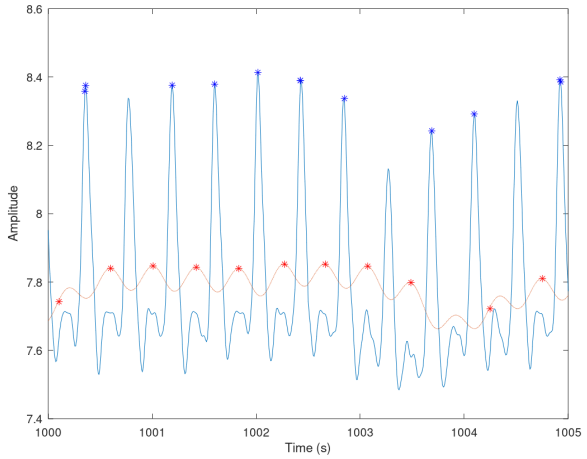

Fig. 10. Plot of percussion peaks against the lowpass'd signal

Fig.10 shows a segment of the output data from the function as well as the lowpass filtered signal used to calculate the peaks.

IV. DISCUSSION

We can gather from the Results that this is a low performance algorithm. In Fig.10 it's even visible that one of the peaks is mis-labeled and the previous peak was double indexed. Additionally, the algorithm did not produce very accurate results with the more unreliable "icp1" signal as evidenced by the high error percentages in TABLE2. Even though this isn't the most accurate algorithm, it is significant that a simple algorithm can detect percussion peaks of an ICP signal. This could be used to detect changes over time in the signal, as well as measure aspects such as the amplitude and period variance of the peaks.

This double-indexing issue can be resolved by forward-biasing the search algorithm to look forward from a low-pass peak instead of around it. This was attempted after the report was written and the d1 error was improved to be 3.277%. Further tuning of the function in terms of frequency cuttoff points yielded an even greater improvement to 0.28%. It is probable then that this basic algorithm as an idea has more potential than was realized in this report. Ideally, the function would automatically detect the lowest significant frequency and potentially track it over time to provide the most accurate low-pass peaks possible. It also appeared that adding in some component of the second significant frequency had positive effects on the accuracy.

For future research, and besides further improving this algorithm, it would be interesting to do what [1] suggests and attempt to further analyze ICP signals with the help of blood pressure signals of the same timeframe.

V. CONCLUSION

In conclusion, a semi-robust algorithm was designed and implemented in the form of a GNU Octave/Matlab function. This function, PressureDetect, prints out a section of the input signal for analysis if desired, and always returns an array of indexes representing the percussion peaks of the input. For clean signals, the function can be expected to be accurate to less than 3%, and for distorted and extremely time-varient signals it can be expected to be accurate to 20%.

VI. REFERENCES

REFERENCES

[1] M. Aboy, J. McNames, and B. Goldstein, "AUTOMATIC DETECTION ALGORITHM OF INTERCRANIAL PRESSURE WAVEFORM COMPONENTS", 2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society.