Developing Infant Suck Detection Interface

Amy Adair, Abiti Adili, Zachary Bradshaw, Joshua Brock, Brandon Dellucky, Heewon Hah, Margarite LaBorde, Hugo Leira, Miles Robicheaux, Sima Sobhiyeh, Mary Wang, Jerome Weston

*Louisiana State University*

*Supervised by Dr. Wolenski and Dr. Harhad*

I. Introduction

For this project, we collaborated with [Abby Duhé](mailto:Abby.Duhe@pbrc.edu), a researcher at [Pennington Biomedical Research Center](https://www.pbrc.edu/), in order to improve the data analysis program used to detect sucks in signals obtained from infant bottle feedings. The focus was to develop a user-friendly and efficient graphical user interface (GUI) using the Matrix Laboratory (MatLab) programming software. We began by improving the layout of the original GUI used to analyze the signal. Once we designed a more appealing interface, we studied and improved upon the previous signal processing and suck detection methods.

Due to limitations, the previous program was inaccurate and ambiguous at best. To address this, we implemented proper orthogonal decomposition (POD), which is a method of data analysis that creates a database of prototypes resembling typical sucks. Therefore, this new GUI—called the Infant Suck Detection Interface—can now detect distinct variations in initial data. This may be beneficial when researching the effects of age and other factors on infantile feeding.

In addition to improving the accuracy of suck detection, we also introduced filtering and preprocessing procedures. Through Fourier analysis and convolutions, the program now reduces noise efficiently, helping to identify sucks. These improvements, along with the new suck detection process, drastically reduced the run time for the program, allowing researchers to obtain results almost instantly.

The Infant Suck Detection Interface can now be used effectively for research on infant sucking behavior while offering adaptability based on variable conditions. Further instruction on how to operate this program can be found in the [Infant Suck Detection Interface Manual](file:///C:\Users\Amy%20Adair\Dropbox\Summer2016\Manual\Manual%20Rough%20Draft.pdf).

II. Former Project and Subsequent Improvements

The original GUI had multiple issues with aesthetic presentation, consistent suck detection, and run time. Initially, the program presented the user with multiple windows popping up in succession, followed by a wait bar that slowed the program further, before finally presenting the graph. The results of the signal analysis were ambiguous, and the performance time was exceedingly slow.

In contrast, the design we presented has a single window with three tabs. The first tab contains input fields for the infant’s demographics. The second tab holds the parameters, the signal’s graph, and a table with overarching statistics. The third presents the primary data gathered from the signal analysis; previously, this was saved to an excel file without being displayed to the user. The user now has a choice to save the data, using a toolbar icon. Also located on the toolbar are buttons to open the signal file, zoom in on the graphic, and zoom out. The wait bar and cancel option have been removed; instead, the parameter display grays out to indicate that the program is running.

The back-end code has also been overhauled to eliminate slowdown. These changes include the deletion of the wait bar, reordering checks with the program, and reducing the use of recursive loops within the program. The use of nonstandard MatLab packages is no longer necessary, as the functionality of such packages has been maintained with the basic package only.

The greatest improvement to the code was the elimination of the nested while loops previously used to detect sucks. The former code checked every part of the signal and compared it with every possible window size. The current program does not even consider parts of the signal below the threshold. It also begins by recognizing potential suck windows first and only tests these portions. The potential windows are a purpose overestimate to reduce the chance of missing any sucks, and the window size is adjusted for maximum suck correlation.

After all of these improvements, the modern GUI is significantly more user friendly. The program experienced a 22x speed up over the original algorithm. With the added functionality of saving graphics and specialized prototypes, this GUI offers an efficient and convenient method for data analysis.

III. Proper Orthogonal Decomposition

The original GUI used for data processing detected fewer sucks than intuitively should be present within the signal; as such, it became necessary to examine the procedure that detected the sucks to improve fidelity. Initially, the program employed a single “prototype” of the suck—a parabola of the form . When searching for sucks in the input signal, it would compare windows of the signal to the prototype until a sufficiently similar shape occurred in the data. This was labelled as a suck.

The use of a single prototype, particularly one so simplistic, raised concern that certain sucks may be overlooked. A better model would utilize multiple prototypes, preferably from actually suck data rather than an idealized parameter. With this in mind, proper orthogonal decomposition (POD) was a natural solution. The POD method allows the user to examine known signals, store known sucks, and form prototypes based on this data.

The mathematical description of POD is outlined in Appendix I. For our purposes, a PODGUI was developed to create the new prototypes. This allowed for easy execution of the following steps. First, the user loads a signal and selects snapshots of previously known sucks. These snapshots () are column vectors of the form:

These vectors are then collected in a matrix, *U*, such that:

From here, we follow the form of section 3.2.2 in Appendix I to determine the modified correlation matrix and the subsequence matrix *W*. This matrix *W* contains columns of the normalized eigenvectors () corresponding to the first *m* eigenvalues of the modified correlation matrix. At this point, the matrix is saved as a prototype for further use in signal processing.

In the Infant Suck Detection Interface GUI, the user can then choose to select a prototype and choose the aforementioned *W* creation. Then, the program projects sample of potential sucks (*s*) into the subspace of the new prototype. This projection is compared to a user-specified cutoff (q). Should the normalized projection be greater than or equal to the cutoff, the data sample is counted as a suck. The projection is calculate as follows:

This method of developing prototypes boasts the advantages of using multiple, more accurate prototypes that could later be customized to reflect particular subsets of infants.

IV. Preprocessing

In addition to refining the suck detection method, a preprocessing protocol was implemented to reduce noise and remove the baseline of the signal. Previously, the noise was removed using a MatLab function found only in an additional CurveFitting toolbox; this posed problems for users with only the basic package. Furthermore, the baseline was formerly a user input—allowing for a high degree of inaccuracy. The new preprocessing function utilizes a band-pass filter functionality to remove both of these without the need for user input or additional packages.

To accomplish this, a discrete Fourier transform—implemented through a fast Fourier transform (FFT)—was performed on the original signal. Then, the high-amplitude frequencies around zero were eliminated. The resultant function was then put through an inverse FFT to regain a signal in the time domain. This eliminated the baseline.

Next, a moving average convolution was preformed to extract high-frequency noise. This smoothed the input signal. Since this function was written completely using the basic MatLab package, the additional CurveFitting toolbox was now unnecessary.

Preprocessing occurs now whenever a data set is loaded into the GUI. No user input is required, and the entire process is quick and allows for more accurate suck detection overall.

V. Conclusion

The original goal of this project was to recreate the original GUI and improve its user interface and data analysis methods. In regards to this, we have achieved a 22x speed up over the original program, developed a more accurate detection method, and added user functionality such as saved graphics and the removal of additional toolboxes. The preprocessing function eliminates noise and user-input baselines. The implementation of proper orthogonal decomposition to develop prototypes means more accurate suck detection. These are all vast improvements over the original status of the program.

For further work, we intend to add a weighting system to the POD method to form a spectral POD. Additional, specific prototypes tailored to infants of specific gender or other characteristics could be developed to measure infants against others of their demographic. This would allow for more pertinent comparisons for each child.

In the short term, the current GUI has been delivered to Pennington and is in operation there. The current product has shown to be much improved over the original; however, we hope to continue exploring this project and its applications.

VI. Acknowledgements

We would like to thank Dr. Wolenski and Dr. Harhad for their guidance throughout this project. We would also like to thank Abby Duhé and LeAnne Redman at Pennington Biomedical Center for the opportunity to work with them and also their willingness to communicate and work with us throughout the whole of the project.

VII. References

Adnani, F. (2006). *Detection, Identification and Classification of Suck, Swallow and Breathing Activity in Premature Infants During Bottle-Feeding.* Virginia Commonwealth University, Biomedical Engineering and Bioengineering Commons. Retrieved from http://scholarscompass.vcu.edu/etd/1505

Buljak, V. (2012). Chapter 3: Proper Orthogonal Decomposition and Radial Basis Functions for Fast Simulations. In V. Buljak, *Inverse Analyses with Model Reduction* (pp. 99-153). Berlin: Springer-Verlag.