Fall 2019 ME459 Final Project Report
University of Wisconsin-Madison

Implementation of Shear Wave Tensiometry Processing Algorithm in C

Jonathon Blank
Dylan Schmitz

December 14, 2019

Abstract

Shear wave tensiometry utilizes two miniature accelerometers to compute the shear wave speed in a soft tissue, which can, in turn, be used to compute the axial soft tissue stress [1]. The Neuromuscular Biomechanics Lab uses a variety of algorithms to process outputs from these two accelerometers to compute wave speed. Some of these include speckle tracking of B-mode ultrasound images, interpretation of radiofrequency (RF) ultrasound data, fast Fourier transformations, and, for the purpose of this project, correlations between the two accelerometer signals. However, all are written in MATLAB, which is an interpreted language, thus making the algorithms timely to execute and infeasible to use in real-time. The purpose of this project is to translate the post processing algorithms used to compute wave speed to C. By doing so, we can enhance the speed of the real-time computing of shear wave speed.

Git Repo: https://github.com/jlblank5/ME459_FinalProject_BlankSchmitz.git

Contents

1. General information

- Home department: Mechanical Engineering
- Current status: PhD students
- Individuals working on Final Project: Jonathon Blank

  Dylan Schmitz

- I am not interested in releasing my code as open source code.

2. Problem statement

The purpose of this project was to convert an algorithm that is frequently used in our lab from Matlab to C. We did this because of the time efficiency that algorithms written in C can provide. To summarize, the code that we are converting is a post-processing algorithm that computes the time delay in a correlated acceleration in response to a mechanical excitation passing by the successive accelerometers. Specifically, we will focus on implementing a code wrapper that filters and computes a normalized cross-correlation of two data signals, before making computations in response to the sorted mechanical tap signal. By doing so, we can compute a shear wave speed, which we use as a proxy to gauge tendon and ligament stress in humans. Having a version of this code written in C could potentially allow our group to gauge these stresses in real-time due to the timing efficiencies that we can leverage.

3. Solution description

We began by constructing a wrapper file that the end-user can use to interface with the library of functions that we created. We then implemented a read function that loaded a LabVIEW file as a data matrix with three columns: the tap signal, the first accelerometer signal, and the second accelerometer signal. The next function implemented was a low and high pass Butterworth filter to eliminate noise in the accelerometer signals that would interfere with our cross-correlation. Then, we utilized a sorting algorithm to determine when “push” and “pull” tap events occurred, which corresponded to when a piezoelectric actuator pushed into and pulled away from the skin superficial to the tendon, respectively. Finally, we implemented a cross-correlation algorithm to compute the time delay in accelerometer wave arrival, which in turn was used to compute the shear wave speed for a known physical distance between the accelerometers.

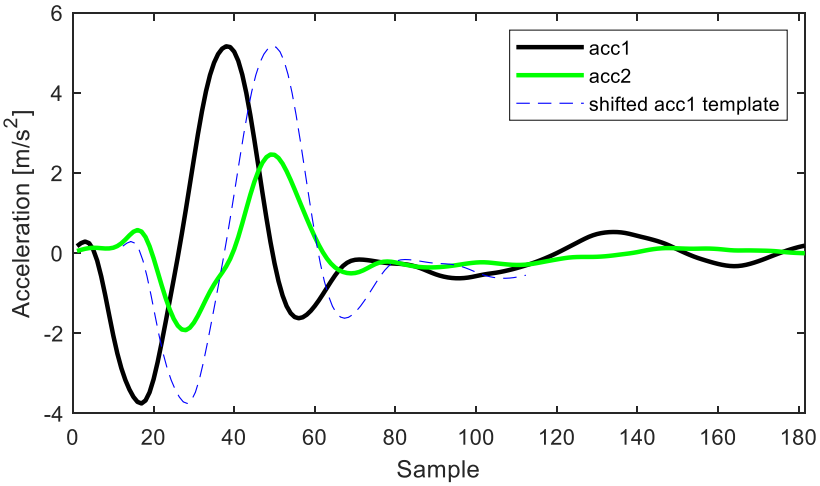

Figure 1: Two filtered accelerometer signals are parsed into distinct tap events (single event shown). The temporal cross-correlation between a template taken from acc1 and the reference signal acc2 gives a maximum when the time delay corresponds to the shear wave arrival at each measurement point. From this, we compute wave speed as the distance divided by this time shift.

To check whether the resulting wave speed calculation was correct, we compared its solution against the current Matlab algorithm using visual inspection. To determine which of the two programs was more efficient, we timed the execution of both programs using the same example .lvm file. Finally, we explored the use of Doxygen to document our program, however we felt that this was unnecessary to pursue beyond an introductory level at this point in the project.

4. Overview of results. Demonstration of your project

The computations returned by our wave speed algorithm written in C are correct. When comparing against the matlab solution, we have minimal error between the two algorithms (Fig. 2).

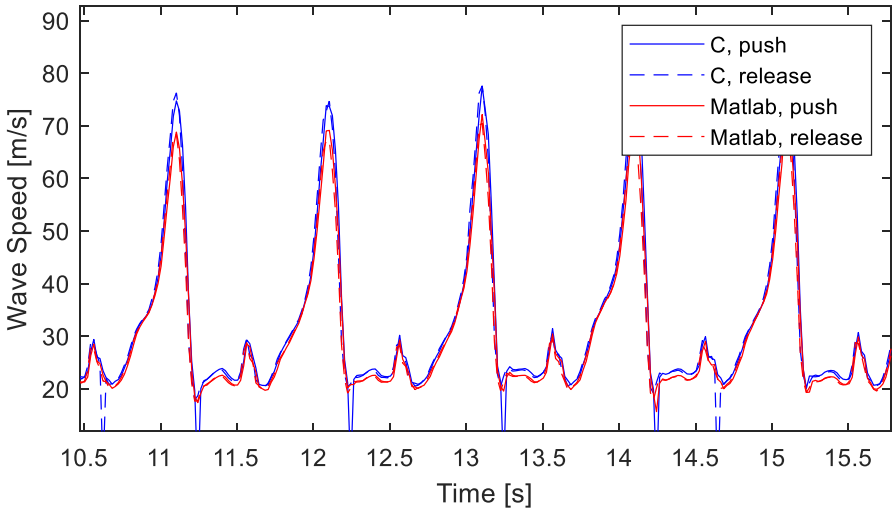

Figure 2: Achilles tendon wave speeds computed for walking with an active ankle exosuit.

The error that occurs at the peak of the wave speed signal is likely due to the additional post-computation filter that is applied in the Matlab algorithm and not within our C algorithm.

Additionally, the Matlab and C algorithms had very comparable execution times: 6.8 seconds for Matlab and 8.5 seconds for C.

To further improve our algorithm, we will take advantage of temporal locality by making more efficient use of variables in the cache. Spatial locality has already been considered, as our increments are small within array indexing our data matrices are handles in row-major order. Our goal is to implement this executable into our lab within the next semester.

5. Deliverables:

We have provided a fully functional wave speed code. Inside of the git repository, the wrapper function, **wrapper.c**, is used to specify user parameters and call the extensive amount of functions that we have implemented, including:

- **readLVM.c**
- **writeCSV.c**
- **filtfilt.c**
- **butterLP.c**
- **butterHP.c**
- **sort.c**
- **xcorr.c**

The following libraries provide the function above. Included below are necessary external libraries:

- **fileIO.h**
- **filter.h**
- **sort.h**
- **xcorr.h**

To run the code, the user simply needs to have the sample data **check2.lvm** inside of the current directory. The compile command is as follows:

gcc wrapper.c filtfilt.c butterLP.c butterHP.c sort.c xcorr.c readLVM.c writeCSV.c -Wall -O3 -o ./wrapper.exe

To execute the code, simply run wrapper.exe.

6. Conclusions and Future Work

The code that we have provided is a useful tool that can be utilized to improve research efficiency within the Neuromuscular Biomechanics Lab. Continued work on this will seek to improve runtime efficiency so that it can be used as an executable in real-time with live tensiometry collections, which will provide a live visualization of results during experiments.

References

[1] Martin, J.A., Brandon, S.C.E., Keuler, E.M. *et al.* Gauging force by tapping tendons. *Nat Commun* **9,** 1592 (2018) doi:10.1038/s41467-018-03797-6.