

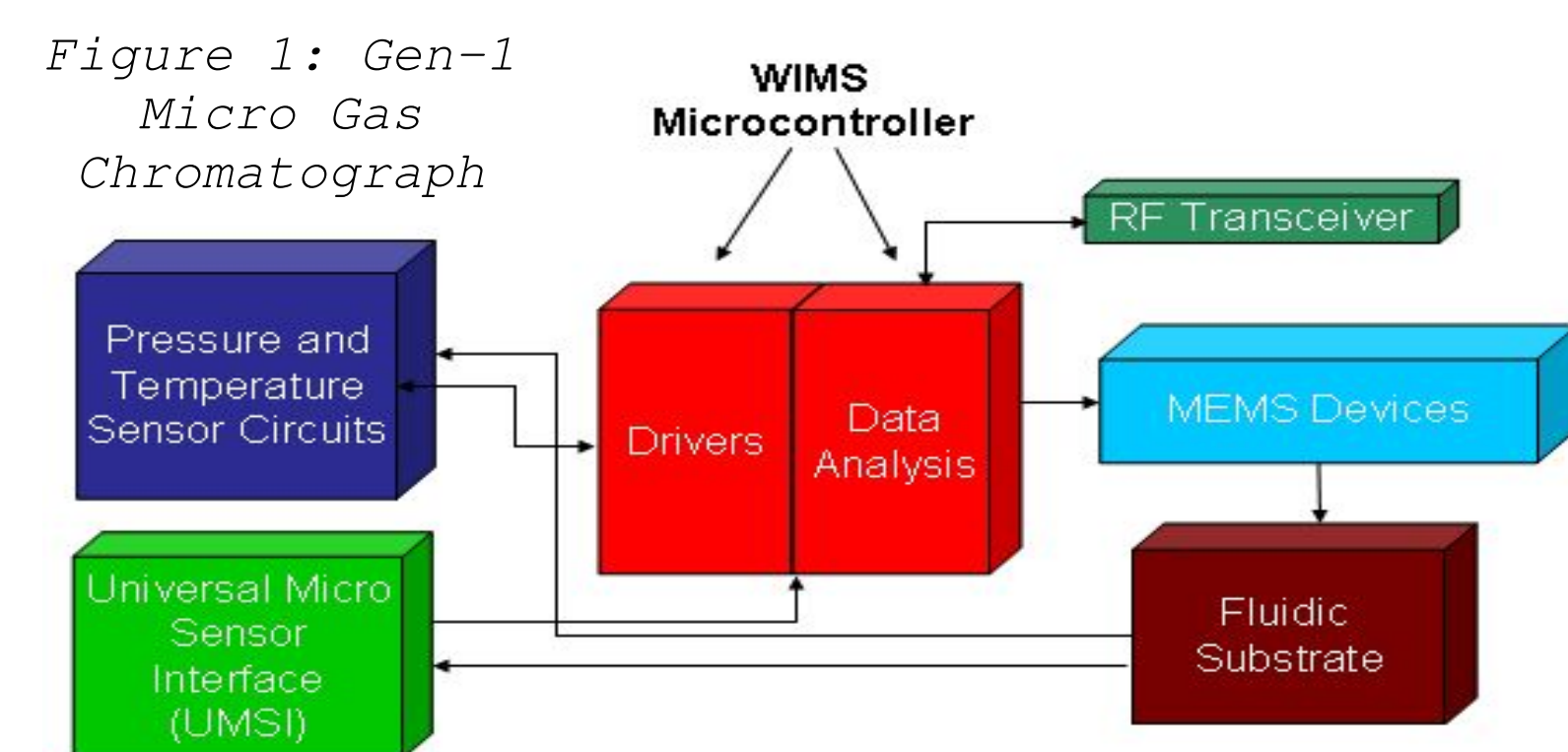
# Development of Application Code to Support the WIMS MCU Interface to the EMT

Students: Giselle Agosto, José J. Díaz, Louis García, Ramón Mercado, Carlos Pérez  
Faculty Advisor: Nayda G. Santiago, Ph.D.

Electrical and Computer Engineering Department  
University of Puerto Rico – Mayaguez Campus  
Mayaguez, Puerto Rico 00681-9042

## Introduction

This project aims to develop application code for the WIMS Microcontroller. This application code will control the Micro Gas Chromatograph ( $\mu$ GC), as well as collect and analyze its data. Figure 1 depicts the  $\mu$ GC, here you can see the central role that the WIMS Microcontroller (MCU) plays on the  $\mu$ GC.



The first part of our work consists on supporting the WIMS MCU interface to the  $\mu$ GC by generating all the necessary drivers for a successful interface. These drivers are codes that provide the microcontroller the ability to interact with all the other modules on the  $\mu$ GC. The driver's functions ranges from device manipulation, to control the  $\mu$ GC, all the way to communication for collecting the data generated by the  $\mu$ GC.

Figure 2 shows a normal  $\mu$ GC plot, where each peak represent one constituent of the gas mixture sample by the  $\mu$ GC. The second part of our work is to analyze this data and determine each peak's location, height and area. This information is sufficient to fully characterize any gas mixture.

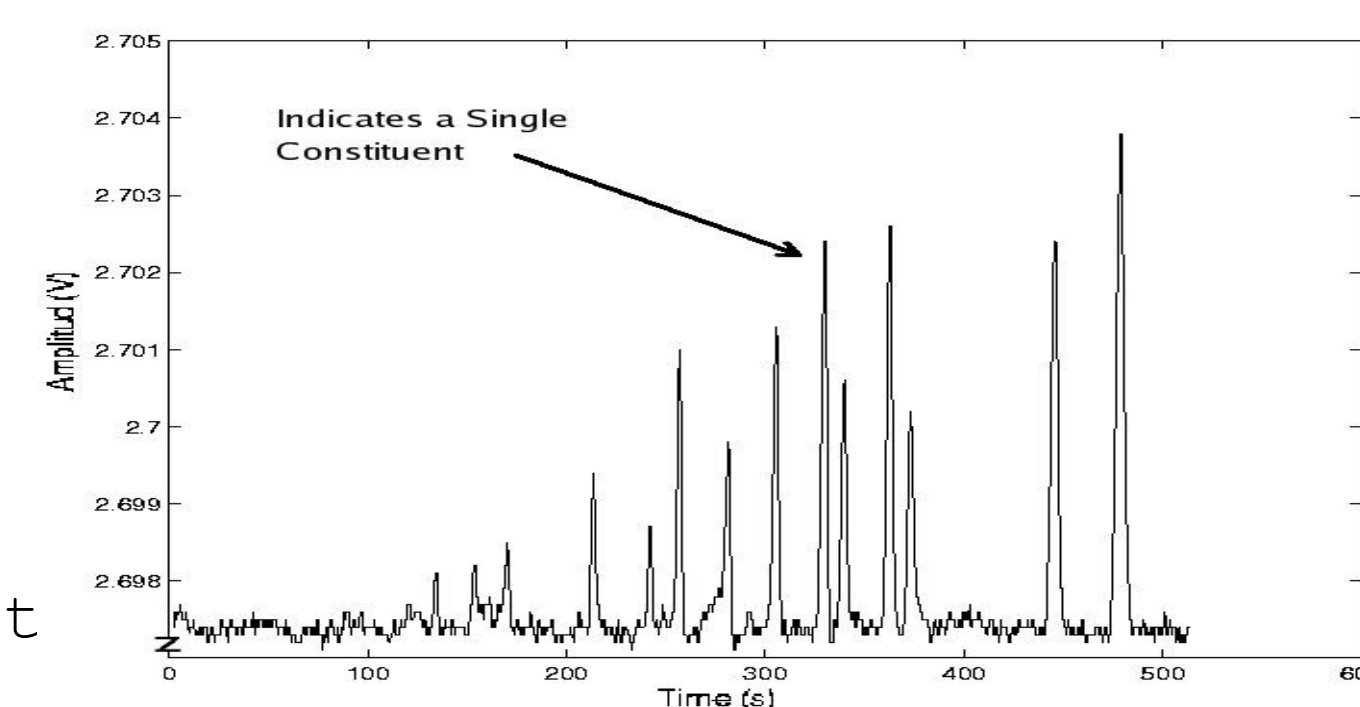


Figure 2: Micro Gas Chromatograph Plot

## Methodology

The application code is written in C programming language. The data analysis code is compiled with the GNU Compiler Collection (GCC), while the driver codes are compiled with WIMS' own low power compiler.

As shown in Figure 3 the data analysis code is divided into three main processes: smoothing, peak detection and area calculation.

The smoothing process filters the data as while also compressing it. The  $\mu$ GC data for this system has a characteristic high frequency noise component. To filter this noise the smoothing process utilizes a moving average low pass filter.

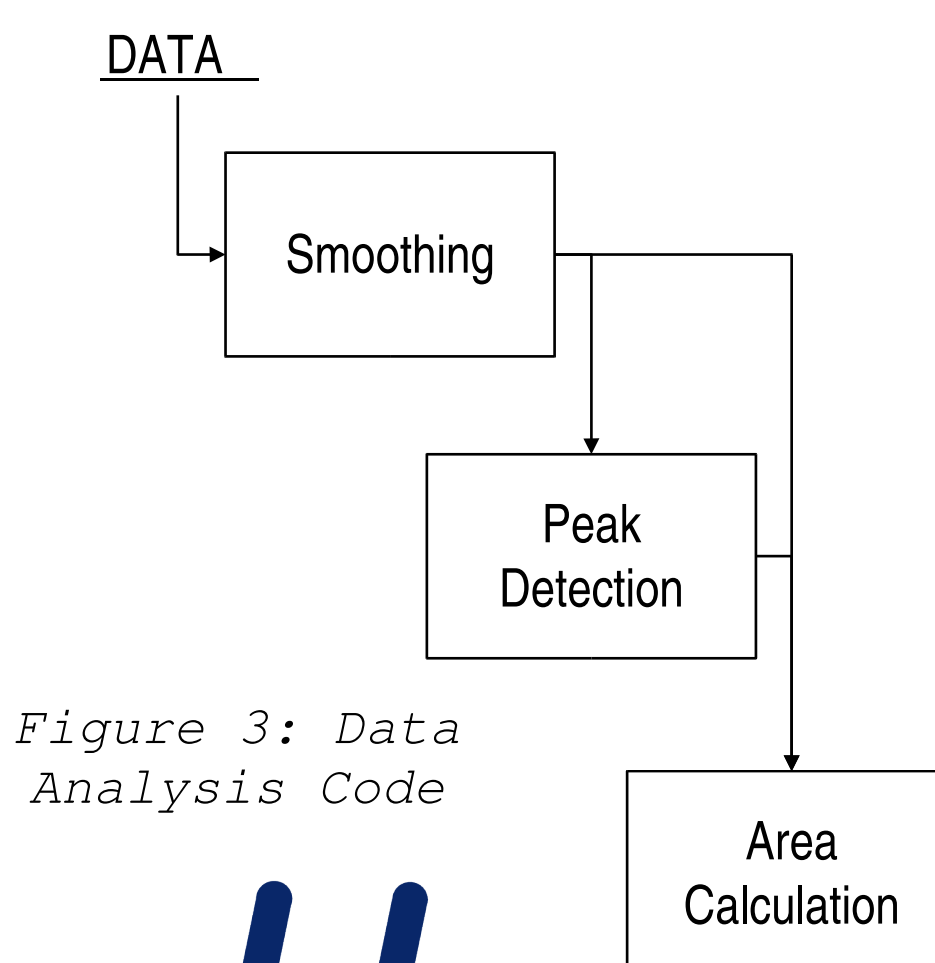


Figure 3: Data Analysis Code

For data compression a simple line fit algorithm is used. Each  $n$  points of original data are replaced by a linear fit of that segment. This provides compression in the way that each  $n$  point segments is substituted with only two pieces of data: the end points of the linear fit and its slope.

The benchmark for the data analysis code is the analytical software, Galactic Grams32. Galactic Grams32 is a commercially available software tool. By comparing the data generated by Galactic Grams32 to data produced by the data analysis code, the validity of the code can be effectively determined.

As mentioned before, the drivers for the  $\mu$ GC are compiled with the WIMS low power compiler. This is the case for two reasons: the low level nature of the drivers and to test the WIMS compiler. Drivers are short pieces of code that do one thing when they are called. Mainly they control, or set-up the MCU so certain parameters could be used by another application to use. That is why they are of low level nature.

The WIMS low power compiler is still under development and the  $\mu$ GC drivers offer a great opportunity for testing and debugging. The WIMS compiler translate the C language drivers to assembly language for the WIMS MCU. This code is then run on the WIMS MCU Verilog model and compared against some hand written assembly code. This comparison allows for the verification of the WIMS compiler's performance.

## Current Work

Currently a preliminary version of the smoothing process is completed. A block diagram of this process is shown on Figure 5, on the next column. At this time more testing is being perform on this process to improve its performance. At the same time a rough version of the peak detection process have been partially generated. Figure 4 shows the block diagram of the proposed peak detection process. This peak detection process is based on an algorithm introduced by Steven L. Horowitz.

The peak detection process takes its input directly from the smoothing process. The lexer block utilizes the base line estimator to determine which segments of the compressed data are over the baseline. With this data the lexer builds a string that contains one character per segment. This character identifies each segment according to the segment position and slope, relative to the baseline. In turn the parser block takes this string and identifies where a peak is present. The outputs of the parser are the peak's base coordinates, the highest point coordinate and the actual height of the peak relative to the baseline segment under that peak.

On the interface side, the Timer and SPI drivers have been generated. The Timer code is currently undergoing testing with the WIMS Compiler and the Verilog Simulator in order to obtain accurate information regarding the effectiveness of the generated code.

The SPI code is also undergoing testing. We are in the process of compiling the code with the WIMS compiler. The next stage is to verify the assembly output with the Verilog Simulator to obtain performance data needed to rate the codes effectiveness.

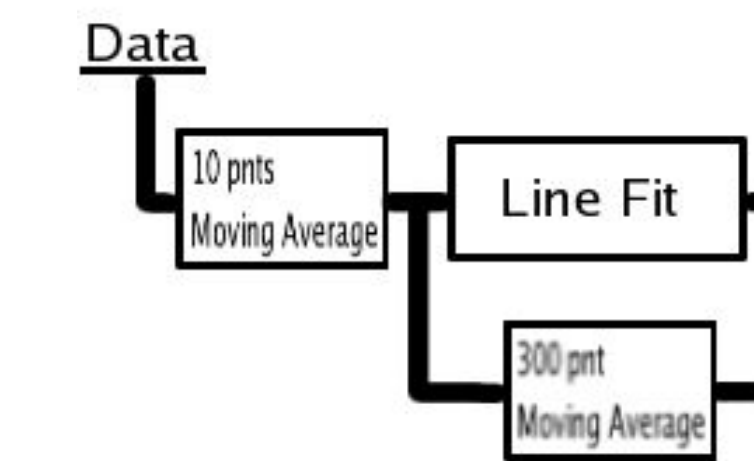
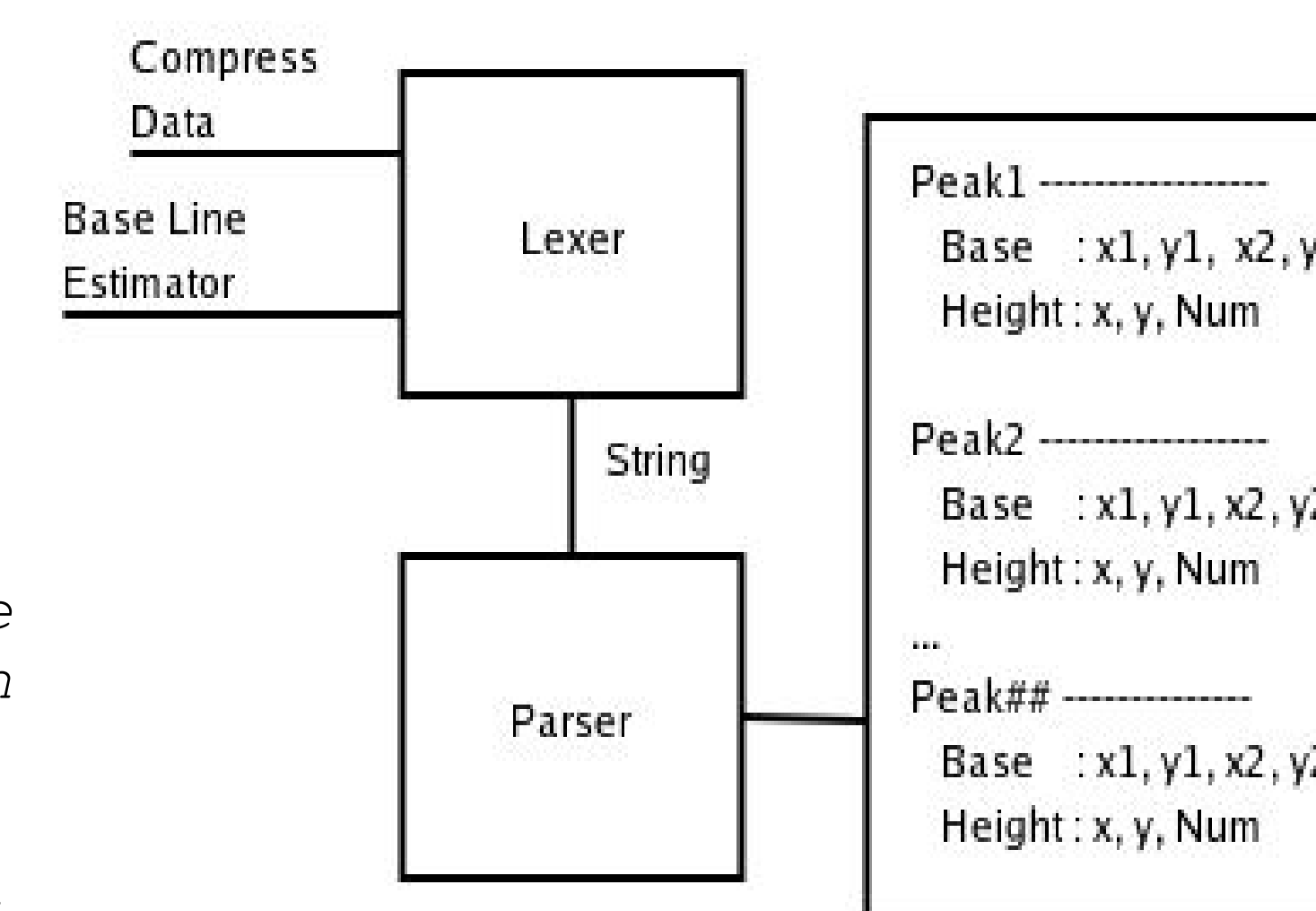


Figure 5: Smoothing Process Block Diagram

The data smoothing process is virtually completed. Figure 4 is a block diagram of the smoothing process. The first stage is a 10-point moving average low pass filter. This provides filtering, but no data compression. The two stages in parallel are the linear fit block and another moving average low pass filter. The linear fit provides the desired data compression, with compression ratios of 100:1. The second moving average block is a 300-point moving average low pass filter. This filter is used to create a baseline parameter estimator for the peak detection process.

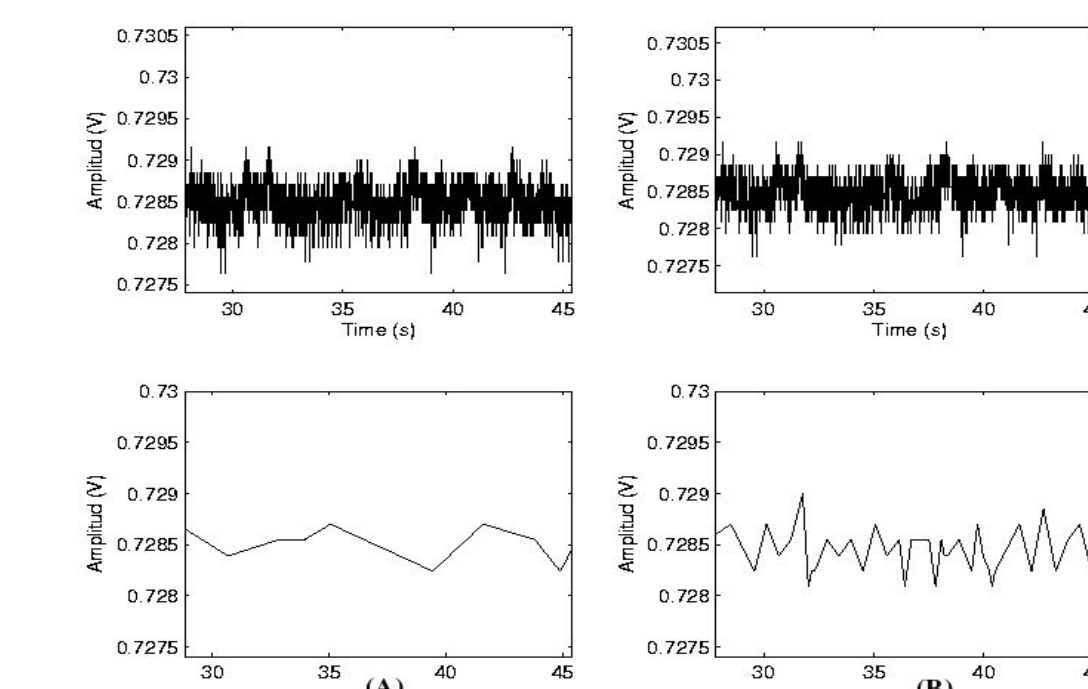


Figure 6: Line-fit Filtering. (A) Heavy Smoothing (B) Light Smoothing

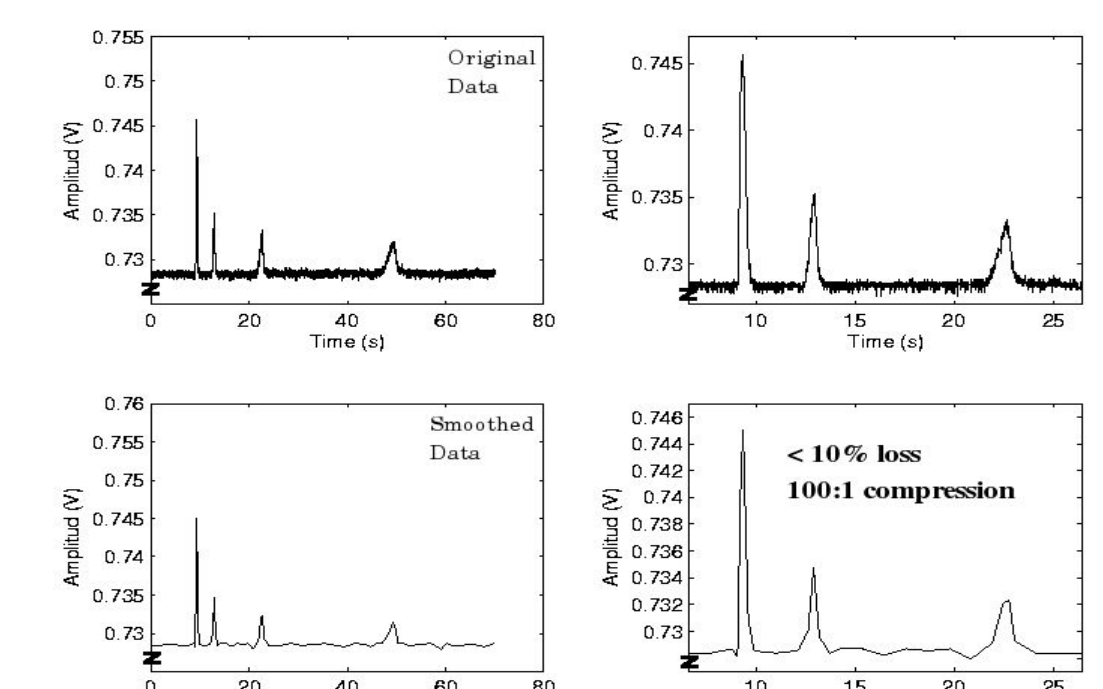


Figure 7: Line-fit Compression. (A) Full Dataset (B) Zoom on Peaks

Figures 6 and 7 show data processed by the line fit algorithm on the smoothing process. Figure 6 shows how the line fit effectively remove high frequency noise. Figure 7 depicts the compression ratios that are possible with the same algorithm.

## Future Work

More testing will yield more information about the validity of the codes developed. This is an extremely important step as these codes lay the groundwork for future development of the EMT as well as the microprocessor unit itself. We will continue developing the interface drivers as well as the analyzing code to provide the necessary support for the new generation of the micro gas chromatograph.

## Acknowledgments

This work was supported by the Engineering Research Centers Program of the National Science Foundation (NSF) under Award No. EEC-9986866.

We would also like to thank Robert M. Senger, Eric Marsman, Rajiv Ravindran, Ganesh Dasika, Dr. Joshua Whiting and Dr. Richard Brown for their invaluable support and time.

