

Signal Analysis

1 Objectives

The objective of this laboratory exercise is to explore basic signal analysis procedures such as the spectral analysis and the peak detection. A particular emphasis is put on studying the effect of the sampling rate on the acquisition of the time-varying signals. The exercises include data acquisition and processing of audible harmonic signals, noise, and measurement of the sound speed in air using a pair of microphones.

2 Theoretical Background

In the process of measurement, a sensor system outputs an analog signal describing a particular physical phenomenon. Often, the measurement changes in time, leading to periodic and non-periodic waveforms. Different waveforms can be analyzed by decomposing the original signal into a set of known *basis functions*.

$$x(t) = \sum_{n=-\infty}^{\infty} c_n \cdot \psi_n(t), \quad (1)$$

Any periodic signal permits description in terms of harmonic basis functions. This description serves as a foundation for the spectral analysis.

2.1 Introduction to spectral analysis

A direct representation of a periodic waveform in terms of harmonic functions can be achieved using the *Fourier Series*. A mathematical expression illustrating this process is presented below.

$$x(t) = \frac{a_0}{2} + \sum_{n=-\infty}^{\infty} a_n \cdot \cos(n \cdot \omega t) + b_n \cdot \sin(n \cdot \omega t), \quad (2)$$

where $a_0 = \frac{2}{T} \int_0^T x(t) dt$, $a_n = \frac{2}{T} \int_0^T x(t) \cdot \cos(n \cdot \omega t) dt$, $b_n = \frac{2}{T} \int_0^T x(t) \cdot \sin(n \cdot \omega t) dt$, $n = 1, 2, \dots$

The exponential form of Equation (2) is

$$x(t) = \sum_{n=-\infty}^{\infty} X_n \cdot e^{i \cdot n \cdot \omega t}, \quad (3)$$

and $X_n = \frac{(a_n - ib_n)}{2} = \frac{1}{T} \int_0^T x(t) \cdot e^{-i \cdot n \cdot \omega t} dt$ are the Fourier coefficients.

Noticeably, signal $x(t)$ in Equations (2) and (3) is a continuous signal. Approximating the continuous signal with its discrete equivalent at the time increments Δt , the expression for Fourier coefficients can be reformulated.

$$X_n = \frac{1}{N} \sum_{k=0}^{N-1} x_k \cdot e^{-i \cdot 2\pi n \cdot k / N}, \quad n = 0, 1, \dots, N-1. \quad (4)$$

where $t = k \cdot \Delta t$, $T = N \cdot \Delta t$, and $\omega = 2\pi n / T$. Then, the discrete time signal is

$$x_k = \sum_{n=0}^{N-1} X_n \cdot e^{i \cdot 2\pi n \cdot k / N}, \quad k = 0, 1, \dots, N-1. \quad (5)$$

A pair of equations (4) and (5) describe a *Digital Fourier Transform* or DFT. DFT is implemented in a many software packages and is used to obtain a frequency domain representation of a signal $x(t)$.

Consider a Matlab example of utilizing DFT to determine the frequency content of a signal.

```

clc; clear all;
%
%-----
%SIGNAL PARAMETERS
N=256; n=1:N; % number of points
T=0.1; %record length
t=linspace(0,T,N);
Ts=t(2)-t(1); T0=T/N; %time instants
fs=1/Ts; fn=1/(2*Ts); % sampling and Nyquist frequencies
ws=2*pi*fs; % sampling frequency in rad/sec
Fs=1/T; %minimum frequency resolution bandwidth
%
%-----
% SIGNAL GENERATION
f1=200; w1=2*pi*f1; % frequency
a1=ones(1,N); %initial amplitude
x1=a1.*sin(w1*t);
%
%-----
% COMPUTING FFT
F=fft(x); Fconstant=N;
Fp=F(2:N/2+1)*Ts ;% computes the positive frequency components
                % and multiplies by Ts=T/N to approximate F(w)
Fc=Fconstant*0.5*Ts;
f=fs*(1:N/2)/N;% build the frequency axis
                % from zero the the Nyquist frequency Ws/2
FM=abs(Fp);
FB=20*log10(abs(Fp)/(abs(Fc)));
%
%-----
% PLOTTING
figure(1)
subplot(311)
plot(t,x1,'-k. ');
xlabel('Time, s'); ylabel('Amplitude, V')

subplot(312)

```

```

plot(f,FM,'-k. ');
xlabel('Frequency, Hz'); ylabel('|F(f)|')

subplot(313)
plot(f,FB,'-k. ');
xlabel('Frequency, Hz'); ylabel('PSD, dB')
    
```

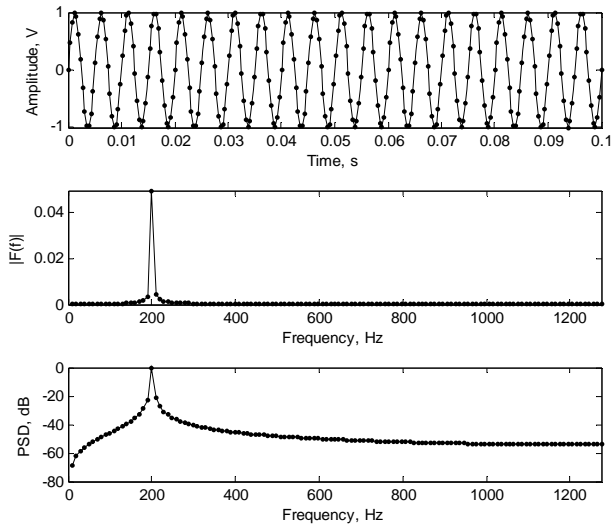


Figure 1 Time-domain and frequency – domain (amplitude and dB re 1 V) representations of a harmonic signal.

Example of a multi-frequency signal

Matlab input

```

x1=(1*sin(2*pi*200*t)+0.07*cos(2*pi*500*t)+0.4*sin(2*pi*700*t))./max(x1);
    
```

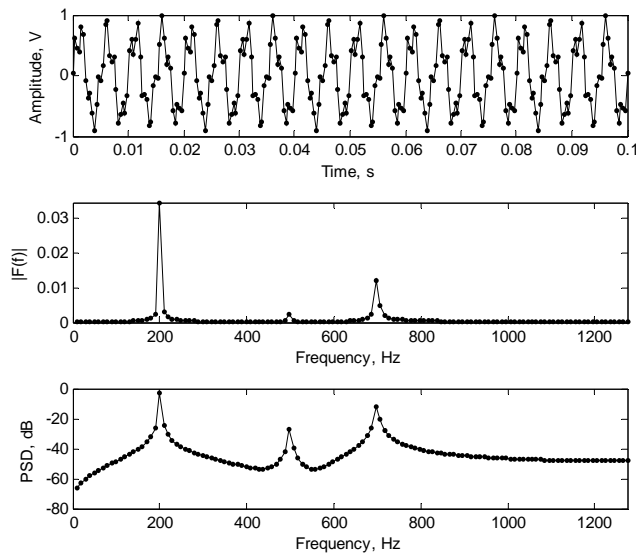


Figure 2 Time-domain and frequency – domain (amplitude and dB re 1 V) representations of a harmonic signal containing 3 frequency components.

Example of a non-harmonic signal.

Matlab input `x1=1*sawtooth(2*pi*200*t);`

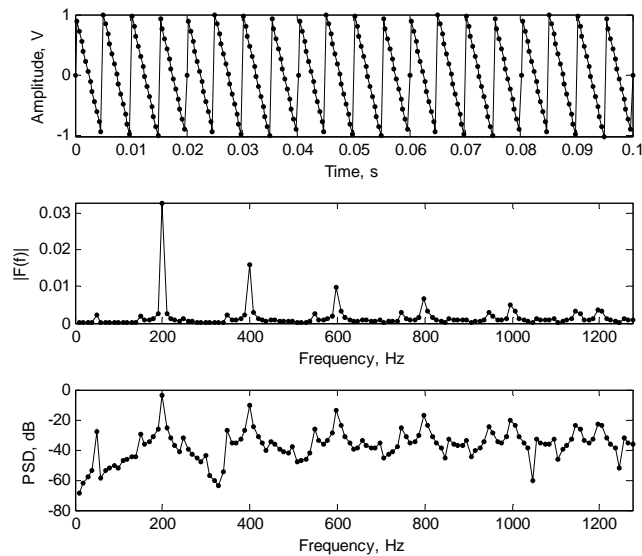


Figure 3 Time-domain and frequency – domain (amplitude and dB re 1 V) representations of a sawtooth signal.

2.2 Sample rate

To represent a time-varying signal correctly, one should consider a sample rate. The sample rate determines a speed of digitizing the waveform and, therefore, affects the distance between two neighboring point on a time scale. The higher the sample rate, the more points are recorded per waveform's period. If a signal changes drastically between the sample points it may not be represented correctly. The adequately digitized signal must have at least two data points per the shortest period. This rule is mathematically expressed as follows:

$$f_s = 2 \cdot f_{\max} \quad (6)$$

where f_s , is the sample frequency and f_{\max} , is the maximum frequency in a waveform. If a signal is sampled at the frequency below $2f_{\max}$, *aliasing* occurs. The aliased signal falsely appears as a signal of a lower frequency than the original waveform and thus must be avoided. Illustration of the *aliasing* is presented below.

The harmonic signal presented in Figure 1 is sampled at $\Delta t = 362.16 \mu\text{s}$, which gives 256 points per 0.1 s waveform. The sample frequency is respectively 2550 Hz. The highest frequency of the signal is 200 Hz and, as it can be noticed from Figure 1, the number of points per period is well over 2 (actually 12.8). For $f_{\max} = 200 \text{ Hz}$, $f_s =$ should be not less than 400 Hz.

Let us decrease the sample rate four times. This would correspond to 64 points per whole waveform and 3.2 points per period of the 200 Hz frequency. Although, as can be seen in Figure 4, the time-domain representation of the signal has changed significantly, still it is possible to resolve the 200 Hz frequency component. When the sample rate is decreased even more (by a factor of 2), the original 200 Hz waveform falsely appears as a low frequency signal with a peak at 107 Hz.

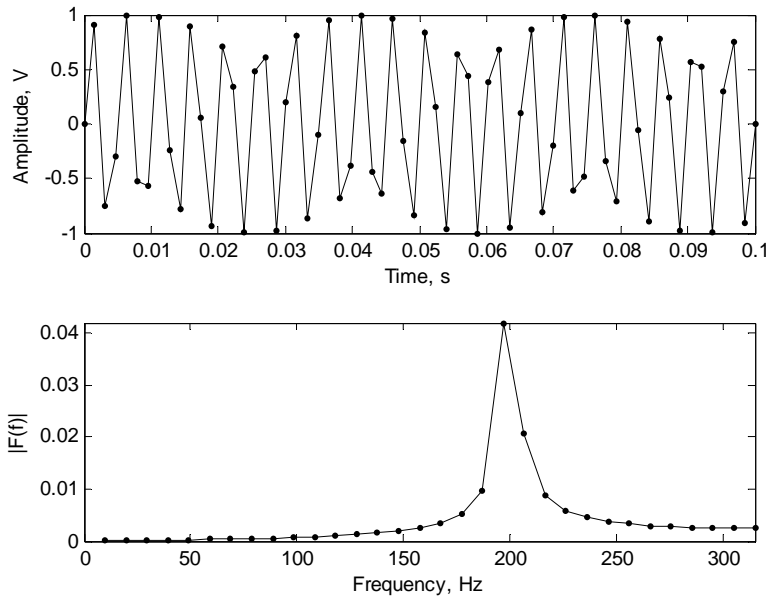


Figure 4 Time-domain and frequency – domain representations of a 200 Hz signal sampled at 630 Hz.

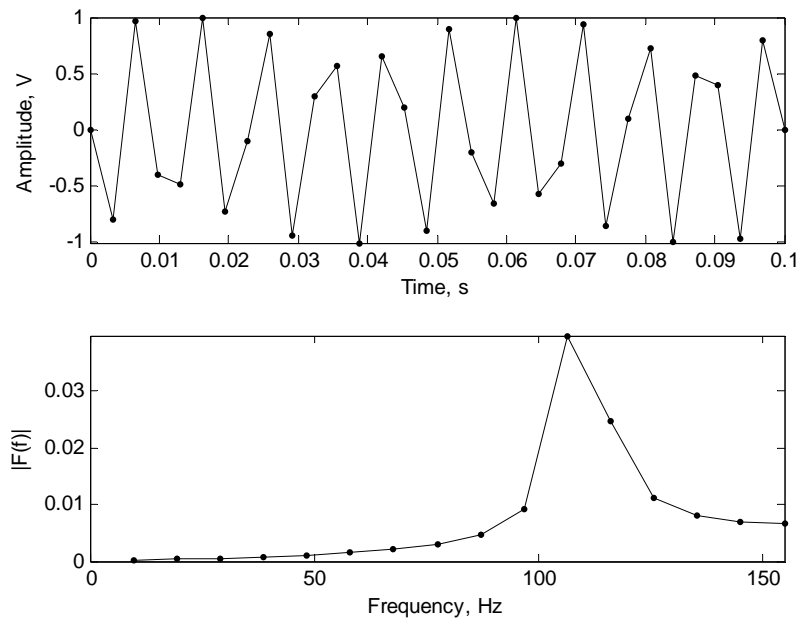


Figure 5 Time-domain and frequency – domain representations of a 200 Hz signal sampled at 310 Hz.

Illustrations in Figures (4) and (5) emphasize that it is very important to ascertain that the measurement data is sampled above the highest frequency of interest.

2.3 Peak detection

Detection of a peak amplitude in the impulse signal can be achieved via Hilbert transform. The Hilbert transform translates a real waveform into the analytical signal with real and complex components. The real component corresponds to the original data and the complex component represents an original waveform shifted by 90 degrees. This property of the Hilbert Transform can be conveniently used in peak detection. To obtain an envelope of the signal, one needs to take an absolute value of its Hilbert transform. The peak detection algorithm is illustrated with the Matlab instruction and Figure 6 below.

```
x2=abs(hilbert(x1)); % x1 is your original signal, an impulse in Figure 6.
```

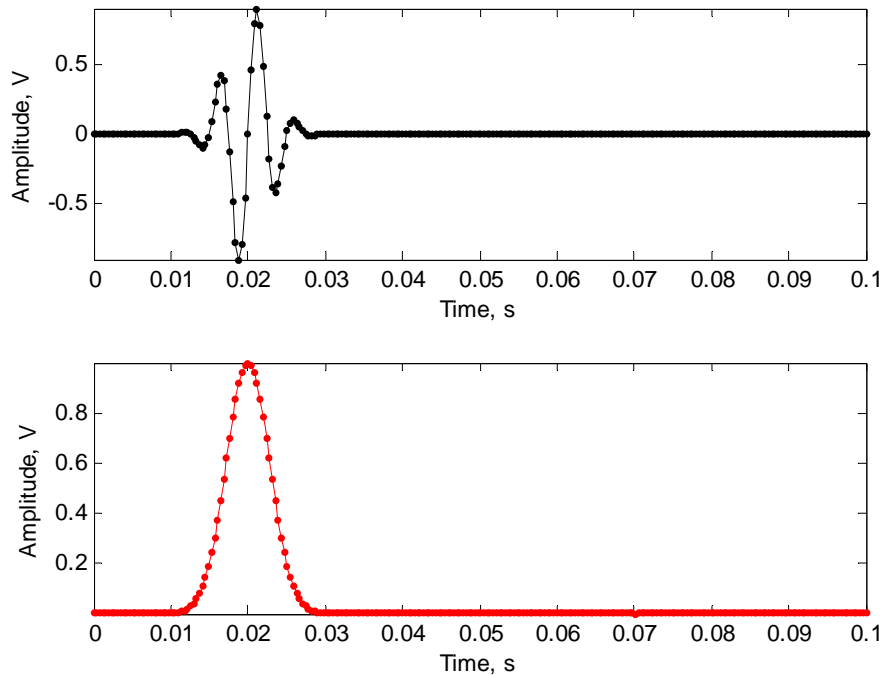
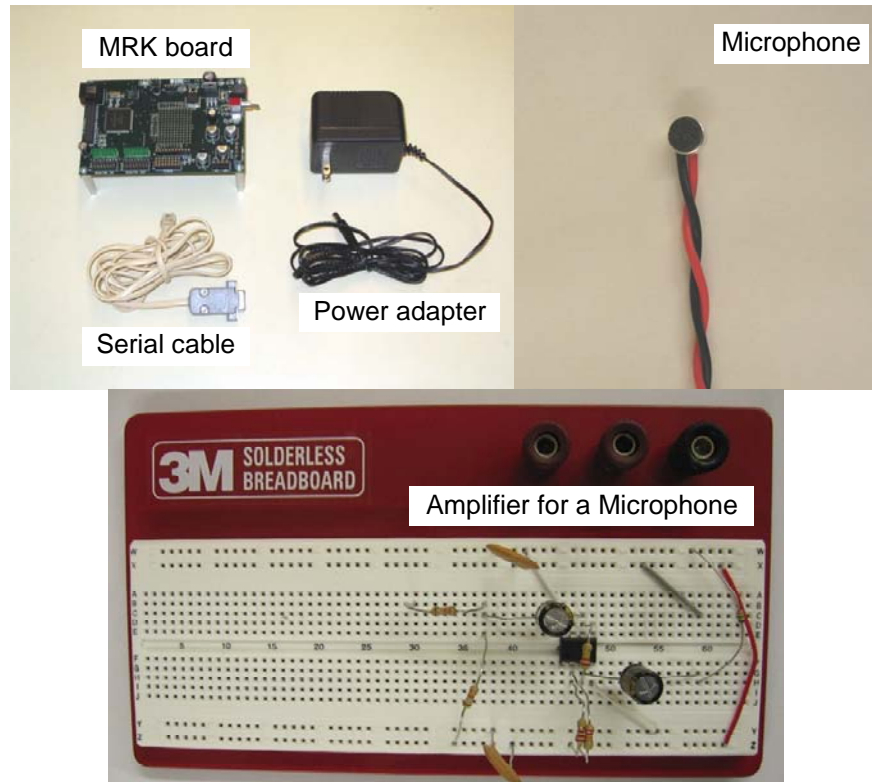


Figure 6 Peak detection using a Hilbert transform: upper graph – original signal, lower graph – the envelope of the signal obtained by applying a Hilbert transform.

To reduce effect of noise and smooth the data, a number of smoothing algorithms can be employed; most popular being the “moving average”. Matlab’s function `x3=smooth(x1,5);` smoothers the signal with the “moving average” window of 5 point, i.e. 5 neighboring points are averaged.

3 Equipment



- MRK board
- Serial communication cable
- Power adapter
- Speakers and dedicated software
- Microphone
- Amplifier for Microphone

4 Procedures

Exercise #1:

Use the provided Matlab code to plot a set of waveforms representing a harmonic signal sampled at different sample rates. Use a sinusoidal waveform with the frequency of $f = 400$ Hz. Change the total number of point per whole signal to 1024, 256, 128 and 64. Calculate a sample rate for each setting; plot waveforms and spectra. Comment on your graphs.

Exercise #2:

You will be using a computer software to generate an audible signal. To open the signal generating software on your computer, click the start button, go to all programs, and open Realtime Analyzer. Once Realtime Analyzer is open click the signal generator button. In the signal generator you will use the tone tab. Set the tone to 400 Hz and the sampling rate to 11.025 kHz (in the tone panel). Press the start button when you are ready collect the signal data.

Try to use about a 5 – 10 on the output level setting with your speakers turned all of the way up. Please, do not forget to turn the speakers back down when you are done with the exercise. This will make your neighbor's results better by lowering the amount of reflected waves in the room.

Begin a new Eclipse project; choose to store the program in EEPROM (starting address 0x4000). Next, you must change the memory.x file so that your program will be stored in EEPROM and the data from the signals will be stored in SRAM. Memory.x is not a program but a configuration file that is in the project folder for every Eclipse project. Edit memory.x to start your data from the location 0x1000 and set a length of 0x2000. You will need to save memory.x after changing it. Please note that in Eclipse unsaved files have a * after the name. These procedures must be done before compiling the C program. As a result, your compiled *.s19 file will contain directions to put the program at 0x4000 and the data at 0x1000. The required memory.x file is presented below. To run your program, you will have to type GO 4000 instead of GO 1000.

```
MEMORY {
page0 (rwx) : ORIGIN = 0x0, LENGTH = 256
text  (rx)  : ORIGIN = 0x4000, LENGTH = 0x1000
data   : ORIGIN = 0x1000, LENGTH = 0x2000
}
PROVIDE (_stack = 0x3C00);
```

The program below will receive the waveform data from the microphone setup. It will be used for both one and two channel signal acquisition with the MRK Board.

```
#include "MRK.h"

#define ARRAY_SIZE 2048

int rate, get_data;

void do_a2d(void) {
    TIME0 = TIMER + rate;
    get_data = 1;
}

int main() {
    int data1[ARRAY_SIZE];
    int data2[ARRAY_SIZE];
    int i, channels;
    rate = 0;
    analog_in(RESOLUTION, 10);
    setup_timer(ENABLE, 0);
    setup_timer(OUTPUT_COMPARE, 0, NOTHING, &do_a2d);
    fprintf(SCI0, "Number or channels (1 or 2):\n\r");
    i = fgetchar(SCI0);
    if (i == '1') channels = 1;
```

```

else channels = 2;
fprintf(SCI0,"Using %d channels.\r\n\n",channels);
fprintf(SCI0,"Select sample rate:\n\r");
fprintf(SCI0,"'1' - 1kHz\n\r");
fprintf(SCI0,"'2' - 10kHz\n\r");
fprintf(SCI0,"'3' - 25kHz (default)\n\r");
i = fgetchar(SCI0);
if (i == '1') rate = 24000;
else if (i == '2') rate = 2400;
else rate = 960;

fprintf(SCI0,"Press any key to sample input.\n\r");
fgetchar(SCI0);
i = 0; get_data = 0;
if (channels == 1) {
    while (i < ARRAY_SIZE) {
        if (get_data) {
            data1[i] = analog_in(IN,3);    // channel 3
            i++;
            get_data = 0;
        }
    }
}
else {
    while (i < ARRAY_SIZE) {
        if (get_data) {
            data1[i] = analog_in(IN,3);    // channel 3
            data2[i] = analog_in(IN,10);   // channel 10
            i++;
            get_data = 0;
        }
    }
}
fprintf(SCI0,"\nData stored.  To receive data, go to\n\r");
fprintf(SCI0,"'Transfers->Receive Text File...'\n\r");
fprintf(SCI0,"\nWhen ready, press any key.  When done,
hit\n\r");
fprintf(SCI0,"stop button and then press any key.\n\r");
fgetchar(SCI0);
for (i=0;i<ARRAY_SIZE;i++) {
    if (channels == 1) fprintf(SCI0,"%u\n",data1[i]);
    else fprintf(SCI0,"%u %u\n",data1[i],data2[i]);
}
fgetchar(SCI0);
}

```

Connect the microphone setup to the analog channel 3 on the MRK Board. Place your microphone close to the speaker. Run the data acquisition program and set the number of channels to 1. Choose the sample rate of 1 kHz. Start your signal generator, and in the MRK terminal program press any key to collect the data. Stop the signal generator after the data has been stored. Follow the instructions on the MRK terminal to receive the text file containing the data. Repeat the data acquisition procedure to collect signals sampled at 10 kHz and 25 kHz sample rates. Once you have 3 data files, plot and analyze your results.

Exercise #3:

Repeat exercise #2 with a sinusoidal wave of frequency $f = 1500$ Hz. During experiments, try to keep your output level low to keep the reflected wave level in the room down.

Exercise #4:

Repeat exercise #2 with noise. To generate the noise click the Noise tab on the Signal Generator. Use the white noise and monaural settings. Please, keep your output level low.

Exercise #5:

For this exercise you will have to join forces with another group because it requires two op-amp/microphone setups. You only need one MRK Board, but you will use two ANALOG IN ports to receive the two different signals. Connect one amp/microphone to analog channel 3 and connect the other to analog channel 10. The purpose of this exercise is to measure the speed of sound in the lab's air. You will need to place the first microphone approximately 2 inches from the speaker and the second microphone approximately 20 – 25 inches behind the first microphone. Record the distance between the two microphones. This will be important in calculating your speed of sound values.

Open the Signal Generator program; select the Pulse as a signal type. Set the pulse repetition period to 40 ms and pulse width to 0.91 ms. You will hear repetitive “clicks” produced by speakers. During this exercise, you may increase your volume slightly to get a good signal, but still keep your sound level within reason. Use the same program as before to collect the data; select a 2 channel data acquisition and the 25 kHz sample rate.

Plot and analyze your signals. Run this experiment at least twice and calculate your sound speed. You may want to use the smoothing “moving average” window (3-5 points) to smooth out the data. This will make the signals easier to tell apart and see. Also, apply the Hilbert transform to obtain the impulse envelope. You will need this to calculate the sound speed. The example is shown in section 2.3 of this lab.

5 Report and analysis requirements

5.1 Theory

Explain the effect of a sampling rate on the ATD conversion of the signal.

Discuss measurement of the sound speed using the impulse technique. What is the sound speed? What is it dependent on? Provide formulation you utilized to calculate the sound speed. What factors affected your measurement. What data analysis procedures can be implemented to simplify calculation of the sound speed?

5.2 Results and analysis

Answer all questions in previous sections of these laboratory instructions.

Present and discuss you results.

How fast does one need to sample a waveform to obtain correct digital approximation of the analog signal? What is the minimum acceptable sample frequency for the harmonic waveforms you obtained in the experiment? How fast does one need to sample a broad band noisy signal to achieve an adequate digital representation?

What is the purpose of a signal conditioning module between the microphone and the MRK controller board?

In the sound speed measurements, if you place a microphone several feet apart, what would happen to the signal coming form the microphone positioned far away from the speaker? What is a physical mechanism influencing the signal change? Suggest methods to alleviate this problem.

What are the other factors that may affect the sound speed measurements? Discuss influence of any detrimental factors during the experiment.

What data analysis procedures can be implemented to simplify calculation of the sound speed? Describe your experience in applying these procedures. How did they change your signal?

*Last updated
October 16, 2006.*