ECEn 487 Digital Signal Processing Laboratory

Lab 1
Finite Impulse Response Filtering
2023

General Laboratory Procedures

Welcome to ECEn 487 Laboratory! I look forward to a safe and productive semester of lab work. The following instructional items include general guidelines:

- **Lab facilities:** Starting January 10 we will meet weekly, Mondays 3:00 – 5:50 p.m., in our lab room, 438 EB, where you will have TA support for your work. Your lab projects will be developed principally in MATLAB code on the workstation PCs in the room. We will be doing real-time (as opposed to batch) processing of sampled signals. Additional accessories including microphonses, mic preamp/mixer, speakers, headphones, cables and adaptors are available in the room to check out for each workstation.

- **Teams:** You will self-select to form two-student lab teams. Though a team may share a board and team members consult together, all lab books, pass-offs, final submissions, and lab grading will be done individually. Both team members are expected to contribute approximately equally to the development of the final working lab project. Lab books are to be kept and submitted separately for each team member. You may share plots, data, and computer codes, but each individual is responsible for compiling his or her own lab book. All responses to questions in the lab handout, all observations and analysis entries, and all conclusions sections must be individually your own. Submission of essentially identical lab books will result in each partner getting half credit.

- **Experiment Pass Off:** At the completion of some lab experiments, you will be directed to present your work, including Lab Book entries, to a TA for "pass off." This should be done before proceeding to the next experiment, though if the TA response is delayed, you may move on to the next session while you await pass off. You will also be asked to show your lab book entries which document your work. A lab is not considered ready for grading until all pass offs are completed and recorded by the TA. It is also useful to insure that you are on the right track, and will not lose points in your final submission because you did not correctly complete the experiment or did not understand the principles being taught.

- **Lab Books** will be kept electronically by you, using your favorite document editor (e.g. MS Word). Pages from your Lab Book may be presented for TA experiment "pass off," and TAs may ask to see your lab book at any time so they can see how you are progressing and help you solve problems. You will keep separate documents for each of the Laboratory Assignments.

  Lab Books are not formal report documents, but are intended to be a verbose, exhaustive, real-time, stream of consciousness, diary-like, un-formatted, sequential record of all you do, observe, measure, design, code, display, etc. during your lab work. It is like a technical diary. *The only formatting style requirement are: 1) that you label entries by Experiment number and procedure Step number, e.g. "Experiment 3-Step 2 ..." This enables the TA to locate entries for specific work;* 2) plots and other illustrations must be clear, easy to read, and well labeled with axis descriptions and titles. Generally, you don't delete Lab Book entries if you later discover your approach was wrong, you just insert a later entry with a description of the changes.

  A Lab Book can include written notes, photos, screen shots, MATLAB plots, algorithm signal flow diagrams, scanned drawings, codes snippets or whole program listings, and more. *It must also include written answers to every question asked by the lab handout.* Someone else should be able to reconstruct everything you did by reading your Lab Book. Sufficient evidence must be included in the lab book that you completed each experiment. You will not be considered as having completed the lab if you have no lab book entries for any experiment.

  Add a conclusions section at the end of your lab book before it is submitted via Learning Suite as your final document for grading. This should include a brief summary of what you accomplished and your assessment of what were important points you learned. Your grade will be based on having completed the Pass -Off steps, and on how well your lab book provides convincing and clear evidence that you correctly completed and <u>understood</u> the significance of each experiment.

Lab Schedule and Submission Requirements:

Due to the MLK Jr. holiday, and to give us time to set up the lab for you, we will not hold lab sessions the first 2 weeks of class. Our first lab meeting will be Monday, January 19th. This is a three-week lab, i.e. you will have three laboratory class periods to complete this assignment. All TA check off must be completed before the start of lab class, on the due date. The three experiments take varying amounts of time, with the second experiment probably taking the most amount of time. Submission via Learning Suit of a copy of your lab book for this lab is due by the following date and time:

Due 2:00 p.m., Monday February 13

This is just prior to the beginning of the lab session where your next lab assignment will begin. Any late submission receives a 25% penalty in grading.

Learning Objectives:

The purpose of this lab is for each student to design an FIR digital bandpass filter using MATLAB and use it to extract a desired Morse code signal from interfering signals at nearby audio frequencies. The extracted message will then be decoded. Students will become familiar with practical aspects of simple filter design techniques and gain an appreciation for its possible uses.

Reading Assignment

1. Textbook, Oppenheim and Schafer, Sections 7.0 , 7.5, 7.6.1, and 8.7.3 .

2. MATLAB online help and Signal Processing Toolbox User's Guide tutorial and documentation on the signal processing toolbox (e.g. “help signal”) functions “audioDeviceReader,” “audioDeviceWriter,” “audioplayer,” "freqz,", "fft," "conv," "filter," "fir1," "hamming," "bartlett,"blackman," "boxcar," "kaiser," and "chebwin."

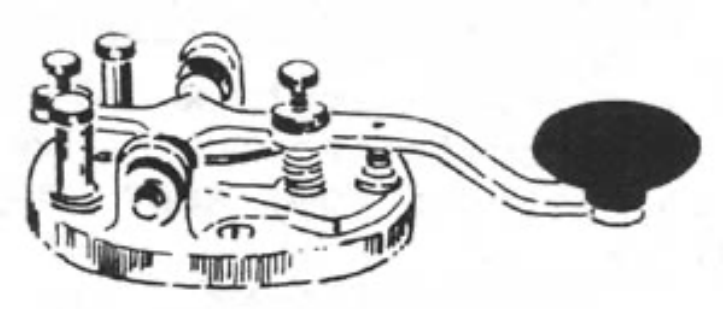
Introduction

In many real-world DSP applications, a specialized microprocessor called a “Programmable Digital Signal Processor” is used to implement the structured, repetitive, high speed mathematical operations needed for “real-time” operation. These processors have very specialized architecture elements including parallel arithmetic-logic units (ALUs), single clock cycle multipliers, and deep pipelined structure. Other high-speed DSP processor options which enable massively parallel-pipelined computations using many (thousands!) of processor cores include field programmable gate array (FPGA) implementations, or graphical processing units (GPUs) which were originally designed for high speed graphic display calculations. For lower sample rates (i.e. in the audio range) and fewer input/output channels (e.g. stereo) most modern PCs are capable of keeping up with some significant signal processing in real time using their general purpose processing architecture; that is what we will be doing for most of the labs experiments this semester. Also, many application-specific integrated circuits (ASICs) are designed to do many dedicated digital signal processing operations, but these are not programmable to allow changing the algorithm.

One of the most important and practical real-time DSP operations is digital filtering of sampled signals. Though IIR filters are sometimes used, for this lab we will concentrate on FIR filter implementations because they are always stable, can be designed to be exactly linear phase (no phase distortion or dispersion), and are easy to understand and design to given filter specifications. Your goal in this lab will be to extract a particular Morse code signal from a hopelessly scrambled recording so that you can decode the message. This simulates a scenario that an amateur radio operator might encounter when trying to copy a weak overseas signal in a channel packed with local interference from other amateurs. Your only hope is that the interfering signals are centered on slightly different audio frequencies, so that a frequency selective filter may help reject them. (My Icom IC7610 ham radio transceiver has exactly this

kind of digital filter built in for selecting the desired CW Morse code signal from a jumble. You can tune the filter's center frequency and passband width with 2 knobs). We will use the windowed filter design technique from your textbook to design the filter.

Morse code is an early binary on-off keying method used to manually communicate text characters. The transmitting operator uses a switch by hand to generate a series of longer "dash" tones, and shorter "dot" tones to form a character code. Characters are separated by longer spaces than the dots and dashes within a character. One ingenious aspect of the code is that more frequently used characters (in English) are assigned shorter codes. This anticipates the later practice of entropy coding for efficient digital communications. Today, the code is used primarily by amateur radio operators for fun and emergency communications. You can find a code table (which you will need to decode the message) at many websites, like http://www.qsl.net/4f5aww/module3c.htm. With a little work on google you can find programs to practice sending and receiving code (though not required for the lab).



A classic Morse Code key

The corrupted Morse code signal you must filter is in the file "mix10.wav" available on the class Learning Suite web page. This is a single channel (mono) signal sampled at 16.2 kHz with 8 bits per sample. You can read it into MATLAB with the command [y,fs] = audioread('mix10.wav'), and play it with the commands myPlayer =audioplayer(y,Fs), and play(myPlayer).

Note: The experiments below are not intended to each be one week long. There are four experiments for a three-week lab, but not all experiments are equally challenging. If you finish one before the end of the lab period, start on the next. The latter experiments are harder and take longer!

***Experiment 1* Familiarization with MATLAB ADC - DAC Operation: loopback program.**

In this task you will write a MATLAB script to sample an audio function generator and CD stereo signal using the built-in workstation sound card and play the signal out to speakers. This loopback program performs the same function as a simple audio patch cable (but with long delays) but will be the basis for future code development.

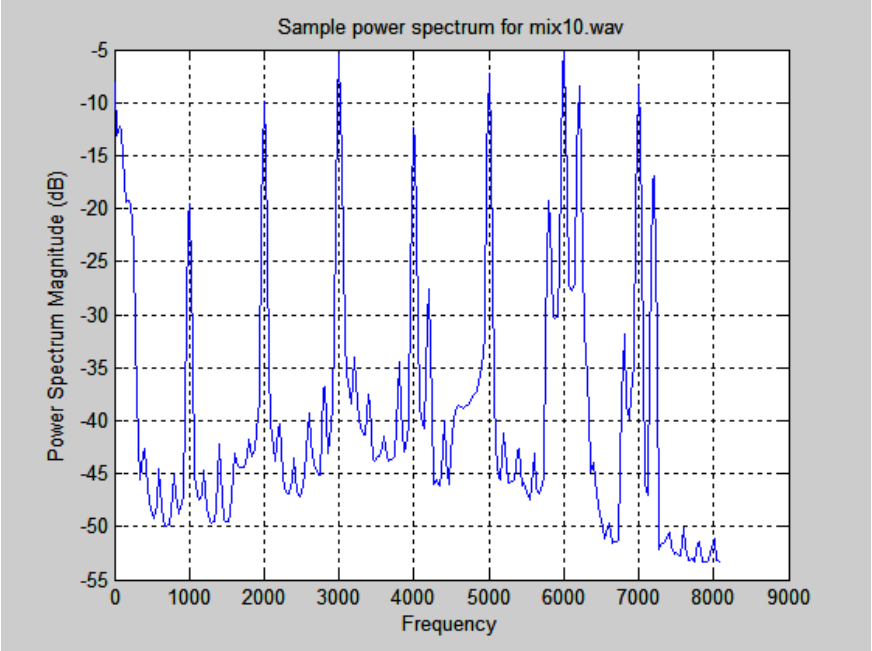
Procedure

1. Your workstations are configured with the Ubuntu Linux operating system. Log in with your CAEDM credentials. Navigate to your user directory where you will be developing your lab codes. Right-click on the desktop and select to open a terminal window for Linux. Startup MATLAB by entering the Linux command line entry: "matlab" Also, make sure that you write and save all your scripts and data to either a removable USB memory stick, or to your personal CAEDM account.
2. Use the provided "loopback2022.m" MATLAB script which samples the line input in stereo, then plays that block through the DAC to amplified speakers. Download this from the Learning Suite => Content => Laboratory web page. (NOTE: We recommend you use the Chrome browser under Ubuntu rather than FireFox which can lock you out if you log out of Ubuntu with your browser open.) You will modify your code to adjust the specified sample rate, sample window block length, and how long it will run before terminating. Read the MATLAB help documentation on the audioDeviceReader and audioDeviceWriter functions.
3. There are two mini-stereo phono jacks on the back panel of your lab PC, located in the lower left corner when viewed from the rear (if the chassis in laid on the table horizontally): Line-in (top socket) and Line-out (bottom socket). The line-in connection will be used in all labs as the input to your analog to digital converter (ADC, or A/D). You must keep signal levels low (below 1 Volt peak) to avoid damaging the sound card. Connect your amplified speakers to the green line-out connection. When you are done with this lab, leave your cables plugged into the back of the computer for future use.
4. Evaluate how the loopback program operates using the function generator as you input. Use the sound card sample rate of 48 ksamp/s. Try different sinusoidal frequencies and other waveforms. For this task, and all experiments below, observe your input and output signals simultaneously with two channels of the oscilloscope. Note differences you observe.
5. Provide a CD audio or MP3 player input to the line connection on the line-in back panel connector. Connect the amplified speakers the to line-out speaker output. Select the sound card sample rate to be 48 ksamp/s. Verify proper digital-audio loop input and output and note what volume levels lead to distortion. Can you detect any degradation in sound quality through the loopback? Should there be any noticeable change? Document your observations in your lab book. Have the TA check off your operating program and enter your pass-off in Learning Suite.

***Experiment 2* Design an FIR bandpass filter**

Procedure

1. Generate a set of specifications for a filter to extract the desired Morse code signal while rejecting interfering neighbors. Specification must include passband and stopband edge frequencies and maximum ripple levels. Base your design on the prior knowledge that the desired signal is centered on 3.0 kHz, and that there are about 7 interfering signals present. All 8 (1 desired and 7 interferers) signals are spread fairly uniformly in frequency from 250 Hz to 7 kHz, as shown in the figure below. Of course, you must translate the analog frequencies (like 1.0 kHz) to the corresponding discrete-time radian/sample frequencies, ω, based on the sample rate. Specify passband and stopband corner frequencies and stopband attenuation levels (in dB). There is no single correct answer here; be creative.



2. Use the approach described in the textbook, Section 7.5, to design an FIR filter $h[n]$ using the windowed method so that it matches your specification. You may use any of the listed window types that you think will work best (e.g. rectangular, Bartlett, Hanning, and Hamming, Chebychev, etc.). Note that MATLAB has built-in functions to generate each of these windows. Also note that the text only talks about designing a lowpass filter. You can compute the $h_d[n]$ (desired ideal impulse response) for a bandpass filter as the difference between two *same length* lowpass filters which have different corner frequencies. Let $h_{1,d}[n]$ be the impulse response for an ideal lowpass filter with corner frequency ω_1, and $h2_d[n]$ likewise with corner $\omega_2 > \omega_1$. $h_d[n] = h_{2,d}[n] - h_{1,d}[n]$ is the ideal bandpass filter impulse response with corners ω_2 and ω_1. The ideal response equation which you will window is

thus

$$h_d[n] = \frac{\sin[\omega_2(n-M/2)]}{\pi(n-M/2)} - \frac{\sin[\omega_1(n-M/2)]}{\pi(n-M/2)}$$
$$= \frac{\omega_2}{\pi}\,\mathrm{sinc}\left[\frac{\omega_2(n-M/2)}{\pi}\right] - \frac{\omega_1}{\pi}\,\mathrm{sinc}\left[\frac{\omega_1(n-M/2)}{\pi}\right] \quad (1)$$

where $N = M + 1$ is the filter length. Note that the second form may be easier to implement in MATLAB, because the built-in sinc function handles the division by zero that occurs when $n = M/2$.

In your design, make N as short as is practical so that the DSP computational requirements are minimized. For a given window choice, N controls transition bandwidth. For a given choice of N, the window shape controls stopband attenuation level. In general, windows with lower stopband ripple also have wider transition bands. Table 7.2 and equations 7.73-7.76 may help with picking N and selecting the window. Also, in MATLAB, type "help signal" to look at a list of all the built-in signal processing functions. Check out the sections titled "Digital Filters > FIR filter design," and "Windows" to see if there is anything helpful. You may not use the function "fir1" which does a complete windowed filter design, or any other filter design toolset. Your code must show that you generated the desired ideal impulse response as in equation (1) and multiplied it by a window of your choice.

3. Plot the impulse response and frequency response of your design (using "freqz.")

4. If the filter frequency response does not look acceptable, repeat steps 1-3.

5. Show your filter design and frequency response plot to a TA for check-off and enter your pass-off in Learning Suite. indicate completion of this experiment.

Experiment 3 Build a block mode FIR filtering program in MATLAB.

Procedure

1. Write a simple FIR filter loop function in MATLAB which uses your FIR impulse response designed in experiment 1. Use a "for" loop structure which increments the output sample index with each pass through the loop to implement the convolution sum directly:

$$y[n] = \sum_{k=0}^{L-1} h[k]\, x[n-k].$$

2. Verify basic operation by filtering short pure sinusoid sequences you generate (within MATLAB, i.e. not sampled) at three frequencies: one in the filter passband, and one for each of the two stopbands. Compute and record the total power levels at each frequency, and

compare them to see if the desired stopband attenuation, in dB was achieved.

3. Read in (using "wavread") the entire Morse Code signal. Your "for" loop filter implementation is too slow to filter it in a reasonable time. Try it, but stop operation with ^c (control c) when you get bored. MATLAB uses faster frequency domain FFT based implementation in the built in functions "conv" and "filter." You will develop a similar code in Experiment 4.

4. Filter the entire corrupted Morse code signal in memory (using "conv" or "filter" functions) and play the result through the sound card (using "sound" or "wavplay"). This is referred to as "batch" mode processing, where all the data of interest is in memory at one time, and you process (filter) it as a single batch.

   If your filter is designed correctly, you should hear a single signal with very little interference. If the filter performs poorly in rejecting interference, or if it is too long (too many taps) to run in a reasonable time, go back to experiment 1 and redesign appropriately.

5. What is the coded message? Have the TA verify your result and enter your pass-off in Learning Suite.

<u>Experiment 4</u> Build a real-time continuous FIR filtering program in MATLAB.

1. Modify your code from experiment 3 so that even though the sample data resides in memory all at once as a single input batch (data matrix), you break it into smaller 1024-sample blocks for successive calls to "filter" in order to build up the output batch data matrix one block at a time. In other words, you will write a loop that calls "filter" on each pass with 1024 new samples, then build the output data matrix by concatenating the filters result blocks together.

2. Design a new FIR bandpass filter to work with a sample rate of 48 ksamp/s and to pass some band of interest to you for a high quality audio signal (e.g. you could pass only the telephony speech band of 300 Hz to 3 kHz, or only 3 kHz to 10 kHz to emphasize percussion tenor instruments, or any other range you are interested in.)

3. Using techniques similar to the loopback2022.m program of Experiment 1, and building on the code from step 1 in this experiment, write a continuously running real-time program to filter signals seen on the ADC input, and play them back through the DAC output. Download and study the file “FIR filtering with block data.pdf” to help you understand how to break the data stream into frames (or blocks) for successive calls to a filtering subroutine without causing distortion or discontinuities in your output signal. Use "filter" as the filtering code, with the "Zi" and "Zf" optional parameters to carry over filter memory from block to block. Work with changing the block length (“framesize”), sample rates, and if necessary your filter length, *N*, to insure that you can filter the signal

continuously without dropping samples or creating data-framing discontinuity distortion.

4. Using the filter design from step 2, verify that your filter's corner frequencies and maximum sidelobe amplitude are as you designed, by sweeping the frequency of a signal generator as the input signal, and observing the output amplitude on an oscilloscope. Record at least three amplitude points in each band (one passband and two stopbands). Hint: If you have trouble with excessive delays between blocks your filter may be too long for the PC to keep up with computation at the specified sample rate.

5. Using the continuously running filter-loop program, listen to the filter output with a CD music signal input. How does it compare to no filter, but using the same sample rate? Demonstrate your running program to the TA for sign off.

6. Drive your filter with a periodic, very narrow analog pulse (ask the TA how to set the function generator to do this. Note that sometimes a lowpass filter will produce easier to understand results in this experiment so you may want to design one rather than use your bandbass filter). Observe the output on the scope. What are you looking at?

7. Have the TA verify your results and lab book entries, and enter your pass-off in Learning Suite.

Conclusions

Document your design procedures and results, including plots of the designed impulse and frequency responses and listings of the MATLAB scripts and/or functions you wrote. Record the Morse code message you decoded (part of your grade on this project will depend on whether you received the correct message.) Write a paragraph or two of conclusions for your lab experience. Discuss any additional implications of what you observed. Describe what you feel are the important principals demonstrated in this lab, and note anything that you learned unexpectedly. What debug and redesign procedures did you need to perform to get it to work?

Questions (Due at beginning of second lab session)

1. List two advantages of using FIR filters rather than IIR filters.

2. Assume a filter is specified with maximum allowable passband ripple of $\Delta_p = 1.0\,\text{dB}$ and stopband ripple of $\Delta_s = -40\,\text{dB}$. What are the corresponding linear scale ripple levels, δ_p and δ_s?

3. Which of the window types discussed in the text (excluding the Kaiser window, which is adjustable) requires the shortest filter length, *N*, for a given transition bandwidth? Which has the highest stopband attenuation?.

4. What MATLAB function can do everything described in Experiment 1 step 2 for you? (Note that you are not permitted to use this function in the lab!!!! We can't make life too easy for you.)

5. Give an equation, and cite sources from the text (page and equation numbers) for the impulse response, $h_d[n]$, of an ideal *high pass* filter.