ME 333: Introduction to Mechatronics

Assignment 6: Introduction to Digital Filtering with the PIC32

Electronic submission due **before** 11:00 a.m. on February 28th

1 Introduction

In this assignment we will be learning some basics of simple digital filtering. We will be using MATLAB to analyze and design filters for an arbitrary noisy signal. We will then implement these same filters on the same signal on the PIC32. **When asked to submit a plot, please make sure to give the plot an x-axis label, y-axis label, and a plot title.**

2 Generating and Analyzing a Signal with MATLAB

The following questions are based on the reading in Chapter 11, *Digital Signal Processing*, of the ME 333 course notes packet. You should base most of your MATLAB code on the sample code included in the chapter.

1. What are the major frequency components of $\sin^4(20\pi t)$? Generate a "sampled" version of this signal in MATLAB assuming a 100 Hz sampling rate and a sampling time of 1 second. Submit a magnitude plot of the signal's FFT. How small of a frequency can the plot detect and how does this relate to the spacing between neighboring frequency components?
2. Plot the magnitude response plot of a seventh-order (eight-sample) MAF in Matlab. What is the cutoff frequency? MATLAB defines the cutoff frequency at -6 dB. What fraction of a sinusoid at 45% of the sampling frequency makes it through the filter? If the signal is a sinusoid at 1.8 times the Nyquist frequency, what is the filter's gain for this signal?
3. Plot the coefficients of a 54th-order FIR LPF designed by Matlab for a cutoff frequency of 0.25 times the Nyquist frequency. Use stem, not plot, to show the coefficients.
4. You can use fir1 to create high-pass, band-stop (notch), and band-pass filters, too. Consult Matlab documentation and design a 50th-order band-stop (notch) filter to remove 50 or 60 Hz noise. Your stop band will be 40 Hz to 70 Hz, and your sampling frequency is 400 Hz. Turn in the command you used to create the filter coefficients, a stem plot of the coefficients, and the frequency response showing the stop band.

3 Using the DSP Libraries on the PIC32

The PIC32 provides several functions for filtering signals, like `mips_fir16`, and performing an FFT on them (`mips_fft16` and `mips_fft32`). The major challenge is understanding the fixed-point representation that Microchip uses internally to perform the operations. A fixed-point number representation treats an integer as if it were a real number. This is done by placing an imaginary decimal point after a certain bit. For example, π might be represented as `double pi = 3.14159` or its fixed-point equivalent, `int pi = 314159`, where the decimal place is implicitly after the 3. A standard fixed-point format used in DSP libraries is known as the

Qn fixed-point representation, where the n represents the number of bits used to represent the fractional part. For example, the Q15 format uses 15 bits as the fractional part of the fixed-point representation. The resolution is then $\frac{1}{2^n} = \frac{1}{2^{15}} \approx 3.05 \times 10^{-5}$. All Qn representations can represent a range of values between -1 and $(1 - \frac{1}{2^n})$. We all know that integers cannot represent values less than 1, so how are, for example, Q15 numbers stored in an int? That's straightforward enough. Any Qn number requires $n + 1$ bits of storage. The MSB is treated as the sign bit. For Q15 that means we can represent integer numbers from -32,768 to 32,767 (the range of a signed 16-bit number or a `short` in C). These numbers serve as the numerator of our fractional representation. The denominator is always $2^{15} = 32768$ (or for general Qn, it's 2^n). For Q15, we see that the range of values that can be represented are between $\frac{-32{,}768}{2^{15}} = -1$ and $\frac{32{,}767}{2^{15}} = 1 - \frac{1}{32{,}768}$, as stated earlier. Hence, we only need to store the numerator when writing our numbers in the Q15 format.

In NU32DSP.c, we've taken care of the scaling down to the Q15 format and scaling back up to the original values that would typically be done in a DSP application. Feel free to take a look at this library to get an idea of how to implement a FIR filter and compute the FFT of a signal. We've provided even more examples, like how to use the 32-bit version of the FFT, online on the ME333 Sample Code wiki page.

4 Programming Digital Filters on the PIC32

Congratulations! You've recently been hired by NU Corp as their new software developer. Although many in the company see you as untested with little experience, the CEO has decided to take a chance with you after a major falling-out with the previous developer. The company is looking for someone to implement a digital low-pass filter. The design specs are that the filter must pass frequencies below 52 Hz with a gain of at least 0.5 and attenuate signals with frequencies above 235.52 Hz with a gain of at most 0.1. It appears as if the previous programmer has made a lot of progress on the design, which uses a 7$^{\text{th}}$ order moving average filter (MAF) and a sampling rate of 1024 Hz. The main reason you got the job is because of your claim of being able to write a more efficient MAF that takes fewer operations than the current implementation.

1. The previous programmer implemented an n^{th}-order MAF as

$$\texttt{output}[k] = \sum_{j=0}^{n} \frac{\texttt{input}[k-j]}{n+1}$$

   You claimed you could write a more efficient version than this, which for an n^{th} order MAF requires $n + 1$ multiplications and n additions. How would you write a function, `void MAF(double input[], int input_length, double output[], int n)`, that computes the values of an element in the output array, i.e., `output`$[k]$, efficiently using only two multiplications, one addition, and one subtraction for a MAF of any order n? One thing to consider when writing this function is how to implement the first few terms of the MAF. A common approach is to assume that the sampled input signal is zero for all $t < 0$. Submit your solution as a typed response.

2. NU Corp has provided you with a sample signal, `noisySignal` in NoisySignal.h, to test your filter on, but they forgot to tell you the fundamental frequencies of the signal. Using the built-in FFT function on the PIC, what are the major frequency components of the test signal? Luckily you kept around your ME 333 reference code, `dsp_sampling.c`, that essentially sets up the FFT for you. After computing the FFT, the program sends the result, along with several other values, over serial. Use a terminal program to save these values to a text file[1] and plot the results in a program, like MATLAB or Excel. If you use MATLAB, then the following snippet of code will read in the text file. Make sure to first remove any lines of text at the top of the file. The file should only consist of numbers. The code also documents what signal or FFT each column pertains to.

---

[1] Most terminal programs are capable of doing this. If you can't figure out how in your preferred terminal program, the NU32_Utility debug terminal can do this.

```
name = 'temp.txt';
DATA = load(name);
% first column is noisySignal
SIG = DATA(:,1);
% second is MAF
MAF = DATA(:,2);
% third is FIR
FIR = DATA(:,3);
% fourth is FFT of orig
FFT_SIG = DATA(1:length(DATA)/2+1,4);
% fifth is FFT of MAF
FFT_MAF = DATA(1:length(DATA)/2+1,5);
% sixth is FFT of FIR
FFT_FIR = DATA(1:length(DATA)/2+1,6);
```

3. Implement a MAF in `dsp_sampling.c` on your PIC and test it on `noisySignal` in NoisySignal.h. You should use the efficient MAF implementation, but you are allowed to implement a different version. Again, consider how to implement the first few terms of the MAF. A common approach is to assume that the sampled input signal is zero for all $t < 0$. Did your MAF remove the high frequency noise according to the NU Corp specifications? Prove it by taking the FFT on the PIC of the filtered signal. Submit a plot of the results of the FFT.

4. NU Corp only gave you one test case to test your filter on, which you know isn't enough to declare your filter a success. You also have this feeling that you've seen the frequency response of this MAF filter before. If you have not already done so, how could you use MATLAB's `freqz` to compute the frequency response of your digital filter? You should realize that it does not meet the design specifications. What is the gain of a sinusoid at 0.6 of the Nyquist frequency?

5. You bring your findings up at the next meeting and suggest that instead of a MAF the company should be using a finite-impulse response (FIR) filter. The company still wants the computations to be done quickly, so you cannot choose an arbitrarily high-order FIR. At the same time, you will eventually need to implement your FIR on the PIC and its digital signal processing (DSP) library requires that all of its FIR filters be multiples of four with a minimum 3rd-order filter. Design the lowest-order FIR filter that meets or exceeds NU Corp's specs, but can still be implemented on the PIC. What is the order of your FIR? Submit a stem plot of the coefficients. Also submit the magnitude response of this filter and label the cutoff frequency and 0.1 gain line. After you've designed your filter, copy the coefficients of your FIR filter into the array `FIR` in dsp_sampling.c, replacing the current values. Don't forget to update the value of `FIR_LENGTH` to reflect the new length of the modified array.

6. Some of your peers at NU Corp are suspicious about the PIC's DSP library. They claim that there must be significant error due to the fact that everything is done using integer math. MATLAB must be more accurate because it does everything using floating point math. As an initial response to their claims, you compare the PIC's and MATLAB's results of the FIR and FFT of `noisySignal`. Generate and submit the following plots:[2]

   (a) A plot of the signal after going through a FIR filter in MATLAB and on the PIC, along with the original signal. Generate the plot with the sample number on the x-axis, and signal amplitude on the y-axis.

   (b) A stem plot of MATLAB's and the PIC's FFT of the FIR filtered signal. On the x-axis, plot the normalized frequency (should range from 0 to 1, where 1 is the Nyquist frequency), and on the y-axis plot, amplitude.

---

[2] You may use whatever plotting tool that you are comfortable with if you don't like importing the text files into MATLAB.

What would you say to your peers regarding MATLAB's results when compared to the PIC's? Do your results support or refute their claim?

5 What to Turn In

There will be no in-class demo for this assignment. Simply submit your code and responses in a zip file before the assignment due date using the naming convention lastname_firstname_a6.zip.