ECE 5630: Digital Signal and Image Processing - Multi-Rate Processing

Due: November 20, 2018

Name (Print):

Objectives

This programming assignment has three objectives:

1. Give you practice in using a program to perform digital signal processing.
2. Help to solidify your understanding of multi-rate processing, and in particular the concepts of decimation and interpolation.
3. Determine the specifications required for a digital filter.

Instructions

1. Use a sample rate conversion of $\frac{L}{M} = \frac{3}{4}$ for this assignment.
2. The program should be written in C or C++. No external libraries should be used – you are writing the entire convolution from scratch.
3. Use the `.wav` file `ghostbustersray.wav` to test your program. This file is available on Canvas and is sampled at 11,025 Hz.
4. Use the Matlab tool `fdatool` to design your FIR filter.
5. For all of the frequency response plots, the normalized frequency running along the x-axis in your plot should be scaled to units of cycles/sample and should cover the range from 0 to 1.
6. For all of the magnitude frequency response plots, the y-axis should be scaled to units of dB by computing $(20 \log_{10} |H(f)|)$.
7. Write out both the final processed results in a `.wav` file for the TA to listen to. Make sure the samples are scaled high enough so that the signal can be played back with a comfortable volume level. Upload the files with your report.
8. Include your work, answers, plots, and code within a `.pdf` document report and upload to the Canvas website. Include your observations and any other appropriate comments. The report should include an introduction, a section describing your approach and findings, and a conclusion.

---

1. Design a linear phase FIR low-pass (prototype) filter $H(z)$ to be used in your sample-rate converter. Be sure that the cutoff of the filter is appropriate for the conversion.
   (a) Plot the impulse response $h(n)$ of the filter.
   (b) Plot the magnitude and phase response.
   (c) What is the length of the filter?
   (d) What method did you use to design the filter?
   (e) What are the pass-band and stop-band edge frequencies in the filter you designed?

(f) What is the size of the ripple in the pass-band (make this small ≤ 0.001 dB)?

(g) What is the peak side-lobe level (in dB) (make this ≥ 80 dB down from the passband)?

2. Using the simple and noble identities, derive the signal flow graph for a polyphase filter that can change your sample rate by a factor of $\frac{3}{4}$ using the minimum number of operations/sample. Draw the final signal flow graph that will be implemented in your program. Assume an input sample rate of 11.025 kHz.

   (a) What is the final sample rate of the output?

   (b) What is the number of operations (multiplies and adds) required per input sample time?

3. If the prototype filter had an impulse response $h(n)$ of length K, show how you would separate the coefficients to the $M \times L$ polyphase type filters $R_{l,m}(z), l = 0, \ldots, L, m = 0, \ldots, M$.

4. In C or C++, write code to implement the straightforward realization for decimation ($\frac{L}{M}$) by up-sampling by L, filtering with the low-pass filter with response $H(z)$, and downsampling by M. Your program should be able to handle any length signal (possibly infinite) without buffering the entire signal first. Then do the following.

   (a) Generate a cosine signal with frequency f_0, $x(n) = \cos(2\pi f_0 n)$ for $n = 0, 1, \cdots, N-1$. Let N be chosen so that it is at least 30 times the length of the $h(n)$ filter. If possible, let N be 100 times longer than the length of your prototype filter. Pass $x(n)$ through the decimator. Let $y(n)$ be the output.

   (b) Plot the 512-point DFT magnitude of $x(n)$ and $y(n)$ on the same plot.

   (c) Repeat this for each of the frequencies f_0 in the table below. Record in the table the frequency $\hat{f}_0$ of the decimated signal found from the DFT.

f_0	$\hat{f}_0$
$\frac{1}{16}$	
$\frac{1}{8}$	
$\frac{1}{4}$	

5. In C or C++, write code to implement the polyphase filter bank realization from part 2 above for decimation (by $\frac{L}{M}$) using the low-pass filter with response $H(z)$. Your program should be able to handle any length signal (possibly infinite) without buffering the entire signal first. Then repeat 4(a)–(c).

f_0	$\hat{f}_0$
$\frac{1}{16}$	
$\frac{1}{8}$	
$\frac{1}{4}$	

6. Use the Matlab function `wavread()` to generate the samples of the file `ghostbustersray.wav`. Decimate the signal using your programs from parts 4 and 5. Time how long it takes to decimate the signal with each of the methods. Import the resulting files back into Matlab, and play the original signal (using `sound()`) at a sampling rate of 11,025 Hz, and the processed signals from both methods at the decimated sample rate. How do the execution times compare? Do they all sound the same? To pass this off, have the TA listen and confirm your results.

   Write out the final processed results in a `.wav` file for the TA to listen to. Make sure the samples are scaled high enough so that the signal can be played back with a comfortable volume level. Upload the files with your report.