ECE 340 – Computer Assignment #2

Interpolation and Matched Filtering

Due by 5 pm on Wednesday, Dec. 7

The second computer assignment will explore signal reconstruction and matched filtering. In the first part of the project, you will write a script to interpolate a discrete-time signal to a sample rate that is L times higher than the input signal. The input signal and interpolation factor will be inputs to your program. As a test signal, you will interpolate a common radar waveform called a linear frequency modulated (LFM) pulse. In the second part of the project, you will pass this signal through a specific type of filter called a matched filter. The matched filter is important because it gives the highest possible signal-to-noise ratio (SNR) at the output of the filter.

I will talk you through the steps of the assignment using this handout. You should observe the results of your simulations, and you should make (well-labeled) plots to demonstrate important details about the project. Use those observations to prepare a typed report explaining your methods and results for each section. There is not a specific format for the report, but be sure to include details that show you did the work and that you understand what is happening. Include plots only when they convey something significant. I will be looking for insight that shows a depth of understanding about these experiments. At a minimum, your report should include an introduction describing what the project is about and what the objectives are; a section discussing the techniques or methods that you used (including theoretical results if appropriate); a section describing your results; and a section with your conclusions. Any questions that I ask in this assignment should be answered, but more importantly these questions are designed to get you started in thinking about the assignment. The best reports will demonstrate analysis and thought beyond the basic questions posed in this handout. Listings of your Matlab code, appropriately labeled and commented, should be included.

You should work with **one** partner and turn in a joint report. I prefer for everyone to have a partner rather than working on your own. Groups may talk to each other about the assignment, **but all Matlab code and report materials must be the result of your own group's work**. Code copied from another group is unacceptable.

Part I: Interpolation

We have already covered the basics of interpolation in class – the discussion can be found in Section 3.2. We know that to increase the sample rate of a signal by a factor of L, we need to interpolate $L - 1$ new samples between every sample that we already have. We also know from our discussion of reconstructing continuous-time signals from discrete-time samples that the ideal interpolation function is a *sinc* function (see Section 8.2 of your textbook). If we can use the sinc function to interpolate a continuous function of time, then certainly we can also use it to interpolate $L - 1$ new samples to create a higher-rate discrete-time signal.

Suppose that a discrete-time signal has been captured by sampling a continuous-time signal at a uniform sampling rate of $F_s = 1/T$. The ideal interpolation formula for reconstructing the continuous-time signal is

$$x(t) = \sum_{n=-\infty}^{\infty} x[n]\, h(t - nT) \tag{1}$$

where $h(t) = \mathrm{sinc}(\pi t/T)$. To interpolate a discrete-time signal by a factor of L, we can use this interpolation formula evaluated at $t = mT + (p/L)T = (m + (p/L))T$ for $p = 1, 2, \ldots, L - 1$. For example, suppose we would like to increase the sample rate by a factor of $L = 4$. We already have the samples $x[0] = x(0)$ and $x[1] = x(T)$, and we need to interpolate $x(t = 0.25T)$, $x(t = 0.5T)$, and $x(t = 0.75T)$. These values correspond to $m = 0$, $p = 1$, 2, and 3. Substituting the interpolation times into (1), we have

$$\begin{aligned} x\left(mT + \left(\frac{p}{L}\right)T\right) &= \sum_{n=-\infty}^{\infty} x[n]\, h\left(mT + \left(\frac{p}{L}\right)T - nT\right) \\ &= \sum_{n=-\infty}^{\infty} x[n]\,\mathrm{sinc}\left(\pi\left(m + \frac{p}{L} - n\right)\right) \end{aligned}. \tag{2}$$

As we discussed in class, to implement this interpolation as a filtering operation, we must first inject the new values into the sequence as zeros. To do this, define a new input sequence

$$\tilde{x}[n'] = \begin{cases} x\left[n'/L\right] & n' = \cdots, -2L, -L, 0, L, 2L, \cdots \\ 0 & else \end{cases}. \tag{3}$$

We can re-write (2) using this new input sequence as

$$\begin{aligned} x\left(mT + \left(\frac{p}{L}\right)T\right) &= \sum_{n'=-\infty}^{\infty} \tilde{x}[n']\, h\left(mT + \left(\frac{p}{L}\right)T - \left(\frac{n'}{L}\right)T\right) \\ &= \sum_{n=-\infty}^{\infty} x[n']\,\mathrm{sinc}\left(\pi\left(m + \frac{p - n'}{L}\right)\right) \end{aligned} \tag{4}$$

Next, we manipulate some of the arguments to obtain

$$\begin{aligned} x\left(mL\left(\frac{T}{L}\right) + p\left(\frac{T}{L}\right)\right) &= \sum_{n'=-\infty}^{\infty} \tilde{x}[n']\, h\left(mL\left(\frac{T}{L}\right) + p\left(\frac{T}{L}\right) - n'\left(\frac{T}{L}\right)\right) \\ x\left((mL + p)\left(\frac{T}{L}\right)\right) &= \sum_{n=-\infty}^{\infty} x[n']\,\mathrm{sinc}\left(\pi\left(\frac{mL + p - n'}{L}\right)\right) \end{aligned}. \tag{5}$$

Finally, we define a new sequence variable $k = mL + p$ and the new sampling interval $T' = T/L$ to obtain

$$\begin{aligned} x(kT') = x[k] &= \sum_{n'=-\infty}^{\infty} \tilde{x}[n']\, h(kT' - n'T') \\ &= \sum_{n'=-\infty}^{\infty} \tilde{x}[n']\, h[k - n'] \end{aligned} \tag{6}$$

where $h[n'] = h(n'T') = \text{sinc}(\pi n'/L)$.

Equation (6) is in the form of a discrete-time convolution, which expresses the output of a LTI discrete-time system due to a discrete-time input. Obtaining this expression took several steps because the sampling rates of the input sequence, discrete-time impulse response, and output sequence must all be equal.

In summary, to *upsample* a discrete-time sequence to a rate that is higher than the original by a factor of L, the following steps are required:

1. Create a new input sequence $\tilde{x}[n']$ by inserting $L - 1$ zeros between each sample $x[n]$ of the original input sequence;
2. Define a discrete-time interpolation filter (a discrete-time lowpass filter!) $h[n']$ at the higher sampling rate. The impulse response of this filter is defined by $h[n'] = \text{sinc}(\pi n'/L)$;
3. Perform a discrete-time convolution of $\tilde{x}[n']$ and $h[n']$ to obtain the output sequence.

The ideal interpolation filter for constructing the continuous-time signal was defined above as $h(t) = \text{sinc}(\pi t/T)$. Using our variables that enabled the rate changes, we can see that

$$h[n'] = h(n'T') = h(n'T/L) = \text{sinc}\left(\pi (n'T/L)/T\right) = \text{sinc}(\pi n'/L). \tag{7}$$

In other words, the ideal discrete-time interpolation filter is obtained by sampling the ideal continuous-time interpolation filter at the higher, upsampled rate. This filter has an infinite number of samples, which is why the discrete-time convolution in (6) requires an infinite summation. As we've studied before, we can make a practical, finite-duration, causal filter by delaying the ideal impulse response and applying a finite-duration window function.

I would like you to create a practical discrete-time interpolating filter. Use the lowpass filter techniques we have already developed to approximate the impulse response of a lowpass filter with cutoff frequency equal to $F_s/2$ (half the original sample rate). Remember that when you delay the impulse response by D samples to make it causal, the output will be delayed by the same amount. Your practical interpolation filter should be sampled at a rate of LF_s.

I would like you to test out your interpolation filter and Matlab script on a linear frequency-modulated (LFM) radar waveform. This waveform is defined by:

$$x(t) = \cos\left(2\pi f_0 t + \pi\gamma t^2\right) \tag{8}$$

where f_0 is the starting frequency and γ is known as the *chirp* rate given in units of Hz per second. The waveform's name can be understood by evaluating the instantaneous frequency of the waveform. The instantaneous frequency is given by

$$f_i(t) = \frac{1}{2\pi}\frac{d}{dt}\left(2\pi f_0 t + \pi\gamma t^2\right) = f_0 + \gamma t\ . \tag{9}$$

From (9), we can see that the frequency of the radar pulse is equal to f_0 and $t = 0$. The frequency then increases linearly with time; hence, the name linear FM.

Generate an LFM waveform that begins at $t = 0$ and ends at $t = T_p = 0.1$ ms. Let the chirp rate be 2×10^{10} Hz/s. Let the initial frequency be $f_0 = 0$. Therefore, the frequency at time $t = 0$ is $f_i(0) = 0$ and the frequency at the end of the radar pulse is $f_i(T_p) = \gamma T_p = 2$ MHz. In turns out, that this ending frequency is also approximately the highest frequency component in the signal. Therefore, create a discrete-time representation of this radar pulse at a sampling rate of 5 MHz, which is a little faster than the approximate Nyquist sampling rate.

Once you have generated this pulse, input it into your interpolation script to obtain the radar pulse sampled at a higher rate. Try several different values for the interpolation factor L and plot your results. Since you have the original equation for the continuous-time pulse, you might try comparing the interpolated signal to the signal you would have obtained by evaluating (8) at the higher rate directly.

Part II: Matched Filtering

The matched filter is important because is maximizes average SNR at the output of the filter. No other filter impulse response can give a better average SNR. In this section, you will perform discrete-time convolution of the LFM radar pulse with a time-reversed version of itself. You will then add noise to the input signal and view the matched filter output.

The matched filter has the same shape is the signal that it is being matched to. Therefore, if you are matching to a LFM waveform as described above in (8), the impulse response of the matched filter is

$$h_m(t) = x(T_0 - t) = \cos\left(2\pi f_0 (T_0 - t) + \pi\gamma (T_0 - t)^2\right). \tag{10}$$

It turns out that the value T_0 is arbitrary – it's purpose is to delay the filter's impulse response to make it a causal filter, much in the same way that we've been adding delays to our lowpass filters. We won't worry about this parameter for now. Instead, once you have a discrete-time version of the LFM waveform from the previous part of this assignment, just make your matched

filter by flipping the array it is stored in (check out the *fliplr* and *flipud* commands). Once you have the LFM waveform and the matched filter, convolve the two sequences using the *conv* command and plot the output. Although I'm not worried about the absolute time delay, your horizontal axis should have the correct time scale.

What does the output of the matched filter look like? How wide is the main lobe of the output? What happens to the width of this main lobe if you change the bandwidth of the radar pulse? In radar, we sometimes refer to this processing step as *pulse compression* – do you have any explanations why this name makes sense? Remember that you might need to change the sampling rate if you increase the bandwidth, and every time you change a parameter of the radar pulse, you must also change the matched filter.

Finally, add noise to the input radar pulse (not to the matched filter impulse response) until it becomes difficult to see the shape of the pulse. You can add noise using the *randn* command in Matlab. For example, if your input pulse is N samples long, you can create noise with the command

```
noise = sqrt(Pn)*randn(N,1);
```

The variance of this noise is Pn. Add this noise array to your input pulse and comment on how the output has changed. Can you speculate on the value of the matched filter for detecting signals that are obscured by noise? How do things change if you change the radar chirp rate or the radar pulse duration, but keep the same noise variance?