Lab 7
April 19, 2016
Sample Rate Conversion and Image Filtering

INSTRUCTIONS:
All lab submissions include a written report and source code in the form of an m-file. The report contains all plots, images, and figures specified within the lab. All figures should be labeled appropriately. Answers to questions given in the lab document should be answered in the written report. ***The written report must be in PDF format***. Submissions are done electronically through **my.ECE**.

1 Discussion

Sample rate conversion is, by definition, changing the sample rate of a discrete signal. Upsampling is the addition of samples to a signal to effectively increase the sample rate of the signal. Conversely, downsampling is the decimation of a signal to effectively decrease the sample rate. In either case, the total length of the signal (in time) does not change. When the number of samples increases, the sample rate decreases such that the total time remains the same. For example, music can be sampled at a low bitrate is the same length as music sampled at a high bitrate. A higher bitrate just corresponds to more samples per second which corresponds to higher frequencies captured.

2 Upsampling and Downsampling

A discrete signal must be band limited in order for the signal to be upsampled. A signal which is not bandlimited can still be upsampled, but the results will be undesirable. Upsampling is a 2 step process. For example, upsampling by U corresponds to inserting $U-1$ zeros between each sample of the original signal. More precisely, given that $x(n)$ is the original signal with length N and $x_{\uparrow}(n)$ is the upsampled signal,

$$x_{\uparrow}(m) = \begin{cases} x(n) & m = Un \\ 0 & \text{else} \end{cases} \tag{1}$$

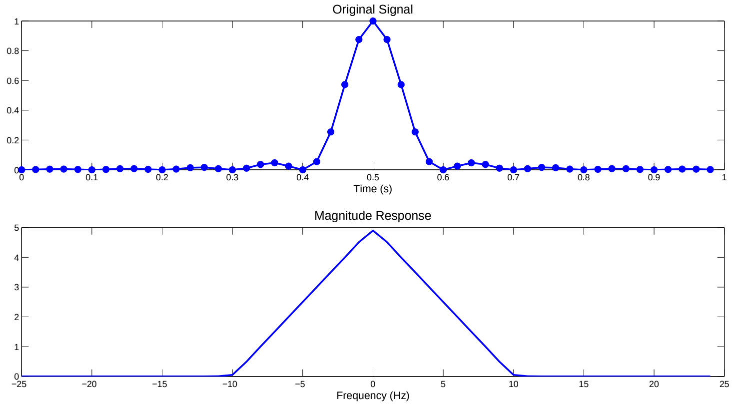

Figure 1: The original signal $x(n)$ and its magnitude response.

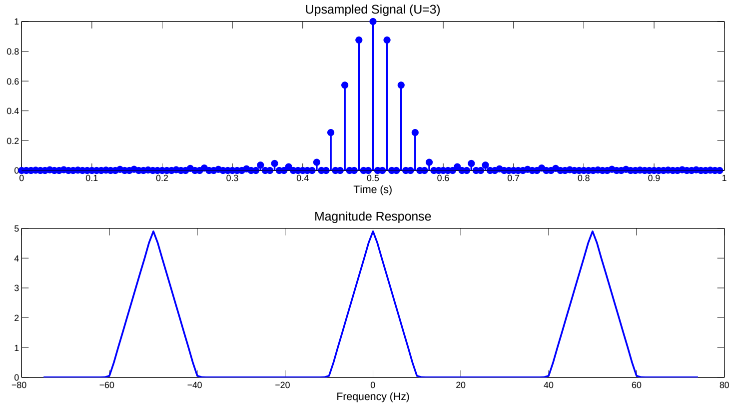

Figure 2: $x(n)$ upsampled by $U = 3$. This increases the sampling frequency by 3 and also introduces copies in the frequency domain due to the scrunching of the frequency axis.

The insertion of $U-1$ zeros between samples of $x(n)$ causes the frequency axis to scrunch by a factor of U. However, this scrunching is only apparent in the DTFT, where $-\pi < \omega < \pi$. In the Ω domain, there is no scrunching. That's because $\Omega = \omega \cdot f_s'$ where $f_s' = f_s U$. So the scrunching in the ω domain comes from the fact that f_s' for $x_{\uparrow}(n)$ is larger than f_s. Pay special attention to the time axis of each graph. The duration of the signal from a time perspective does not change.

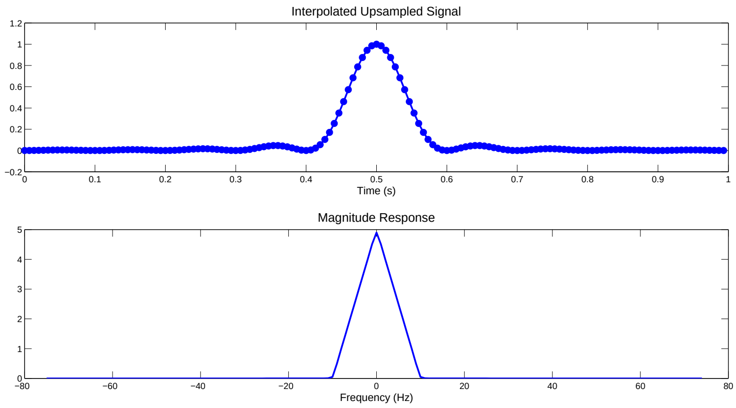

Figure 3: Applying an ideal filter LPF to the frqeuency to remove the copies.

The spectrum of $x_{\uparrow}(n)$ has the same shape as the spectrum of $x(n)$, barring the additional copies. To remove the additional copies, we can apply an ideal LPF. In MATLAB, this can be done by letting $X_{\uparrow}(f) = 0$ for $f > f_s/2$ and $f < -f_s/2$ where $X_{\uparrow}(f)$ is the Fourier transform of $x_{\uparrow}(n)$. Additionally, one must also vertically scale $X_{\uparrow}(f)$ by U such that the energy in $x_{\downarrow}(t)$ and $x(t)$ are the same. The result is shown in Figure 3. Figure 4 superimposes $x(t)$ and $x_{\uparrow}(t)$ to show that one is the upsampled version of the other.

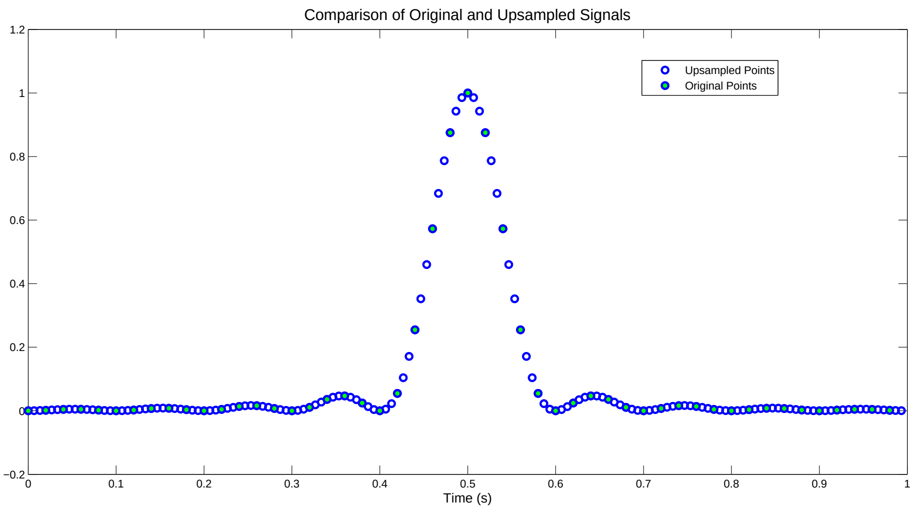

Figure 4: Comparison of the original signal $x(n)$ to its upsampled version $x_{\uparrow}(n)$.

By comparison, downsampling is much simpler than upsampling. The downsampling operation can be expressed as

$$x_{\downarrow}(n) = x(nD) \tag{2}$$

where D is an integer greater than 1. The choice of D is restricted by Nyquist sampling rate, which can be violated if D is chosen to be too large. Upsampling and downsampling can be combined to achieve the desired sample rate. The final sample rate can be expressed as $f_s' = \frac{U}{D} f_s$.

Report Item: An audio technician accidentally set the sample rate to 66,150 Hz during an important recording session. This is 1.5 times the standard 44,100 Hz. Having already said goodbye to the musicians, he happens to come across an ECE311 student who understands sample rate conversion. Read the sound *audioclip.wav* using **audioread**. What values of U and D can you choose such that the final sample rate is 44,100 Hz. Plot the magnitude spectrum (in dB) of the upsampled signal before applying the lowpass filter, after applying the lowpass filter, and post downsampling. Be sure to scale the frequency axis Ω with respect to the proper sampling frequency. Use **sound** to play the original sound and the sound after sample rate conversion. Is the original sound faster or slower compared to the new sound?

3 2-D DFT

The definition of 2-D DFT is

$$X(u,v) = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} x(m,n) e^{-i2\pi(mu/M+nv/N)} \tag{3}$$

Like the 1-D DFT, the 2-D DFT uses sinusoidal basis functions—except they're in 2-D. In 1-D, we're used to dealing with time and frequency. 2-D DFT is often used for image processing where we deal with space and, correspondingly, spatial frequency. Assume that $x(m,n)$ represents an $M \times N$ image where $m = 0, \ldots, M-1$ and $n = 0, \ldots, N-1$. Then u indexes variations in the vertical direction and v indexes variations in the horizontal direction. Physics students may be more familiar with the wavenumber notation

$$k_x = 2\pi v/N, \tag{4}$$
$$k_y = 2\pi u/M \tag{5}$$

where k_x represents variations in the horizontal direction and k_y represents variations in the vertical direction.

The 2-D DFT can be re-written as

$$X(u,v) = \sum_{m=0}^{M-1} e^{-i2\pi mu/M} \sum_{n=0}^{N-1} x(m,n) e^{-i2\pi nv/N} \tag{6}$$

which can be implemented using a series of FFTs. The number of operations is approximately $\mathcal{O}(MN \log N) + \mathcal{O}(NM \log M)$ or $\mathcal{O}(NM \log NM)$.

Report Item: Create a function called *myDFT2* that implements the 2-D DFT using the 1-D FFT. Verify your results using **fft2**.

$$f(m,n) = \cos(k_y m + k_x n)$$

Then implement $f(m,n)$ using **meshgrid**. Use **imagesc** to plot $f(m,n)$ for $(k_x, k_y) = (\pi/4, 0)$ and its 2-D DFT (magnitude only). Repeat for $(k_x, k_y) = (0, \pi/4)$, $(k_x, k_y) = (\pi/4, \pi/4)$, and $(k_x, k_y) = (\pi/4, -\pi/4)$. Make sure to label both k_x and k_y axes.

One should notice that, like 1-D Fourier transforms, the 2-D Fourier transform increases in frequency further away from the origin. However, there is an additional component to 2-D Fourier transforms not seen in 1-D: the idea of *rotation*. So not only do we have magnitude, phase, and frequency, we also have rotation angle.

4 Image Filtering

An image can be filtered using 2-D convolution, which can be expressed as

$$Y(m,n) = \sum_{m'} \sum_{n'} X(m',n') h(m-m', n-n') \tag{7}$$

where $h(m,n)$ is the filtering kernel. An example of a low-pass filter is

$$h(m,n) = \begin{bmatrix} 1/8 & 1/16 & 1/8 \\ 1/16 & 1/4 & 1/16 \\ 1/8 & 1/16 & 1/8 \end{bmatrix} \tag{8}$$

There are several ways to recognize that this is a lowpass filter. First, using the fact that the DC component of the Fourier transform is proportional to the sum of the coefficients in $h(m,n)$, it can be seen that $\sum_m \sum_n h(m,n) = 1$. Since the DC component is non-zero, it is likely a lowpass filter. Another clue is the fact that the filter looks Gaussian. Since a Gaussian in the spatial domain is also a Gaussian in the frequency domain, the filter is a lowpass.

Report Item: Load *image1.jpg* using **imread**. This image contains what is known as salt and pepper noise. Filter this image using the lowpass filter given in (8) and plot the result using **imagesc**. Use **conv2** to apply the filter.

In contrast, the Laplacian filter is a highpass filter. This can be deduced from the fact that the sum of its coefficients equals zero, which suggests that it cannot be a lowpass. A highpass filter can be used for edge detection in an image.

$$h(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad \text{(Laplacian Filter)} \tag{9}$$

Report Item: Load *image2.jpg* using **imread**. Apply the Laplacian filter and plot the resulting figure using **imagesc** with **colormap('gray')**. Use **conv2** to apply the filter.