ECE 310 Fall 2023

Lecture 32

Upsampling and interpolation

Corey Snyder

Learning Objectives

After this lecture, you should be able to:

- Describe how upsampling affects a signal in the time-domain and frequency-domain.
- Explain the structure and reasoning for using an interpolation filter in an upsampling system.
- Implement a non-integer rate conversion scheme using interpolator and decimator systems.

Recap from previous lecture

In the previous lecture, we introduced the concept of digital rate conversion and explored the first of two operations: downsampling. We demonstrated how downsampling changes a signal in both the time-domain and frequency-domain. We will discuss the other fundamental rate conversion in this lecture, known as upsampling. We will again show the effects in the time-domain and frequency-domain and other practical considerations

1 Upsampling

1.1 Upsampling and the time-domain

We saw in the previous lecture how downsampling reduced the length of a discrete-time signal and implicitly decreased the sampling rate of our digital signal. We now will discuss the complementary rate conversion process known as *upsampling*. Upsampling is the discrete-time process by which we increase the implicit sampling rate of the signal by an integer factor U. We accomplish upsampling by inserting $U-1$ zeros after each sample. Let $y[n]$ be the result of upsampling $x[n]$ by U. We may write $y[n]$ as follows:

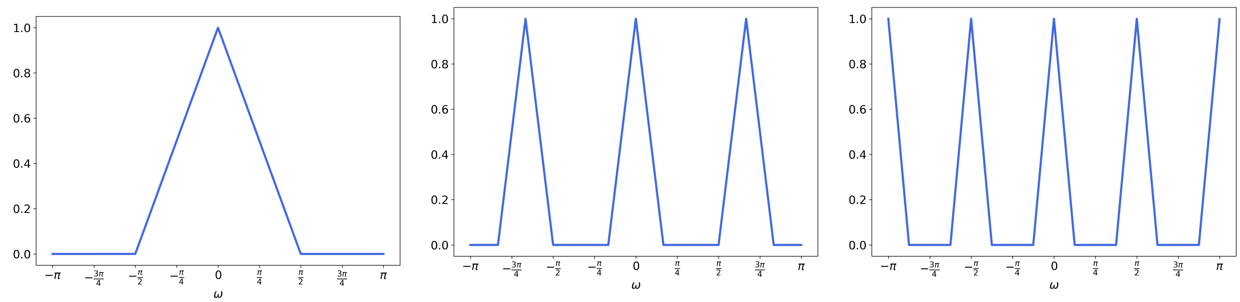
$$y[n] = \begin{cases} x\left[\frac{n}{U}\right], & n = 0, U, 2U, \dots \\ 0, & \text{else} \end{cases}. \tag{1}$$

For example, let U=3. Below, we show an example of upsampling a simple signal by U:

$$x[n] = \{3, 1, 1, 4\} \tag{2}$$
$$y[n] = \{3, 0, 0, 1, 0, 0, 1, 0, 0, 4, 0, 0\}. \tag{3}$$

Like with downsampling, we can intuitively show how upsampling increases the sampling rate of our digital signal. Suppose again we record a piece of audio given by $x(t)$ for 5 seconds at $f_s = 1$ kHz to obtain $x[n]$. The discrete-time signal $x[n]$ will have 5,000 samples that follow the relation

$$\begin{aligned} x[n] &= x(nT) &(4)\\ &= x\left(\frac{n}{1000}\right),\ 0 \leq n < 5{,}000. &(5) \end{aligned}$$



(a) $X(\omega)$ (b) $Y_1(\omega)$ (U=3) (c) $Y_2(\omega)$ (U=4)

Figure 1: Examples of effects of upsampling in the frequency domain. Plot (b) shows the resulting spectrum from upsampling the given $x[n]$ by U=3 to obtain $y_1[n]$. Plot (c) shows the result from upsampling $x[n]$ by U=4 to compute $y_2[n]$.

If we upsample $x[n]$ by $U = 4$ to obtain $y[n]$, we will have $y[n]$ is length-20,000. These 20,000 samples still correspond to the same 5-second long audio clip. Thus, we have a new sampling frequency of

$$\frac{\text{20,000 samples}}{\text{5 seconds}} = 4 \text{ kHz}. \tag{6}$$

This demonstrates how upsampling increases the sampling rate by a factor of U. We can also say that upsampling reduces the sampling period T by a multiple of U to achieve new sampling period T_U:

$$T_U = \frac{1}{U} T. \tag{7}$$

1.2 Upsampling and the frequency-domain

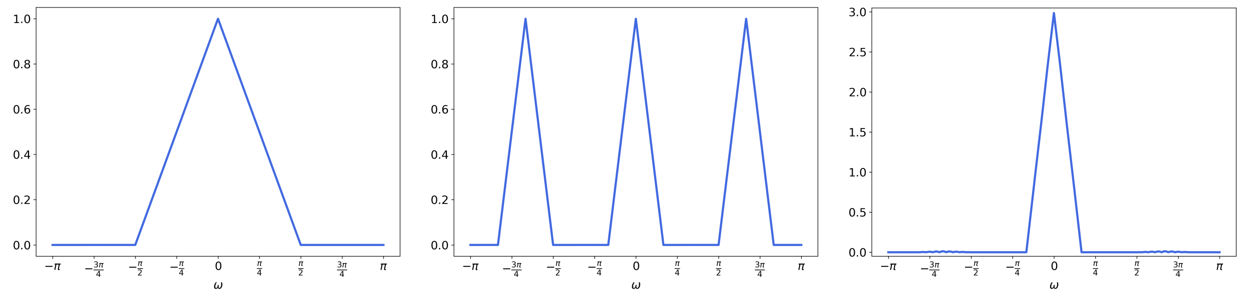
Next, we must look at how upsampling changes the DTFT of a discrete-time signal. Let $x[n]$ and $y[n]$ be the original and upsampled signals, respectively.

$$\begin{align} Y(\omega) &= \sum_{n=-\infty}^{\infty} y[n] e^{-j\omega n} \tag{8} \\ &= \sum_{m=-\infty}^{\infty} y[Um] e^{-j\omega Um} \tag{9} \\ &= \sum_{m=-\infty}^{\infty} x[m] e^{-j\omega Um} \tag{10} \\ &= X(U\omega). \tag{11} \end{align}$$

Above, line 9 follows from the substitution of $n = Um$ and acknowledging that $y[n]$ is zero for all indices that are not multiples of U. Thus, we obtain the important relation that upsampling $x[n]$ by the factor U to obtain $y[n]$ yield the DTFT

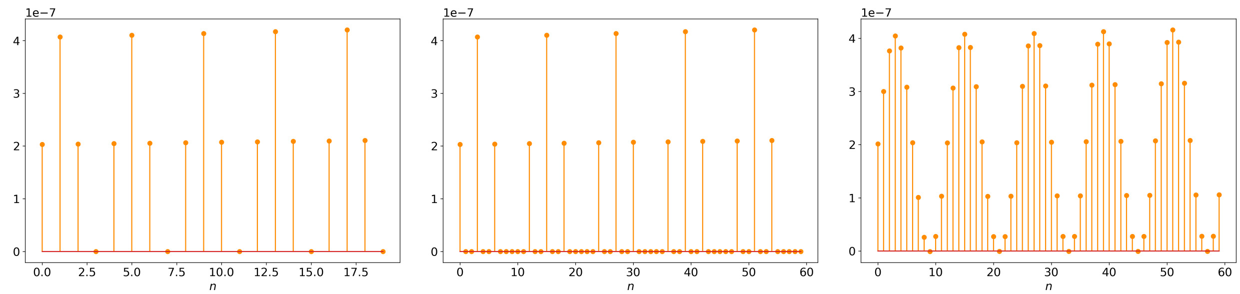
$$Y(\omega) = X(U\omega). \tag{12}$$

In words, Eqn. 12 tells us that upsampling compresses the frequency spectrum by the upsampling rate U. This provides a nice intuitive contrast to downsampling where we stretched the DTFT spectrum. Now, with upsampling, we make the frequency axis tighter. Figure 1 shows examples of upsampling a given signal with two different values of U. We see in the figure that compressing the frequency axis causes adjacent spectral copies to appear in the central $[-\pi, \pi]$ range. For example, when $U = 3$, we see the copy centered at 2π for $X(\omega)$ now appears at $\frac{2\pi}{3}$ in the $Y(\omega)$ spectrum.



(a) $X(\omega)$ (b) $Y(\omega)$, upsampling by U=3 (c) $Z(\omega)$, after interpolation filter

Figure 2: Example of entire interpolator scheme in the frequency-domain. Plot (a) shows the original spectrum $X(\omega)$, plot (b) gives the spectrum after upsampling $Y(\omega)$, and plot (c) gives the final result after the interpolation filter $Z(\omega)$.



(a) $x[n]$ (b) $y[n]$, upsampling by U=3 (c) $z[n]$, after interpolation filter

Figure 3: Example of entire interpolator scheme in the time-domain. Plot (a) shows the original signal $x[n]$, plot (b) gives the signal after upsampling $y[n]$, and plot (c) gives the final result after the interpolation filter $z[n]$.

1.3 Interpolation filter

The plots in Figs. 1b and 1c show how upsampling introduces spectral copies in the central $[-\pi, \pi]$ range. If we performed digital-to-analog conversion with our new sampling period T_U, these spectral copies would be present in the recovered analog signal. Thus, we would like to only retain the original center copy after performing upsampling.

Before upsampling, we know that the central copy of the DTFT is contained between $\omega \in [-\pi, \pi]$. After upsampling, the frequency axis compresses by the factor U and thus the original central copy is now held in $\omega \in \left[-\frac{\pi}{U}, \frac{\pi}{U}\right]$. We can then extract only the original central copy by applying a low-pass *interpolation filter*, $H_U(\omega)$ with frequency response

$$H_U(\omega) = \begin{cases} U, & |\omega| \leq \frac{\pi}{U} \\ 0, & |\omega| > \frac{\pi}{U} \end{cases}. \tag{13}$$

Notice how the interpolation filter has a similar structure to the anti-aliasing filter from lecture 29. Both filters are low-pass filters with cutoff frequencies determined by their rate-conversion factors, D or U. The key difference is that $H_U(\omega)$ has gain of U in its passband instead of 1 to ensure the original samples from $x[n]$ will not be rescaled after interpolation. Figures 2 and 3 show the full interpolator scheme in the frequency-domain and time-domain, respectively. Note how the interpolation filter "fills in the blanks" that the upsampling operation introduces in the time-domain. Finally, Fig. 4 summarizes the interpolator system that first upsamples by U then applies the appropriate interpolation filter.

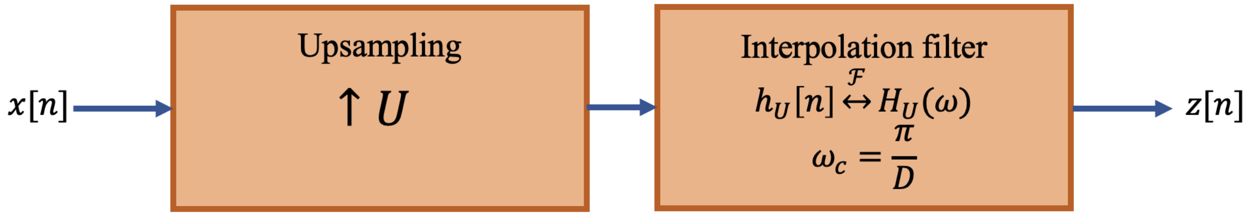


Figure 4: Overall interpolator scheme from upsampling to interpolation filter.

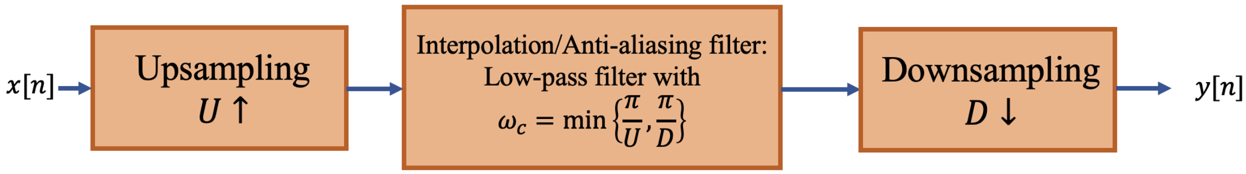


Figure 5: Cascaded rate conversion scheme for rate conversion by non-integer factor $R = \frac{U}{D}$.

2 Rate conversion by a non-integer factor

Thus far, we have considered how to increase or decrease the sampling rate of a discrete-time signal by an integer factor via upsampling and downsampling, respectively. What if we would like to change the sampling rate by non-integer factor?

We may achieve non-integer rate conversion by cascading interpolator and decimator systems. Concretely, if we would like to change the sampling rate by a factor of $R = \frac{U}{D}$, we can use an interpolator with upsampling factor U and a decimator with downsampling factor D. The remaining question is in which order should we cascade the two systems? Intuitively, you may want to put the interpolator first since we can condense the interpolation filter and anti-aliasing filter of the decimator into one filter. And this is correct! However, we should note that there is a more principled reason for doing this than just saving one digital filter.

To see why, let $U = 3$, $D = 5$ (rate conversion by $R = 3/5$) and consider if we downsample first: we will anti-alias filter then stretch the spectrum. The anti-aliasing filter may remove some frequency content from the input signal to avoid aliasing if the input spectrum has information outside $|\omega| = \frac{\pi}{5}$. Conversely, if we upsample first, the spectrum will first be compressed by a factor of three. Then the combined anti-aliasing/interpolation filter will still remove frequencies outside $|\omega| = \frac{\pi}{5}$. However, since we upsampled first, this cutoff of $\frac{\pi}{5}$ corresponds to $\omega = \frac{3\pi}{5}$ in the original spectrum. Thus, we can keep all frequency content up to $\frac{3\pi}{5}$ from the original spectrum in this scheme! This shows that we should always interpolate before decimating in such a cascaded system to minimize information loss due to anti-alias filtering.

In general, a non-integer rate conversion system by factor $R = \frac{U}{D}$ will proceed in the following three steps:

1. Upsample by factor of U (just inserting zeros).

2. Apply low-pass filter with cutoff frequency $\omega_c = \min\{\frac{\pi}{U}, \frac{\pi}{D}\}$.

3. Downsample by factor of D.

Figure 5 illustrates this cascaded rate conversion scheme.