ECE 438 - Laboratory 4
Sampling, Reconstruction, Interpolation and Decimation

Last updated on May 9, 2022

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import soundfile as sf
import IPython.display as ipd
from scipy import signal
```

In [2]:
```
# make sure the plot is displayed in this notebook
%matplotlib inline
# specify the size of the plot
plt.rcParams['figure.figsize'] = (16, 6)

# for auto-reloading extenrnal modules
%load_ext autoreload
%autoreload 2
```

1. Introduction

It is often desired to analyze and process continuous-time signals using a computer. However, in order to process a continuous-time signal, it must first be digitized. This means that the continuous-time signal must be sampled and quantized, forming a digital signal that can be stored in a computer. Analog systems can be converted to their discrete-time counterparts, and these digital systems then process discrete-time signals to produce discrete-time outputs. The digital output can then be converted back to an analog signal, or *reconstructed*, through a digital-to-analog converter. Figure 1 illustrates an example, containing the three general components described above: a sampling system, a digital signal processor, and a reconstruction system.

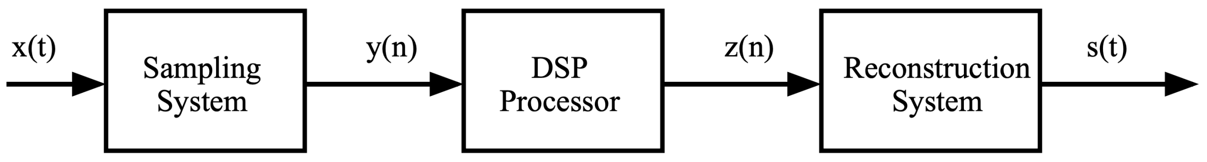

Figure 1: Example of a typical digital signal processing system

When designing such a system, it is essential to understand the effects of the sampling and reconstruction processes. Sampling and reconstruction may lead to different types of distortion, including low-pass filtering, aliasing, and quantization. The system designer must insure that these distortions are below acceptable levels, or are compensated through additional processing.

2. Sampling Overview

Sampling is simply the process of measuring the value of a continuous-time signal at certain instants of time. Typically, these measurements are uniformly separated by the sampling period, T_s. If $x(t)$ is the input signal, then the sampled signal, $y[n]$, is as follows:

$$y[n] = x(t)|_{t=nT_s}.$$

A critical question is the following: What sampling period, T_s, is required to accurately represent the signal $x(t)$? To answer this question, we need to look at the frequency domain representations of $y[n]$ and $x(t)$. Since $y[n]$ is a discrete-time signal, we represent its frequency content with the discrete-time Fourier transform (DTFT), $Y(e^{j\omega})$. However, $x(t)$ is a continuous-time signal, requiring the use of the continuous-time Fourier transform (CTFT), denoted as $X(f)$. Fortunately, $Y(\omega)$ can be written in terms of $X(f)$:

$$\begin{aligned} Y(\omega) &= \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X(f)|_{f=\frac{\omega-2\pi k}{2\pi T_s}} \\ &= \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X\left(\frac{\omega-2\pi k}{2\pi T_s}\right). \end{aligned} \quad (1)$$

Consistent with the properties of the DTFT, $Y(\omega)$ is periodic with a period 2π. It is formed by rescaling the amplitude and frequency of $X(f)$, and then repeating it in frequency every 2π. The critical issue of the relationship in (1) is the frequency content of $X(f)$. If $X(f)$ has frequency components that are above $1/(2T_s)$, the repetition in frequency will cause these components to overlap with (i.e. add to) the components below $1/(2T_s)$. This causes an unrecoverable distortion, known as *aliasing*, that will prevent a perfect reconstruction of $X(f)$. We will illustrate this later in the lab. The $1/(2T_s)$ "cutoff frequency" is known as the ***Nyquist frequency***.

To prevent aliasing, most sampling systems first low pass filter the incoming signal to ensure that its frequency content is below the Nyquist frequency. In this case, $Y(\omega)$ can be related to $X(f)$ through the $k = 0$ term in (1):

$$Y(\omega) = \frac{1}{T_s} X\left(\frac{\omega}{2\pi T_s}\right) \quad \text{for } \omega \in [-\pi, \pi]$$

Here, it is understood that $Y(e^{j\omega})$ is periodic with period 2π. Note in this expression that $Y(e^{j\omega})$ and $X(f)$ are related by a simple scaling of the frequency and magnitude axes. Also, note that $\omega = \pi$ in $Y(e^{j\omega})$ corresponds to the Nyquist frequency, $f = 1/(2T_s)$ in $X(f)$.

Sometimes after the sampled signal has been digitally processed, it must then converted back to an analog signal. Theoretically, this can be done by converting the discrete-time signal to a sequence of continuous-time impulses that are weighted by the sample values. If this continuous-time "impulse train" is filtered with an ideal low pass filter, with a cutoff frequency equal to the Nyquist frequency, a scaled version of the original low pass filtered signal will result. The spectrum of the reconstructed signal $S(f)$ is given by

$$S(f) = \begin{cases} Y(2\pi f T_s) & \text{for } |f| < \frac{1}{2T_s} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

3. Sampling and Reconstruction Using Sample-and-Hold

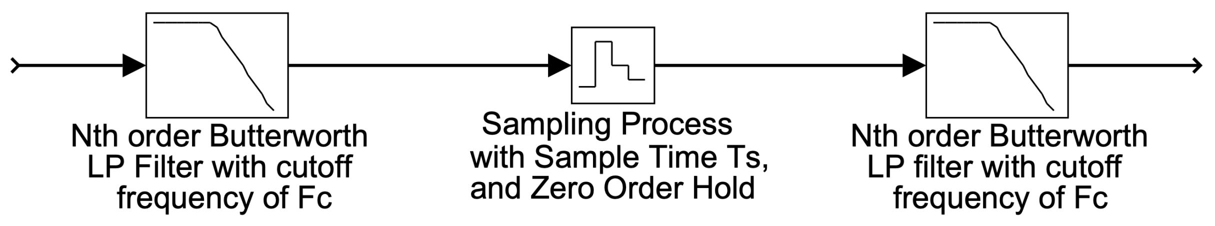
Nth order Butterworth LP Filter with cutoff frequency of Fc
Sampling Process with Sample Time Ts, and Zero Order Hold
Nth order Butterworth LP filter with cutoff frequency of Fc

Figure 2: Sampling and reconstruction using a sample-and-hold

In practice, signals are reconstructed using digital-to-analog converters. These devices work by reading the current sample, and generating a corresponding output voltage for a period of T_s seconds. The combined effect of sampling and D/A conversion may be thought of as a single sample-and-hold device. Unfortunately, the sample-and-hold process distorts the frequency spectrum of the reconstructed signal. In this section, we will analyze the effects of using a zero^{th}-order sample-and-hold in a sampling and reconstruction system. Later in the laboratory, we will see how the distortion introduced by a sample-and-hold process may be reduced through the use of discrete-time interpolation.

Figure 2 illustrates a system with a low-pass input filter, a sample-and-hold device, and a low-pass output filter. If there were no sampling, this system would simply be two analog filters in cascade. We know the frequency response for this simpler system. Any differences between this and the frequency response for the entire system is a result of the sampling and reconstruction. Our goal is to compare the two frequency responses using Python. For this analysis, we will assume that the filters are N^{th} order Butterworth filters with a cutoff frequency of f_c , and that the sample-and-hold runs at a sampling rate of $f_s = 1/T_s$.

We will start the analysis by first examining the ideal case. Consider replacing the sample-and-hold with an ideal impulse generator, and assume that instead of the Butterworth filters we use perfect low-pass filters with a cutoff of f_c . After analyzing this case we will modify the results to account for the sample-and-hold and Butterworth filter roll-off.

If an ideal impulse generator is used in place of the sample-and-hold, then the frequency spectrum of the impulse train can be computed by combining the sampling equation (1) with the reconstruction equation (2).

$$\begin{aligned} S(f) &= Y(2\pi f T_s) \\ &= \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X\left(\frac{2\pi f T_s - 2\pi k}{2\pi T_s}\right) \\ &= \frac{1}{T_s}\sum_{k=-\infty}^{\infty} X(f - kf_s), \quad \text{for } |f| \le \frac{1}{2T_s} \\ S(f) &= 0 \quad \text{for } |f| > \frac{1}{2T_s}. \end{aligned}$$

If we assume that $f_s > 2f_c$, then the infinite sum reduces to one term. In this case, the reconstructed signal is given by

$$S(f) = \frac{1}{T_s}X(f) \tag{3}$$

Notice that the reconstructed signal is scaled by the factor $\frac{1}{T_s}$.

Of course, the sample-and-hold does not generate perfect impulses. Instead it generates a pulse of width T_s, and magnitude equal to the input sample. Therefore, the new signal out of the sample-and-hold is equivalent to the old signal (an impulse train) convolved with the pulse

$$p(t) = \text{rect}\left(\frac{t}{T_s} - \frac{1}{2}\right).$$

Convolution in the time domain is equivalent to multiplication in the frequency domain, so this convolution with $p(t)$ is equivalent to multiplying by the Fourier transform $P(f)$ where

$$|P(f)| = T_s|\text{sinc}(f/f_s)|. \tag{4}$$

Finally, the magnitude of the frequency response of the N-th order Butterworth filter is given by

$$|H_b(f)| = \frac{1}{1 + \left(\frac{f}{f_c}\right)^N}. \tag{5}$$

We may calculate the complete magnitude response of the sample-and-hold system by combining the effects of the Butterworth filters in equation (5), the ideal sampling system in equation (3), and the sample-and-hold pulse width in equation (4). This yields the final expression

$$\begin{aligned} |H(f)| &= |H_b(f)P(f)\frac{1}{T_s}H_b(f)| \\ &= \left(\frac{1}{1 + \left(\frac{f}{f_c}\right)^N}\right)^2 |\text{sinc}(f/f_s)|. \end{aligned}$$

Notice that the expression $|\text{sinc}(f/f_s)|$ produces a roll-off in frequency which will attenuate frequencies close to the Nyquist rate. Generally, this roll-off is not desirable.

Exercise 3.1

Let $T_s = 1$ sec, $f_c = 0.45$ Hz, and $N = 20$.

1. Compute and plot the magnitude response of the system in Figure 2 without the sample-and-hold device. Let the frequencies be `f = np.linspace(-1, 1, 2001)` .

In [3]:
```
# write your code here
```

2. Compute and plot the magnitude response of the complete system in Figure 2. Let the frequencies be `f = np.linspace(-1, 1, 2001)`.

In [4]:
```
# write your code here
```

3. Comment on the shape of the two magnitude responses. How might the magnitude response of the sample-and-hold affect the design considerations of a high quality audio CD player?

write your answer here

4. Simulink Overview

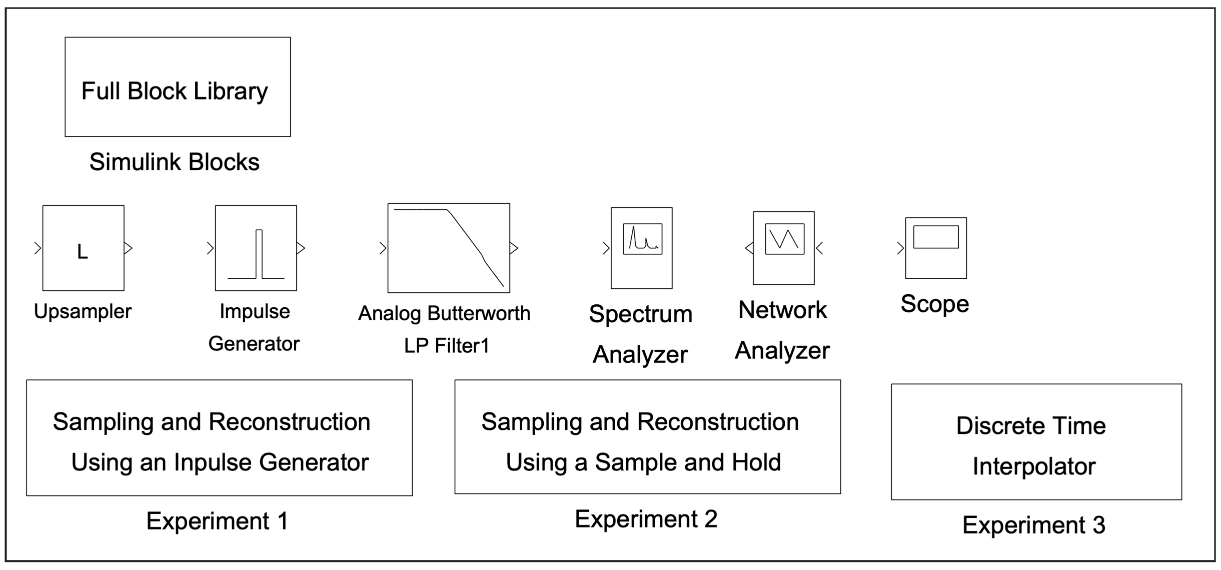
Full Block Library
Simulink Blocks
L
Upsampler
Impulse Generator
Analog Butterworth LP Filter1
Spectrum Analyzer
Network Analyzer
Scope
Sampling and Reconstruction Using an Inpulse Generator
Experiment 1
Sampling and Reconstruction Using a Sample and Hold
Experiment 2
Discrete Time Interpolator
Experiment 3

Figure 3: Simulink utilities for lab 4

In this lab we will use Simulink to simulate the effects of the sampling and reconstruction processes. Simulink treats all signals as continuous-time signals. This means that "sampled" signals are really just continuous-time signals that contain a series of finite-width pulses. The height of each of these pulses is the amplitude of the input signal at the beginning of the pulse. In other words, both the sampling action and the zero-order-hold reconstruction are done at the same time; the discrete-time signal itself is never generated. This means that the impulse-generator block is really a "pulse-generator", or zero-order-hold device. Remember that, in Simulink, frequency spectra are computed on continuous-time signals. This is why many aliased components will appear in the spectra.

5. Sampling and Reconstruction with an Impulse Generator

In this section, we will experiment with the sampling and reconstruction of signals using a pulse generator. This pulse generator is the combination of an ideal impulse generator and a perfect zero-order-hold device. In order to run the experiment, open Matlab, change current directory to `Lab4Utilities`, and then type `Lab4`. A set of Simulink blocks and experiments will come up as shown in Fig 3.

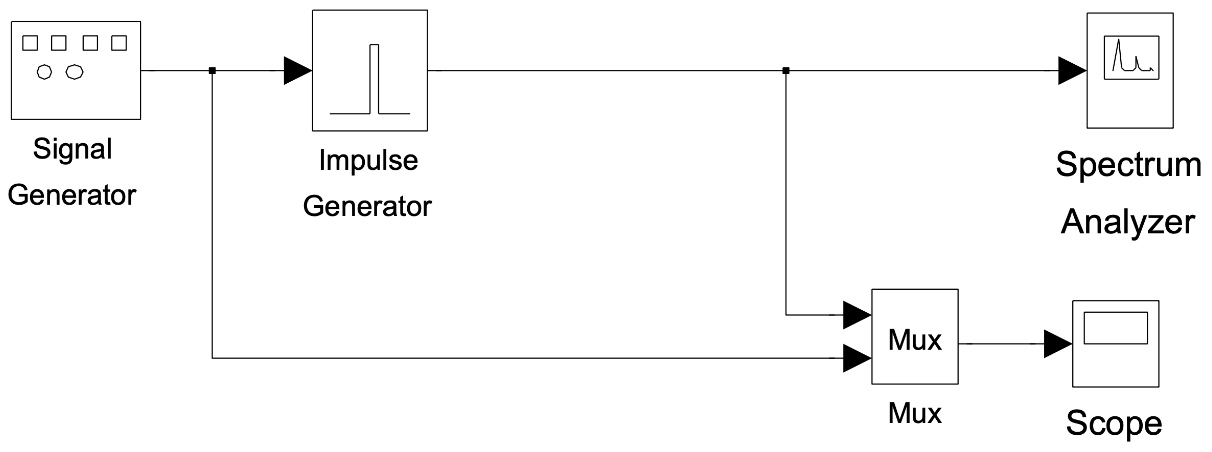

Figure 4: Simulink model for sampling and recosntruction using an impulse generator

Before starting this experiment, use the MATLAB command `close all` to close all figures other than the Simulink windows. Double click on the icon named ***Sampling and Reconstruction Using An Impulse Generator*** to bring up the first experiment as shown in Figure 4. In this experiment, a sine wave is sampled at a frequency of 1 Hz; then the sampled discretetime signal is used to generate rectangular impulses of duration 0.3 sec and amplitude equal to the sample values. The block named ***Impulse Generator*** carries out both the sampling of the sine wave and its reconstruction with pulses. A single ***Scope*** is used to plot both the input and output of the impulse generator, and a ***Spectrum Analyzer*** is used to plot the output pulse train and its spectrum.

Exercise 5.1

First, run the simulation with the frequency of input sine wave set to 0.1 Hz (initial setting of the experiment). Let the simulation run until it terminates to get an accurate plot of the output frequencies. Then print the output of ***Scope*** and the ***Spectrum Analyzer***. Be sure to label your plots.

1.Submit the plot of the input/output signals and the plot of the output signal and its frequency spectrum. On the plot of the spectrum of the reconstructed signal, circle the aliases, i.e. the components that do NOT correspond to the input sine wave.

insert your plots here

Ideal impulse functions can only be approximated. In the initial setup, the pulse width is 0.3 sec, which is less then the sampling period of 1 sec. Try setting the pulse width to 0.1 sec and run the simulation. Print the output of the ***Spectrum Analyzer***.

2.Submit the plot of the output frequency spectrum for a pulse width of 0.1 sec. Indicate on your plot what has changed and explain why.

Insert your plots here

Set the pulse width back to 0.3 sec and change the frequency of the sine wave to 0.8 Hz. Run the simulation and print the output of the ***Scope*** and the ***Spectrum Analyzer***.

3.Submit the plot of the input/output signals and the plot of the output signal and its frequency spectrum. On the frequency plot, label the frequency peak that corresponds to the lowest frequency (the fundamental component) of the output signal. Explain why the lowest frequency is no longer the same as the frequency of the input sinusoid.

Insert your plots here

Leave the input frequency at 0.8 Hz. Now insert a filter right after the impulse generator. Use a 10th order Butterworth filter with a cutoff frequency of 0.5 Hz. Connect the output of the filter to the ***Spectrum Analyzer*** and the ***Mux***. Run the simulation, and print the output of ***Scope*** and the ***Spectrum Analyzer***.

4.Submit the plot of the input/output signals and the plot of the output signal and its frequency spectrum. Explain why the output signal has the observed frequency spectrum.

Insert your plots here

6. Sampling and Reconstruction with Sample and Hold

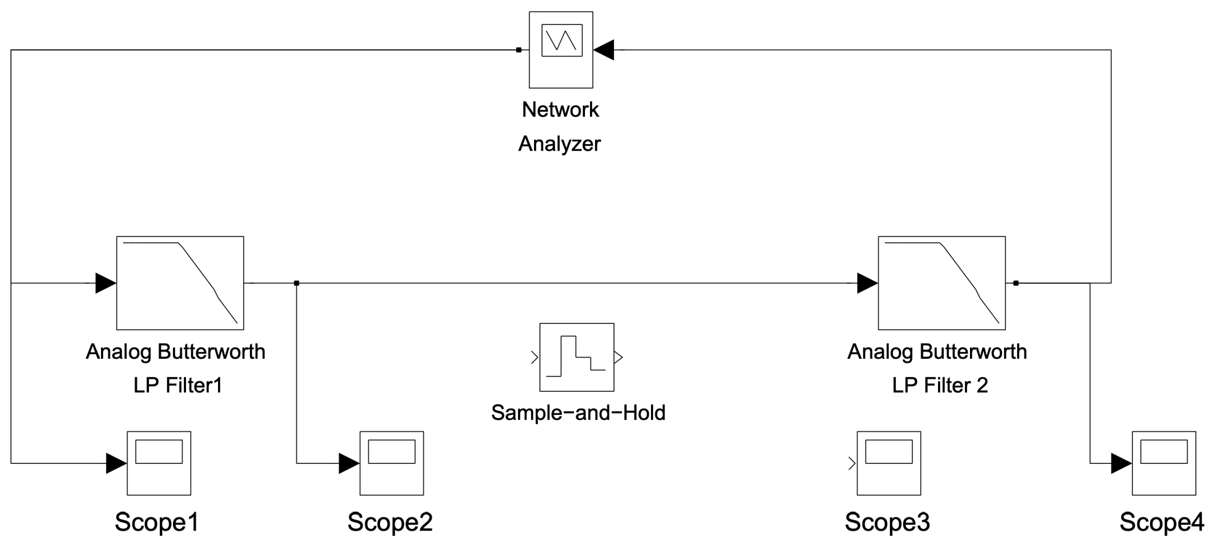

Figure 5: Initial Simulink model for sampling and reconstruction using a sample-and-hold. This system only measures the frequency response of the analog filters.

In this section, we will sample a continuous-time signal using a sample-and-hold and then reconstruct it. We already know that a sample-and-hold followed by a low-pass filter does not result in perfect reconstruction. This is because a sample-and-hold acts like a pulse generator with a pulse duration of one sampling period. This “pulse shape” of the sample-and-hold is what distorts the frequency spectrum (see Sec 1.2).

To start the second experiment, double click on the icon named ***Sampling and Reconstruction Using A Sample and Hold***. Figure 5 shows the initial setup for this exercise. It contains 4 Scopes to monitor the processing done in the sampling and reconstruction system. It also contains a ***Network Analyzer*** for measuring the frequency response and the impulse response of the system.

The ***Network Analyzer*** works by generating a weighted chirp signal (shown on Scope 1) as an input to the system-under-test. The frequency spectrum of this chirp signal is known. The analyzer then measures the frequency content of the output signal (shown on Scope 4). The transfer function is formed by computing the ratio of the output frequency spectrum to the input spectrum. The inverse Fourier transform of this ratio, which is the impulse response of the system, is then computed.

In the initial setup, the ***Sample-and-Hold*** and ***Scope 3*** are not connected. There is no sampling in this system, just two cascaded low-pass filters. Run the simulation and observe the signals on the ***Scopes***. Wait for the simulation to end.

Exercise 6.1

1.Submit the figure containing plots of the magnitude response, the phase response, and the impulse response of this system. Use the tall mode to obtain a larger printout by typing `orient('tall')` directly before you print.

Insert your plots here

Double-click the ***Sample-and-Hold*** and set its ***Sample time*** to 1. Now, insert the ***Sample-and-Hold*** in between the two filters and connect ***Scope 3*** to its output. Run the simulation and observe the signals on the ***Scopes***.

2.Submit the figure containing plots of the magnitude response, the phase response, and the impulse response of this system. Explain the reason for the difference in the shape of this magnitude response versus the previous magnitude response. Give an analytical expression for the behavior of the magnitude plot for frequencies below 0.45 Hz.

Insert your plots here

7. Discrete-Time Interpolation

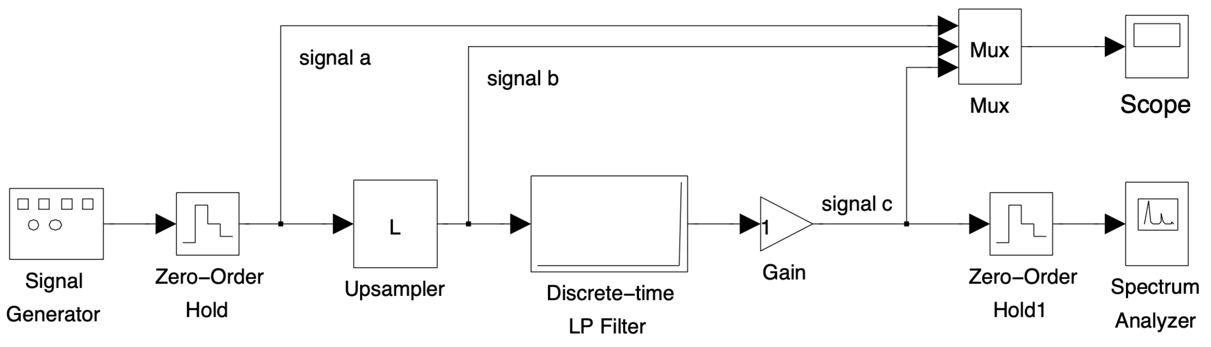

Figure 5: Simulink model for discrete-time interpolation.

In the previous experiments, we saw that the frequency content of a signal must be limited to half the sampling rate in order to avoid aliasing effects in the reconstructed signal. However, reconstruction can be difficult if the sampling rate is chosen to be just above the Nyquist frequency. Reconstruction is much easier for a higher sampling rate because the sampled signal will better “track” the original analog signal.

From another perspective, the analog output filter must have a very sharp cutoff in order to accurately reconstruct a signal that was sampled just above the Nyquist rate. Such filters are difficult and expensive to manufacture. Alternatively, a higher sampling rate allows the use analog output filters that have a slow roll-off. These filters are much less expensive. However, a high sampling rate is not practical in most applications, as it resultsin unnecessary samples and excessive storage requirements.

A practical solution to this dilemma is to *interpolate* the digital signal to create new (artificial) samples between the existing samples. This may be done by first upsampling the digital representation, and then filtering out unwanted components using a discrete-time filter. This discrete-time filter serves the same purpose as an analog filter with a sharp cutoff, but it is generally simpler and more cost effective to implement.

Upsampling a signal by a factor of L is simply the process of inserting $L-1$ zeros in between each sample. The frequency domain relationship between a signal $x(n)$ and its upsampled version $z(n)$ can be shown to be the following

$$Z(\omega) = X(\omega L). \tag{6}$$

Therefore the DTFT of $z(n)$ is simply $X(\omega)$ compressed in frequency by a factor of L. Since $X(\omega)$ has a period of 2π, $Z(\omega)$ will have a period of $2\pi/L$. All of the original information of $x(n)$ will be contained in the interval $[-\pi/L, \pi/L]$ of $Z(\omega)$, and the new aliases that are created in the interval $[-\pi, \pi]$ are the unwanted components that need to be filtered out. In the time domain, this filtering has the effect of changing the inserted zeros into artificial samples of $x(n)$, commonly known as *interpolated* samples.

Figure 5 shows a Simulink model that demonstrates discrete-time interpolation. The interpolating system contains three main components: an upsampler which inserts $L-1$ zeros between each input sample, a discrete-time low pass filter which removes aliased signal components in the interpolated signal, and a gain block to correct the magnitude of the final signal. Notice that **signal a** is the input discrete-time signal while **signal c** is the final interpolated discrete-time signal.

Open the experiment by double clicking on the icon labeled ***Discrete Time Interpolator***. The components of the system are initially set to interpolate by a factor of 1. This means that the input and output signals will be the same except for a delay. Run this model with the initial settings, and observe the signals on the ***Scope***.

Simulink represents any discrete-time signal by holding each sample value over a certain time period. This representation is equivalent to a sample-and-hold reconstruction of the underlying discrete-time signal. Therefore, a continuous-time ***Spectrum Analyzer*** may be used to view the frequency content of the output **signal c**. The ***Zero-Order Hold*** at the **Gain** output is required as a buffer for the ***Spectrum Analyzer*** in order to set its internal sampling period.

The lowest frequency component in the spectrum corresponds to the frequency content of the original input signal, while the higher frequencies are aliased components resulting from the sample-and-hold reconstruction. Notice that the aliased components of **signal c** appear at multiples of the sampling frequency of 1 Hz. Print the output of the ***Spectrum Analyzer***.

Exercise 7.1

1.Submit your plot of signal c and its frequency spectrum. Circle the aliased components in your plot.

insert your answer here

Next modify the system to upsample by a factor of 4 by setting this parameter in the ***Upsampler***. You will also need to set the ***Sample time*** of the DT filter to 0.25. This effectively increases the sampling frequency of the system to 4 Hz. Run the simulation again and observe the behavior of the system. Notice that zeros have been inserted between samples of the input signal. After you get an accurate plot of the output frequency spectrum, print the output of the ***Spectrum Analyzer***.

Notice the new aliased components generated by the upsampler. Some of these spectral components lie between the frequency of the original signal and the new sampling frequency, 4 Hz. These aliases are due to the zeros that are inserted by the upsampler.

2.Submit your plot of signal c and its frequency spectrum. On your frequency plot, circle the first aliased component and label the value of its center frequency. Comment on the shape of the envelope of the spectrum.

insert your answer here

Notice in the previous ***Scope*** output that the process of upsampling causes a decrease in the energy of the sample-and-hold representation by a factor of 4. This is the reason for using the ***Gain*** block.

Now determine the gain factor of the ***Gain*** block and the cutoff frequency of the ***Discrete-time LP filter*** needed to produce the desired interpolated signal. Run the simulation and observe the behavior of the system. After you get an accurate plot of the output frequency spectrum, print the output of the ***Spectrum Analyzer***. Identify the change in the location of the aliased components in the output signal.

3.Submit your plot of signal c and its frequency spectrum. Give the values of the cutoff frequency and gain that were used. On your frequency plot, circle the location of the first aliased component. Explain why discrete-time interpolation is desirable before reconstructing a sampled signal

insert your answer here

---

8. Discrete-Time Decimation

In the previous section, we used interpolation to increase the sampling rate of a discretetime signal. However, we often have the opposite problem in which the desired sampling rate is lower than the sampling rate of the available data. In this case, we must use a process called *decimation* to reduce the sampling rate of the signal.

Decimating, or *downsampling*, a signal $x[n]$ by a factor of D is the process of creating a new signal $y[n]$ by taking only every Dth sample of $x[n]$. Therefore $y[n]$ is simply $x[Dn]$. The frequency domain relationship between $y[n]$ and $x[n]$ can be shown to be the following:

$$Y(\omega) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(\frac{\omega - 2\pi k}{D}\right). \qquad (7)$$

Notice the similarity of (7) to the sampling theorem equation in (1). This similarity should be expected because decimation is the process of sampling a discrete-time signal. In this case, $Y(\omega)$ is formed by taking $X(\omega)$ in the interval $[-\pi, \pi]$ and expanding it in frequency by a factor of D. Then it is repeated in frequency every 2π, and scaled in amplitude by $1/D$. For similar reasons as described for equation (1), aliasing will be prevented if in the interval $[-\pi, \pi]$, $X(e^{j\omega})$ is zero outside the interval $[-\pi/D, \pi/D]$. Then (7) simplifies to

$$Y(\omega) = \frac{1}{D} X\left(\frac{\omega}{D}\right) \quad \text{for } \omega \in [-\pi, \pi]. \qquad (8)$$

A system for decimating a signal is shown in Figure 6. The signal is first filtered using a low pass filter with a cutoff frequency of $\pi/2$ rad/sample. This insures that the signal is band limited so that the relationship in (8) holds. The output of the filter is then subsampled by removing every other sample.

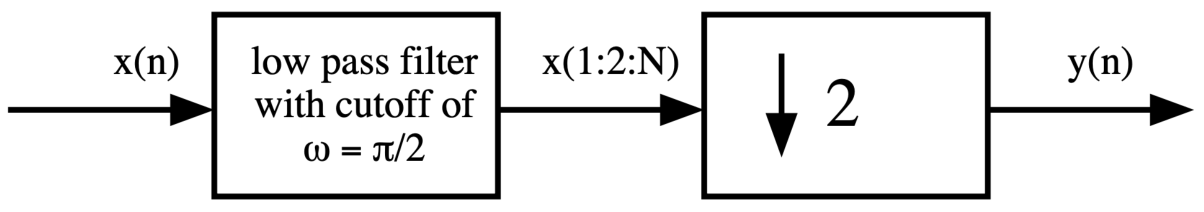

Figure 6: This system decimates a discrete-time signal by a factor of 2

Exercise 8.1

1. Read in the signal contained in `music.au` using `music, fs = sf.read("music.au")` , and then play it back with sound using `ipd.Audio(music, rate=fs)` .

```
In [5]: # insert your code here
```

The signal contained in `music.au` was sampled at 16 kHz, so it will sound much too slow when played back at the default 8 kHz sampling rate.

2. To correct the sampling rate of the signal, form a new signal, `sig1` , by selecting every other sample of the music vector. Play the new signal, and listen carefully to the new signal.

```
In [6]: # insert your code here
```

---

3. Compute a second subsampled signal, `sig2` , by first low pass filtering the original music vector using a discrete-time filter of length

20, and with a cutoff frequency of $\pi/2$. Then decimate the filtered signal by 2, and listen carefully to the new signal.

Hint: You can create a low-pass filter by using the function `signal.firwin` (https://docs.scipy.org/doc/scipy/reference/reference/generated/scipy.signal.firwin.html) from `scipy` and use the following syntax:

```
signal.firwin(M, W/np.pi)
```

which return a length `M` low-pass filter with cutoff frequency `W` rad/sample.

Use `np.convolve` (https://numpy.org/doc/stable/reference/generated/numpy.convolve.html) to get the convolution of two signals.

In [7]:
```
# insert your code here
```

---

4. Comment on the quality of the audio signal generated by using the two decimation methods. Was there any noticeable distortion in sig1? If so, describe the distortion.

insert your answer here