"I said the hounds of Spring are
on Winter's traces—but let it pass, let it pass!"

EE 311 **April 12, 2019**
Lecture 22

Chapter 7
Sample Rate Conversion

Intro
Down sampling
Up sampling

7.1 Integer Decimation

• Integer down-sampling can be done by simply removing samples. So for example, to down-sample by a factor 3, we would remove two samples out of every three as shown in Figure 7.1.

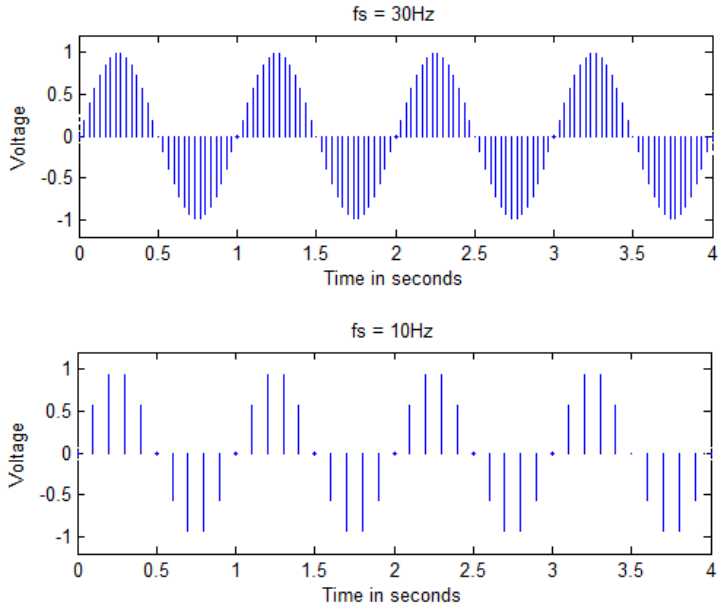


Figure 7.1
Down-sampling by a factor of 3 is done by removing two samples out of every three.

• If the original sample frequency is f_s and we down-sample for a factor D then the anti-aliasing filter must eliminate those frequencies above $f_s/(2D)$. The ideal anti-aliasing filter will have the following characteristics:

$$H_{AA}(f)=\begin{cases}1 & 0\le f\le f_s/(2D)\\ 0 & \text{otherwise}\end{cases}$$ where D is the decimation factor.

Example 7.1
Use MATLAB® to create a chirp signal that ranges in frequency from 10 Hz to 2000 Hz and is sampled at 11025 Hz. Down-sample this signal by a factor of 2 and a factor of 4 without filtering to show the effects of aliasing.

Solution

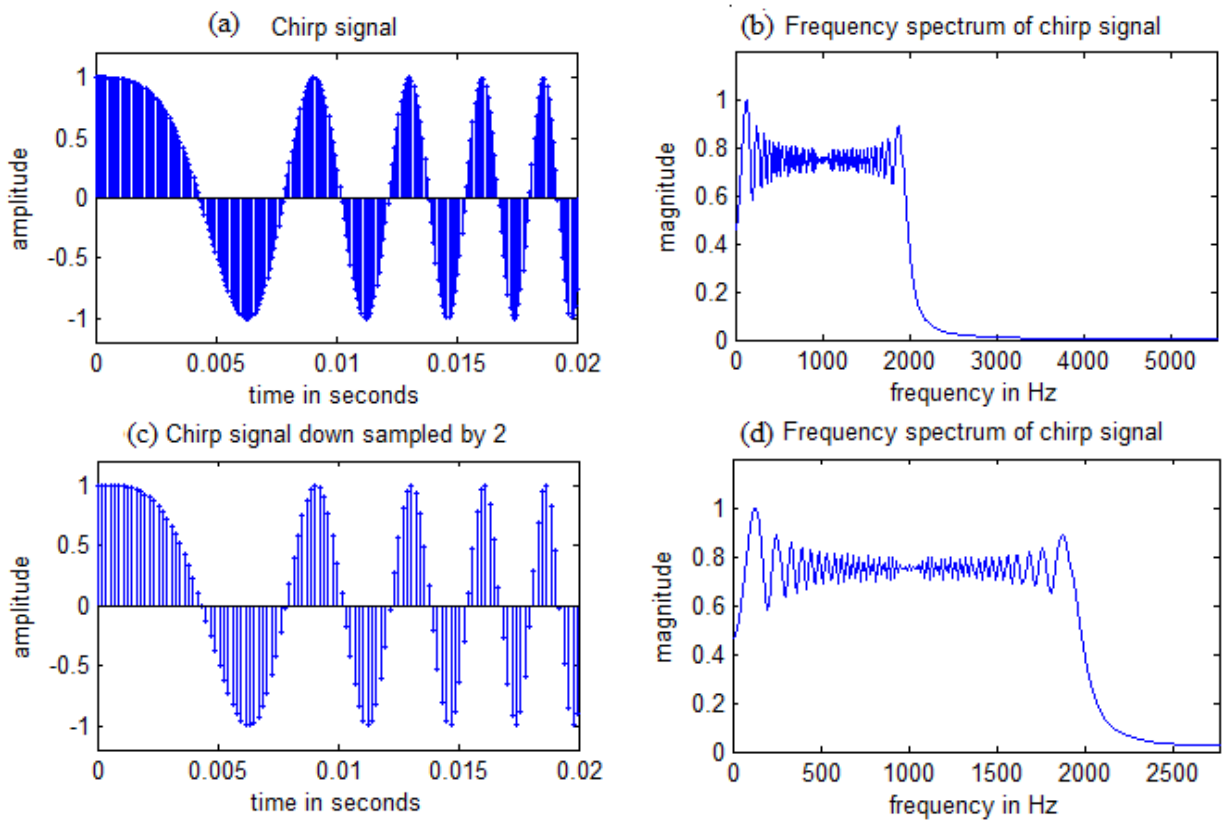
The original signal can be generated by the following lines in MATLAB®.

```
N = 1000;
fs = 11025;T = 1/fs;
t = 0:T:(N-1)*T;
y = chirp(t, 10, (N-1)*T, 2000);
subplot(1, 2, 1);
stem(t, y, 'MarkerSize', 1);
title('Chirp signal');
subplot(1, 2, 2);
yDFT = fft(y);
f = 0:fs/N:fs-fs/N;
yDFTNorm = abs(yDFT)/max(abs(yDFT));
plot(f, yDFTNorm);
```

In MATLAB® we can use the down-sample function to reduce the sample frequency without filtering. The syntax is

```
yDn = downsample(y, 2);  %Down-sample by a factor of 2.
```

The results are shown in Figure 7.2.



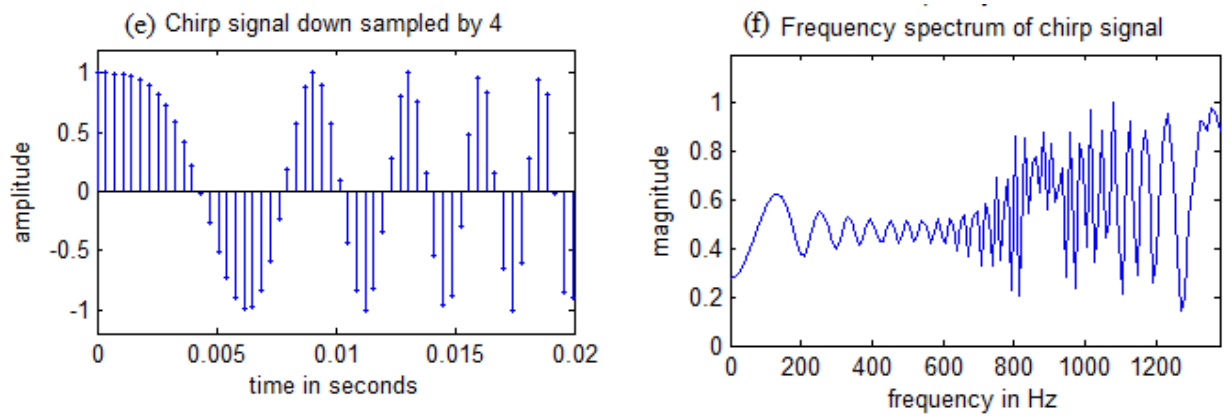


Figure 7.2
(a) is the original chirp signal which is sampled at 11025 Hz. (b) is the frequency spectrum of the signal. Figures (c) and (d) show the same signal down-sampled by a factor of two. There is little aliasing since most of the frequency spectrum is below the original $f_s/2$. In (d)) and (e) the signal has been down-sampled by a factor of 4. The frequency spectrum shows the clear effects of aliasing.

Frequency spectrum of the down sampled signal

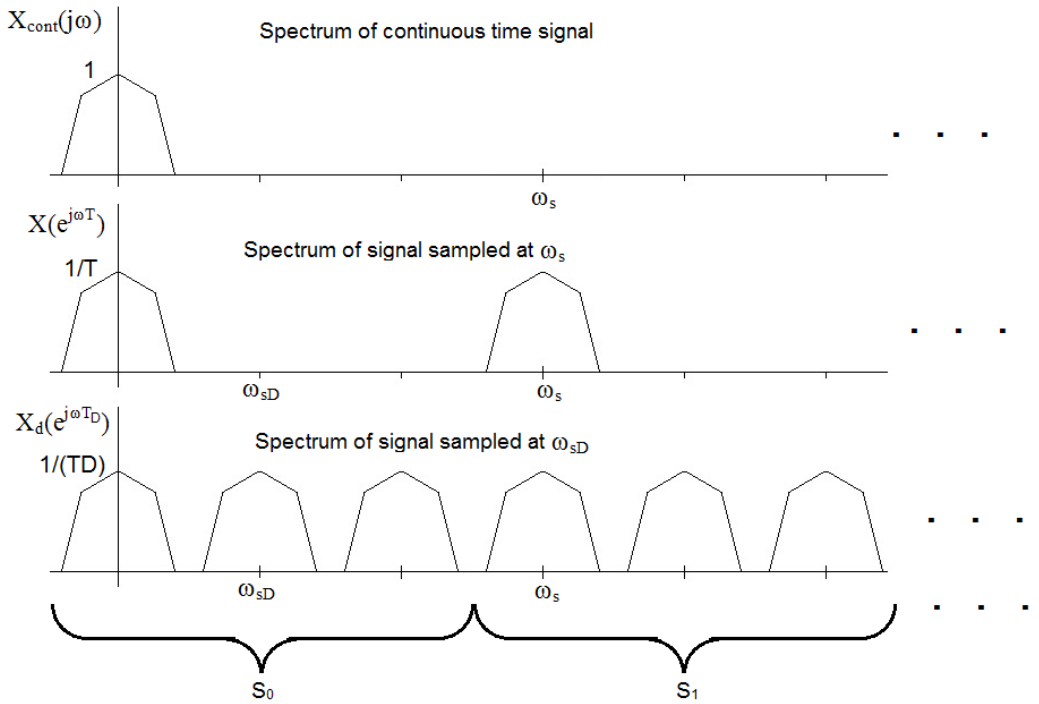


Figure 7.3
Frequency spectrum of (top) continuous signal, (middle) sample signal, and (bottom) down sampled signal

Cascaded Decimation

In many practical applications it is necessary to do decimation by a large factor, say greater than 10. In such cases it becomes practical to implement the decimation in stages since cascading stages can be computationally more efficient than doing the decimation in a single stage. A cascaded decimator with *N* stages looks like that shown in Figure 7.4.

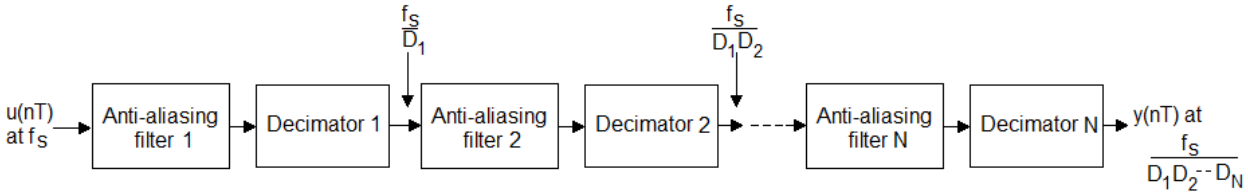


Figure 7.4
A cascaded decimator structure of *N* stages. Each stage decimates the signal by an integer factor and each stage has its own anti-aliasing filter.

Example 7.2
Consider a signal which has a pass band of 0 to 100 Hz and a transition band from 100 Hz to 150 Hz. The sample frequency is 20,000 Hz and we wish to reduce this sample frequency to just 1000 Hz, or a factor of D = 20. We want to design an anti-aliasing filter for a decimator with a pass band ripple of $r_p = 0.01$ and a stop band ripple of $r_s = 0.001$.

Solution
We can use MATLAB® to calculate the order for a filter to do this in one stage. The code below shows the order for a Parks-McClelland filter to be 127.

```
fs = 20000;
fpass = 100;
fstop = 500;
rpass = 0.01;
rstop = 0.001;
F = ([fpass fstop]);
M = ([1 0]);
Dev = [rpass rstop];
[N F A W] = firpmord(F, M, Dev, fs);
disp(N);
```

Consider solving this same problem with a two stage decimator. We have some choices as to what the two stages will be: 2-10, 10-2, 4-5, or 5-4. For purposes of illustration we choose the two stage 5-4 arrangement shown in Figure 7.5.

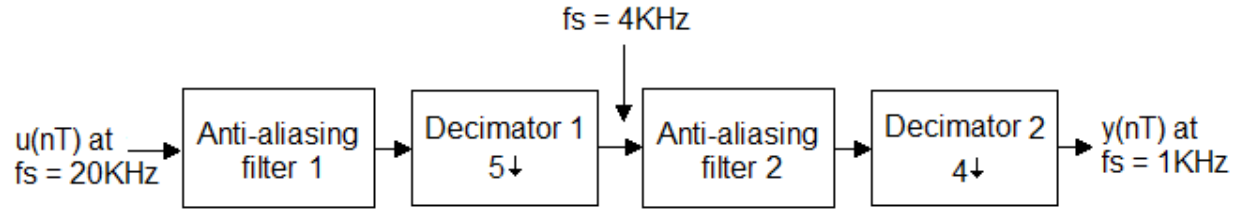


Figure 7.5
Two stage cascaded decimators. The first cuts the sample frequency by a factor of 5 from 20 KHz to 4 KHz. The second cuts the 4 KHz sample frequency by a factor of 4 to 1 KHz.

The first anti-aliasing filter receives a signal at a 20 KHz sample frequency and outputs a signal at a 4 KHz sample frequency. It must filter out those frequencies that can be aliased back into

the pass band. This includes those frequencies above 2 KHz. For the first anti-aliasing filter we have f_{pass} = 0 to 100 Hz and f_{stop} = 2 KHz to $f_s/2$.

Unfortunately, the pass band ripple gain of the two anti-aliasing filters will multiply so we need to reduce the ripple of each of these filters so that the total ripple figure is maintained. The pass band gain for the cascaded anti-aliasing filters will be $(1 - r_{p1})^2 = 1 - r_p$. This gives:
$1 - 2r_{p1} + r_{p1}^2 = 1 - r_p$.
Since the ripple is very small we neglect the squared term to get
$r_{p1} = r_p/2$

For this case the stop band ripple need not be changed. Using these specifications and the MATLAB® code above we determine that a Parks-McClelland filter of order 38 is needed.

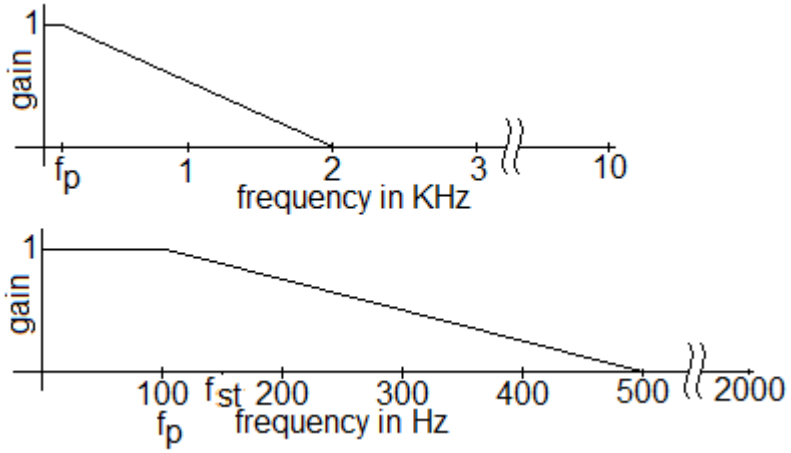


Figure 7.6
The magnitude plots for the two anti-aliasing filters. For the top figure the pass band remains at 100 Hz but the stop band need not start until f_{st1} = 2 KHz .

The second decimation filter takes in a signal sampled at 4 KHz and outputs the signal sampled at 1 KHz. The specifications for the second filter are: pass band from 0 to 100 Hz, stop band from 500 Hz to $f_s/2$. Pass band ripple = 0.005 and stop band ripple = 0.001. This gives an FIR Parks-McClelland filter of order 27. The magnitude plots for the two anti-aliasing filters are shown in Figure 7.6.

7.2 Integer Interpolation

Interpolation is the counterpart to decimation and is used to *increase* the sampling rate by some integer value. The term *interpolation* generally conjures images of drawing a straight line between two points and calculating intermediate locations along that line. But an interpolation filter in DSP is a filter which inserts zeros between sample points to increase the sample rate and passes the result through a low pass filter to eliminate the image that is produced, leaving the original signal sampled at a higher rate. The technique is counterintuitive, so consider what happens to a signal in the frequency domain if we insert zeros between all of the sample points.

In Figure 7.7, a 1 Hz sinusoid that is sampled at 10 Hz is up-sampled to 20 Hz by inserting zeros between the samples. The sampled sinusoid is shown in both the time and frequency domain. In the frequency domain we have shown the image frequencies which appear between $f_s/2$ and f_s. Figure 7.7 (a) is the original 1 Hz sinusoid. Figure 7.7 (b) shows the frequency domain if the sinusoid is sampled at 10 Hz. The 1 Hz sinusoid has an image at 9 Hz. In Figure 7.7 (c) Zeros have been inserted between the sample points the effectively doubling the sample frequency. Figure 7.7 (d) shows the frequency response of the up-sampled signal. The response shows the original signal plus its image at 9 Hz.

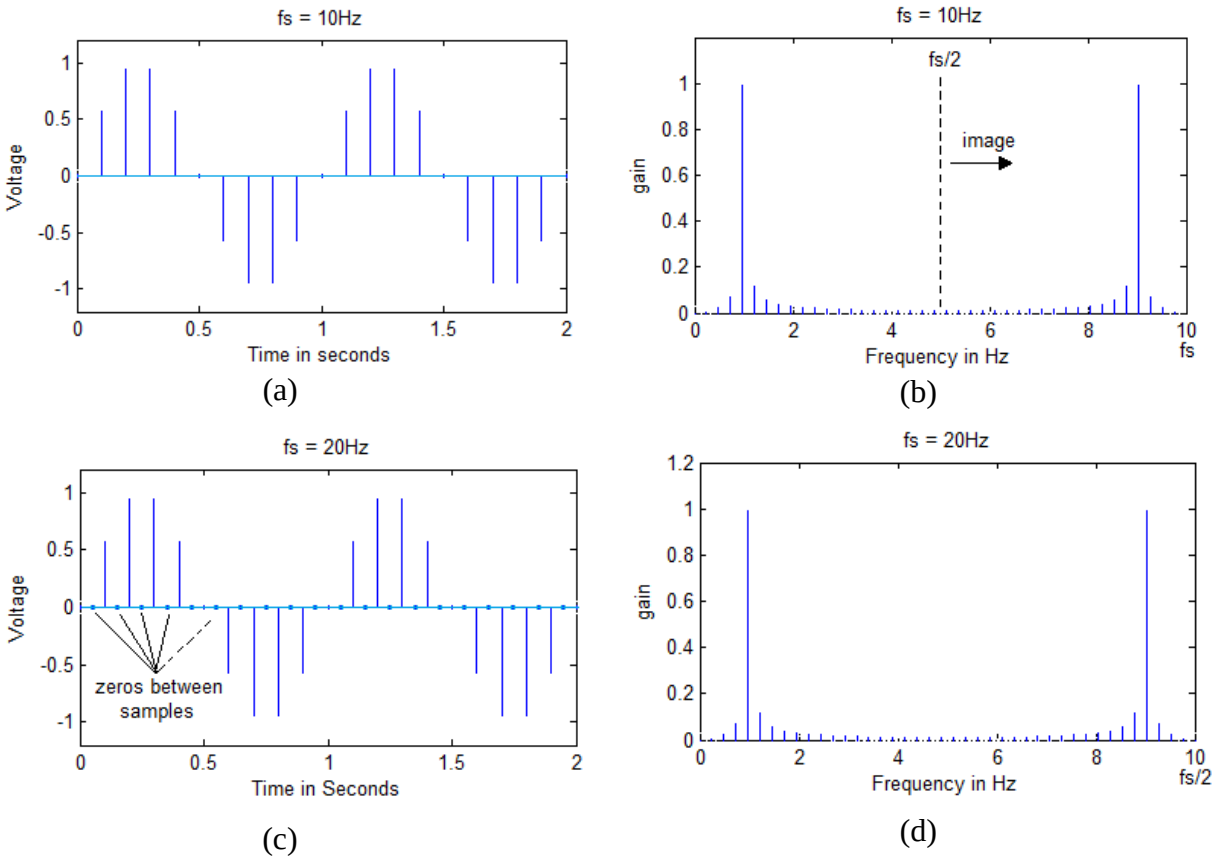


Figure 7.7
(a) A 1 Hz sinusoid sampled at 0.1 seconds. (b) The same sinusoid in the frequency domain. The sinusoid includes a frequency image after fs/2. (c) The sinusoid from part A has zeros inserted between the sample points effectively doubling the sample period to 0.05 seconds. (d) The effect of adding the zeros is to double the sample rate and capture the original sinusoids image at 9 Hz.

The effect of inserting zeros between samples in Figure 7.7 (c) and getting the frequency response of Figure 7.7 (d) is not intuitive. But Figure 7.8 shows that if we begin by adding a 1 Hz and a 9 Hz sinusoid and sampling the result at 20 Hz we do indeed get the same time-domain signal as shown in Figure 7.7 (c).

```
fsig = 1;              %signal
fs = 20;T = 1/fs;
n = 0:4*fsig/T;
u = sin(2*pi*fsig*n*T);
u1 = sin(2*pi*fsig*9*n*T);
subplot(2, 1, 1);
 t = 0:.05:4;
 plot(t, u);
 hold on;
 plot(t, u1, 'r');
 xlabel('Time in seconds');
 ylabel('voltage');
 title('Sinusoids at 1 Hz and 9
Hz');
subplot(2, 1, 2);
 plot(t, u+u1);
 xlabel('Time in seconds');
 ylabel('voltage');
 title('Sinusoids added');
```

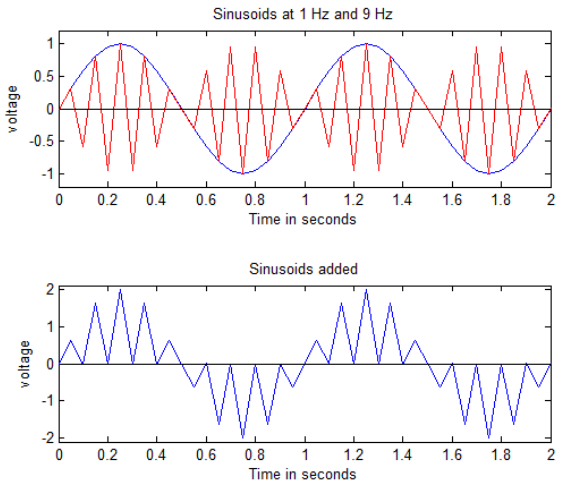


Figure 7.8
Top shows a 1 Hz and a 9 Hz sinusoid and bottom is their sum. If the bottom sum is sampled at 0.05 Hz we get sample points identical to those of a 1Hz sinusoid sampled at 0.1 Hz with zeros inserted between samples.

Thus, interpolation consists of inserting zeroes between the samples of a signal and passing the result through a low pass filter to eliminate the image frequencies. This is much the same process and an anti-aliasing filter. The filter used for this purpose is called an *interpolation filter*.

The ideal interpolation filter will have the following characteristics:

$$H_{Ideal}(f) = \begin{cases} U & 0 \le f < f_s/(2U) \\ 0 & f_s/(2U) \le f \le f_s/2 \end{cases}$$

where f_s is the fast sample rate and U is the up-sample factor. The ideal interpolator has a gain of U to compensate for the drop in the average value of the signal by the insertion of U zeros between samples.

Example 7.3
Suppose you have two identical signals which are limited to a small band of frequencies around 100 Hz. The first signal is sampled at 1 KHz and the second is sampled at 1.5 KHz. If we up-sample both signals to 3 KHz what will be the difference in the frequency characteristics of the two.

Solution
The first signal needs to be up-sampled by a factor of 3 so there will be three images in the range 0 to 3 KHz. The second signal needs to be up-sampled by a factor of 2 so there will only be 2 images in the range 0 to 3 KHz. This is illustrated in Figure 7.10. The original signal at 100 Hz will be the same after filtering but the signal that was up-sampled by a factor of 2 will have less stringent requirements for the interpolation filter than will the signal that was up-sampled by a factor of 3.

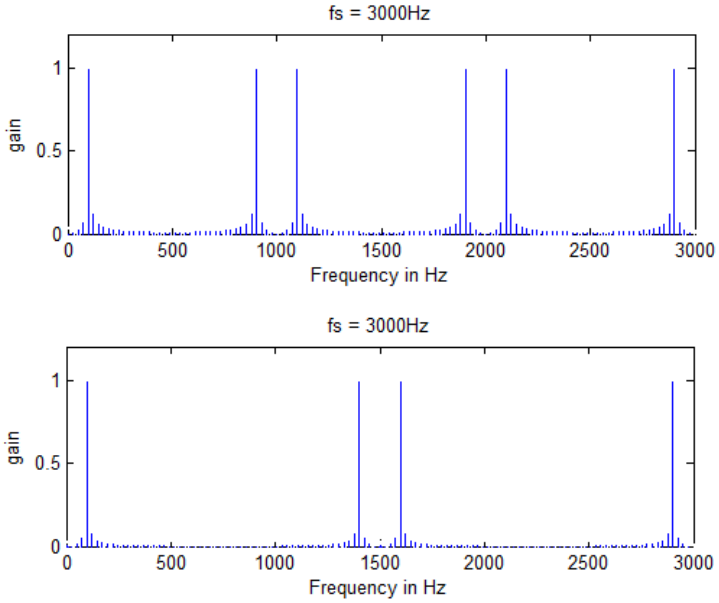


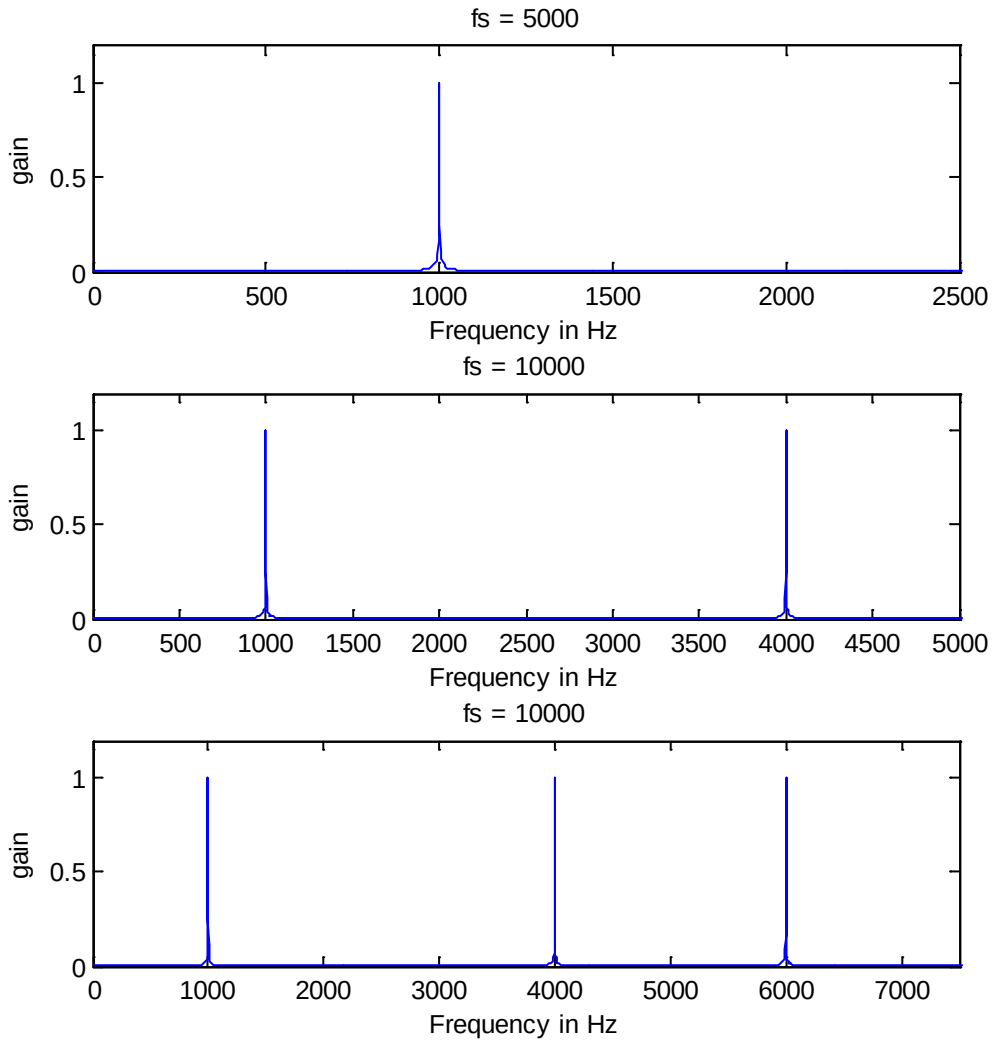
Figure 7.10
Top is the frequency plot for a 100 Hz signal sampled at 1 KHz and up-sampled to 3 KHz. Bottom is the same 100 Hz signal sampled at 1.5 KHz and up-sampled to 3 KHz. The low pass filter specifications to eliminate the images are less stringent for the second case.

Example 7.4
A signal which is limited in frequency to the band from 0 to 3 KHz is sampled at 11,025 Hz. If a 10-bit A/D converter is being used and we want to up-sample this signal to 22,050 Hz, what are reasonable specifications for the interpolation filter?

Solution
The 0 to 3 KHz signal sampled at 11025 Hz will present an image in the range 8,025 Hz to 11,025 Hz which needs to be removed. Since we have a 10-bit A/D converter it is reasonable to limit band edges to half a bit or less. This gives a pass band from 0 to 3 KHz with a gain of 1.0 ±0.000488 and a stop band from 8,025 to $f_s/2$ with a gain of .000488 or 66 DB of attenuation. These specifications can be met with an elliptic filter of order 5 or a Parks-McClellan FIR filter of order 17.

EE 311 **April 9, 2018**
Up sample
SOLUTION

7.2. The upsample function in MATLAB® inserts zeros between samples to get a signal which is sampled at a higher rate. Suppose I have a sinusoid that has a frequency of 1000 Hz that is sampled at 5000 Hz. Predict what the spectrum will look like if the upsample function is applied to the signal to (a) raise the sample frequency by a factor of 2 and (b) raise the sample frequency by a factor of 3. Verify your prediction using MATLAB®.



```
                    fs = 5000;T = 1/fs;
fsig = 1000;
t = 0:T:.4;
x = sin(2*pi*fsig*t);
yup2 = upsample(x, 2);
yup3 = upsample(x, 3);
figure(1);clf;
subplot(3, 1, 1);
xfft = fft(x);
xfft = xfft/max(abs(xfft));
k = 0:length(x)-1;
plot(k*fs/length(x), abs(xfft)/max(abs(xfft)));
axis([0 fs/2 0 1.2]);
xlabel('Frequency in Hz');
ylabel('gain');
title(['fs = ' num2str(fs)]);
%
subplot(3, 1, 2);
yup2fft = fft(yup2);
yup2fft = yup2fft/max(abs(yup2fft));
k = 0:length(yup2)-1;
plot(k*2*fs/length(yup2),
abs(yup2fft)/max(abs(yup2fft)));
axis([0 fs 0 1.2]);
xlabel('Frequency in Hz');
ylabel('gain');
title(['fs = ' num2str(2*fs)]);
%
subplot(3, 1, 3);
yup3fft = fft(yup3);
yup3fft = yup3fft/max(abs(yup3fft));
k = 0:length(yup3)-1;
plot(k*3*fs/length(yup3),
abs(yup3fft)/max(abs(yup3fft)));
axis([0 3*fs/2 0 1.2]);
xlabel('Frequency in Hz');
ylabel('gain');
title(['fs = ' num2str(2*fs)]);
```

Cascaded Interpolators

As in the case of decimators, interpolators are often cascaded when the interpolation factor is large, say greater than 10. The reason for this can be seen by looking at the frequency response of cascaded systems as shown in Figure 7.11. Up-sampling by *U* provides *U* images in the frequency space and only the first image represents the signal we want to keep. As a result the specifications for the anti-imaging filter get tighter as the value of *U* increases. Cascading two or more up-samplers together can alleviate this problem.

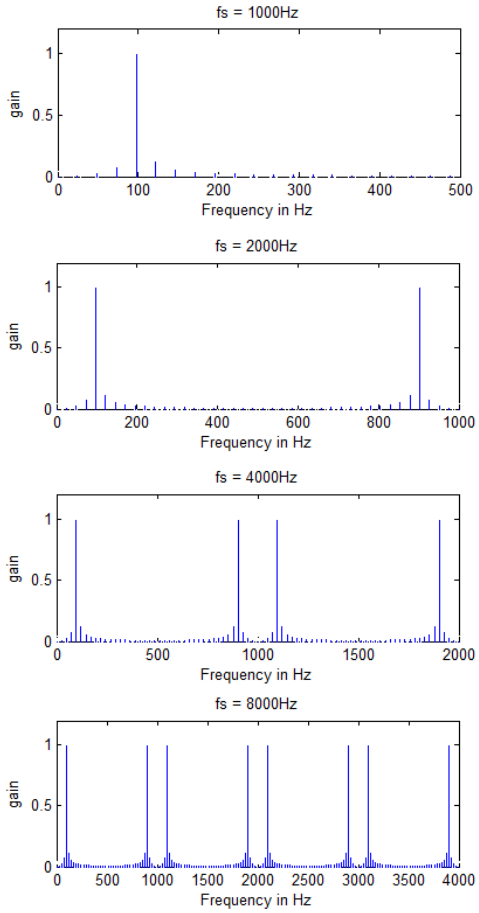


Figure 7.11

The spectrum of a sinusoid at 100 Hz sampled at 1000 Hz (top) is resampled at 2000 Hz, 4000 Hz, and 8000 Hz.

Example 7.5

Consider a sinusoid at 100 Hz which is sampled at 500 Hz. We want to resample the signal at 50,000 Hz. The pass band and stop band ripple should be 0.001. Compare the anti-aliasing filter specifications for an interpolator that increases the sample rate by a factor of 100 to two cascaded interpolators that accomplish the same thing.

Solution

With a sample frequency of 500 Hz and a signal at 100 Hz the first image will be at $f_s - 100 =$ 400 Hz. The anti-imaging filter will have a pass band from 0 to 100 Hz with a ripple of 0.001

and a stop band from 400 Hz to 25,000 Hz with a ripple of 0.001. This can be done with a Parks-McClelland filter of order 543.

To up-sample using two interpolators in cascade by a factor of 100 we have a number of options. For purposes of illustration we choose two interpolators each with a multiplier of 10. For the first anti-aliasing filter the specifications for the pass band and stop band is identical to those used when we up-sampled by a factor of 100. But the sample rate has changed from 50,000 Hz to just 5,000 Hz which makes the normalized transition width smaller. This filter can be implemented as a Parks-McClelland of order 54.

For the second anti-aliasing filter the sample frequency is 50,000 Hz but the signal is sampled at 5,000 Hz. The signal with at 100 Hz will be aliased at 4,900 Hz so the requirements for the second filter are: pass band 0 to 100 Hz, stop band from 4,900 Hz to 25,000 Hz. The ripple remains as before. This second Parks-McClelland filter will have an order of 33.

The two anti-imaging filter magnitude responses are shown in Figure 7.12.

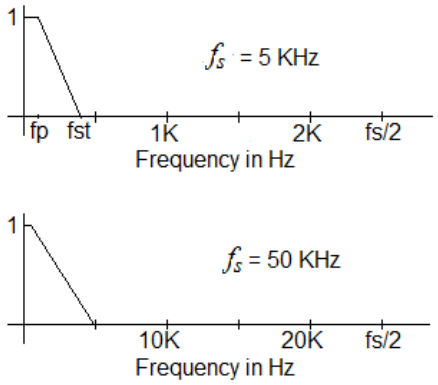


Figure 7.12
Magnitude plots for the two anti-imaging filters.

There are a number of other options to try including 2:50, 4:25, and 5:20 and the inverse of these. Likewise, other low pass filter's may be used including some IIR filters which can be done at significantly lower order. Table 7.1 lists the order for various options using a Parks-McClellan filter.

Multiplier	P-M Order		Multiplier	P-M Order
1:100	543		10:10	54:33
2:50	8:204		50:2	272:2
4:25	21:91		25:4	136:11
5:20	26:71		20:5	109:15

Table 7.1
Cascaded up-sample factor and anti-imaging filter order for Parks-McClelland low pass filter.