12.1 Multirate Digital Signal Processing Basics

In many areas of digital signal processing (DSP) applications—such as communications, speech, and audio processing—rising or lowering of a sampling rate is required. The principle that deals with changing the sampling rate belongs essentially to multirate signal processing (Ifeachor and Jervis, 2002; Porat, 1997; Proakis and Manolakis, 1996; Sorensen and Chen, 1997). As an introduction, we will focus on sampling rate conversion; that is, sampling rate reduction or increase.

12.1.1 Sampling Rate Reduction by an Integer Factor

The process of reducing a sampling rate by an integer factor is referred to as downsampling of a data sequence. We also refer to downsampling as "decimation" (not taking one of ten). The term "decimation" used for the downsampling process has been accepted and used in many textbooks and fields. To downsample a data sequence $x(n)$ by an integer factor of $M$, we use the following notation:

$$y(m) = x(mM), \quad (12.1)$$

where $y(m)$ is the downsampled sequence, obtained by taking a sample from the data sequence $x(n)$ for every $M$ samples (discarding $M - 1$ samples for every $M$ samples). As an example, if the original sequence with a sampling period $T = 0.1$ second (sampling rate = 10 samples per sec) is given by

$x(n): 8 \ 7 \ 4 \ 8 \ 9 \ 6 \ 4 \ 2 \ -2 \ -5 \ -7 \ -6 \ -4 \ ...$

and we downsample the data sequence by a factor of 3, we obtain the downsampled sequence as
\[ y(m) = 8 \ 8 \ 4 \ -5 \ -6 \ldots , \]

with the resultant sampling period \( T = 3 \times 0.1 = 0.3 \) second (the sampling rate now is 3.33 samples per second). Although the example is straightforward, there is a requirement to avoid aliasing noise. We will illustrate this next.

From the Nyquist sampling theorem, it is known that aliasing can occur in the downsampled signal due to the reduced sampling rate. After downsampling by a factor of \( M \), the new sampling period becomes \( MT \), and therefore the new sampling frequency is

\[ f_{sm} = 1/(MT) = f_s / M, \quad (12.2) \]

where \( f_s \) is the original sampling rate.

Hence, the folding frequency after downsampling becomes

\[ f_{sM}/2 = f_s/(2M). \quad (12.3) \]

This tells us that after downsampling by a factor of \( M \), the new folding frequency will be decreased \( M \) times. If the signal to be downsampled has frequency components larger than the new folding frequency, \( f > f_s/(2M) \), aliasing noise will be introduced into the downsampled data.

To overcome this problem, it is required that the original signal \( x(n) \) be processed by a lowpass filter \( H(z) \) before downsampling, which should have a stop frequency edge at \( f_s/(2M) \) (Hz). The corresponding normalized stop frequency edge is then converted to be

\[ \Omega_{\text{stop}} = 2\pi \left( f_s/(2M) \right) T = \pi/M \text{ radians}. \quad (12.4) \]

In this way, before downsampling, we can guarantee that the maximum frequency of the filtered signal satisfies

\[ f_{\text{max}} < f_s/(2M), \quad (12.5) \]

such that no aliasing noise is introduced after downsampling. A general block diagram of decimation is given in Figure 12-1, where the filtered output in terms of the z-transform can be written as

\[ W(z) = H(z)X(z), \quad (12.6) \]

where \( X(z) \) is the z-transform of the sequence to be decimated, \( x(n) \), and \( H(z) \) is the lowpass filter transfer function. After anti-aliasing filtering, the downsampled signal \( y(m) \) takes its value from the filter output as:

\[ y(m) = w(mM). \quad (12.7) \]

The process of reducing the sampling rate by a factor of 3 is shown in Figure 12-1. The corresponding spectral plots for \( x(n) \), \( w(n) \), and \( y(m) \) in general are shown in Figure 12-2.
Figure 12-1. Block diagram of the downsampling process with M = 3:

Figure 12-2. Spectrum after downsampling.

**Downsampling example**

To verify this principle, let us consider a signal $x(n)$ generated by the following:

$$x(n) = 5 \sin \left(\frac{2\pi \times 1000n}{8000}\right) + \cos \left(\frac{2\pi \times 2500n}{8000}\right), \quad (12.8)$$
with a sampling rate of \( f_s = 8,000 \) Hz, the spectrum of \( x(n) \) is plotted in the first graph in Figure 12-3a, where we observe that the signal has components at frequencies of 1,000 and 2,500 Hz. Now we downsample \( x(n) \) by a factor of 2, that is, \( M = 2 \). According to Equation (12.3), we know that the new folding frequency is \( 4000/2 = 2000 \) Hz. Hence, without using the anti-aliasing lowpass filter, the spectrum would contain the aliasing frequency of \( 4 \text{ kHz} - 2.5 \text{ kHz} = 1.5 \text{ kHz} \) introduced by 2.5 kHz, plotted in the second graph in Figure 12-3a.

![Figure 12-3A. Spectrum before downsampling and spectrum after downsampling without using the anti-aliasing filter.](image)

Now we apply a finite impulse response (FIR) lowpass filter designed with a filter length of \( N = 27 \) and a cutoff frequency of 1.5 kHz to remove the 2.5-kHz signal before downsampling to avoid aliasing. How to obtain such specifications will be discussed in a later example. The normalized cutoff frequency used for design is given by

\[
\Omega_c = 2\pi \times 1500 \times (1/8000) = 0.375\pi.
\]

Thus, the aliasing noise is avoided. The spectral plots are given in Figure 12-3b, where the first plot shows the spectrum of \( w(n) \) after anti-aliasing filtering, while the second plot describes the spectrum of \( y(m) \) after downsampling. Clearly, we prevent aliasing noise in the downsampled data by sacrificing the original 2.5-kHz signal. Program 12-1 gives the detail of MATLAB implementation.
Figure 12-3B. Spectrum before downsampling and spectrum after downsampling using the anti-aliasing filter.

Program 12.1. MATLAB program for decimation.

close all; clear all;
% Downsampling filter (see Chapter 7 for FIR filter design)
H=[0.00074963181416 0.00074963033476 0.00074963444416 -0.00044044128500 ... -0.00910635730626 0.00000000000000 0.000367631506 -0.0023971562885 ... -0.00171263672800 -0.00062620094567 -0.0003506790030210 -0.01060980550088 ... -0.002901490103794 0.35000000000000 0.2901490103794 0.1060980550088 ... -0.03590790035210 -0.06376620469567 -0.0171263672810 0.0223710562885 ... -0.00203567631506 0.00000000000000 -0.00910635730626 -0.0044044128500 ... -0.00146936649416 0.00074963033476 0.00074963181416];
% Generate the 2048 samples
fs=8000; % Sampling rate
N=2048; % Number of samples
M=2; % Downsample factor
n=0:1:N-1;
X = 5*sin(n*pi/4)+cos(5*n*pi/8); % Compute the single-sided amplitude spectrum
% AC component will be doubled, and DC component will be kept the same value
X = 2*abs(fft(x,N))/N*X(1)=X(1)/2;
% Map the frequency index up to the folding frequency in Hz
f=[0:1:N/2-1]/fs;
% Downsampling
y=x(1:M:N);
% Length of the downsampled data
% Compute the single-sided amplitude spectrum for the downsampled signal
Y = 2*abs(fft(y,N))/length(y);Y(1)=Y(1)/2;
% Map the frequency index to the frequency in Hz
fM=[0:1:N/2-1]/fs/2;
subplot(2,1,1);plot(f,abs(y(1:N/2)));grid;xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fs*Y(1:N/2));grid;xlabel('Frequency (Hz)');
figure
w=filter(B,1,x); % Anti-aliasing filtering
% Compute the single-sided amplitude spectrum for the filtered signal
W = 2*abs(fft(w,N))/N*Y(1)=Y(1)/2;
% Downsample
y=w(1:M:N);
% Compute the single-sided amplitude spectrum for the downsampled signal
Y = 2*abs(fft(y,N))/N*Y(1)=Y(1)/2;
% Plot spectra
subplot(2,1,1);plot(f,abs(y(1:N/2)));grid;xlabel('Frequency (Hz)');
subplot(2,1,2);plot(fs*Y(1:N/2));grid;xlabel('Frequency (Hz)');

(Click to enlarge)
Program 12-1. MATLAB program for decimation.

Now we focus on how to design the anti-aliasing FIR filter, or decimation filter. We will discuss this topic via the following example.

**Example 12.1.**
Given a DSP downsampling system with the following specifications, determine the FIR filter length, cutoff frequency, and window type if the window method is used:

- Sampling rate = 6,000 Hz
- Input audio frequency range = 0–800 Hz
- Passband ripple = 0.02 dB
- Stopband attenuation = 50 dB
- Downsampling factor $M = 3$,

Solution: Specifications are reorganized as:

- Anti-aliasing filter operating at the sampling rate = 6000 Hz
- Passband frequency range = 0–800 Hz
- Stopband frequency range = 1–3 kHz
- Passband ripple = 0.02 dB
- Stopband attenuation = 50 dB
- Filter type = FIR.

The block diagram and specifications are depicted in Figure 12-4.

![Block diagram for Example 12.1](image)

*Figure 12-4. Filter specifications for Example 12.1.*

The Hamming window is selected, since it provides 0.019 dB ripple and 53 dB stopband attenuation. The normalized transition band is given by

$$\Delta f = (f_{\text{stop}} - f_{\text{pass}})/f_s = (1000 - 800)/6000 = 0.033.$$  

The length of the filter and the cutoff frequency can be determined by

$$N = 3.3/\Delta f = 3.3/0.033 = 100.$$  

We choose the odd number, that is, $N = 101$, and

$$f_c = (f_{\text{pass}} + f_{\text{stop}})/2 = (800 + 1000)/2 = 900 \text{ Hz}.$$  

**Sampling Rate Increase by an Integer Factor**

12.1.2 Sampling Rate Increase by an Integer Factor
Increasing a sampling rate is a process of upsampling by an integer factor of $L$. This process is described as follows:

$$y(m) = \begin{cases} x(m/L) & m = nL, \\ 0 & \text{otherwise}, \end{cases} \quad (12.9)$$

where $n = 0, 1, 2, \ldots$, $x(n)$ is the sequence to be upsampled by a factor of $L$, and $y(m)$ is the upsampled sequence. As an example, suppose that the data sequence is given as follows:

$x(n): 8 \ 8 \ 4 \ -5 \ -6 \ldots$

After upsampling the data sequence $x(n)$ by a factor of 3 (adding $L - 1$ zeros for each sample), we have the upsampled data sequence $w(m)$ as:

$w(m): 8 \ 0 \ 0 \ 8 \ 0 \ 0 \ 4 \ 0 \ 0 \ -5 \ 0 \ 0 \ -6 \ 0 \ 0 \ldots$

The next step is to smooth the upsampled data sequence via an interpolation filter. The process is illustrated in Figure 12-5a.

Similar to the downsampling case, assuming that the data sequence has the current sampling period of $T$, the Nyquist frequency is given by $f_{\text{max}} = f_s/2$. After upsampling by a factor of $L$, the new sampling period becomes $T/L$, thus the new sampling frequency is changed to be $f_s L = Lf_s$. (12.10)

This indicates that after upsampling, the spectral replicas originally centered at $f_s$, $2f_s$, $\ldots$ are included in the frequency range from 0 Hz to the new Nyquist limit $Lf_s = 2$ Hz, as shown in Figure 12-5b. To remove those included spectral replicas, an interpolation filter with a stop frequency edge of $f_s = 2$ in Hz must be attached, and the normalized stop frequency edge is given by...
\[ \Omega_{\text{ups}} = 2\pi \left( \frac{f_s}{2} \right) \times \left( \frac{T}{L} \right) = \pi/L \text{ radians. (12.11)} \]

After filtering via the interpolation filter, we will achieve the desired spectrum for \( y(n) \), as shown in Figure 12-5b. Note that since the interpolation is to remove the high-frequency images that are aliased by the upsampling operation, it is essentially an anti-aliasing lowpass filter. **Upsampling example**

To verify the upsampling principle, we generate the signal \( x(n) \) with 1 kHz and 2.5 kHz as follows:

\[
x(n) = 5 \sin \left( \frac{2\pi \times 1000n}{8000} \right) + \cos \left( \frac{2\pi \times 2500n}{8000} \right),
\]

with a sampling rate of \( f_s = 8,000 \) Hz. The spectrum of \( x(n) \) is plotted in Figure 12-6. Now we upsample \( x(n) \) by a factor of 3, that is, \( L = 3 \). We know that the sampling rate is increased to be \( 3 \times 8000 = 24,000 \) Hz. Hence, without using the interpolation filter, the spectrum would contain the image frequencies originally centered at the multiple frequencies of 8 kHz. The top plot in Figure 12-6 shows the spectrum for the sequence after upsampling and before applying the interpolation filter.
Figure 12-6. (Top) The spectrum after upsampling and before applying the interpolation filter; (bottom) the spectrum after applying the interpolation filter.

Now we apply an FIR lowpass filter designed with a length of 53, a cutoff frequency of 3,250 Hz, and a new sampling rate of 24,000 Hz as the interpolation filter, whose normalized frequency should be

\[ \Omega_c = 2\pi \times 3250 \times (1/24000) = 0.2708\pi. \]

The bottom plot in Figure 12-6 shows the spectrum for \( y(m) \) after applying the interpolation filter, where only the original signals with frequencies of 1 kHz and 2.5 kHz are presented. Program 12-2 shows the implementation detail in MATLAB. Now let us study how to design the interpolation filter via Example 12.2.
Example 12.2.
Given a DSP upsampling system with the following specifications, determine the FIR filter length, cutoff frequency, and window type if the window design method is used:

- Sampling rate = 6,000 Hz
- Input audio frequency range = 0–800 Hz
- Passband ripple = 0.02 dB
- Stopband attenuation = 50 dB
- Upsample factor $L = 3$

Solution: The specifications are reorganized as follows:

- Interpolation filter operating at the sampling rate = 18,000 Hz
- Passband frequency range = 0–800 Hz
- Stopband frequency range = 3–9 kHz
- Passband ripple = 0.02 dB
- Stopband attenuation = 50 dB
- Filter type = FIR.
We choose the Hamming window for this application. The normalized transition band is

$$\Delta f = (f_{\text{stop}} - f_{\text{pass}})/f_{\text{sl}} = (3000 - 800)/18000 = 0.1222.$$ 

The length of the filter and the cutoff frequency can be determined by

$$N = 3.3/\Delta f = 3.3/0.1222 = 27,$$

and the cutoff frequency is given by

$$f_c = (f_{\text{pass}} + f_{\text{stop}})/2 = (3000 + 800)/2 = 1900 \text{ Hz}.$$ 

Part 2 shows how to change the sampling rate by a non-integer factor. It also looks at multistage decimation and polyphase filters. It will be published Monday, April 28.

Related articles
- Design: Basics of ADCs and DACs, part 1
- Design: Comment: What is a DSP engineer?
- Paper: Q5 Upsampling / Sample Rate Conversion Technology