MEHRAN UNIVERSITY OF ENGINEERING & TECHNOLOGY, JAMSHORO
INSTITUTE OF INFORMATION & COMMUNICATION TECHNOLOGIES

ADVANCED DIGITAL SIGNAL PROCESSING

Lab 10: Multirate Digital Signal Processing

Objective

The objective of this lab is to alter the sampling rate of signal.

1 Introduction

In many practical applications of digital signal processing, one is faced with the problem of changing the sampling rate of a signal, either increasing it or decreasing it by some amount. The process of converting a signal from a given rate to a different rate is called *sampling rate conversion*. In turn, systems that employ multiple sampling rates in the processing of digital signals are called *multirate digital signal processing systems*.

Let us think of an underlying (or original) analog signal $x_a(t)$ that was sampled using the sampling rate of $F_s = \frac{1}{T}$ *samples/second* to produce a discrete signal $x(n)$. The resulting digital signal $x(n)$ is subsequently filtered using a lowpass filter (LPF) with a cutoff frequency of ω_c.

Thus, the output signal y(n) has all its energy in the band $0 \leq \omega \leq \omega_c = 2\pi f_c$. According to the sampling theorem, such a signal may be represented by the rate of $^{2f_c}/_T$ samples/second instead of its existing rate of $F_s = \frac{1}{T}$. Note that $|f_c| \leq 0.5$. However, if $f_c \ll 0.5$, then $^{2f_c}/_T \ll F_s$. Hence it would seem more advantageous to lower the sampling frequency to a value closer to $^{2f_c}/_T$ and perform signal processing operations at this lower rate.

Other applications include the need for an optimal interpolation in computer tomography and efficient multistage designs of narrowband lowpass filters.

The MATLAB Signal Processing toolbox may be used to carry out a variety of operations associated with sampling rate conversion and multirate processing. The key functions are listed below,

1. `decimate` % resample data at a lower rate after lowpass filtering
2. `resample` % change the sampling rate of a signal
3. `interp` % resample data at a higher rate using lowpass interpolation
4. `upfirdn` % upsample, apply a specified FIR filter, and downsample a signal

Use on line MATLAB help to find out more about these functions. Following examples illustrate the use of these functions.

2 Decimation

The basic operation required in decimation is the downsampling of the high-rate signal into a low-rate signal. MATLAB provides the function `[y] = downsample(x,D)` that downsamples input array `x` into output array `y` by keeping every D^{th} sample starting with the first sample. An optional third parameter "phase" specifies the sample offset which must be an integer between `0` and `(D-1)`. For example,

```
≫ x = [1,2,3,4,3,2,1]; y = downsample(x,2)

y =
   1   3   3   1
```

downsamples by a factor of 2 starting with the first sample. However,

```
≫ x = [1,2,3,4,3,2,1]; y = downsample(x,2,1)

y =
   2   4   2
```

produces an entirely different sequence by downsampling, starting with the second sample (i.e., offset by 1).

2.1 Example

Let $x(n) = \cos(0.125\pi n)$. Generate a large number of samples of $x(n)$ and decimate them using D = 2, 4, and 8 to show the results of decimation.

MATLAB provides the function `y = decimate(x,D)` that resamples the sequence in array `x` at `1/D` times the original sampling rate. The resulting resampled array `y` is `D` times shorter i.e., `length(y) = length(x)/D`. Designing an ideal lowpass filter is not possible in the MATLAB implementation; however, fairly accurate approximations are used. The default lowpass filter used in the function is an 8th-order Chebyshev type-I lowpass filter with the cutoff frequency of $^{0.8\pi}/_D$. Using additional optional arguments, the filter order can be changed or an FIR filter of specified order and cutoff frequency can be used.

We will plot the middle segments of the signals to avoid end-effects due to the default lowpass filter in the `decimate` function. The following MATLAB script shows details of these operations

```
clc, clear all, close all
n = 0:2048; k1 = 256; k2 = k1+32; m = 0:(k2-k1)
    ;
Hf1 = figure('units','inches','position'
    ,[1,1,6,4], ...
    'paperunits','inches','paperposition'
        ,[0,0,6,4]);
% (a) Original signal
x = cos(0.125*pi*n);
subplot(2,2,1);
```

```
Ha = stem(m,x(m+k1+1),'g','filled');
axis([-1,33,-1.1,1.1]); set(Ha,'markersize',2);
ylabel('Amplitude'); title('Original Sequence x
    (n)');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (b) Decimation by D = 2
D = 2; y = decimate(x,D);
subplot(2,2,2);
Hb = stem(m,y(m+k1/D+1),'c','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hb,'markersize',2); ylabel('Amplitude');
title('Decimated by D = 2');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (c) Decimation by D = 4
D = 4; y = decimate(x,D); subplot(2,2,3);
Hc = stem(m,y(m+k1/D+1),'r','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hc,'markersize',2); ylabel('Amplitude');
title('Decimated by D = 4');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
% (d) Decimation by D = 8
D = 8; y = decimate(x,D); subplot(2,2,4);
Hd = stem(m,y(m+k1/D+1),'m','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hd,'markersize',2); ylabel('Amplitude');
title('Decimated by D = 8');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
```

We observe that the decimated sequences for D = 2 and D = 4 are correct and represent the original sinusoidal sequence $x(n)$ at lower sampling rates. However, the sequence for D = 8 is almost zero because the lowpass filter has attenuated $x(n)$ prior to downsampling. Recall that the cutoff frequency of the lowpass filter is set to $0.8\pi/D = 0.1\pi$ which eliminates $x(n)$. If we had used the downsampling operation on $x(n)$ instead of decimation, the resulting sequence would be $y(m) = 1$, which is an aliased signal. Thus, the lowpass filtering is necessary.

3 Interpolation

An increase in the sampling rate can be accomplished by interpolating new samples between successive values of the signal. This process can be accomplished in two steps. The first step creates an intermediate signal at the high rate by interlacing zeros in between nonzero samples in an operation called *upsampling*. In the second step, the intermediate signal is filtered to "fill in" zero-interlaced samples to create the interpolated high-rate signal.

MATLAB provides the function `[v] = upsample(x,I)` that upsamples input array `x` into output `v` by inserting `(I-1)` zeros between input samples. An optional third parameter, "phase," specifies the sample offset, which must be an integer between `0` and `(I-1)`. For example,

```
>> x = [1,2,3,4]; v = upsample(x,3)

v =
   1   0   0   2   0   0   3   0   0   4   0
   0
```

upsamples by a factor of 2 starting with the first sample. However,

```
>> v = upsample(x,3,1)

v =
   0   1   0   0   2   0   0   3   0   0   4
   0

>> v = upsample(x,3,2)

v =
   0   0   1   0   0   2   0   0   3   0   0
   4
```

produces two different signals by upsampling, starting with the second and the third sample (i.e., offset by 1), respectively. Note that the lengths of the upsampled signals are `I` times the length of the original signal.

3.1 Example

Let $x(n) = \cos(\pi n)$. Generate samples of $x(n)$ and interpolate them using I = 2, 4, and 8 to show the results of interpolation.

MATLAB provides the function `[y,h] = interp(x,I)` that resamples the signal in array `x` at `I` times the original sampling rate. The resulting resampled array `y` is `I` times longer i.e., `length(y) = I*length(x)`. The ideal lowpass filter is approximated by a symmetric filter impulse response, `h`, which is designed internally. It allows the original samples to pass through unchanged and interpolates between so that the mean square error between them and their ideal values is minimized. The third optional parameter `L` specifies the length of the symmetric filter as `2*L*I+1`, and the fourth optional parameter cutoff specifies the cutoff frequency of the input signal in π units. The default values are `L = 5` and `cutoff = 0.5`. Thus, if `I = 2`, then the length of the symmetric filter is `21` for the default `L = 5`.

We will plot the middle segments of the signals to avoid end-effects due to the default lowpass filter in the `interp` function. The following MATLAB script shows details of these operations

```
clc, clear all, close all
n = 0:256; k1 = 64; k2 = k1+32; m = 0:(k2-k1);
Hf1 = figure('units','inches','position'
    ,[1,1,6,4], ...
    'paperunits','inches','paperposition'
        ,[0,0,6,4]);
```

```
% (a) Original signal
x = cos(pi*n);
subplot(2,2,1);
Ha = stem(m,x(m+k1+1),'g','filled');
axis([-1,33,-1.1,1.1]); set(Ha,'markersize',2);
ylabel('Amplitude'); title('Original Sequence x
    (n)');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (b) Interpolation by I = 2
I = 2; y = interp(x,I);
subplot(2,2,2);
Hb = stem(m,y(m+k1/I+1),'c','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hb,'markersize',2); ylabel('Amplitude');
title('Interpolated by I = 2');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (c) Interpolation by I = 4
I = 4; y = interp(x,I); subplot(2,2,3);
Hc = stem(m,y(m+k1/I+1),'r','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hc,'markersize',2); ylabel('Amplitude');
title('Interpolated by I = 4');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
% (d) Interpolation by I = 8
I = 8; y = interp(x,I); subplot(2,2,4);
Hd = stem(m,y(m+k1/I+1),'m','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hd,'markersize',2); ylabel('Amplitude');
title('Interpolated by I = 8');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
```

We observe that the interpolated sequences for all three values of `I` are appropriate and represent the original sinusoidal signal $x(n)$ at higher sampling rates. In the case of $I = 8$, the resulting sequence does not appear to be perfectly sinusoidal in shape. This may be due the fact the lowpass filter is not close to an ideal filter.

3.2 Example

Examine the frequency response of the lowpass filter used in the interpolation of the signal in Example 3.1.

The second optional argument in the `interp` function provides the impulse response from which we can compute the frequency response, as shown in the following MATLAB script.

```
clc, clear all, close all
n = 0:256; x = cos(pi*n); w = [0:100]*pi/100;
Hf1 = figure('units','inches','position'
    ,[1,1,6,4], ...
    'paperunits','inches','paperposition'
        ,[0,0,6,4]);

% (a) Interpolation by I = 2, L = 5;
I = 2; [y1,h1] = interp(x,I); H1 = freqz(h1,1,w
    ); H1 = abs(H1);
subplot(2,2,1); plot(w/pi,H1,'g'); axis([0,1,0,
    I+0.1]); ylabel('Magnitude');
title('I = 2, L = 5');
set(gca,'xtick',[0,0.5,1]); set(gca,'ytick'
    ,0:1:I);
% (b) Interpolation by I = 4, L = 5;
I = 4; [y2,h2] = interp(x,I); H2 = freqz(h2,1,w
    ); H2 = abs(H2);
subplot(2,2,2); plot(w/pi,H2,'g'); axis([0,1,0,
    I+0.2]); ylabel('Magnitude');
title('I = 4, L = 5');
set(gca,'xtick',[0,0.25,1]); set(gca,'ytick'
    ,0:1:I);
% (c) Interpolation by I = 8, L = 5;
I = 8; [y3,h3] = interp(x,I); H3 = freqz(h3,1,w
    ); H3 = abs(H3);
subplot(2,2,3); plot(w/pi,H3,'g'); axis([0,1,0,
    I+0.4]); ylabel('Magnitude');
title('I = 8, L = 5'); xlabel('\omega/\pi','
    fontsize',10)
set(gca,'xtick',[0,0.125,1]); set(gca,'ytick'
    ,0:2:I);
% (d) Interpolation by I = 8, L = 7;
I = 8; L = 7; [y4,h4] = interp(x,I,L); H4 =
    freqz(h4,1,w); H4 = abs(H4);
subplot(2,2,4); plot(w/pi,H4,'g'); axis([0,1,0,
    I+0.4]);
ylabel('Magnitude');
title('I = 8, L = 7'); xlabel('\omega/\pi','
    fontsize',10)
set(gca,'xtick',[0,0.125,1]); set(gca,'ytick'
    ,0:2:I);
```

The first three frequency response plots are for $L = 5$ and, as expected, the filters are all lowpass with passband edges approximately around π/I frequencies and the gain of `I`. Also note that the filters do not have sharp transitions and thus are not good approximations to the ideal filter. The last plot shows the response for $L = 7$, which indicates a more sharp transition, which is to be expected. Any value beyond $L = 7$ results in an unstable filter design and hence should be avoided.

4 Sampling Rate Conversion

Having discussed the special cases of decimation (downsampling by a factor `D`) and interpolation (upsampling by a factor `I`), we now consider the general case of sampling rate conversion by a rational factor `I/D`. Basically, we can achieve this sampling rate conversion by first performing interpolation by the factor `I` and then decimating the output of the interpolator by the factor `D`. In other words, a sampling rate conversion by the rational factor `I/D` is accomplished by cascading an interpolator with a decimator. We emphasize that the importance of performing the interpolation first and the decimation second is to preserve the desired spectral characteristics of $x(n)$.

4.1 Example

Consider the sequence $x(n) = \cos(0.125\pi n)$ discussed in Example 2.1. Change its sampling rate by 3/2, 3/4, and 5/8.

MATLAB provides the function `[y,h] = resample(x,I,D)` that resamples the signal in array `x` at `I/D` times the original sampling rate. The resulting resampled array `y` is `I/D` times longer (or the ceiling of it if the ratio is not an integer) i.e., `length(y) = ceil(I/D)*length(x)`. The function approximates the anti-aliasing (lowpass) filter by an FIR filter, `h`, designed (internally) using the Kaiser window. It also compensates for the filter's delay.

The length of the FIR filter `h` that resample uses is proportional to the fourth (optional) parameter `L` that has the default value of 10. For `L = 0`, resample performs a nearest-neighbor interpolation. The fifth optional parameter beta (default value 5) can be used to specify the Kaiser window stopband attenuation parameter β. The filter characteristics can be studied using the impulse response `h`.

The following MATLAB script shows the details.

```
clc, clear all, close all
n = 0:2048; k1 = 256; k2 = k1+32; m = 0:(k2-k1)
    ;
Hf1 = figure('units','inches','position'
    ,[1,1,6,4],...
    'paperunits','inches','paperposition'
        ,[0,0,6,4]);
% (a) Original signal
x = cos(0.125*pi*n); subplot(2,2,1);
Ha = stem(m,x(m+k1+1),'g','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Ha,'markersize',2); ylabel('Amplitude');
title('Original Sequence x(n)');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (b) Sample rate Conversion by 3/2: I= 3, D =
    2
I = 3; D = 2; y = resample(x,I,D); subplot
    (2,2,2);
Hb = stem(m,y(m+k1*I/D+1),'c','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hb,'markersize',2); ylabel('Amplitude');
title('Sample Rate I/D: I = 3, D = 2');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
% (c) Sample rate Conversion by 3/4: I= 3, D =
    4
I = 3; D = 4; y = resample(x,I,D); subplot
    (2,2,3);
Hc = stem(m,y(m+k1*I/D+1),'r','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hc,'markersize',2); ylabel('Amplitude');
title('Sample Rate I/D: I = 3, D = 4');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
% (d) Sample rate Conversion by 5/8: I= 5, D =
    8
I = 5; D = 8; y = resample(x,I,D); subplot
    (2,2,4);
Hd = stem(m,y(m+k1*I/D+1),'m','filled'); axis
    ([-1,33,-1.1,1.1]);
set(Hd,'markersize',2); ylabel('Amplitude');
title('Sample Rate I/D: I = 5, D = 8');
set(gca,'xtick',[0,16,32]); set(gca,'ytick'
    ,[-1,0,1]);
xlabel('n');
```

The original $x(n)$ signal has 16 samples in one period of the cosine waveform. Since the first sampling rate conversion by 3/2 is greater than one, the overall effect is to interpolate $x(n)$. The resulting signal has $16 \times 3/2 = 24$ samples in one period. The other two sampling rate conversion factors are less than one; thus, overall effect is to decimate $x(n)$. The resulting signals have $16 \times 3/4 = 12$ and $16 \times 5/8 = 10$ samples per period, respectively.

5 Exercise

1. A continuous time signal is characterized by the following function,

$$x(t) = A\cos(2\pi f_1 t) + B\cos(2\pi f_2 t)$$

   (a) Using MATLAB, generate a discrete time equivalent of the signal. Assume a sampling frequency of 1 kHz, $f_1 = 50$ Hz, $f_2 = 100$ Hz, $A = 1.5$, $B = 1$.

   (b) Interpolate the discrete time signal by a factor of 4 using the `interp` command.

   (c) Decimate the output of the interpolator in step (b) by a factor of 4 using the decimate function.

   (d) Plot the original, interpolated and decimated discrete time signal.