Digital Signal Processing Ⅰ

15th Week EXPERIMENT

Report

(10th report of DSP1 course)

Subject	Digital Signal Processing Ⅰ
Professor	Joon-Hyuk Chang
Submission Date	June 13th, 2021
University	Hanyang University
School	College of Engineering
Department	Department of Computer Science & Engineering
Student ID	**Name**
2019009261	최가온(CHOI GA ON)

-Contents-

Ⅰ. Abstraction

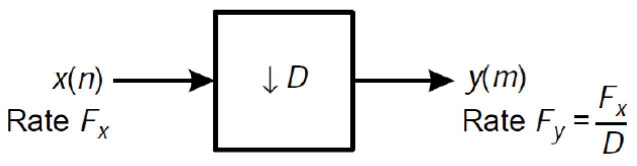
1. System representation in the z-domain

1-1. The Downsampler

The basic operation required in decimation is the downsampling of the high-rate signal $x[n]$ into a low-rate signal $y[m]$.

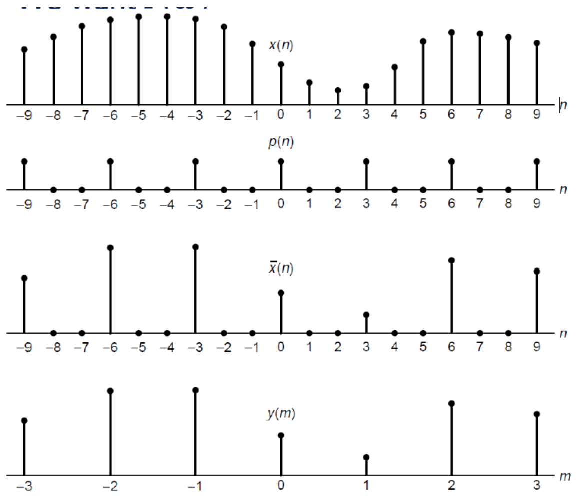
The down-sampled signal $y[m]$ is obtained by selecting one out of D samples of $x[n]$ and throwing away the other $(D-1)$ samples out of every D samples.

$$y[m] = x[n]|_{n=mD} = x[mD]\,; \quad n, m, D \in \{integers\}$$



1-2. Frequency-domain representation of the downsampled signal $y[m]$

$$Y(z) = \sum_{m=-\infty}^{\infty} x[m]\left[\frac{1}{D}\sum_{k=0}^{D-1} e^{\frac{j\pi mk}{D}}\right] z^{-\left(\frac{m}{D}\right)} = \frac{1}{D}\sum_{k=0}^{D-1}\sum_{m=-\infty}^{\infty} x[m]\left(e^{-\frac{j2\pi k\,1}{D_z\;D}}\right)^{-m}$$
$$= \frac{1}{D}\sum_{k=0}^{D-1} X\left(e^{-\frac{j2\pi k\,1}{D_z\;D}}\right)$$



Expressing $Y(w)$ in terms of $X(w)$,

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

To avoid aliasing erros, one nedds the spectrum $X(w_x)$ to be less than full band or band limited.

In fact, we must have

$$X(w_x) = 0 \quad for \quad \frac{\pi}{D} \le |w_x| \le \pi$$

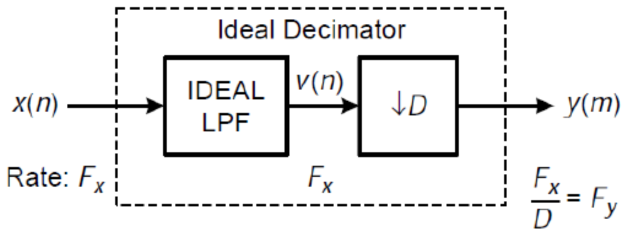
1-3. The ideal decimator

To avoid aliasing, we must first reduce the bandwidth of $x[n]$ to $F_{x,max} = \frac{F_x}{2D}$ or, equivalently $w_{x,max} = \pi/D$.

The decimation process is illustrated as below.

1) The input sequence $x[n]$ is passed through a low pass filter.

2) $v[n]$ is downsampled by the factor D.

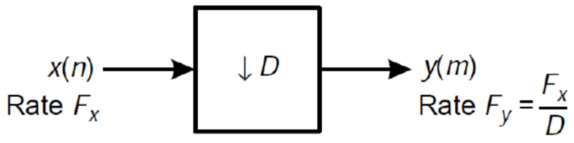


1-4. The upsampler

An increase in the sampling rate by an integer factor of I can accomplished by interpolating $I-1$ new samples between successive values of the signal.

The downsampled signal $y[m]$ is obtained by selecting one out of D samples of $x[n]$ and throwing away the other $(D-1)$ samples out of every D samples.

$$y[m] = x[n]|_{n=mD} = x[md] \;\; ; \quad n, m, D \in \{integers\}$$



This process can be accomplished in two steps.

1) Create an intermediate signal at the high rate F_y by interlacing zeros in between nonzero samples in an operation called up-sampling.

2) The intermediate signal is filtered to “fill in” zero-interlaced samples to create the interpolated high-rate signal.

1-5. Frequency-domain representation of the upsampled signal $y[m]$

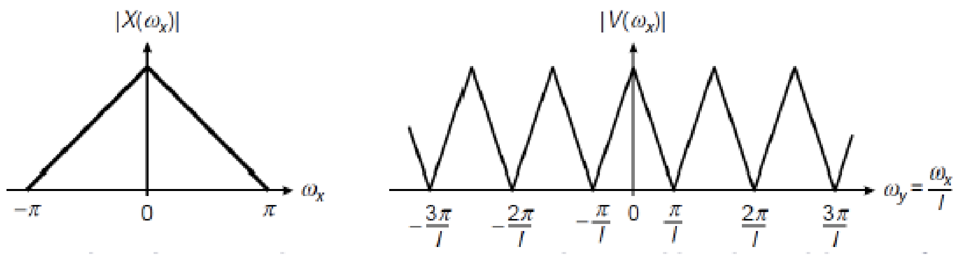
The spectrum of $v[m]$ is obtained by

$$V(w_y) = X(w_y I)$$

where w_y denotes the frequency variable relative to the new sampling rate F_y.

The frequency variables w_x and w_y are related according to the formula

$$w_y = \frac{w_x}{I}$$

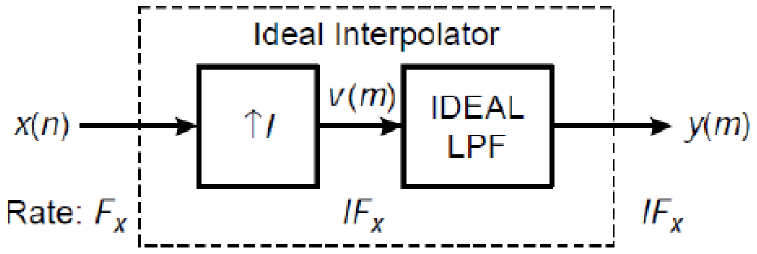


We observe that the sampling rate increase, obtained by the addition of $I-1$ zero samples between successive values of $x[n]$, results in a signal whose spectrum $V(w_y)$ is an $I-fold$ periodic repetition of the input signal spectrum $X(w_x)$.

1-6. The ideal interpolator

Since only the frequency components of in the range are unique, the images of above should be rejected by passing the sequence through a lowpass filter with a frequency response that ideally has the characteristic

$$H_I(w_y) = \begin{cases} C, & 0 \le |w_y| \le \pi/I \\ 0, & otherwise \end{cases}$$



Ⅱ. Exercises

In this part, there are four exercise questions. Each exercise consists of code and its result. All documents including Matlab code, result, and this report are uploaded in this website:

https://github.com/Gaon-Choi/ELE3076/tree/main/10_sampling

Example 1

Using $D = 2$ and $x[n] = \{1, 2, 3, 4, 3, 2, 1\}$, verify that the down-sampler is time varying.

The downsampled signal is $y[m] = \{1, 3, 3, 1\}$. If we now delay $x[n]$ by one sample, we get $x[n-1] = \{0, 1, 2, 3, 4, 3, 2, 1\}$. The corresponding downsampled signal is $y_1[m] = \{0, 2, 4, 2\}$, which is different from $y[m-1]$.

[y] = downsample(x, D)

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).

(MATLAB Code)

```
x1 = [1, 2, 3, 4, 3, 2, 1]
y1 = downsample(x1, 2)

x2 = [1, 2, 3, 4, 3, 2, 1];
y2 = downsample(x2, 2, 1)
```

The first code downsamples by a factor 2 starting with the first sample.

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

(Result)

```
x1 =

     1     2     3     4     3     2     1

y1 =

     1     3     3     1

y2 =

     2     4     2
```

Example 2

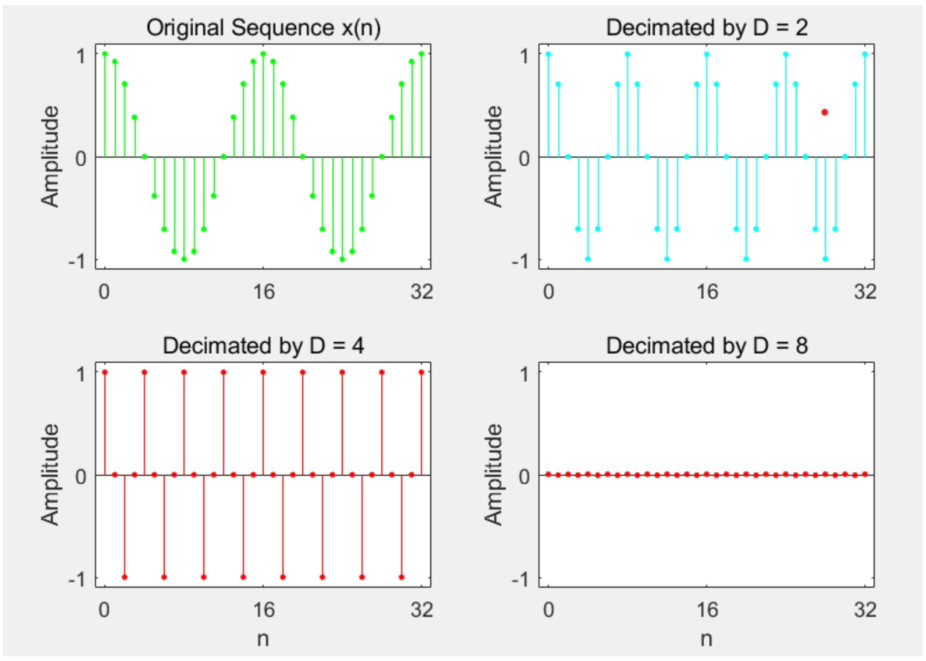
Let $x[n] = \cos(0.125\pi n)$. Generate a large number of samples of $x[n]$ and decimate them using $D = 2, 4$, and 8 to show the results of decimation.

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

(MATLAB Code)

```
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), "r", "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");
```

(Result)



Example 3

Let $I = 2$ and $x[n] = \{1, 2, 3, 4\}$. Verify that the upsampler is time varying.

The upsampled signal $v[m] = \{\,1, 0, 2, 0, 3, 0, 4, 0\,\}$. If we now delay $x[n]$ by one sample, we get $x[n-1] = \{\,0, 1, 2, 3, 4\,\}$. The corresponding upsampled singal is $v_1[n] = \{\,0, 0, 1, 0, 2, 0, 3, 0, 4, 0\,\} = v[m-2]$ and not $v[m-1]$.

[v] = upsample(x, I)

upsamples input array x into output v by inserting $(I-1)$ zeros between input samples. An optional 3rd parameter, “phase” specifies the sample offset, which must be an integer between 0 and $(I-1)$.

(MATLAB Code)

```
x = [1, 2, 3, 4]; v1 = upsample(x, 3)
v2 = upsample(x, 3, 1)
v3 = upsample(x, 3, 2)
```

(Result)

```
v1 =

     1     0     0     2     0     0     3     0     0     4     0     0

v2 =

     0     1     0     0     2     0     0     3     0     0     4     0

v3 =

     0     0     1     0     0     2     0     0     3     0     0     4
```

Example 4

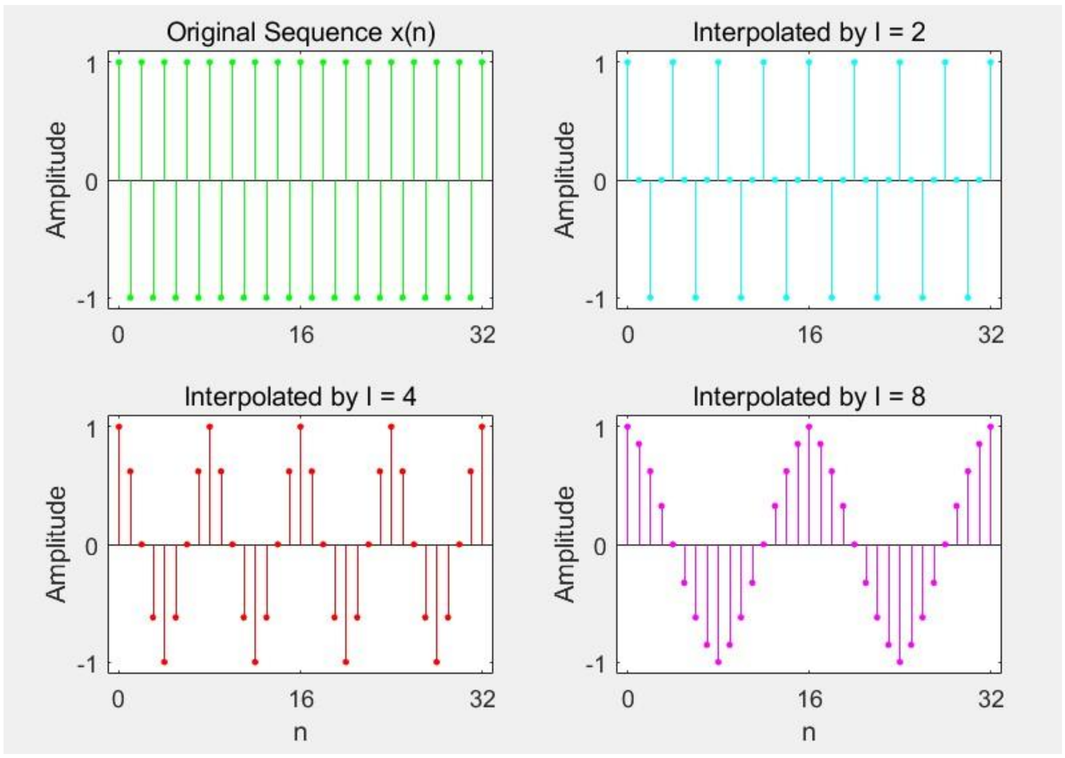
Let $x[n] = \cos(\pi n)$. Generate samples of $x[n]$ and interpolates them using $I = 2$, 4, and 8 to show the results of interpolation.

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.

(MATLAB Code)

```
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");
```

(Result)


Original Sequence x(n)
Interpolated by I = 2
Interpolated by I = 4
Interpolated by I = 8
Amplitude
n
1
0
-1
0
16
32

References

[1] Matlab Official Reference

https://kr.mathworks.com/help/matlab/ref/movegui.html