Lab 5 Sampling and Reconstruction

Objective

- Learn to convert an analog signal to a discrete-time sequence via sampling.
- Be able to reconstruct an analog signal from a discrete-time sequence.
- Understand the conditions when a sampled signal can uniquely represent its analog counterpart.

Content

Sampling

A continuous-time signal can be processed by processing its samples through a discrete-time system. For reconstruction of the continuous-time signal from its discrete-time samples without any error, the signal should be sampled at a sufficient rate that is determined by the sampling theorem.

Nyquist Sampling Theorem

If a signal is band limited and its samples are taken at a sufficient rate, then the samples uniquely specify the signal and the signal can be reconstructed from those samples. This is known as the Nyquist sampling theorem.
When a real signal $\mathrm{x}(t)$ is sampled in the time domain, the sampled signal can be represented as:

$$x_s(t) = x(t)\delta_T(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT)$$

Since the impulse $\delta_T(t)$ is a periodic signal of period T, it can be expressed as a trigonometric Fourier series as follows (Fourier series expansion is discussed in Fourier Analysis).

$$\delta_T(t) = \frac{1}{T}[1 + 2\cos\omega_s t + 2\cos 2\omega_s t + 2\cos 3\omega_s t + \cdots] \qquad \omega_s = \frac{2\pi}{T} = 2\pi f_s$$

Therefor

$$x_s(t) = x(t)\delta_T(t) = \frac{1}{T}[x(t) + 2x(t)\cos\omega_s t + 2\,x(t)\cos 2\omega_s t + 2\,x(t)\cos 3\omega_s t + \cdots]$$

According to the characteristics of Fourier theory, transform $x_s(t)$ in the time domain into $X_S(\omega)$ in frequency domain term by term as listed in Table 1.

Table 1 Fourier Transform of Sampled Signal

Time Domain	Frequency Domain
$\mathrm{x}(t)$	$\mathrm{X}(\omega)$
$2\mathrm{x}(t)\cos\omega_s t$	$\mathrm{X}(\omega+\omega_s) + X(\omega-\omega_s)$

$2x(t)\cos 2\omega_s t$	$X(\omega+2\omega_s)+X(\omega-2\omega_s)$
$2x(t)\cos 3\omega_s t$	$X(\omega+3\omega_s)+X(\omega-3\omega_s)$
…	…

From the table, it is easy to find out that the spectrum $X_S(\omega)$ consists of $X(\omega)$ repeating periodically with period ω_s. Therefor $X_S(\omega)$ can be expressed as follows:

$$X_S(\omega) = \frac{1}{T}\sum_{n=-\infty}^{\infty} X(\omega - n\omega_s)$$

To reconstruct $x(t)$ from $x_s(t)$, it should be possible to recover $X(\omega)$ from $X_S(\omega)$. The recovery is possible if there is no overlap between successive cycles of $X_S(\omega)$. Thus as long as the sampling frequency ω_s is greater than twice the signal bandwidth ω_b, $X_S(\omega)$ will consist of no overlapping repetitions of $X(\omega)$. In this case $x(t)$ can be recovered from its samples $x_s(t)$. $2\omega_b$ is called the Nyquist rate and ω_s must exceed it in order to avoid aliasing. Show in Figure 1.

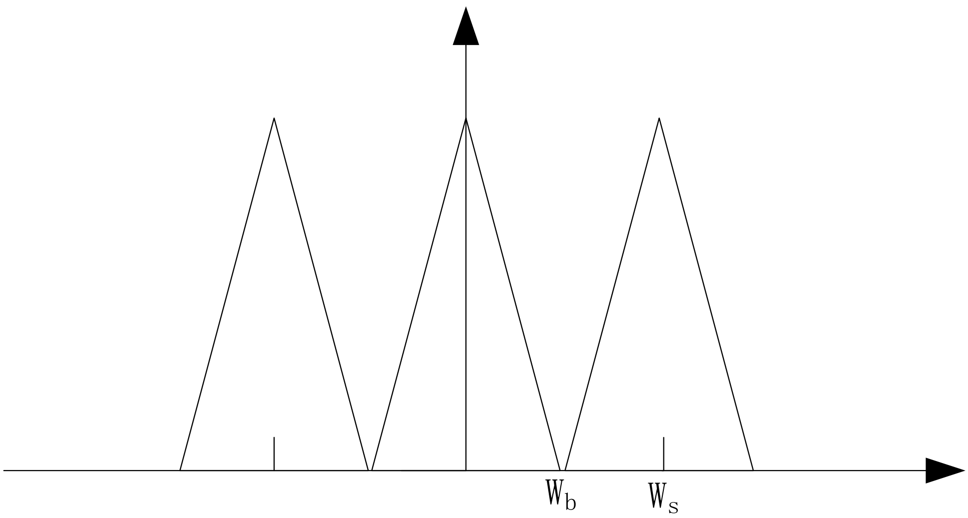


Figure 1 Nyquist frequency

Sampling of Non Band Limited Signal

In most cases, $x(t)$ may not be bandlimited. At this moment an anti-aliasing filter is needed.
An anti-aliasing filter is a filter that is used before a signal sampler, to restrict the bandwidth of a signal to approximately satisfy the sampling theorem. The aim of the filter is to eliminate the frequency components beyond $f_s/_2$ from $x(t)$ before sampling $x(t)$. Usually the relationship between the cutoff frequency of anti-aliasing filter f_c and the sample rate f_s is as follows:

$$f_c = {f_s}/_{2.56}$$

Function **filter(b,a,x)** can be used as an anti-aliasing filter. Here we take the moving-average filter and Butterworth filter as examples. Other kinds of filters will be discussed in Digital Signal Processing in detail.

Moving-average filter

A moving-average filter slides a window of length **windowSize** along the data, computing averages of the data contained in each window. The following difference equation defines a moving-average filter of a vector x:

$$\mathrm{y(n)} = \frac{1}{windowSize}(x(n) + x(n-1) + \cdots + x(n-(windowSize-1)))$$

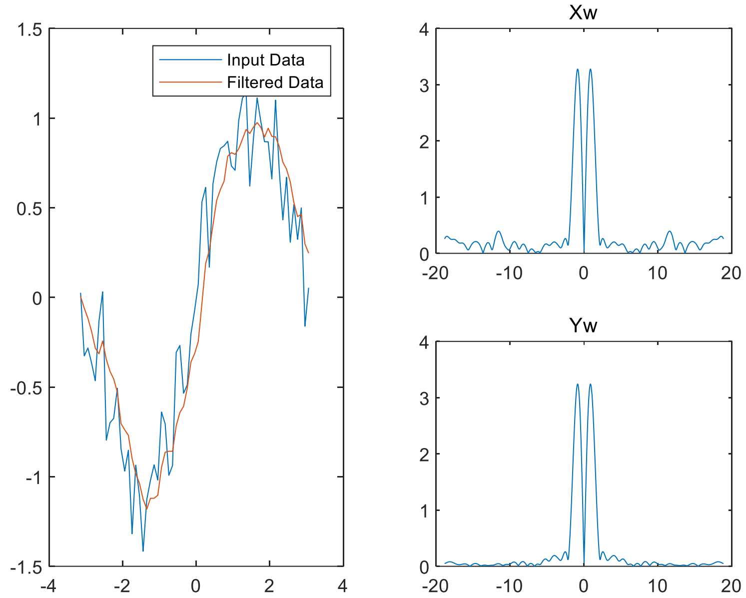
An example of a moving-average filter with windowSize=5 is given below.

```
% signal with noisy
ds = 0.1;
t = -pi:ds:pi;
x = sin(t)+0.25*randn(size(t));

% set the filter
windowSize = 5;
b = (1/windowSize)*ones(1,windowSize);
a = 1;

% show the result
y = filter(b,a,x);
subplot(2,2,[1 3]);
plot(t,x); hold on;
plot(t,y); legend('Input Data','Filtered Data');

N = 300;
w1 = 2*pi*3; k=-N:N; w=k*w1/N;
Xw = abs(ds*x*exp(-1i*t'*w));
Yw = abs(ds*y*exp(-1i*t'*w));
subplot(2,2,2); plot(w,Xw); title('Xw');
subplot(2,2,4); plot(w,Yw); title('Yw');
```



Butterworth filter

Function **butter** is used to design a Butterworth filter. Details are as follows:

[b, a]=butter(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in vectors **b** (numerator) and **a** (denominator). The cutoff frequency Wn must be 0.0<Wn<1.0, with 1.0 corresponding to half the sample rate. That means $w_n = \frac{f_c}{f_s/2}$, f_c is the cutoff frequency of the filter and f_s is the sampling frequency.

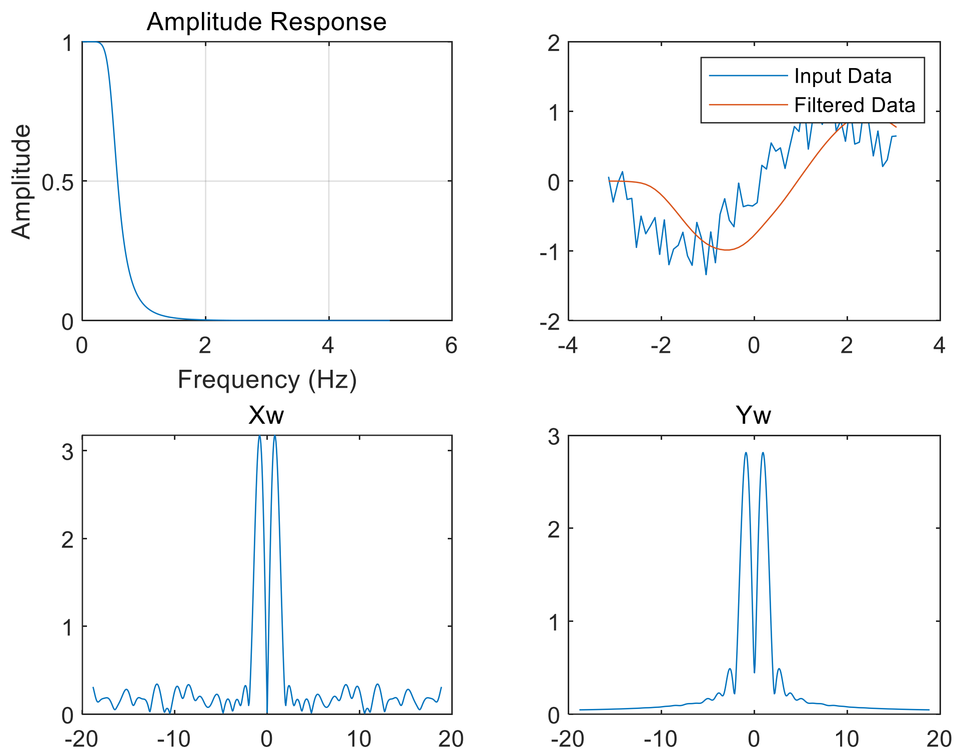
After getting the filter coefficients with function **butter**, filter the input data with function **filter**. Here is an example:

```
% signal with noisy
clear; clf;
ds = 0.1;
fs = 1/ds;     % sample rate
t = -pi:ds:pi;
x = sin(t)+0.25*randn(size(t));  % the signal frequency is 1/(2*pi)
fc = 0.5;  % set cutoff frequency to 0.5Hz

% set the filter
[b a]=butter(4,fc/(fs/2));  % normalize the cutoff frequency with
fs/2
[h,w]=freqz(b,a);
subplot(2,2,1);
plot(w/pi*fs/2,abs(h));  % spectrum of the Butterworth filter
grid;
title('Amplitude Response');
xlabel('Frequency (Hz)'); ylabel('Amplitude');
```

```
% filter the signal and show the result
y = filter(b,a,x);
subplot(2,2,2);
plot(t,x); hold on;
plot(t,y); legend('Input Data','Filtered Data');

N = 300;
w1 = 2*pi*3; k=-N:N; w=k*w1/N;
Xw = abs(ds*x*exp(-1i*t'*w));
Yw = abs(ds*y*exp(-1i*t'*w));
subplot(2,2,3); plot(w,Xw); title('Xw');
subplot(2,2,4); plot(w,Yw); title('Yw');
```



Reconstruction

A band limited signal $x(t)$ can be reconstructed from its samples. To reconstruct the signal, pass the sampled signal through an ideal low pass filter with the bandwidth of ω_c, where ω_c should satisfy: $\omega_b < \omega_c < \omega_s - \omega_b$. The transfer function of the filter is expressed as follows:

$$H(\omega) = \begin{cases} T, & -\omega_c < \omega < \omega_c \\ 0, & otherwise \end{cases}$$

The relationship is displayed in Figure 2.

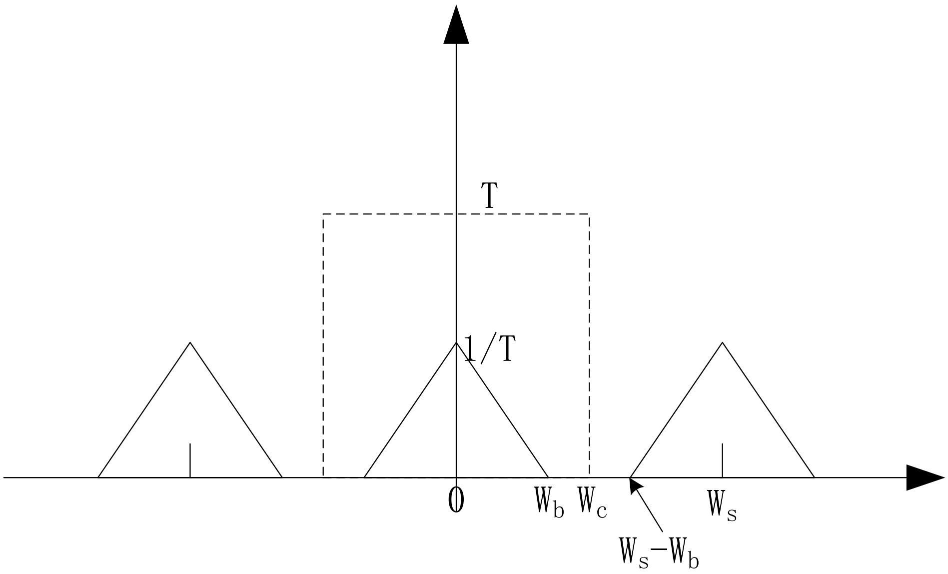


Figure 2 Relationship of $\mathrm{H}(\omega)$ and $X'(\omega)$

So in the frequency domain:

$$X_R(\omega) = X_S(\omega) \cdot H(\omega)$$

Then in time domain:

$$x_r(t) = x_s(t) * h(t)$$

For simplicity, set ω_c as the average of ω_b and ($\omega_s - \omega_b$), that is:

$$\omega_c = \frac{\omega_s}{2} = \frac{\pi}{T_s}$$

Then:

$$\mathrm{h}(t)$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} H(\omega) e^{j\omega t}\, d\omega = \frac{1}{2\pi} \int_{-\pi/T_s}^{\pi/T_s} T_s e^{j\omega t}\, d\omega = \frac{T_s}{2\pi} \cdot \frac{1}{jt} \cdot e^{j\omega t} \Bigg|_{-\pi/T_s}^{\pi/T_s} = \frac{T_s}{\pi t} \sin \frac{\pi t}{T_s} = sinc\left(\frac{t}{T_s}\right)$$

Where

$$\mathrm{sinc}(\mu) = \sin(\pi\mu)/_{\pi\mu}.$$

So

$$x_r(t) = x_s(t) * h(t) = \left(\sum_{n=-\infty}^{\infty} x(nT_s)\delta(t - nT_s) \right) * h(t)$$

$$= \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x(nT_s)\delta(\tau - nT_s) h(t - \tau)\, d\tau$$

$$= \sum_{n=-\infty}^{\infty} x(nT_s) h(t - nT_s) = \sum_{n=-\infty}^{\infty} x(nT_s) sinc\left(\frac{t - nT_s}{T_s}\right)$$

The interpolation formula can be verified at $\mathrm{t} = \mathrm{k}\Delta\mathrm{s}$:

$$x_r(k\Delta s) = \sum_{n=-\infty}^{\infty} x(nT_s)\,\mathrm{sinc}\frac{(k\Delta s - nT_s)}{T_s}$$

$$\mathrm{sinc}(k-n) = \frac{\sin\big((k-n)\pi\big)}{(k-n)\pi}$$

$$= \begin{cases} 0, & \mathrm{k} \neq \mathrm{n} \\ \lim\limits_{m\to 0}\dfrac{\sin(m\pi)}{m\pi} = \lim\limits_{m\to 0}\dfrac{\frac{d\sin(m\pi)}{dm}}{\frac{dm\pi}{dm}} = \lim\limits_{m\to 0}\dfrac{\pi\cos(m\pi)}{\pi} = 1, & \mathrm{k} = \mathrm{n} \end{cases}$$

So $x_r(k\Delta s) = x(nT_s)$, which aligns with $x_r(t) = x(t)$.

An example of sampling and reconstruction is given below.

Eg: x_samples is the sampled signal, ts is the sample time and ds is the sample interval.

```
dr = 0.5;
tr = 0:dr:20;

% Method one: Follow the formula
x_recon = zeros(length(tr),1);
for k = 1:length(tr)
    for n=1:length(x_samples)
        x_recon(k) = x_recon(k)+
x_samples(n)*sinc(((k-1)*dr-(n-1)*ds)/ds);
    end
    plot(tr,x_recon);
end

% Method two: Calculate sample by sample
x_recon = 0;
for n=1:length(x_samples)
    x_recon = x_recon+x_samples(n)*sinc(tr-ts(n));
end
plot(tr,x_recon)
```