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 for reconstruction of the sampled signal.

Content

Sampling

Sometimes we need to convert continuous time signals into discrete time signals before processing. To ensure that the sampled signal can represent the original signal, the original signal should be sampled at a sufficient rate determined by the sampling theorem. In this case, we can reconstruct the original signal from the sampled signal.

Nyquist Sampling Theorem

If the frequency band of the signal is limited and its samples are taken at a sufficient rate, 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:

$$x_s(t) = x(t)\delta_{T_s}(t)$$

Since the impulse $\delta_{T_s}(t)$ 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).

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

Therefor

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

Convert $x_s(t)$ to $X_S(\omega)$ by Fourier transform. According to the characteristics of Fourier theory, we get the result for each term listed in Table 1.

Table 1 Fourier Transform of Sampled Signal

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

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

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:

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

To reconstruct $x(t)$ from $x_s(t)$, it is necessary to be able to get $X(\omega)$ from $X_S(\omega)$, which requires that there is no overlap between consecutive $X_S(\omega)$. Figure 1 shows $X_S(\omega)$, from which it is easy to find that as long as the sampling frequency ω_s is greater than twice the signal bandwidth ω_b, $X_S(\omega)$ will not contain any overlap. 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.

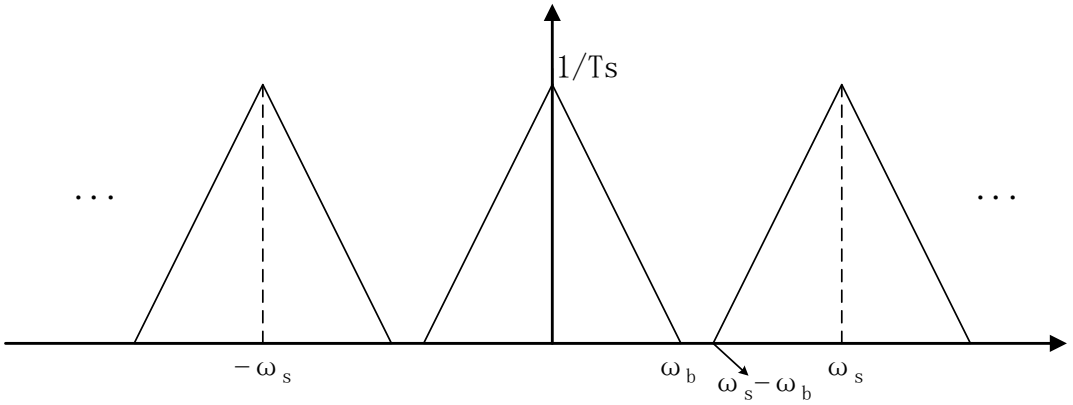


Figure 1 Nyquist Theorem

FYI:

Sometimes you may see both Nyquist rate and Nyquist frequency. The difference between them are briefly described below. Just for your reference.

Nyquist rate is the lower bound of the sampling frequency that satisfies the Nyquist sampling theorem. It is twice the bandwidth or maximum component frequency of the signal.

Nyquist frequency is half the sampling frequency of a discrete signal processing system. It is sometimes called the folding frequency, or the cut-off frequency of a sampling system.

Sampling of Non Band Limited Signal

In most cases, $x(t)$ may not be bandlimited. In this case, an anti-aliasing filter is required.

An anti-aliasing filter is a filter used before a signal sampler, and its purpose is to limit the bandwidth of the signal to satisfy the sampling theorem. Assuming that the main frequency components of the signal are distributed in the bandwidth ω_b, before sampling, a filter should be used to remove the frequency components beyond ω_b from $x(t)$. Usually we take f_c, the cutoff frequency of the filter, a little bigger than ω_b.

Function **filter(b,a,x)** is used to implement filtering, where parameters **a** and **b** describe the characteristics of the system just as we discussed in the former chapters, and parameter **x** is the signal to be filtered. There are many kinds of filters. Here we take Butterworth filter as an example.

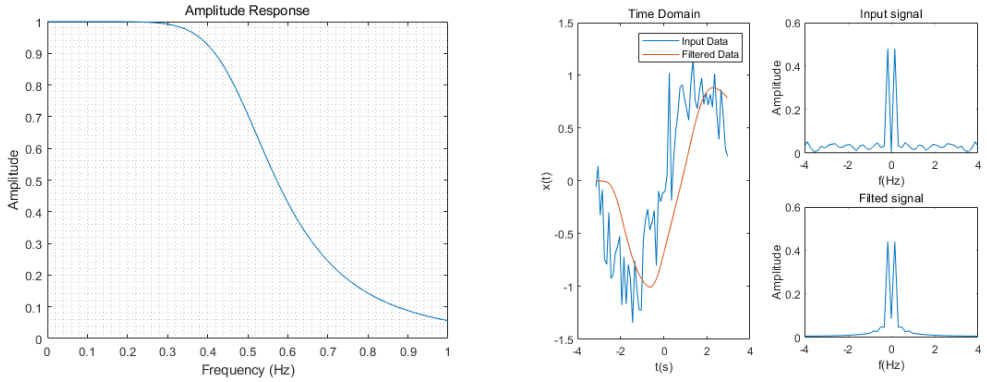
As for other kinds of filters and the design of the filters will be discussed in Digital Signal Processing in detail.

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:

```
clf; clear;
% signal with noisy
dt = 0.1; fs = 1/dt;
t = -pi:dt:pi-dt;
x = sin(t)+0.25*randn(size(t));
fc = 0.5; % cutoff frequency

% set the filter
[b,a] = butter(4,fc/(fs/2));
[H w] = freqz(b,a);
plot((w/pi)*(fs/2),abs(H)); % the unit of the horizontal axis is *pi
rad/sample
xlim([0 1]);grid minor;title('Amplitude Response');
xlabel('Frequency (Hz)'); ylabel('Amplitude');

% show the result in time domain
y = filter(b,a,x);
subplot(2,2,[1 3]);
plot(t,x); hold on;
plot(t,y); legend('Input Data','Filtered Data');
xlabel('t(s)');ylabel("x(t)");title('Time Domain')

% show the result in frequency domain
N = length(x);
f = (-N/2:N/2-1)*fs/N;

Fx = fftshift(fft(x))/N;
subplot(2,2,2); plot(f,abs(Fx));axis([-4 4 0 0.6]);
xlabel('f(Hz)');ylabel('Amplitude');title('Input signal');

Fy = fftshift(fft(y))/N;
```

```
subplot(2,2,4); plot(f,abs(Fy));axis([-4 4 0 0.6]);
xlabel('f(Hz)');ylabel('Amplitude');title('Filted signal');
```



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:

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

The relationship is displayed in Figure 2.

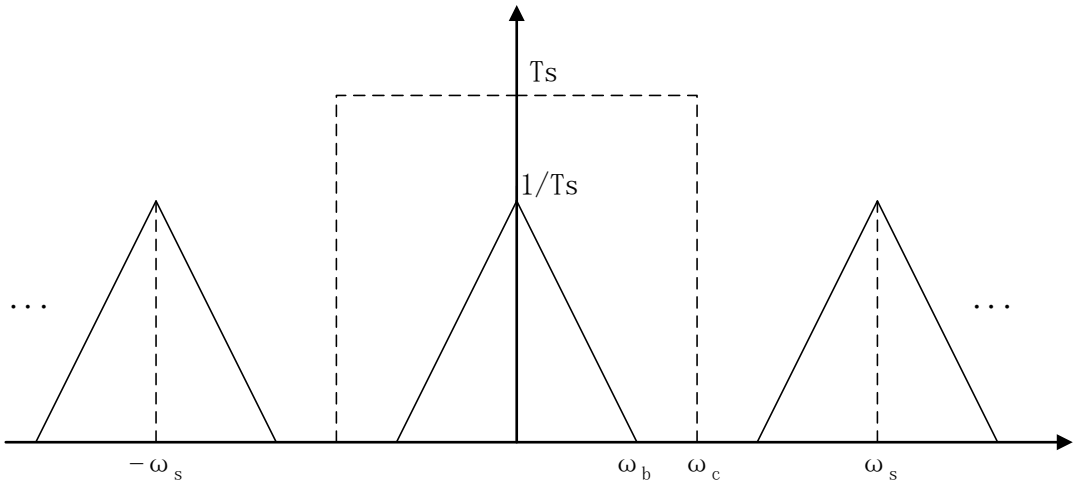


Figure 2 Relationship of $H(\omega)$ and $X'(\omega)$

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:

$h(t)$

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

Where

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

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 $t = k\Delta t$:

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

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

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

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

An example of sampling and reconstruction is given below.

Eg: x_samples is the sampled signal, t_samples is the sample time and Ts is the sample interval.

```
clear;clf;
% original signal
dt = 0.1; t = 0:dt:20;
F1 = 0.1; F2 = 0.2;
x = sin(2*pi*F1*t)+sin(2*pi*F2*t);
x_size = length(x);

% sampling
s = 10;    % sample rate = 1/ (dt*s)
Ts = dt*s;
x_samples = x(1:s:x_size); % gets samples of x.
t_samples = (0:length(x_samples)-1)*Ts;
subplot(3,1,1);plot(t,x);hold on;stem(t_samples,x_samples);
```

```
title('Original/Sampled signal');xlabel('t(s)');ylabel('x(t)/x(n)');
legend('Original signal','Sampled signal');

% Reconstruction Method 1: Follow the formula
x_recon=zeros(length(t),1);
for k=1:length(t)
   for n=1:length(x_samples)
       x_recon(k)=x_recon(k)+x_samples(n)*sinc(((k-1)*dt-(n-1)*Ts)/Ts);
   end
end
subplot(3,1,2);plot(t,x_recon);
title('Reconstruction by time');xlabel('t(s)');ylabel('xr(t)');

% Reconstruction Method 2: Calculate sample by sample
x_recon = 0;
for n=1:length(x_samples)
   x_recon = x_recon+x_samples(n)*sinc((t-t_samples(n))/Ts);
end
subplot(3,1,3);plot(t,x_recon)
title('Reconstruction by sample');xlabel('t(s)');ylabel('xr(t)');
```