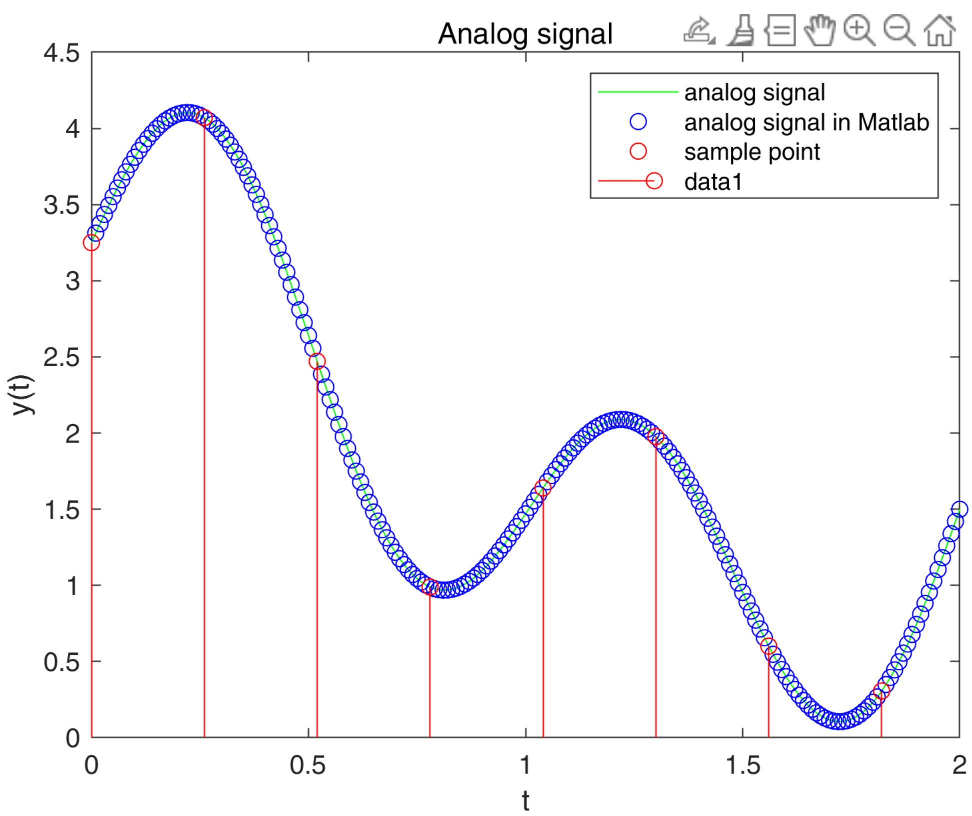
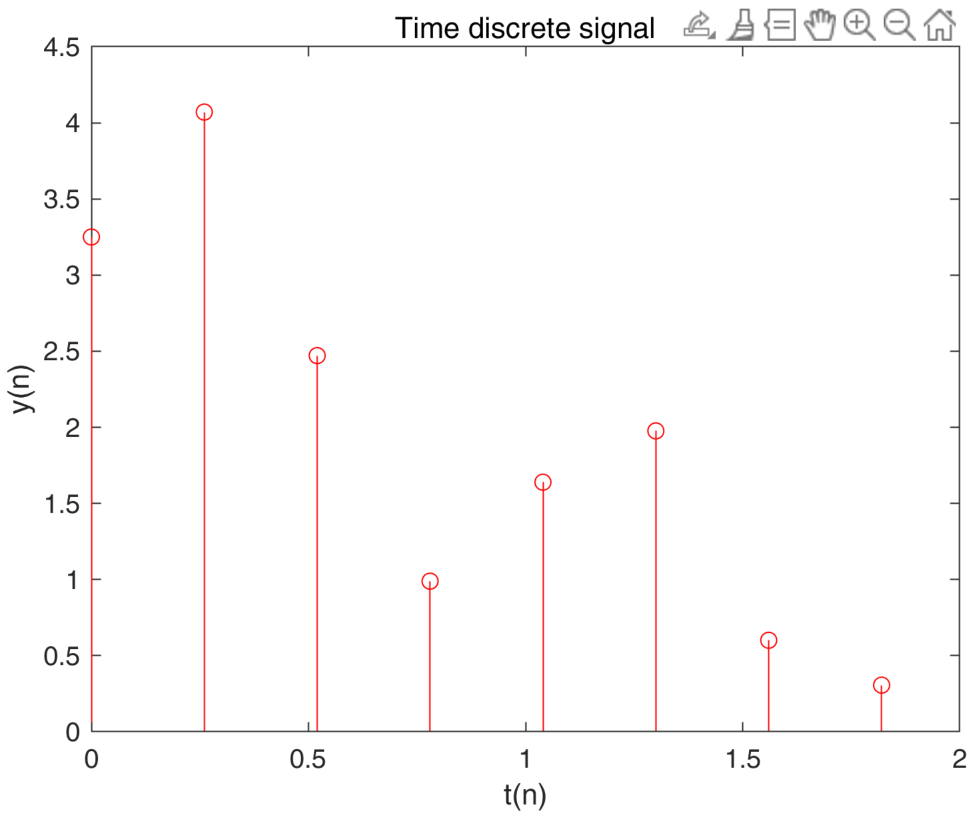
Sampling and Reconstruction

```
clf;clear;
dt = 0.01;
t = 0:dt:2;
x = sin(2*pi*t)+ 0.75*cos(0.75*pi*t)+ 0.5*cos(0.5*pi*t)+2;

sample_interval=26;
x_sample = x(1:sample_interval:end);
t_sample = t(1:sample_interval:end);

plot(t,x,'g');legend('analog signal');
xlabel('t');ylabel('y(t)');title('Analog signal');hold on;axis([0 2 0 4.5]);
waitforbuttonpress();
plot(t,x,'ob'); legend('analog signal','analog signal in Matlab')
waitforbuttonpress();
plot(t(1:sample_interval:end),x(1:sample_interval:end),'or');
legend('analog signal','analog signal in Matlab','sample point');
waitforbuttonpress();
stem(t_sample,x_sample,'r'); hold off;
```




```
waitforbuttonpress(); stem(t_sample,x_sample,'r'); axis([0 2 0 4.5]);
xlabel('t(n)');ylabel('y(n)');title('Time discrete signal')
```

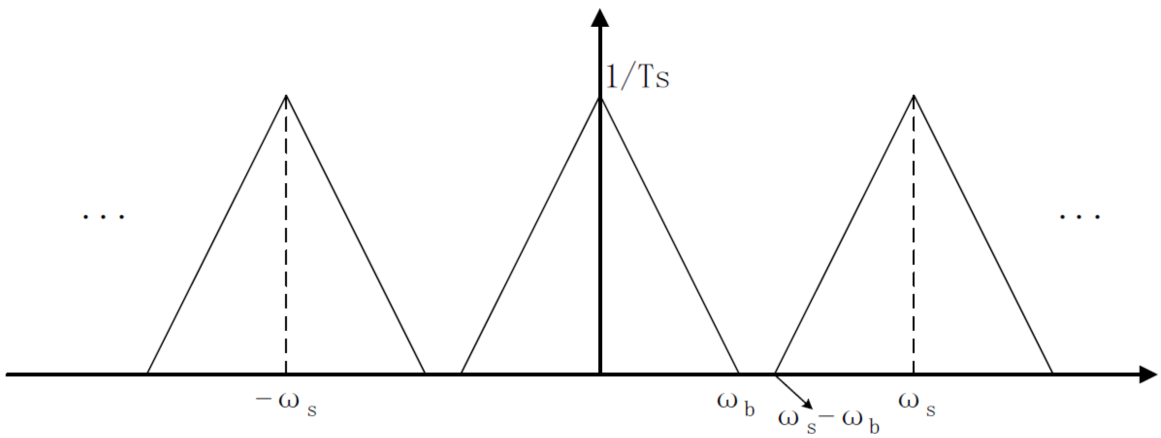


$$x_s(t) = x(t) \cdot \delta_{T_s} = \sum_{n=-\infty}^{\infty} x(nT_s) \cdot \delta(t - nT_s)$$

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

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

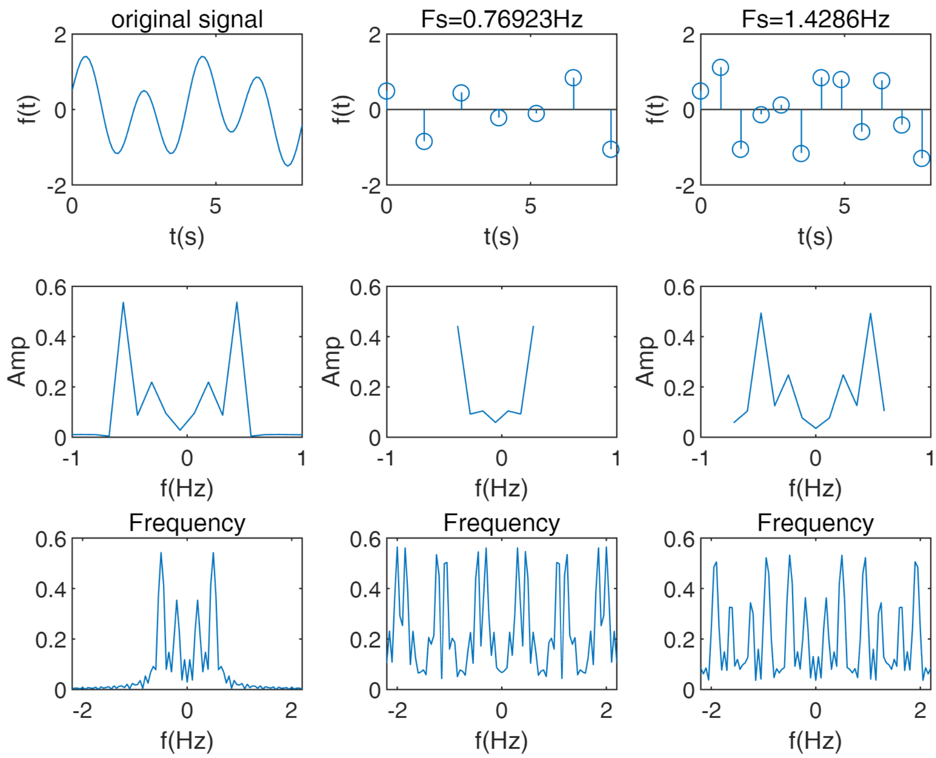
Time Domain	Frequency Domain
$x(t)$	$X(\omega)$
$2x(t)\cos\omega_s t$	$X(\omega + \omega_s) + X(\omega - \omega_s)$
$2x(t)\cos 2\omega_s t$	$X(\omega + 2\omega_s) + X(\omega - 2\omega_s)$
$2x(t)\cos 3\omega_s t$	$X(\omega + 3\omega_s) + X(\omega - 3\omega_s)$
...	...



Sample rate

```
clear; clf;
dt = 0.1;
t = 0:dt:8;
freq = 0.5;
sig = sin(2*pi*freq*t)+0.5*cos(2*pi*0.4*freq*t);

Fs = [1/dt,1.5*freq,2.56*freq]; % sample rate. (1/dt refers to the original signal)
ds = 1./Fs;
sample_interval = floor(ds/dt); % the number of interval points

for i = 1:length(sample_interval)
    % sampled signal
    f_sample = sig(1:sample_interval(i):end);   % data after sampling (ft)
    t_sample = t(1:sample_interval(i):end);     % timeline after sampling (t)
    Ts = sample_interval(i)*dt;                 % acture time interval after sampling (dt)
    sigLen = length(f_sample);                  % data length after sampling
    N = sigLen;
%     f_sample = [f_sample,zeros(1,100)];
%     N = length(f_sample);

    subplot(3,length(sample_interval),i);
    if i==1
        plot(t_sample,f_sample); xlabel('t(s)'); ylabel('f(t)');
        title('original signal');axis([0 t(end) -2 2]);
    else
        stem(t_sample,f_sample); xlabel('t(s)'); ylabel('f(t)');
        title(['Fs=' num2str(1/dt/sample_interval(i)) 'Hz']);axis([0 t(end) -2 2]);
    end

    % fft
    Fs = 1/Ts; df = Fs/N;
    F = fftshift(fft(f_sample))/sigLen;
    f = (-N/2:N/2-1)*df;
    subplot(3,length(sample_interval),i+length(sample_interval));
    plot(f,abs(F)); axis([-1 1 0 0.6]);xlabel('f(Hz)'); ylabel('Amp');
```

```
    % the spectrum of sampled signal
    N = 50;     % set the frequency points number
    w1 = 2*pi*(5*freq); k = -N:N; w = k*w1/N;
    Fw = f_sample*exp(-1j*t_sample'*w)*Ts;
    Fw = abs(Fw)/(t(end)-t(1));
    subplot(3,length(sample_interval),i+length(sample_interval)+3);
    plot(w/(2*pi),Fw); title('Frequency'); xlabel('f(Hz)');xlim([-2.2 2.2])
end
```



Anti-aliasing Filter

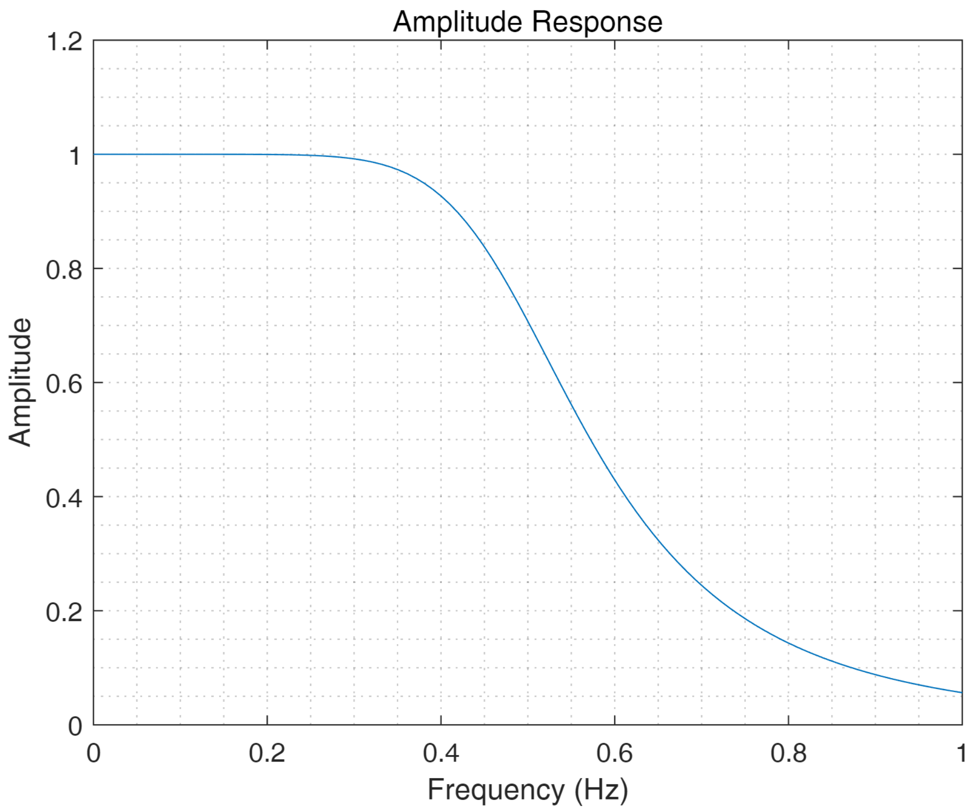
Battworth filter

y = filter(b,a,x);

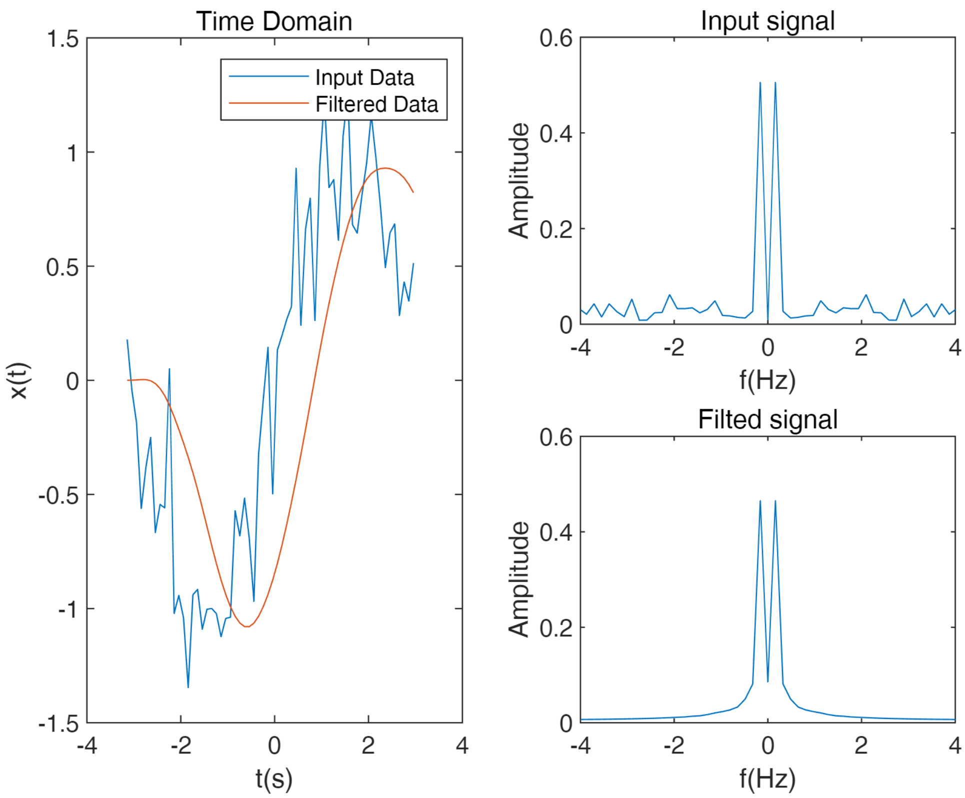
[b,a] = butter(n, Wn); Wn=fc/(fs/2);

```
% signal with noisy
clf;clear;
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(N, fc/(Fs/2))
[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');
```

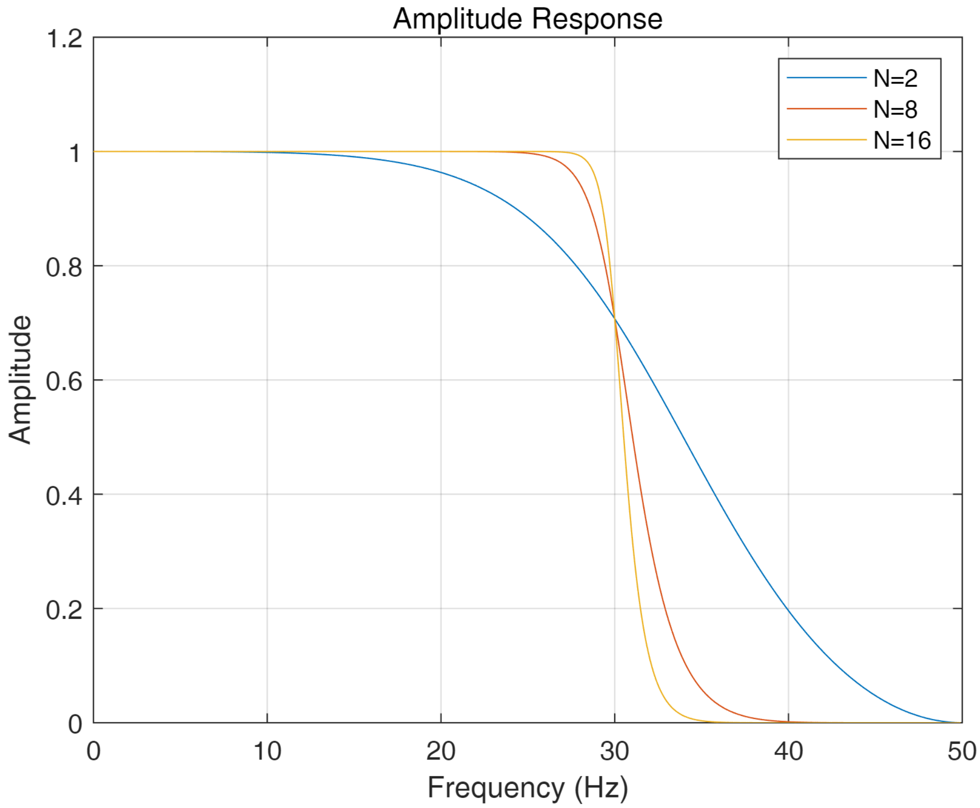



```
% show the result in time domain
y = filter(b,a,x);
subplot(2,2,[1 3]);
plot(t,x); hold on;
plot(t,y); legend('Input Data','Filtered Data');
xlabel('t(s)');ylabel("x(t)");title('Time Domain')

% fft
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');
```

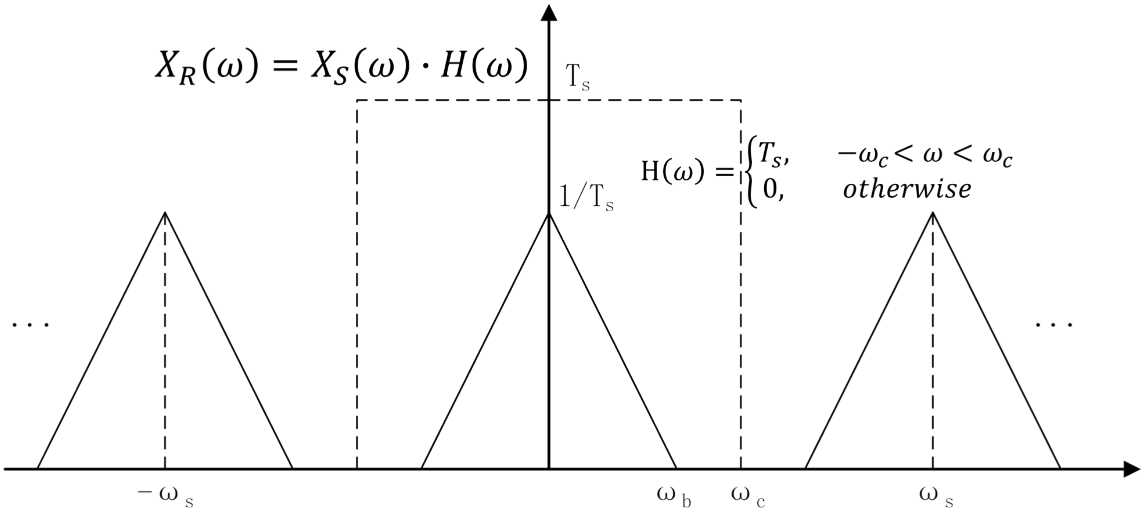


The effect of N and fc

```
clear;clf;
Fs=100;
fc=30; %Cutoff frequency:30Hz
N = [2,8,16];
for i = 1:length(N)
    [b,a]=butter(N(i),fc/(Fs/2));
    [h,w]=freqz(b,a);
    plot(w/pi*Fs/2,abs(h)); grid;
    hold on;
end
legend('N=2','N=8','N=16');
title('Amplitude Response');
xlabel('Frequency (Hz)'); ylabel('Amplitude');
```



Reconstruction



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

$$\mathrm{h}(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(\omega)e^{j\omega t}d\omega = \frac{1}{2\pi}\int_{-\pi/T_s}^{\pi/T_s} T_s e^{j\omega t}\, d\omega = sinc\left(\frac{t}{T_s}\right)$$

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

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

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

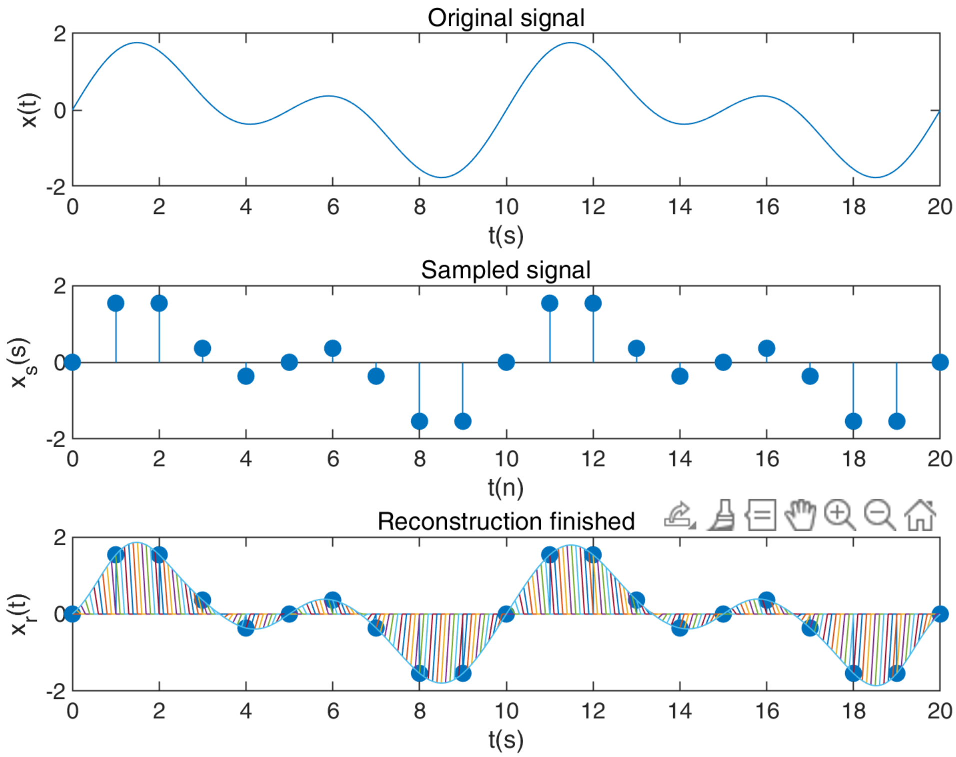
$$x_r(k\Delta t) = \sum_{n=-\infty}^{\infty} x(nT_s)\mathrm{sinc}\frac{(k\Delta t - nT_s)}{T_s}$$

Method 1 Follow the formula

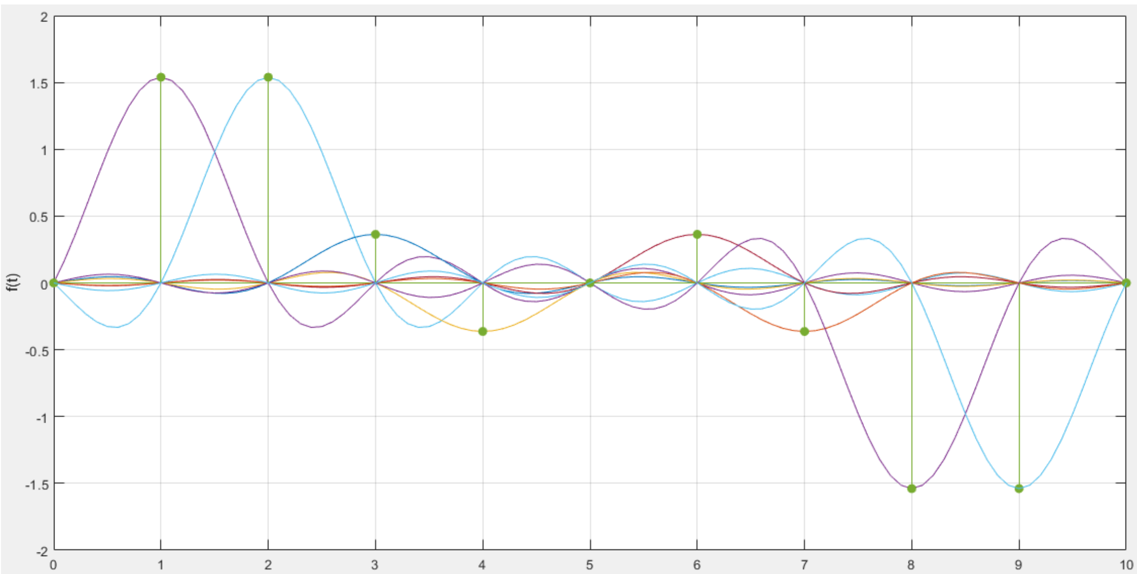
$$x_r(k\Delta t) = \sum_{n=-\infty}^{\infty} x(nT_s)\mathrm{sinc}\frac{(k\Delta t - nT_s)}{T_s}$$

```
% Sampling & reconstruction
% Create  'analog' signal
clear;clf;
set(gcf,'Visible','on');
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);
```

```
%plotting the original analog signal
% figure(1);
subplot(3,1,1);plot(t,x);
title('Original signal');xlabel('t(s)');ylabel('x(t)');

%sampling
sample_interval = 10;   % number of sampling intervals
                        % it should  be caldulated by sampling frequency
Ts = dt*sample_interval;
x_samples = x(1:sample_interval:x_size);
t_samples = (0:length(x_samples)-1)*Ts;
subplot(3,1,2);stem(t_samples,x_samples,'filled');
title('Sampled signal');xlabel('t(n)');ylabel('x_s(s)');

%reconstruction
subplot(3,1,3);
stem(t_samples,x_samples,'filled');
title('Sample by sample reconstruction');xlabel('t(s)');ylabel('x_r(t)');hold on;

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
    plot(t,x_recon);
    pause(0.01);
end
title('Reconstruction finished');
hold off;
```



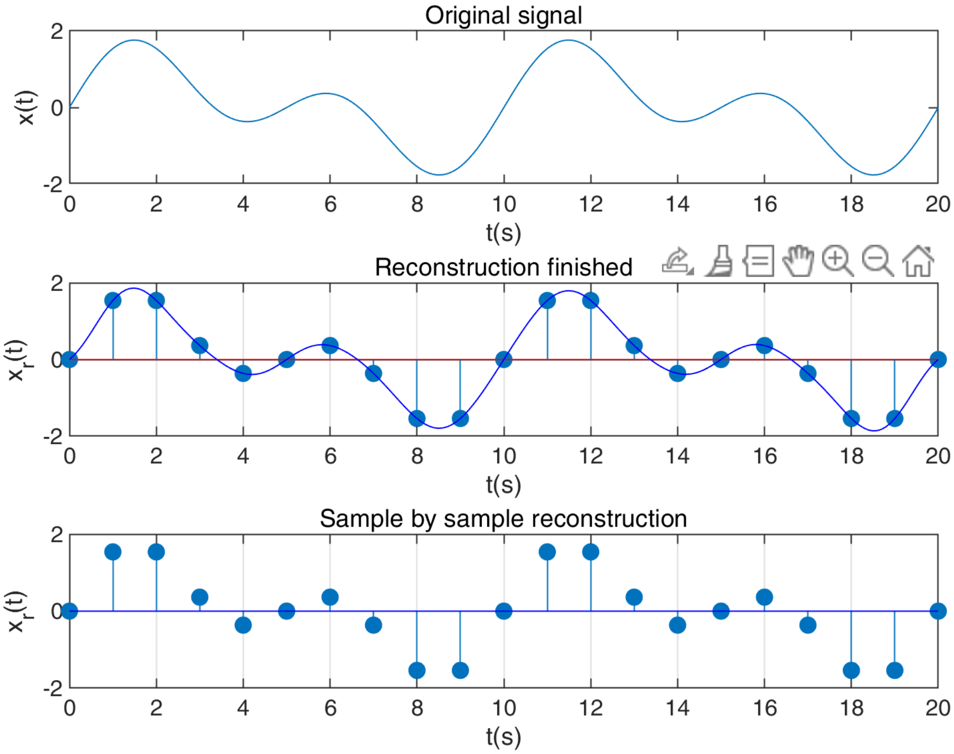
Method 2 Calculate sample by sample



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

```
% Sampling & reconstruction
% Creating  'analog' signal
clear;clf;
set(gcf,'Visible','on');
```

```
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);

% plotting the original signal
figure(1);
subplot(3,1,1);plot(t,x);
title('Original signal');xlabel('t(s)');ylabel('x(t)');
% sampling
sample_interval = 10;
Ts = dt*sample_interval;
x_samples = x(1:sample_interval:x_size);
t_samples = (0:length(x_samples)-1)*Ts;
subplot(3,1,2);stem(t_samples,x_samples,'filled');
title('Sampled signal');xlabel('t(n)');ylabel('x_s(t)');

% reconstruction
x_recon = 0;
subplot(3,1,3);
for n=1:length(x_samples)
    stem(t_samples,x_samples,'filled');
    if n==length(x_samples)
        title('Reconstruction finished');
    else
        title('Sample by sample reconstruction');
    end
    grid on; axis([0 20 -2 2]);

    x_recon = x_recon+x_samples(n)*sinc((t-t_samples(n))/Ts);
    hold on;
    plot(t,x_samples(n)*sinc((t-t_samples(n))/Ts),'r')
    plot(t,x_recon,'b');xlabel('t(s)');ylabel('x_r(t)')
    hold off;

    waitforbuttonpress
end
```


Original signal
x(t)
t(s)
Reconstruction finished
x_r(t)
t(s)
Sample by sample reconstruction
x_r(t)
t(s)