Sampling and Reconstruction

Riley Kenyon

March 14th, 2019

Abstract

This report describes the Shannon Sampling Theorem and how its properties can be applied to the Nyquist frequency using a sample signal. Signal aliasing can be attributed to sampling and reconstruction, which will be proven with an example signal in combination with different filters for convolution.

1 Introduction

Due to advancing technologies, electronics have advanced to the point where computers can control systems using digital methods. Digital processing is necessary due to a handful of components such as analog-to-digital converters and the converse. Although that equipment plays a major role in phase lag and the resolution of a signal, a lot of adjustment in the theory behind a digital controller comes from discrete sampling of an input.

1.1 Sampling and the Comb Function

Take a continuous function $x(t)$ for example, the sampling rate of this system is 'infinite'. In order to do digital processing and real-time digital control, the continuous function is sampled at an interval T_s, or the time between adjacent sampled points $x[k]$ and $x[k-1]$. To represent the sampled data as a continuous function, the comb function

$$\delta_{T_s}(t) = \sum_{k=-\infty}^{\infty} \delta(t - kT_s), \tag{1}$$

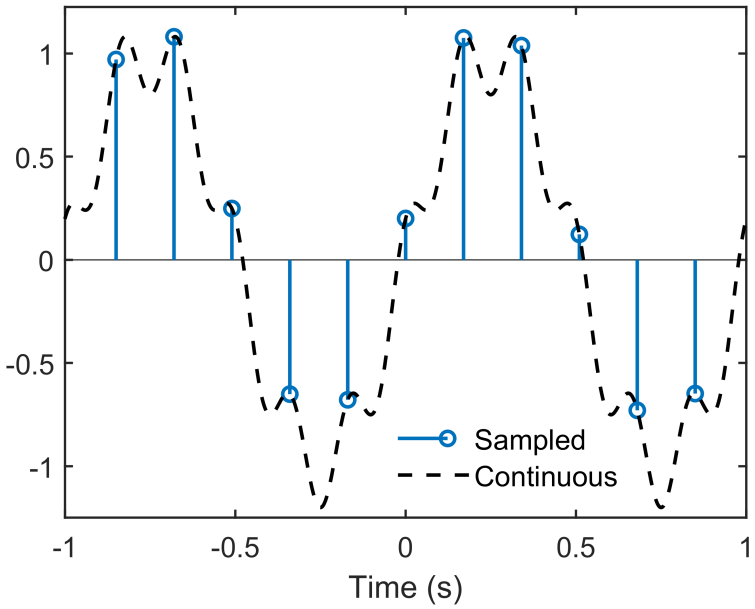

Figure 1: A continuous function being sampled at discrete intervals $T_s = 0.17$ s, the sampling property of the comb function allows for the sampling.

is used to define the relationship between indexed data and the continuous representation of the sample. The sampling property of the dirac-delta function establishes this relationship because it evaluates a function $f(t)$ at the offset of the delta function. For example, at a delta function offset of a the function will be evaluated $f(a)$.

$$\int_{-\infty}^{\infty} f(t)\delta(t-a)dt = f(a) \tag{2}$$

The governing equation that describes the sampling signal $x^*(t)$ as it relates to the continuous function $x(t)$ can be shown to be

$$x^*(t) = x(t) * \delta_{T_s}(t) = \sum_{k=-\infty}^{\infty} x(t)\delta(t-kT_s), \tag{3}$$

where the delta functions are spaced apart at an interval of T_s. Note that the operation occurring with $x(t)$ and δ_{T_s} is their convolution.The equation is visualized in Figure 1 with an example continuous function x(t).

1.2 Frequency Representation

The frequency representation of discrete signals is where things get interesting. A function that is discrete in the time domain is periodic in the frequency domain. This can be shown by looking at the Fourier Series of the signal $x^*(t)$. First, the Fourier Transform of the comb function is defined as

$$\sum_{k=-\infty}^{\infty} \delta(t - kT_s) = \sum_{N=-\infty}^{\infty} C_N \cdot e^{j(\frac{2\pi}{T_s}N)t}, \tag{4}$$

where C_N are the Fourier coefficients which are derived by the synthesis equation,

$$C_N = \frac{1}{T_s} \int_{-T/2}^{T/2} \sum_{k=-\infty}^{\infty} \delta(t - kT_s) \cdot e^{j(\frac{2\pi}{T_s}N)t} = \frac{1}{T_s}. \tag{5}$$

Evaluating this over one period eliminates the summation of k because the only element that exist within that range is $k = 0$. Using the sampling property of the delta function(evaluating the exponential at a value of $t = 0$) the Fourier coefficients simplify to $C_N = 1/T_s$. The next step is to take the Laplace transform of the function $x^*(t)$, by definition the Laplace Transform is equal to the following relation

$$\mathscr{L}\{x^*(t)\} = X^*(s) = \int_{-\infty}^{\infty} x^*(\tau) \cdot e^{-s\tau} d\tau. \tag{6}$$

By plugging in the definition of $x^*(t)$, with the value of the comb function obtained in equation 4 the relation can be simplified to

$$\mathscr{L}\{x^*(t)\} = \frac{1}{T_s} \sum_{N=-\infty}^{\infty} \int_{-\infty}^{\infty} x(\tau) \cdot e^{-(s - jN\omega_s)\tau} d\tau, \tag{7}$$

where $\omega_s = \frac{2\pi}{T_s}$ is the sampling rate. Notice that the form is the same as equation 6 with $s = s - jN\omega_s$ meaning the final representation evaluated at $s = j\omega$ can be displayed as

$$X^*(j\omega) = \frac{1}{T_s} \sum_{N=-\infty}^{\infty} X(j(\omega - N\omega_s)). \tag{8}$$

This gives validity to the statement that a signal which is discrete in time is periodic in frequency (the continuous Fourier Transform is picked up and shifted to the left and right by ω_s. Another way to look at is is the function has a period of ω_s.

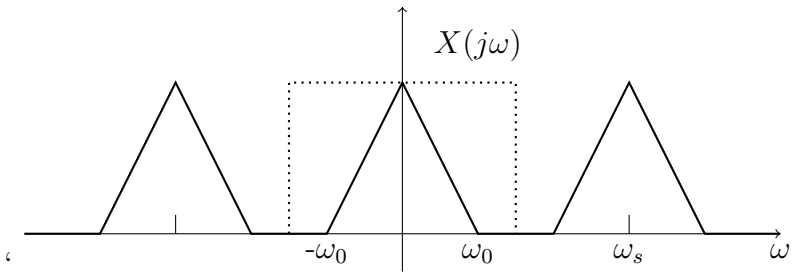

Figure 2: An example Fourier Transform with max frequency content ω_0, and a low-pass filter with cutoff frequency of $\omega_s/2$.

1.3 Reconstruction and Aliasing

The relation obtained in equation 8 also provides intuition to reconstruction and aliasing. The frequency content contained by signal $x(t)$ is bounded by ω_0, the max frequency content. This is visualized with the two scenarios, in Figure 2 the sampling frequency is greater than $2\omega_0$ and Figure 3 where the sampling frequency is less than $2\omega_0$ and aliasing occurs.

After sampling a signal, the points need to be reconstructed to create a function. In order to make the sampled points appear like a continuous signal, the frequency content needs to be passed through a low-pass filter. Figure 2 shows that the frequency content remaining within the low-pass filter is identical to the initial function's frequency content. Under the constraint that $\omega_s > 2\omega_0$, the reconstructed signal $x_R(t)$ will be the same as $x(t)$ when a perfect low-pass filter is used. However, if the sampling rate is less than twice the max frequency content or a non-ideal filter is used to process the sampled signal - aliasing can occur. The metric used to quantify this relation is know as the Nyquist frequency, defined as

$$\omega_n = \frac{\omega_s}{2} > \omega_0. \tag{9}$$

Looking at the first of the possible problems, as seen in Figure 3 the overlapping frequency content will superimpose and create the profile of a different signal, even if an ideal filter is used. Upon reconstruction the signal will be different, or an 'alias'.

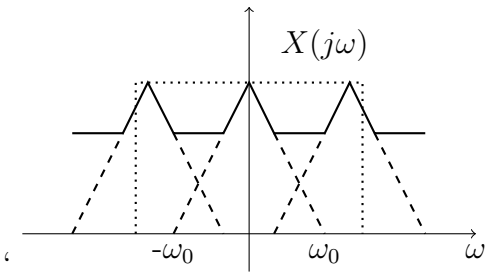

Figure 3: An example Fourier Transform with max frequency content ω_0, and a overlapping frequency signatures due to a poor sampling rate ω_s.

2 Example Signal

To exemplify the theory mentioned in the previous section, an example signal will be sampled at several frequencies and then reconstructed using varying types of filters. The function used throughout the process is

$$x(t) = sin(2\pi t) + 0.2cos(12\pi t), \tag{10}$$

where the frequency content of the signal is discrete, with the max frequency $\omega_0 = 12\pi$. The other frequency occurs at $\omega = 2\pi$. The frequency content can be used later to predict an alias at a specific sampling frequency.

2.1 Building the signal

In order to do this, we will be constructing a signal in Matlab. Due to there not being continuous signals available for simulation, the step-size between points will be minimized to where the signal can be approximated as continuous. The step size used in this account is $DT = 0.0001$, spanning the range of $-20 < t < 20$. At this resolution, the computational needs are not excessive while still achieving the continuous appearance of the function.

2.2 Discretizing the signal

Another work around is performing the numerical convolution needed to reconstruct the sampled signal. The pulse response of the low pass filter is used in convolution with the sampled signal $x^*(t)$, and is created at the same

time interval as the pseudo-continuous function created in Matlab describing equation 10. A caveat to still having discrete points is to make the sampled signal a vector of the same length as the pseudo-continuous function, but with zeros at any point that does not coincide with the time-increment that was sampled. The ramifications of this is that instead of specifying an interval, the time between points will be a multiple of the initial frequency. For example sampling at a $T_s = 0.017$ requires a $N_s = 170$ because of the initial $DT = 0.0001$, and a $T_s = 0.17$ requires a $N_s = 1700$.

3 Reconstruction

In the ideal reconstruction of the signal $x_R(t)$, the function will identically represent the sampled function $x(t)$. Because a discrete signal is periodic in the frequency domain, the frequency content will be cropped when the convolution is performed with the signal and a low-pass filter. The content within the filter will be preserved while the rest of the periodic signal will be chopped, leaving the frequency profile of a continuous signal. The ideal low pass filter (ILP) can be used for an exact result. However, this filter, as seen in the next section is non-causal and not realizable in a physical sense. The alternatives are different types of low pass filters: zero-order hold, first-order hold (also non-causal), and predictive first-order hold. The first order hold is also non-causal but is not infinite, so it can be implemented after data have been collected. These filters are implemented with convolution here, but likely the zero-order hold suffices in real-time control and updates its value once the the next data point is received.

3.1 Ideal Low-Pass Filter (ILP)

The ideal low-pass filter as seen in Figure 4, is equal to

$$h(t) = sinc(\frac{\omega_s t}{2}) = \frac{sin(\omega_s t/2)}{\omega_s t/2}. \tag{11}$$

This is the result of the inverse Fourier Transform of the top-hat needed in the frequency domain. The cutoff of this filter needs to be at a frequency that is greater than ω_0 but less than $\omega_s - \omega_0$. The default value is $\omega_s/2$ as seen in equation 11.

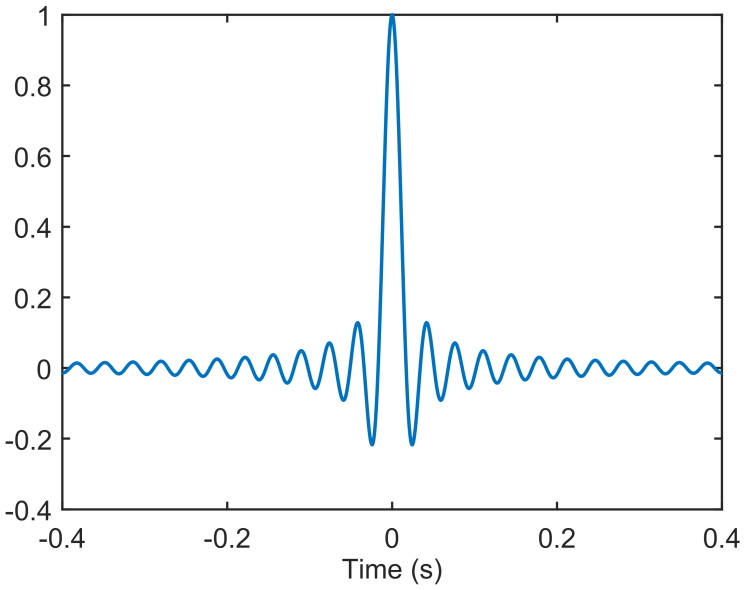

Figure 4: The time domain representation of the ideal low-pass filter, with a cutoff frequency corresponding to $T_s = 0.017s$.

As seen in Figure 5, the reconstruction of the sampled data perfectly recreate the continuous signal for the higher sampling rate. Due to the fact that the data taken by the sampling rate is not of infinite length, the reconstruction can vary at the end points. The plot seen in Figure 5 has data spanning the time scale of [-20,20], because the span is much greater than the range shown, the end points can be assumed to be at negative and positive infinity. Looking at the other sampling rate $T_s = 0.17$ s, the nyquist frequency is not great enough and reconstruction produces aliasing. This will be true for all the other low pass-filters upon reconstruction with the lower sampling rate.

3.1.1 Aliasing due to sampling

The ideal low-pass filter gives the best insight into how different sampling rates affect aliasing because it perfectly constructs whatever signal was sampled. In the case of this analysis, the max frequency content of the signal is equal to $\omega_0 = 37.69$, while the sampling rates of $T_s = 0.17$ s and $T_s = 0.017$ s correspond to frequencies of $\omega_s = 36.96$ and $\omega_s = 369.6$. According to

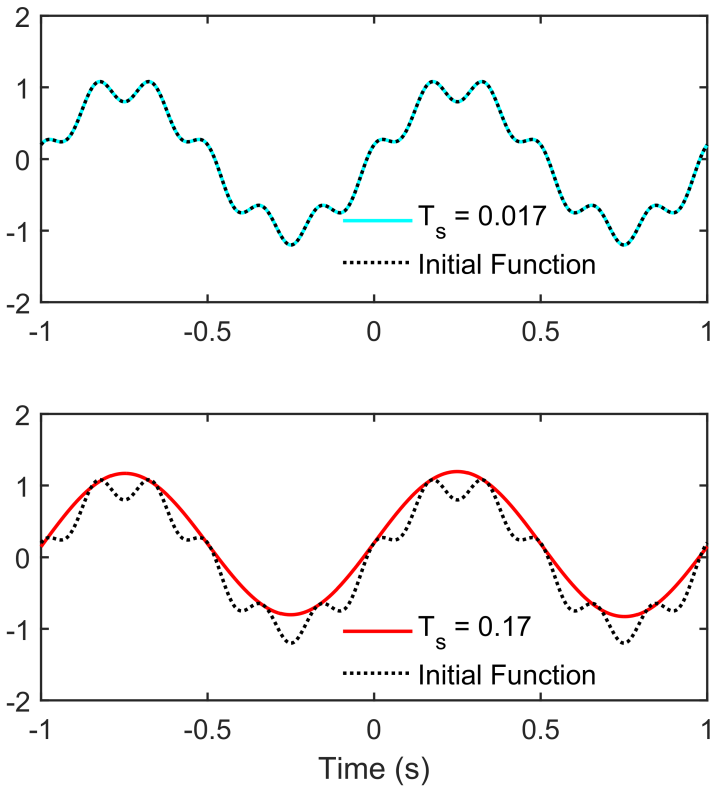

Figure 5: The reconstruction of the sampled data with $T_s = 0.017$s and $T_s = 0.17$s, and the comparison to the original function. The ideal low-pass filter perfectly recreates the initial function for the higher sampling rate.

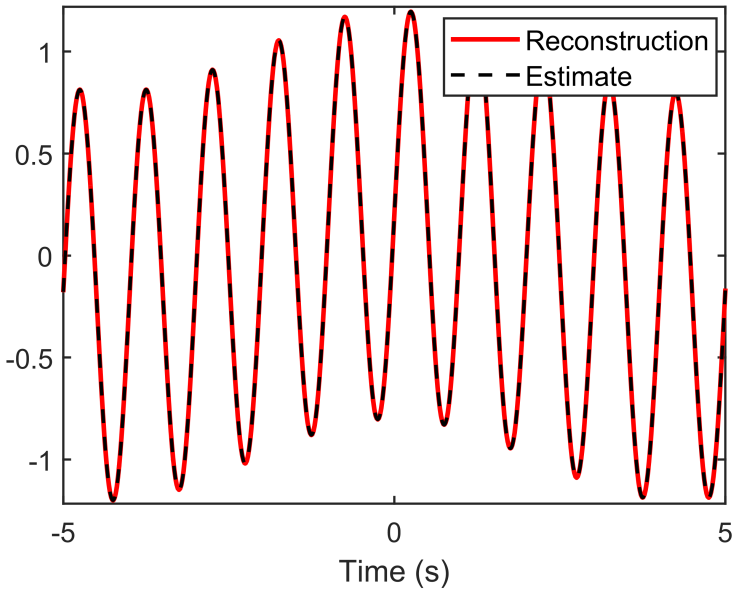

Figure 6: The predicted alias of the function with too low of a sampling rate $T_s = 0.17$s, this resulted in frequency folding and a slower oscillating waveform at a frequency of $\omega = 0.739$.

Nyquist (9) if the sampling rate is not greater than twice the max frequency content, aliasing will occur.

For our $T_s = 0.017$ s, it is sampling at nearly ten times the max frequency content, where $T_s = 0.17$ s does not. An estimate can be made to what the frequency of the aliased signal will be, because the only two frequencies in this signal are the $\omega = 2\pi$ and the max frequency. Frequency folding is where the initial frequency content is shifted from it's initial location by multiples of ω_s, the frequency ω_0 will shift to the left by ω_s which results in a frequency of $\omega = 0.739$, which is a $T = 8.5$ s. In Figure 6, the prediction directly overlaps on the reconstructed signal, confirming the estimated alias due to too low of a sampling rate.

3.2 Zero-Order Hold (ZOH)

The zero order hold is another implementation of a low pass filter. The filter is causal and simple to implement in real-time control due to it holding the current sampled value until the next one is obtained. The zero-order hold is

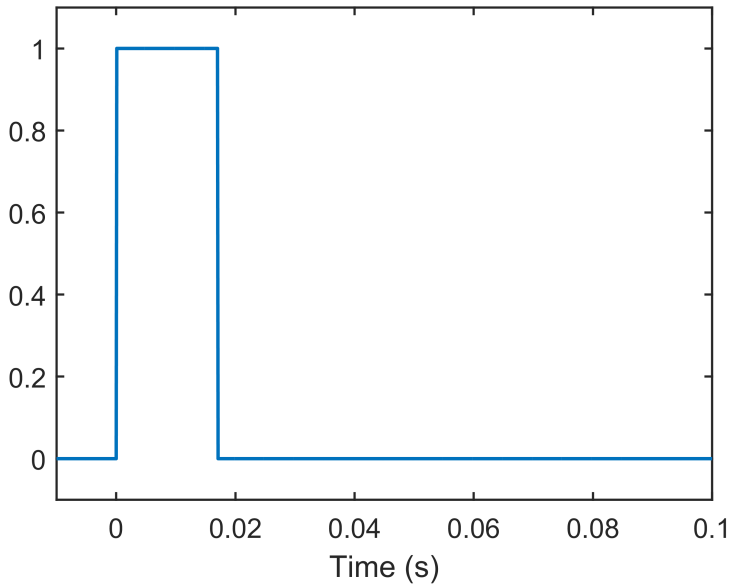

Figure 7: The zero-order hold the current sample value until the next point where it then updates.

arguably the simplest form of reconstruction, and is easiest to implement in code because the value is held constant between point to point. For many control applications this form of reconstruction is adequate and does not add complexity. The pulse response of the zero-order hold looks like Figure 7 which models how the sampled points will be reconstructed.

The convolution of the function $x^*(t)$ (the continuous representation of the discrete data) with the pulse response of the zero-order hold creates a function that is displayed in Figure 8. For the higher sampling rate, with $T_s = 0.017$ the reconstruction follows the shape of the initial function $x(t)$. As mentioned previously, the reconstruction with the lower sampling produces aliasing, which can clearly be seen in the lower plot of Figure 8.

3.3 First-Order Hold (FOH)

The first order hold is non-causal meaning that it needs 'negative' time in order to work, luckily because this reconstruction is not being performed real-time we have all the data needed to use this method. The pulse response of the FOH uses the information of the future points to interpolate it's position.

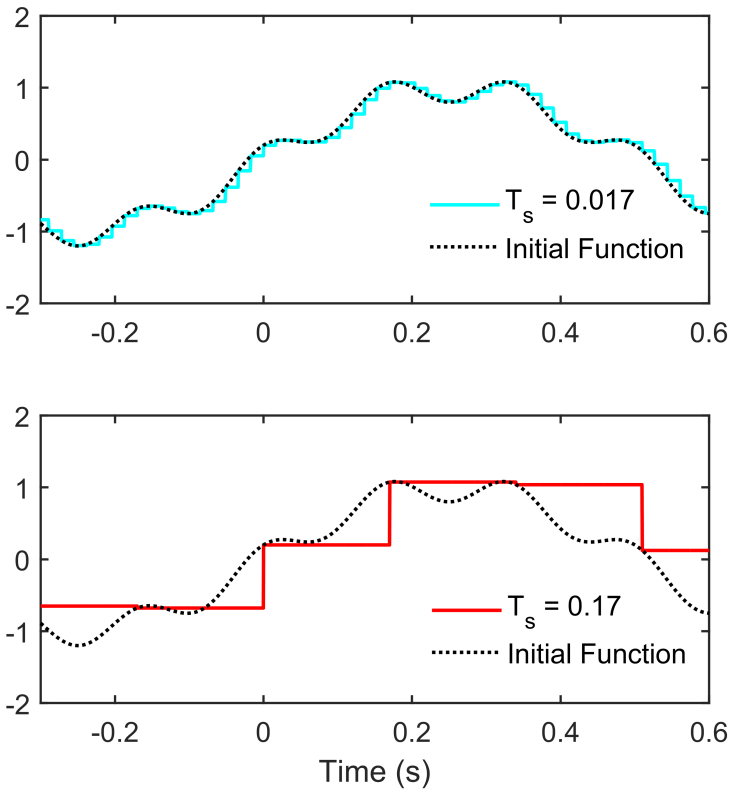

Figure 8: The reconstruction of the sampled data with $T_s = 0.017$s and $T_s = 0.17$s, and the comparison to the original function. The zero-order hold holds the current sample value until the next point where it then updates.

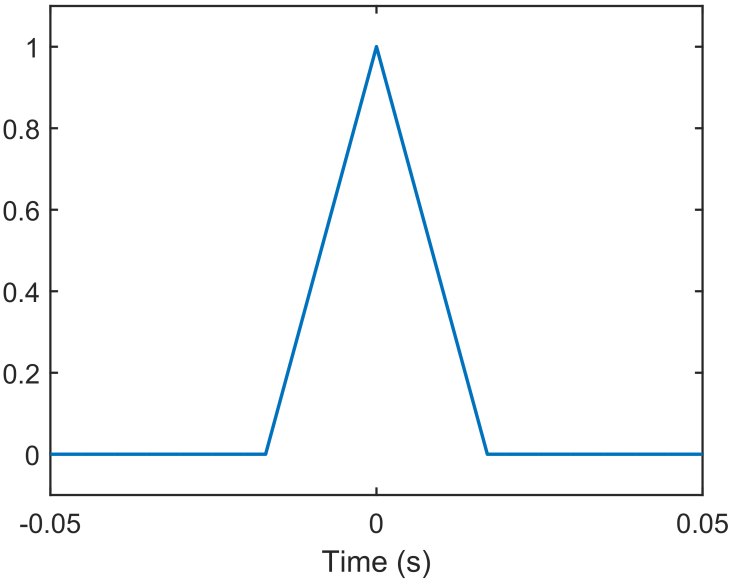

Figure 9: The model of a first order hold, with a sampling rate of $T_s = 0.017$s, it is a model of connecting data directly from the previous value to the next sample in the shortest distance.

This is the equivalent of drawing a line in between adjacent points, Figure 9 shows the pulse response $h(t)$.

The first order hold is the next best in terms of reconstructing the exact signal $x(t)$, like the ideal low-pass filter it is also non causal as can be seen in the pulse response. There is a negative time component to the model, meaning that it requires the next point to determine how to connect the points. As seen in the reconstruction, Figure 8, the reconstruction seems to follow the line very precisely. The lower plot shows the process of connecting points better, the points are taken at a larger T_s producing an aliased signal but showing the method of reconstruction more clearly.

3.4 Predictive First-Order Hold (PFOH)

The predictive first-order hold is a real-time implementation of the FOH, where it takes the previous sample $x[k-1]$ and the current sample $x[k]$, and makes a prediction based off the slope to what the next point will be. If the function is a slow changing or the sampling rate is very high this

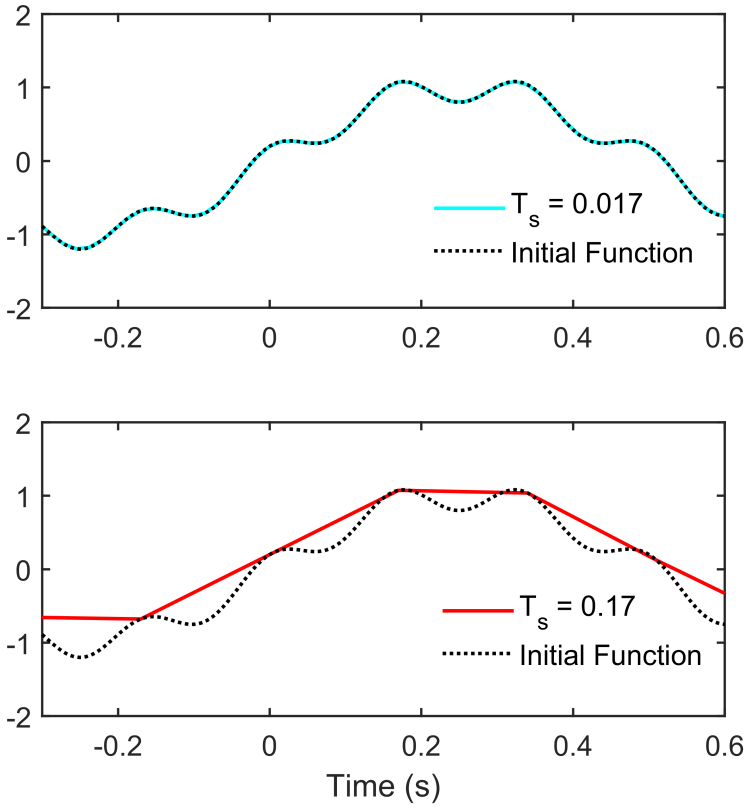

Figure 10: The reconstruction of the sampled data with $T_s = 0.017$s and $T_s = 0.17$s, and the comparison to the original function. The first-order hold is non-causal and interpolates the current sample value to the next point. At this resolution, the FOH looks approximately like the ILP for the higher sampling rate.

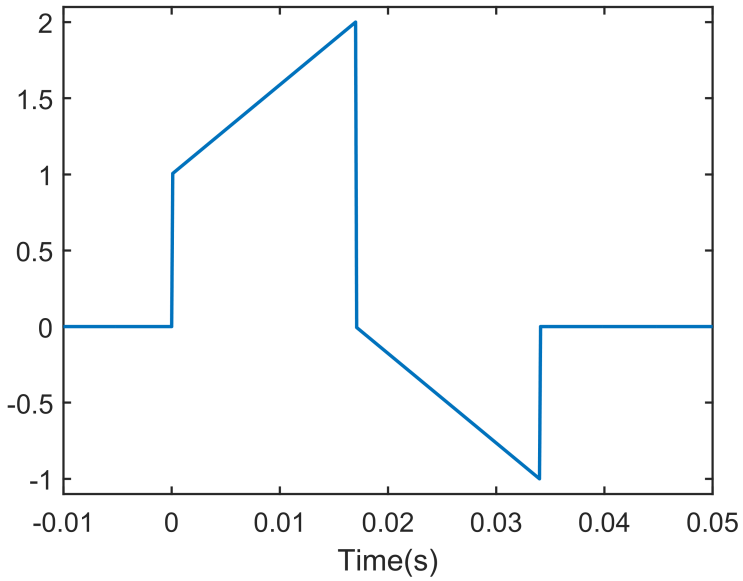

Figure 11: The model of a predictive first order hold, with a causal filter, with a sampling rate of $T_s = 0.017$s. It is a model of predicting the value of the next sample based off the slope from the previous sample and the current sample.

approximation is a good estimate, however for quickly changing functions this method will create sharp spikes at certain points along the reconstruction.

3.4.1 Derivation

The element-wise representation of the predictive first order hold is

$$x[k+1] = x[k] + \frac{x[k] - x[k-1]}{T_s} t, \tag{12}$$

where T_s is the sampling rate and t is the variable that is interpolating between adjacent points. To visualize this, the function $\delta[k]$ (amplitude of 1 at $k = 0$, and 0 otherwise) is plotted and then used with the PFOH. The time between index $[k]$ and $[k+1]$ is T_s, The slope is then $1/T_s$ between points $k = 0$ and $k = 1$, then between points $k = 1$ and $k = 2$ the slope is

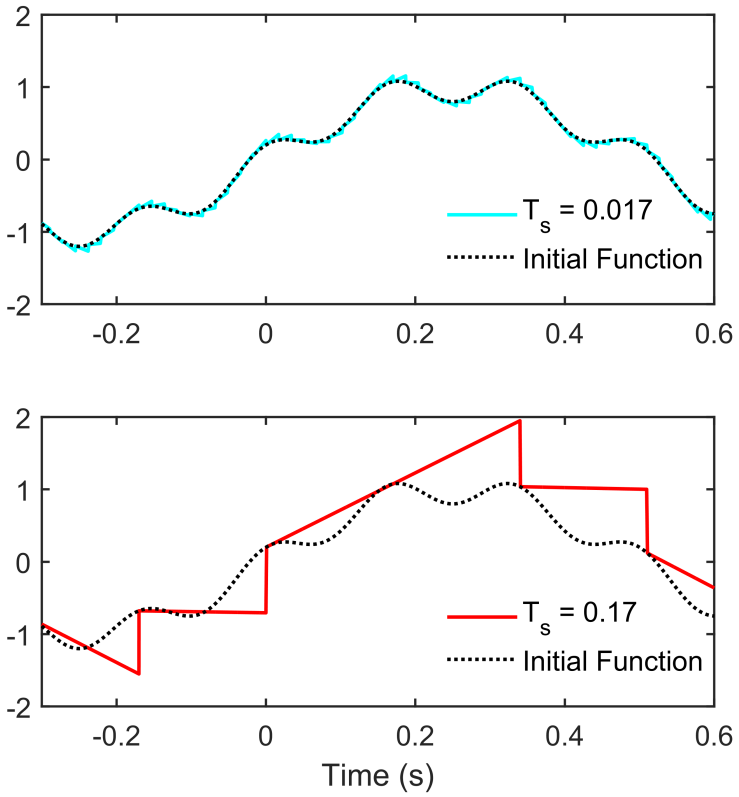

Figure 12: The reconstruction of the sampled data with $T_s = 0.017$s and $T_s = 0.17$s, and the comparison to the original function. The predictive first-order hold is causal, and uses information from the previous point and the current point to make an estimate for the next sample.

$-1/T_s$. The resulting equation for the model of the first order hold is

$$h(t) = \begin{cases} 1 + t/T_s & 0 \leq t < T_s \\ -t/T_s & T_s \leq t < 2T_s \\ 0 & else \end{cases} \tag{13}$$

The predictive first-order hold model $h(t)$ is visualized in Figure 11, with the convolution with the sampled data shown in Figure 12.

4 Frequency-Response Function (FRF)

The Frequency-Response Function of the zero-order hold can be derived by taking the Fourier Transform of the pulse response $h(t)$. The magnitude plot of the FRF provides some insight into the performance of the zero-order hold, as it compares to the ideal low-pass filter. The ideal low-pass filter has a cutoff frequency of $\omega_s/2$, with an immediate drop to zero after that point. The ZOH on the other hand is not as sharp, the Fourier Transform of the zero order hold can be derived from the Laplace transform and plugging in $s = j\omega$. The result of which is the absolute value of the sync function. Notice the duality - the square top-hat in the frequency domain is the sinc function in the time domain, and visa versa for the ZOH. In terms of performance, the ideal low-pass filter cuts exactly at the cutoff frequency where the zero-order hold dies fairly quickly and reverberates back and forth which not only attenuates some of the frequencies that are supposed to be kept but can let in higher frequency noise.

5 Conclusion

The intuitions gained of the Shannon Sampling Theorem are incredibly useful in discrete signal processing. Reconstruction is a major part of understanding the sampled points gathered from an ADC in real-time or when using a DAQ and doing post-processing. The theory behind sampling a continuous function can be attributed to the comb function and the basis of the Shannon Sampling Theorem is built off of the Fourier Transform of the continuous representation of the sampled signal. Recall from the beginning of the report that aliasing can occur when the sampling frequency is not high enough, specifically if it is less than the nyquist frequency. The nyquist frequency (9)

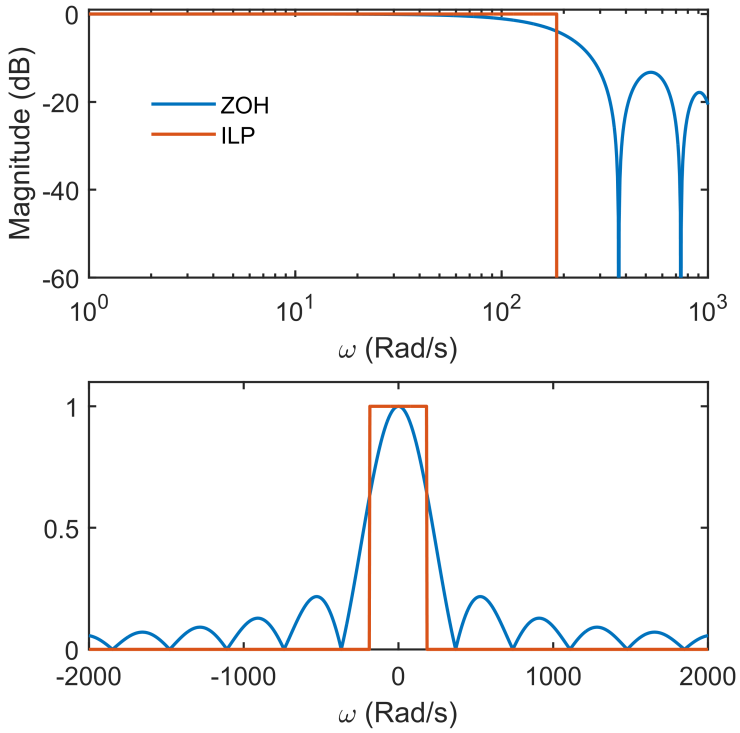

Figure 13: The frequency response function of the ideal low-pass filter (ILP) and the zero-order hold (ZOH) for a sampling frequency of $\omega_s = 369.6$ Rad/s.

is equal to twice the highest frequency content of the signal, and this was proven by looking at the frequency content of a sampled signal. A sampled signal (discrete in the time domain) is periodic in the frequency domain, and the distance the signal content is apart is the sampling frequency ω_s. If overlap occurs, then the amplitudes will super impose, creating a different signature. Even with an ideal low-pass filter, at this point the reconstructed signal is an alias. However, when sampling at higher frequencies, aliasing can still occur upon reconstruction because the perfect low-pass filter is not realizable or causal. Instead, another method of reconstruction will likely be used, for many control applications a zero-order hold is acceptable. As seen from the frequency response function of the zero-order hold, the filter still lets signals through at higher frequencies and attenuates portions that are still within the cutoff. Additionally, if the cutoff frequency of the low-pass filter is too high and the sampling rate sufficient the shifted frequency content could not be cut out entirely. A standard for the filter is a cutoff frequency of $w_s/2$. Using these fundamental principals, sampling rates can be chosen carefully to avoid aliasing and aid in reconstruction of the signal.

A Code

A.1 Creating Plots

```
%% Lab03 - reconstruction
% Riley Kenyon
% 3/14/2019
%-------------------------------------------------------------------------
%% Industrial Automation Lab 3 - Convolution
% Riley Kenyon 3/7/2019
%-------------------------------------------------------------------------
clear all; close all; clc;
% Actual function
DT = 0.0001;
t = -20:DT:20;
t1 =[-20.06:0.017:20.06]; % For T_s = 0.017
t2 = [-20.06:0.17:20.06]; % For T_s = 0.17

% Example function for Reconstruction and Convolution
```

```
funct = @(t) sin(2*pi*t) + 0.2*cos(12*pi*t);
ILP = @(t,T_s) sinc(t/T_s); % sinc is sin(pi*t)/(pi*t)
alias = @(t) sin(2*pi*t) + 0.2*cos(0.739*t);

% Sampling Time: T_s = 0.017
%-------------------------------------------------------------------------
[t_017, x_017] = sample(t1,funct); % Now has interval of DT with data at T_s
ILP_017 = ILP(t,0.017); % Impulse ILP of function with T_s = 0.017, and at interva
ZOH_017 = ZOH(t,0.017); % Impulse ZOH
FOH_017 = FOH(t,0.017); % Impulse FOH
PFOH_017 = PFOH(t,0.017); % Impulse PFOH
% Convolution of filters with signal
y_ILP_017 = conv(ILP_017,x_017,'same');
y_ZOH_017 = conv(ZOH_017,x_017,'same');
y_FOH_017 = conv(FOH_017,x_017,'same');
y_PFOH_017 = conv(PFOH_017,x_017,'same');
%y_017 = convDiscrete(x_017,h_017,0.017);


% Sampling Time: T_s = 0.017
%-------------------------------------------------------------------------
[t_17, x_17] = sample(t2,funct); % Now has interval of DT with data at T_s
ILP_17 = ILP(t,0.17); % Impulse ILPof function with T_s = 0.017, and at interval D
ZOH_17 = ZOH(t,0.17); %Impulse ZOH
FOH_17 = FOH(t,0.17);
PFOH_17 = PFOH(t,0.17);
y_ILP_17 = conv(ILP_17,x_17,'same');
y_ZOH_17 = conv(ZOH_17,x_17,'same');
y_FOH_17 = conv(FOH_17,x_17,'same');
y_PFOH_17 = conv(PFOH_17,x_17,'same');
%y_17 = convDiscrete(x_17,h_17,0.17);

% Figures
%-------------------------------------------------------------------------
figure() %continuous Function
hold on
plot(t,funct(t));
```

```
figure() %Example of sampling property
hold on
stem(t2,funct(t2))
plot(t,funct(t),'k--','LineWidth',1.5);

figure() %expanded sample of t = 0.017
hold on
plot(t_017,x_017);
stem(t1,funct(t1));

figure() % alias prediction
hold on
plot(t,y_ILP_17,'r','LineWidth',1.5);
plot(t,alias(t),'k--','LineWidth',1.5);

figure() % ZOH model
hold on
plot(t,ZOH_017);

figure() %FOH model
hold on
plot(t,FOH_017);

figure() %PFOH model
hold on
plot(t,PFOH_017);

figure() % plot of for ILP reconstructions
subplot(2,1,1);
hold on
plot(t,y_ILP_017,'c','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-1 1])
legend('T_s = 0.017','Initial Function')
subplot(2,1,2);
hold on
plot(t,y_ILP_17,'r','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
```

```
xlim([-1 1])
legend('T_s = 0.17','Initial Function')
xlabel('Time (s)')

figure() % plot of for ZOH reconstructions
subplot(2,1,1);
hold on
plot(t,y_ZOH_017,'c','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.017','Initial Function')
subplot(2,1,2);
hold on
plot(t,y_ZOH_17,'r','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.17','Initial Function')
xlabel('Time (s)')

figure() % plot of for FOH reconstructions
subplot(2,1,1);
hold on
plot(t,y_FOH_017,'c','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.017','Initial Function')
subplot(2,1,2);
hold on
plot(t,y_FOH_17,'r','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.17','Initial Function')
xlabel('Time (s)')

figure() % plot of for PFOH reconstructions
subplot(2,1,1);
hold on
plot(t,y_PFOH_017,'c','LineWidth',1.5)
```

```
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.017','Initial Function')
subplot(2,1,2);
hold on
plot(t,y_PFOH_17,'r','LineWidth',1.5)
plot(t,funct(t),'k:','LineWidth',1.5)
xlim([-0.3 0.6])
legend('T_s = 0.17','Initial Function')
xlabel('Time (s)')


w = -2*pi*1000:0.01:2*pi*1000;
%w = 2*pi./t; % vector of frequencies
w_s = 2*pi/0.017;
ILP_w = abs(ILP(w,w_s));
ZOH_w = ZOH(w,w_s)+ 1e-9;
%ZOH_w = ZOH(t,0.017);
figure()
subplot(2,1,1)
hold on
plot((w),20*log10(ILP_w));
plot((w-w_s/2),20*log10(ZOH_w))
%axis([0 3 -60 1])
subplot(2,1,2)
hold on
plot(w,ILP_w);
plot((w-w_s/2),ZOH_w);
axis([-2000 2000 0 1.1]);
function [t,x] = sample(time, funct)
%------------------------------------------------------------------------
% input:
%       N_s : sampling number multiple of 0.0001
%       funct : function being sampled with deltaT of 0.0001
% output:
%       x : vector length of function sampled at intervals of T_s
%------------------------------------------------------------------------
    DT = 0.0001;
```

```
    T_s = time(2)-time(1); % current difference between points
    N_s = round(T_s/DT);
    x = zeros((length(time)-1)*N_s+1,1);
    t = time(1):DT:time(end);
    for i = 1:N_s:length(x)-1
        x(i) = funct(time((i-1)/N_s + 1));
    end
end
function [y] = convDiscrete(x,h,T_s)
%-------------------------------------------------------------------------
% input:
%       x : discrete function with intermediary spacing of DT = 0.0001
%           and sample period of T_s
%       h : impulse response of perfect low-pass filter (sinc function with
%           DT = 0.0001
% output:
%       y : vector of values corresponding to continuous region of -20<t<20
%-------------------------------------------------------------------------
    DT = 0.0001;
    N_s = round(T_s/DT);
    offset = (length(h)-1)/2;
    y = zeros(length(h),1);
    for N = 1:length(h) % change this to be time
        for k = 1:N_s:length(x)
            if((N-k)>0) % update with offset
                y(N) =  y(N) + x(k)*h(N-k);
            end
        end

    end

end
function [h] = ZOH(t,T_s)
%-------------------------------------------------------------------------
% input:
%       t : time vector with intermediary spacing of DT = 0.0001
%       T_s: sample period of T_s
% output:
```

```
%       h : impulse response vector
%-------------------------------------------------------------------------
h = zeros(length(t),1);
for i = 1:length(t)
    if (t(i)>0 && t(i) <= T_s)
        h(i) = 1;
    end
end
% figure()
% plot(t,h);

end
function[h] = FOH(t,T_s)
%-------------------------------------------------------------------------
% input:
%       t : time vector with intermediary spacing of DT = 0.0001
%       T_s: sample period of T_s
% output:
%       h : impulse response vector
%-------------------------------------------------------------------------
h = zeros(length(t),1);
for i = 1:length(t)
    if (t(i)>-T_s && t(i) <= 0)
        h(i) = 1/T_s*(t(i)+T_s);
    end
    if (t(i)> 0 && t(i) <= T_s)
        h(i) = -1/T_s*t(i)+1;
    end
end
end
function [h] = PFOH(t,T_s)
%-------------------------------------------------------------------------
% input:
%       t : time vector with intermediary spacing of DT = 0.0001
%       T_s: sample period of T_s
% output:
%       h : impulse response vector predictive first order hold
%-------------------------------------------------------------------------
```

```
h = zeros(length(t),1);
for i = 1:length(t)
    if (t(i)>0 && t(i) <= T_s)
        h(i) = 1/T_s*t(i)+1;
    end
    if (t(i)> T_s && t(i) <= 2*T_s)
        h(i) = -1/T_s*(t(i)-T_s);
    end
end

end
```