Coovi Meha
MEid:481-473

Project 3

MCEN 5228-004
March 15, 2019

Abstract

In Engineering, most designs revolve around using analog and digital devices. Thus, the conversion of data from one type to another becomes necessary to achieve a desired goal in building a project. Data which will be refereed to as a signal during in this report can be discrete or analog. Analog signals are continued while discrete signal are discretized data points sampled at a specific frequency (period) by computerize device.
Let assume that we have e system consist of a MyRio and a motor; and we wish to record the speed of the while running a feedback loop. To achieve this, the MyRio must record the voltage output from the motor, take it through the feedback loop then output a new a corrected signal to make the motor run at a constant speed. However, it is important to note that computer runs in discrete time while the motor runs in continues time leading to the necessity to convert the signal between analog (continue time signal) and digital(discrete time signal).
The conversion process is themed signal reconstruction. This report will center on how reconstruction can be performed, the problem that can arise during signal processing with examples and some solution to help be a master at reconstructing signal. To achieve the above we must assume the following:

- The system must be causal: We can not see the future in another word the system response time start at t=0s.
- Linear time invariant system: The system is linear and time independent.

What is signal reconstruction?

Sampling of signals consist of the snapshot the continues time signal at a constant time intervals. The process leads generality to collection data points of different amplitudes with a certain period or frequency called the sampling period. The reconstruction of this signal despite the opposite of the previous process with a continuous signal similar to the original signal sampled. But first what does sampling look like from Engineering prospective?

Sampling

Consider the following signal in continue time domain.

$$f(t) = sin(2\pi t) + 0.2cos(12\pi t) \tag{1}$$

If one wants to plot this function in Matlab for example, it is necessary to generate discrete points by defining the small enough time interval that can give a good resolution to the plot.

This is a discrimination of the continuos time signal function. The similar method is used when sampling a signal. However, instead we define an impulse function

$$f_i(t) = T \sum_{k=-\infty}^{\infty} \delta(x - kT) \tag{2}$$

to slash the continuous signal into discrete data points the impulse signal must have a high of exactly one to allow avery point generated to have the same amplitudes as in the continuous function. The following illustration give a better view of the process As show above, to get a

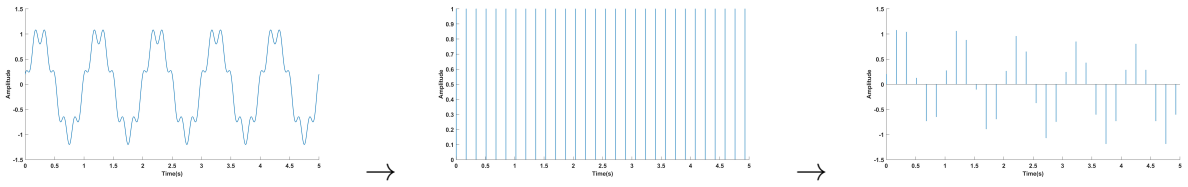

Figure 1: Sampling illustration of a signal. From left to right Signal in continuous time domain, impulse input signal and sampled signal.

discrete time signal the input signal is multiply by an impulse of amplitude 1 shifted by kT with T the period and K the nth shifted of the impulse.

$$f_s(t) = T \sum_{k=-\infty}^{\infty} f_i(kT)\delta(x - kT) \tag{3}$$

One issue during sampling of signal come generally from the sampling period or frequency. In fact, the sampling frequency affects the signal during reconstruction. If the signal is sampled at period very high or slash with an impulse with high period, the discrete points output would be scattered with a pattern that will look nothing like the input signal. Harry Nyquist, an electronic engineer, came up with a know maximum frequency at which a signal can be sampled to still represent the origin input signal as

$$\Omega_s > 2\Omega_{Niq} \tag{4}$$

Ω_{Niq} is known as the Nyquist frequency and Ω_s the sampling frequency

Reconstruction

The signal sampled above can be reconstructed into a continued signal perfectly but there is a catch. Shannon theory stated that to be able to reconstruct a signal, one must not only sampled the input at $\Omega_s > 2\Omega_{Niq}$ but also passed it through an ideal low pass filter. From the above section we known what Nyquist principle is , thus let see what an ideal low pass filter is.

- Perfect Ideal Low Pass Filter
  An ideal low pass filter (ILP) has the following form

$$H(\Omega) = \{T_s; -f_c < t < f_c \tag{5}$$

and zero everywhere else as shown in the figure bellow. The impulse response response

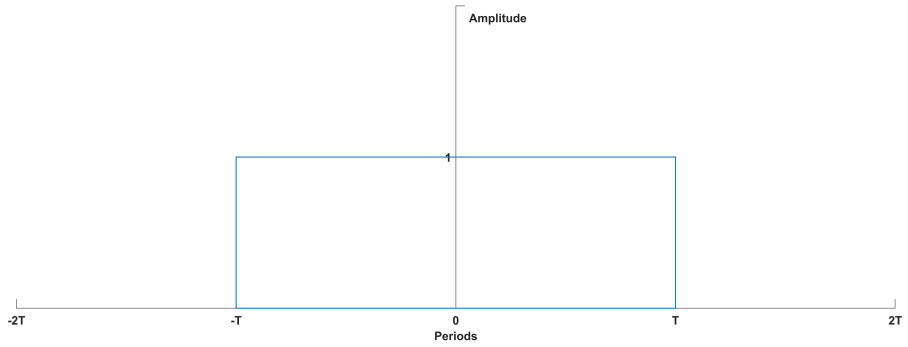

Figure 2: An ideal LP filter with amplitude 1 and cut-off frequency of f_c

of such signal can be found using

$$h(t) = \int_{-f_c}^{f_c} e^{-j\Omega t} d\Omega \tag{6}$$

after performing the integration, the above function become

$$h(t) = \frac{sinc(f_c t)}{f_c t} \tag{7}$$

the sinc function can be plotted in MatLab to obtain the figure bellow. It can be

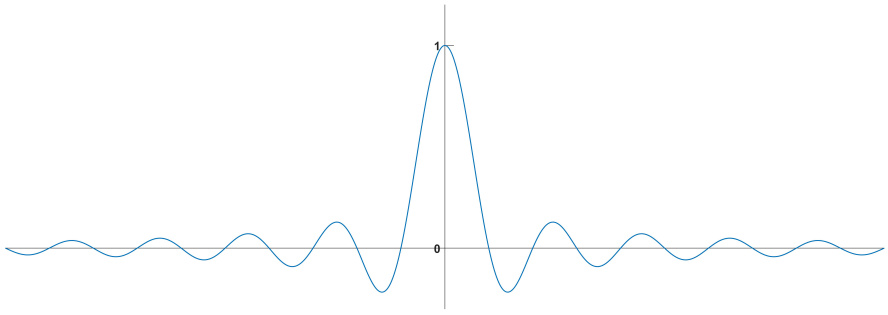

Figure 3: plot of the sinc function derive from the above low pass filter

seen from the figure that, the continuous time signal of a ILP is not causal. In fact, the system responded before even it has bee kicked, in other word before time started as we can not consider negative time. This sort of system is not real and can not be implemented in real life. Thus, Shannon theory can not be used in real system leading to the conclusion that a signal can not be perfectly reconstructed using real life hardwares with this sort of low-pass filter. Nonetheless, we can reconstruct the signal perfectly using MatLab provided that the sampling frequency respect the Nyquist rule. When sampling at the period of 0.017 The signal can be reconstructed to match exactly

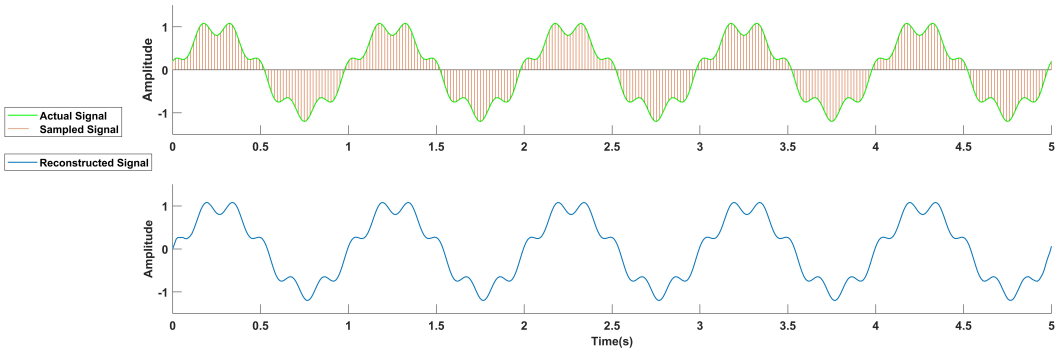

Figure 4: Reconstructed signal sampled at T=0.017s. The top figure is the sampled signal and bottom represent the reconstructed Signal using Matlab and the sinc function

the analog input but the if the signal is sampled at a period ten times slower, it become obvious that the reconstructed signal does look nothing like the input signal. This

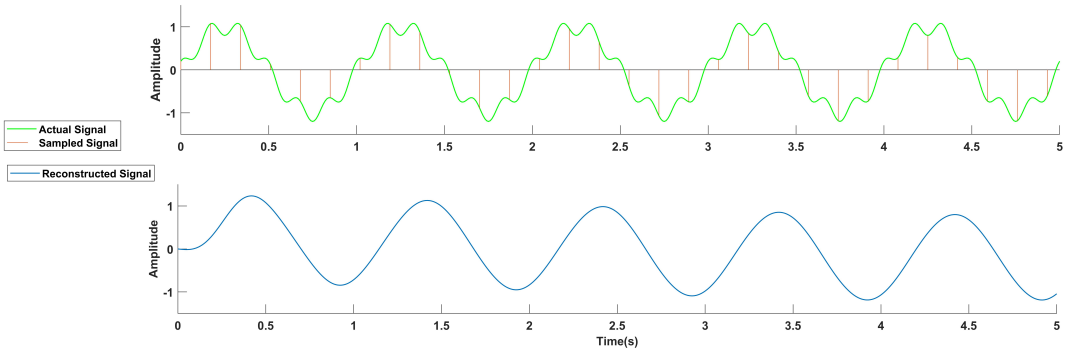

Figure 5: Reconstructed signal sampled at T=0.17s. The top figure is the sampled signal and bottom represent the reconstructed Signal using Matlab and the sinc function. The reconstructed signal is liaised

phenomena is know as liaising. As mentioned above, a perfect low pass filter can not be design in real life, thus engineers have come with many brilliant ideas to design ADC and DAC.

- Zero Order Hold (ZOH) Zero oder hold consist of using the nth value of the sampling signal and hold it until the next value is available. Any signal can then be reconstructed closed to accuracy using this method if the sampled period is small enough. Mathematically it can be showed that the amplitude of the frequency response of ZOH is

$$\mid G_{ZOH}(j\omega) \mid = T \mid \frac{sin(\omega T/2)}{\omega T/2} \mid \tag{8}$$

and the phase shifted is

$$\underline{/G_{ZOH}(j\omega)} = \frac{\omega T}{2} + \underline{/sin(\omega T/2)} \tag{9}$$

The bode plot of the ZOH is as follow Using MatLab, the signal can also be recon-

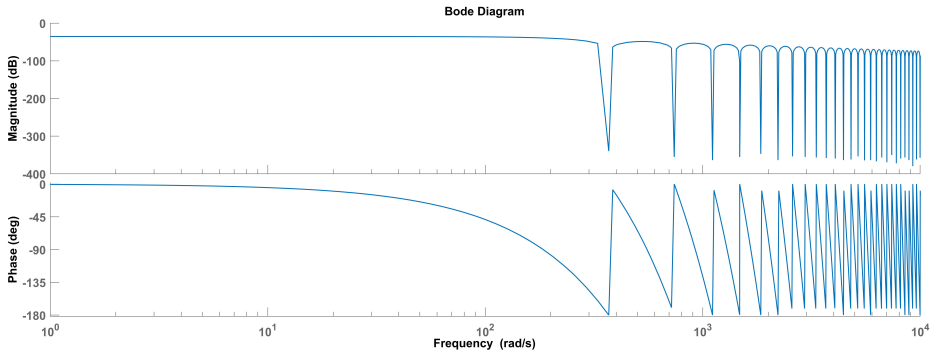

Figure 6: Bode plot of Zero oder hold. The phase shift of the system is zero for frequency lower than 100 rad/s

structed using the ZOH. For each value of the sampled signal, and array of known size can be created and populated with each data point. Doing so for all elements of the sampled data points Matlab will generate an array which plotted should look like the one bellow. As it can be seen on the above plot, the accuracy of the ZOH depend on the sampling period of the Signal

- Acausal First Order Hold(AFOH)
AFOH uses interpolation to Predict the next value of the signal. When reconstructing using AFOH, the n-1 and n are used to found a value in between. Depending of the sampling period, the reconstructed signal has higher resolution than the ZOH. In fact, 2x point is generated with period of Ts/2. The above figure show the AFOH using matlab. The code can be found attached to this report.
Mathematically, if the sampling signal of AFOH is

$$x_s(t) = T \sum_{n=-\infty}^{\infty} \delta(t - nT) \tag{10}$$

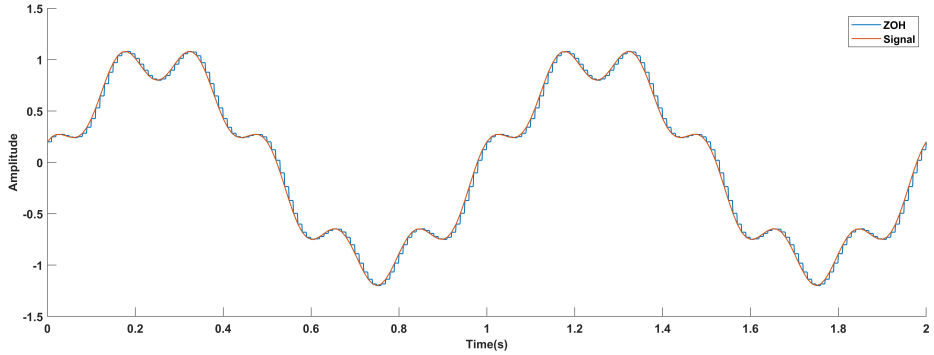

Figure 7: Zero order hold low pass filter, the resolution of the reconstructed in dependent of the sampling period

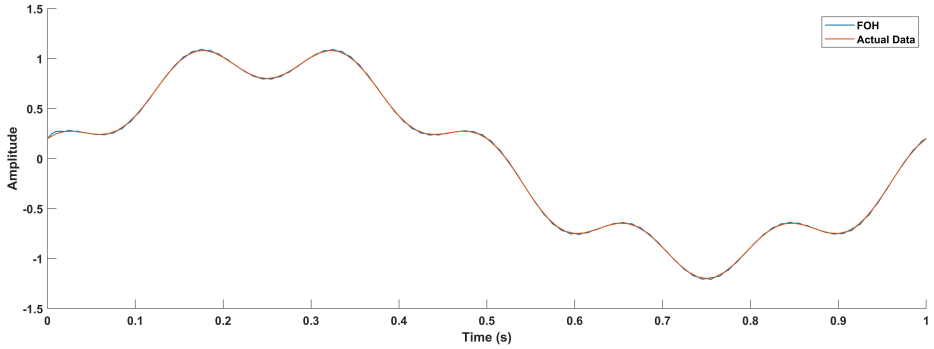

Figure 8: Acausal first order hold signal reconstructed using Matlab. The reconstruction uses interpolation method

Then the laplace domain response of the signal is to be

$$H(s) = (\frac{1 - e^{-sT}}{sT})^2 \tag{11}$$

This type of system is realizable by designing a digital filter of gain $H(z) = 1 - z^{-1}$ The bode plot of the laplace domain transferer function is similar to the ZOH except at high frequency

- Predictive First Order Hold (PFOH) Unlike AFOH, PFOH uses extrapolation. The knowing the values of the n-2 and n-2 data point, extrapolation is used to get a value of the n position before such data point is collected. PFOH has lower resolution than the AFOH. This can be seen on the figures 7 and 8 Both system can be design in real life by creating filter that mimic the same principles.

$$H(s) = (1 + sT)(\frac{1 - e^{-sT}}{sT})^2 \tag{12}$$

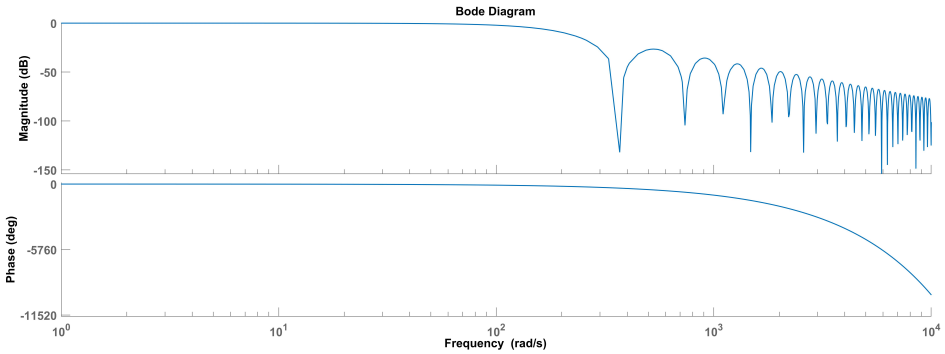

Figure 9: Bode plot of First oder hold. The phase shift of the system is zero for frequency lower than 100 rad/s

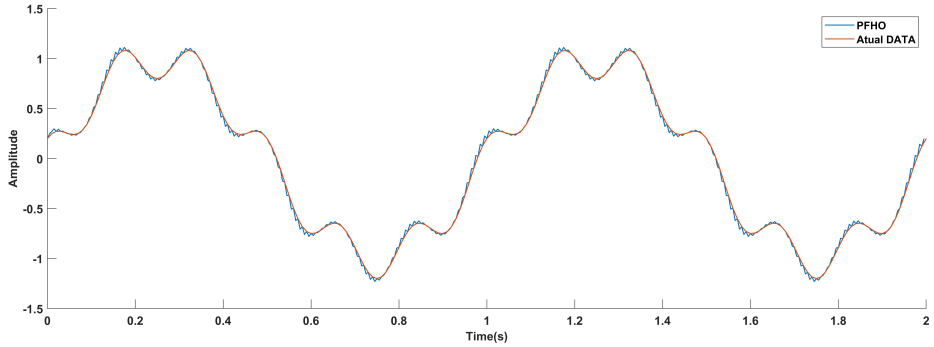

Figure 10: Reconstructed signal of predicted first order hold using extrapolation

The laplace domain equation of a PFOH is shown above. The main different is that PFOH has an extra zero which could make the system complex to design

Conclusion

Any signal can be reconstructed using a giving filter. In the perfect world, an ideal low pass filter can be used to reconstruct any signal perfectly if it is sampled following the Nyquist principle (Shannon theorem). Though such system is impossible to design, other technics such as ZOH, AFOH and PFOH are great tools to reconstruct signal with create accuracy. ZOH is the less accurate but easy to implement. In the other hand AFOH has a greater accuracy and can be easy to design. However, AFOH shift the reconstructed signal due to delay.

References

[1] "First-order Hold." Wikipedia. January 02, 2019. Accessed March 14, 2019.

[2] "Digital Control of Physical system", Chapter 2, page 1-14.

[3] "Documentation." Overleaf, Online LaTeX Editor. Accessed March 14, 2019. https://www.overleaf.com/learn/latex/.

Matlab Code

```
clc,clear,close all
%% Generate sin and cos plot and stem plot .17 s
t = 0:0.0001:5;
y= sin(2*pi*t) + 0.2*cos(12*pi*t);
figure(1)
plot(t,y)
figure(2)
t1 = 0:1700*.0001:5;
y= sin(2*pi*t1) + 0.2*cos(12*pi*t1);
h=stem(t1,y);
set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1)
figure(3)
x=ones(1,30);
t2 = 0:1700*.0001:5;
h=stem(t2,x);
set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1)

%% Generate sin and cos plot and stem plot .017 s
figure(4)
pe=.017 ;
t = 0:170*.0001:5;
N=length(t);
y= sin(2*pi*t) + 0.2*cos(12*pi*t);
h=stem(t,y);
set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1)
figure(5)
x=ones(1,295);
t = 0:170*.0001:5;
h=stem(t,x);
set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1)
```

```
31
32 %% zero oder-hold
33 deltat=0.01;
34 n=0:deltat:2;
35 y= sin(2*pi*n) + 0.2*cos(12*pi*n);
36 N=length(n);
37 ta=0:0.001:2;
38 ZOH=[];
39 for i=1:N-1
40  ZOH=[ZOH ones(1,10)*y(i)];
41 end
42 ZOH=[ZOH y(end)];
43 figure(6)
44 plot(ta, ZOH)
45 hold on
46 plot(n, y);
47
48 %%Perdictive first order hold
49 deltat=0.01;
50 n=0:deltat:2;
51 y= sin(2*pi*n) + 0.2*cos(12*pi*n);
52 N=length(n);
53 ta=0:0.005:2;
54 PFOH=[y(1) y(2) y(3)];
55 for i=3:N-1
56  p=polyfit([n(i-2) n(i-1)],[y(i-2) y(i-1)],1);
57  PFOH=[PFOH polyval([p(1) p(2)], n(i))];
58  PFOH=[PFOH y(i)];
59 end
60 PFOH=[PFOH y(end)];
61 figure(7)
62 ta=ta(1:end-1);
63 plot(ta, PFOH)
64 hold on
65 plot(n, y);
66 %% Acausal First Order hold
67 deltat=0.01;
68 n=0:deltat:2;
69 y= sin(2*pi*n) + 0.2*cos(12*pi*n);
70 N=length(n);
71 ta=0:0.005:2;
72 FOH=[y(1) y(2) y(3)];
```

```
for i=3:N-1
 m=(n(i-1)+n(i))/2;
  p=polyfit([n(i-2) n(i-1)],[y(i-2) y(i-1)],1);
 FOH=[FOH polyval([p(1) p(2)], m)];
 FOH=[FOH y(i)];
end
FOH=[FOH y(end)];
ta=ta(1:end-1);
figure(8)
plot(ta, FOH)
hold on
plot(n, y);

%% LOwpass filter
clc, clear
figure(9)
rectangle('Position',[1 2 5 6])
axis([0 10 0 10])
rectangle('Position',[-1 0 2 1])
axis([-2 2 0 2])
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';

names = {'-2T'; '-T'; '0'; 'T'; '2T'};
set(gca,'xtick',[-2:2],'xticklabel',names)
name1 = {'0'; '1'; '2'};
set(gca,'ytick',[0:2],'yticklabel',name1)

%% Sinc function
clc, clear
x = -10:0.01:10;
y = sinc(x);
figure(10)
plot(x,y)
ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
set(gca,'xtick',[])
set(gca,'xticklabel',[])
name1 = {'0'; '1'};
set(gca,'ytick',[0:2],'yticklabel',name1)
```

```
ylim([-.3,1.2]);

%% Reconstruction at 0.017 sampling
clc, clear
F = 1; %Frequency of analog data
Fs = 1/0.017;%sampling Frequency
t1 = 0;
t2 = 5; %Time range
t = t1:1e-2:t2;
y = sin(2*pi*F*t) + 0.2*cos(12*pi*F*t);
Ts = 1/Fs; %sampling period
ts = t1:Ts:t2;
sampled = sin(2*pi*F*ts) + 0.2*cos(12*pi*F*ts);
recons = zeros(1,length(t));
samples = length(ts);
for i = 1:1:length(t)
    for n = 1:1:samples
        recons(i) = recons(i) + sampled(n)*sinc((t(i)-n*Ts)/Ts); %
            reconstructed using sinc function
    end
end
figure(11)
subplot(2,1,1)
plot(t,y)
hold on
h=stem(ts,sampled);
set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1);
subplot(2,1,2)
plot(t,recons)
%% Reconstruction at 0.17 sampling
clc, clear
F = 1; %Frequency of analog data
Fs = 1/0.17;%sampling Frequency
t1 = 0;
t2 = 5; %Time range
t = t1:1e-2:t2;
y = sin(2*pi*F*t) + 0.2*cos(12*pi*F*t);
Ts = 1/Fs; %sampling period
ts = t1:Ts:t2;
sampled = sin(2*pi*F*ts) + 0.2*cos(12*pi*F*ts);
recons = zeros(1,length(t));
samples = length(ts);
```

```
156 for i = 1:1:length(t)
157     for n = 1:1:samples
158         recons(i) = recons(i) + sampled(n)*sinc((t(i)-n*Ts)/Ts); % reconstructed using sinc function
159     end
160 end
161 figure(12)
162 subplot(2,1,1)
163 plot(t,y)
164 hold on
165 h=stem(ts,sampled);
166 set(h(1),'MarkerFaceColor','blue','MarkerSize',0.1);
167 subplot(2,1,2)
168 plot(t,recons)
169 %% Bode plot ZOH
170 figure(13)
171 s=tf('s');
172 Ts=0.017;
173 sys=(1-exp(-s*Ts))/s;
174 bode(sys)
175 %% Bode plot FOH
176 figure(14)
177 s=tf('s');
178 Ts=0.017;
179 sys=((1-exp(-s*Ts))/(Ts*s))^2;
180 bode(sys)
181 %% Bode plot PFOH
182 figure(15)
183 s=tf('s');
184 Ts=0.017;
185 sys=(1+Ts*s)*((1-exp(-s*Ts))/(Ts*s))^2;
186 bode(sys)
```