

EE 386 Lab 1

Signals and System Review with Matlab

By. Christopher W. Fingers

010099178

**Part 1:**

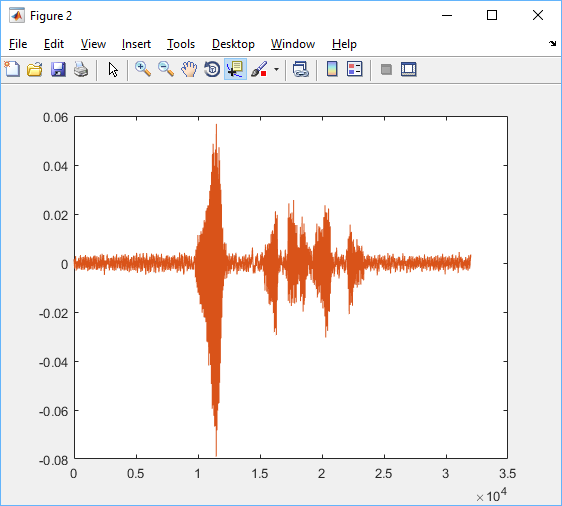


Figure 1: “Hello, Welcome to DSP” plot. Fs=8000Hz

**Part 2:**

1. There are 64000 points that were sampled in voiced recorded data.
2. There is a large initial spike that indicates when I first start talking with 4-5 other smaller similar sized spikes following it.
3. The x-axis represents the time of the recording.
4. Xlabel(‘time(sec)’)
5. The y axis represents the magnitude or amplitude of the recording.
6. After changing Fs to 11025 there are 88200 points that were recorded.
7. The second recording sounds clearer than the first recording with less static noise.
8. The final recording sounds very muffled and much quieter than the first and second recording.

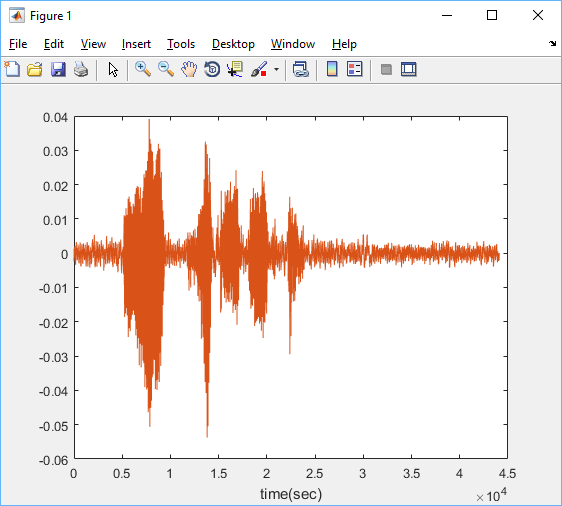


Figure 2: “Hello, Welcome to DSP” plot, Fs = 11025 hZ

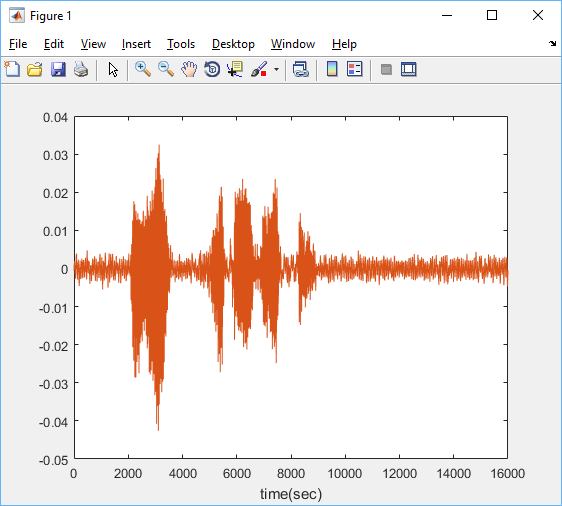


Figure 3: “Hello, Welcome to DSP” plot. Fs = 4000Hz

Section A:

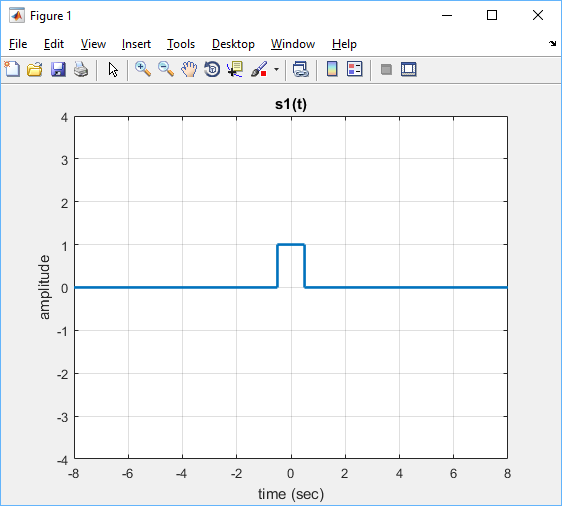


Figure 4 : Plot of S1 = rectpuls(t)

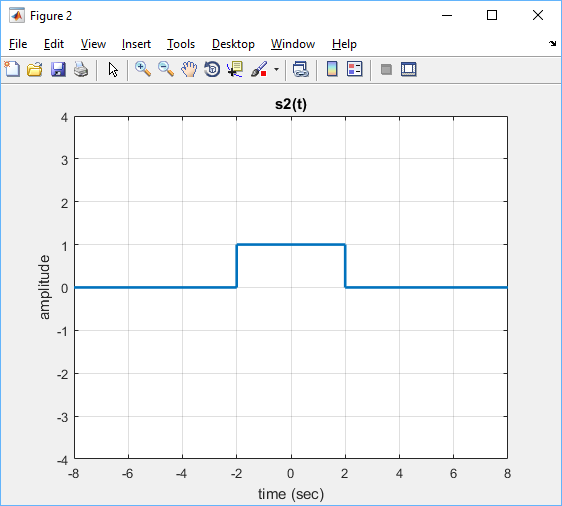


Figure 5 : Plot of S2 = rectpuls(t/4)

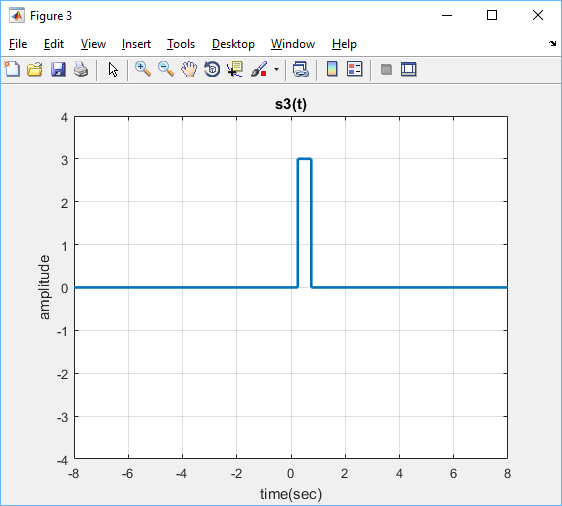


Figure 6: Plot of S3 = 3\*rectpuls(2\*t-1)

Section B:

Area = Height\*Length

Section C:

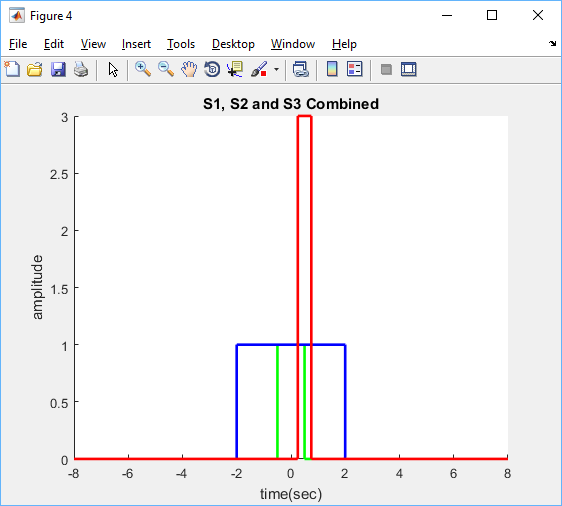


Figure 7: S1, S2 and S3 plotted together.

Section D:

S1 and S2 have y axis symmetry, which makes these two graphs both even functions. S3 is neither an odd or an even function. An even function requires that the if the function has a negative time value that the amplitude of the positive time value is the same. An odd function has it so that a negative time value has an amplitude that is the negative value of the positive time value amplitude.

**MatLab Code:**

t = -8:0.001:8;

s1 = rectpuls(t);

figure

plot(t,s1,'linewidth',2);

grid on;

axis([-8 8 -4 4]);

xlabel('time (sec)');

ylabel('amplitude');

title('s1(t)');

saveas(gcf,'lab1a.png'); %save the figure as an image

% Part A

s2 = rectpuls(t/4);

figure

plot(t,s2,'linewidth',2);

grid on

axis ([-8 8 -4 4]);

xlabel('time (sec)');

ylabel('amplitude');

title('s2(t)');

saveas(gcf,'lab1b.png');

figure

s3 = 3\*rectpuls(2\*t - 1);

plot(t,s3,'linewidth',2);

grid on

axis ([-8 8 -4 4]);

xlabel('time(sec)');

ylabel('amplitude');

title('s3(t)');

saveas(gcf,'lab1c.png');

% Part B

% The rectangular pulse function in matlab creates a pulse with an

% amplitude of 1 that is stretched between the x axis points of -1/2 and

% 1/2.

Area1=polyarea(t,s1)

Area2=polyarea(t,s2)

Area3=polyarea(t,s3)

% Part C

figure

hold on

plot(t,s1,'g','linewidth',2);

hold on

plot(t,s2,'b','linewidth',2);

hold on

plot(t,s3,'r','linewidth',2);

xlabel('time(sec)');

ylabel('amplitude');

title('S1, S2 and S3 Combined');

**Part 3:**

Section A:

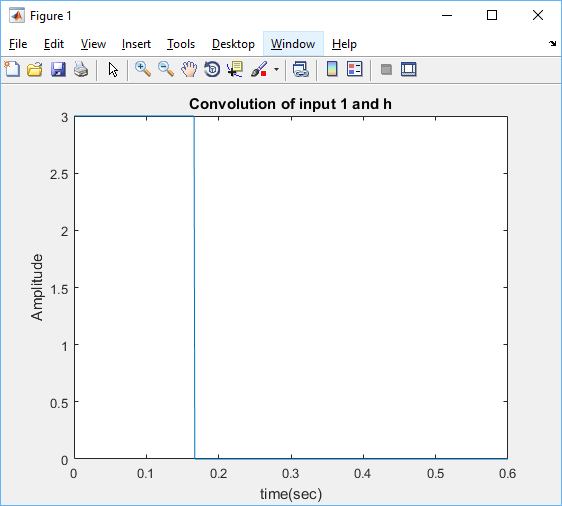


Figure 8: Convolution graph of input 1 and h.

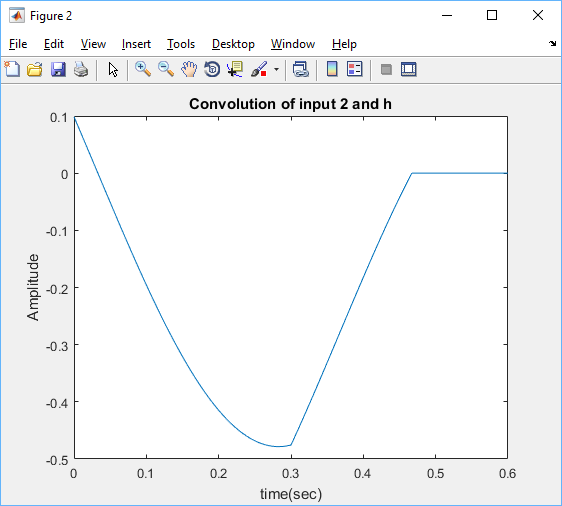


Figure 9: Convolution graph of input 2 and h.

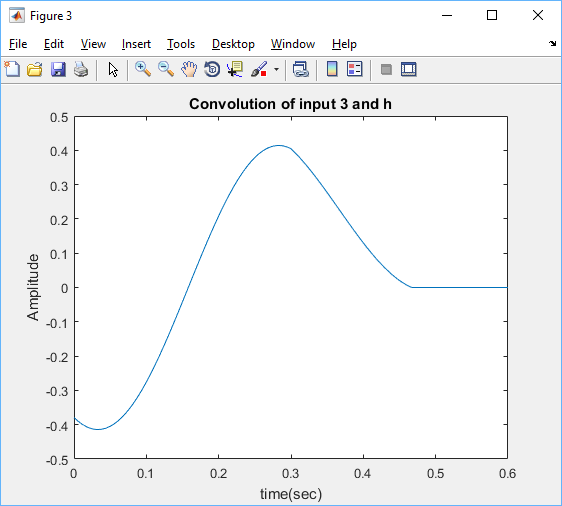


Figure 10: Convolution of input 3 and h.

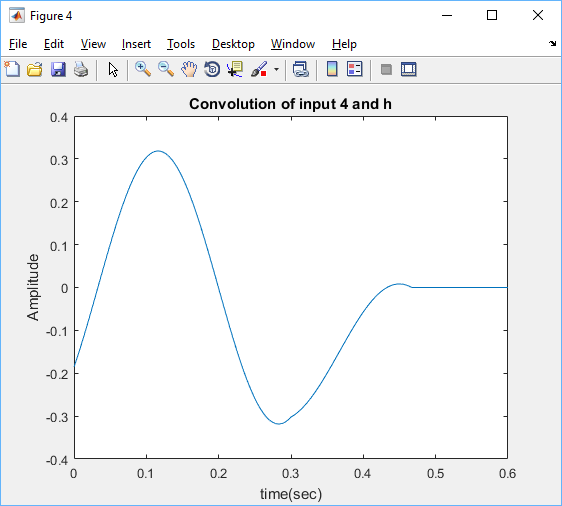


Figure 11: Convolution of input 4 and h.

Section B:

The input frequency in each cause is:

Input 1 = 0 hz

Input 2 = 1 hz

Input 3 = 2 hz

Input 4 = 3 hz

Section C:

With a sampling frequency(fs) = 15hz the number of samples in one period for input signals b through d:

b. n = 15 c. n = 7.5 d. n = 5

The increase in the frequency shows that the period of the function decreases. This causes fewer points to be sampled on the smaller period size.

Section D:

Input 2’s output signal

**Matlab code:**

t=0:0.001:0.6;

% Enter our input values

In1 = 1;

In2 = cos(2\*pi\*t);

In3 = cos(4\*pi\*t);

In4 = cos(6\*pi\*t);

% Transfer function

h = 3.\*rectpuls(3.\*t);

%Using matlab's convolution function, inputs are convolved with the

%transfer function

C1 = conv(In1,h);

C2 = conv(In2,h,'same');

C3 = conv(In3,h,'same');

c4 = conv(In4,h,'same');

%The results of the convolution are then plotted against our set time value

figure

plot(t, C1)

title('Convolution of input 1 and h');

xlabel('time(sec)');

ylabel('Amplitude');

figure

plot(t,C2/1000)

title('Convolution of input 2 and h');

xlabel('time(sec)');

ylabel('Amplitude');

figure

plot(t,C3/1000)

title('Convolution of input 3 and h');

xlabel('time(sec)');

ylabel('Amplitude');

figure

plot(t,c4/1000)

title('Convolution of input 4 and h');

xlabel('time(sec)');

ylabel('Amplitude');

**Part 4:**

Section 1:

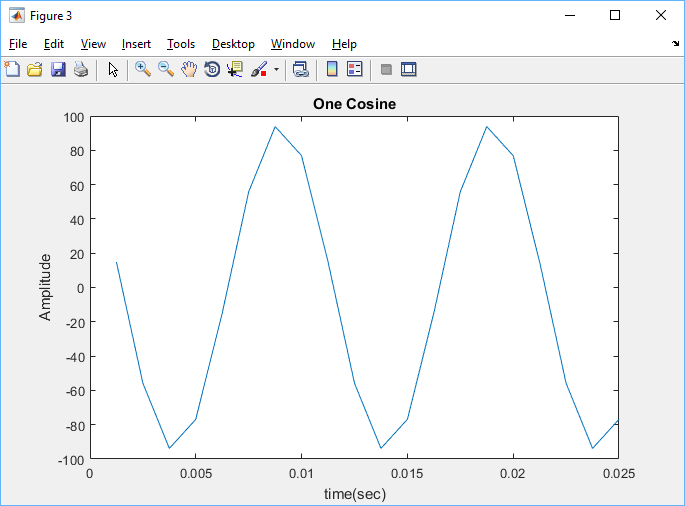


Figure 12: Plot of one\_cos function.

The one\_cos function is just a standard cosine function, however only 20 samples are to be taken over a varying duration. The above figure is to take 20 samples of a cosine function with an amplitude 95, omega 200pi, and a phase angle of pi/5. The duration of this sampling is a max of 0.025 seconds.

**Matlab code:**

Main Code:

A = 95;

w = 200.\*pi;

phi = pi./5;

dur = 0.025;

[X,T] = one\_cos(A,w,phi,dur);

plot(T,X)

xlabel('time(sec)');

ylabel('Amplitude');

title('One Cosine');

Function Code:

function [X,T] = one\_cos(A,w,phi,dur)

%Using the duration and sample size a relation can be made to find

%the sample times. These values are then put into a cosine

%function with the other inputs. This gives us our time values and

%x values.

T = (1:1:20).\*(dur/20);

X = A.\*cos(w.\*T+phi);

end

Section 2:

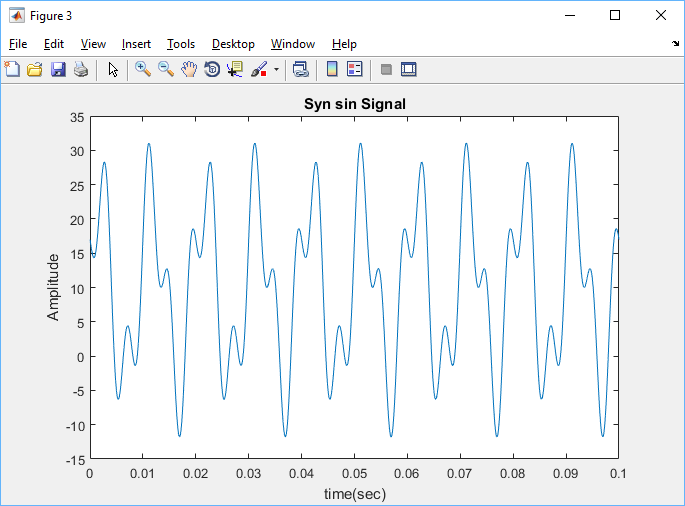


Figure 13: syn\_sin signal generation.

The syn\_sin function generates an output signal that is the combination of multiple different signals that can have varying amplitudes and frequencies. Utilizing built in Matlab functions such as abs and angle, compact amplitudes and phase angles are able to be taken from the complex equations. The amplitudes and phase angles are then placed into an array that sum together in a for loop for the varying frequencies over a given time.

The time vector is found depending on the number of samples requested, the duration of the sampling and the start time. A standard vector that is the length of how many samples is created with values that count up to the max number of samples. A relation of duration divided by sample is then used to keep track of the time a sample is taken plus the original starting time.

Section 3:

Given that it takes 0.1 seconds to finish 5 periods. The time it takes to finish one period is . The periods of the three other sinusoids are undetermined, .01 and .004, given that the frequencies are 0, 100 and 250. The reason why xx0 has a longer period than the other 3 periods is because it is taking the sum of three sinusoids that have varying frequencies. This means that the varying frequencies are being added together to create the longer period.

**Matlab Code:**

Main Code:

[xx0,tt0] = syn\_sin([0,100,250],[10,14\*exp(-j\*pi/3),8\*j],10000,0.1,0);

plot(tt0,xx0)

xlabel('time(sec)');

ylabel('Amplitude');

title('Syn sin Signal')

Function Code:

function [ x,time ] = syn\_sin( fk, Camp, sample, dur, tstart)

% A count variable is created to keep track of for loop for x value

count = 1;

% X variable is loaded in to createa 1 x sample vector for the summed x

% values at specific times.

x=0;

% Determine length of frequency and amplitudes.

l1 = length(fk)

l2 = length(Camp)

% Find incremental time value giving the duration and sample.

IncrementT = dur/sample;

% Set a vector from 1 to sample size as a way to keep track of time value.

Tray = 1:1:sample;

% Find time for which a sample is taken

time = IncrementT\*Tray + tstart;

if nargin < 5, tstart=0, end

%display error message if amplitude and frequency do not work.

if l1 ~= l2

disp('Error, frequency and amplitudes are not the same legnth.')

return;

else

for count = (1:length(fk))

% Using comands angle and abs, the exponential value is used to find

% the phase angle and the amplitude of the complex number.

phi(count) = angle(Camp(count))

Amp(count) = abs(Camp(count))

% The x vector saves the total x values at different phase angles and

% added back together. The real part of the exponential wave function

% is being analysed so cos of the angular frequency and phase shift are

% put into a cosine function and multiplied with the amplitude.

x = x+Amp(count).\*cos(2.\*pi.\*fk(count).\*time + phi(count));

end

end

**Part 5:**

Section A:

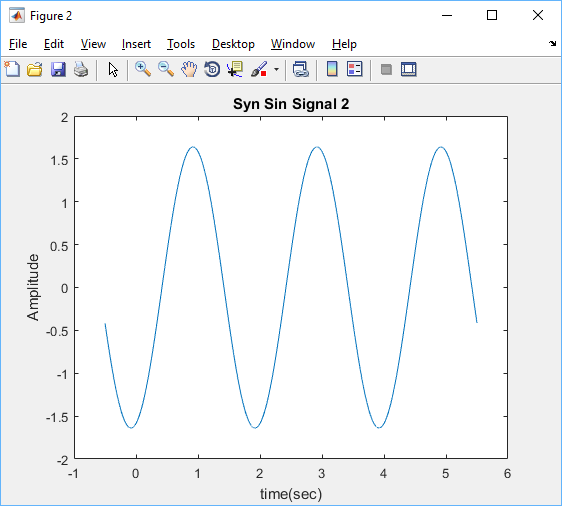


Figure 14: Plot of varying exponential values combined using syn\_sin function

Finding the maximum time duration for 3 full period cycles requires taking the inverse of the frequency (1/2) and multiplying it by 3. This gives the max time interval to be 6 seconds.

Section B:

Using the minimum(-0.5s) and maximum time(5.5s) interval for the sin graph, the frequency can be found by dividing this value by 3 and taking the inverse. This will give the time it takes for the graph to complete one period.

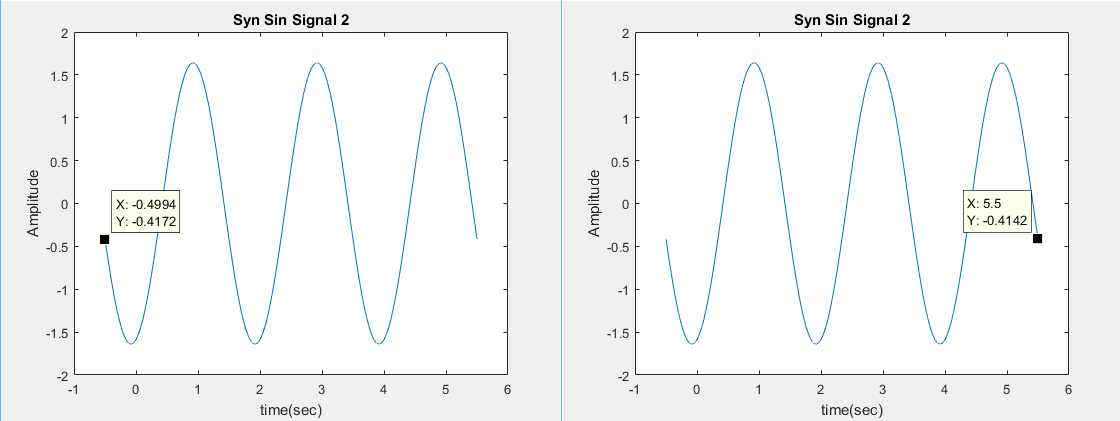


Figure 15: max and min time interval

The phase can be found by finding the time at which the sinusoidal function crosses the t axis. When this is found the equation:

Can be used to calculate the phase angle, however because this is a function using cosine the value pi/2 needs to be added to the multiplication of time and omega.

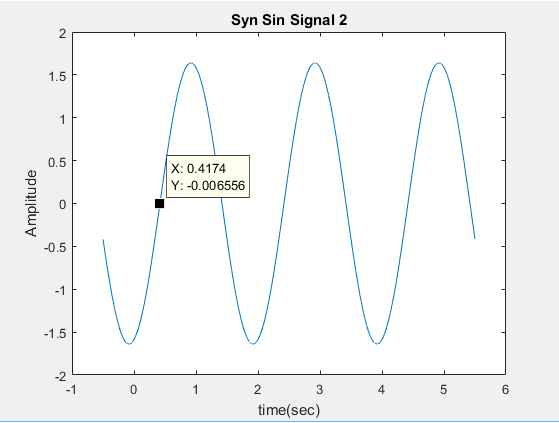


Figure 16: Time at which the graph crosses the t axis.

The Amplitude can be found just by taking the highest y value data point that is represented on the syn sin graph.

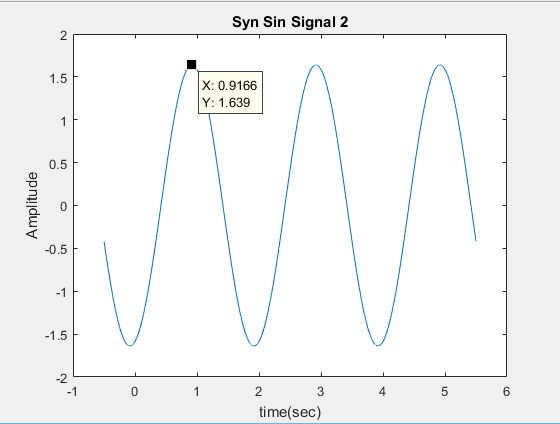


Figure 17: Max amplitude=1.639

Section C:

Using Matlab to add the complex amplitude values and using the built-in functions of abs and angle, the phase angle and magnitude can be found.

Phase = -2.8861 Magnitude = 1.6390.

The calculations in section B has a very similar correlation with section Cs. The main difference is shown in the phase angle, however this is more than likely due to Matlab not taking a sample value at the point the time axis (x-axis) was crossed.

**Matlab Code:**

[xx0, tt0] = syn\_sin([1/2,1/2,1/2],[2.\*exp(j.\*pi),2.\*exp(j.\*pi).\*exp(-j.\*pi.\*1.25),(1-j).\*exp(j.\*pi)], 10000,6.000,-0.5);

plot(tt0,xx0)

xlabel('time(sec)');

ylabel('Amplitude');

title('Syn Sin Signal 2')

%Used to find the phase and Magnitude for section 3

TotalComplex = 2\*exp(j\*pi)+2.\*exp(j.\*pi).\*exp(-j.\*pi.\*1.25)+(1-j)\*exp(j\*pi)

phase = angle(TotalComplex)

Magnitude = abs(TotalComplex)