ELC 433-L1

Lab 1 - Discrete-Time Signals in the Time Domain Using MATLAB

Brian Worts and Chris Jenson

9/22/20



**Introduction:**

This lab covered the basics of MATLAB. Using the command and line, workspace, data types, data structures, indexing, operators, products, and functions effectively to manage the project. The core concept of bandlimited sampling where if the frequency-domain representation of a continuous-time signal is zero for all frequencies above some frequency (fn), then all its info is contained in any set of uniformly spaced samples taken at discrete times, as long as the sampling frequency is twice fn. If a signal is sampled at too low of a sampling rate, then the irreducible distortion of the info is produced by aliasing. Convolution is used to create precise lowpass, bandpass, high pass, all-pass, notch, and arbitrary frequency-selective filters.

**Procedure:**

There were five MATLAB situations to model in this lab. The first was to create and visualize the harmonic series for a square ware, with discrete-time sampling. Using given values, a vector for sampling times, and a single sampled sine wave was created. The 3rd, 5th, 7th, 9th, 11th, and 13th harmonics were sequentially added. The signals were plotted and listened to as created. This process was repeated with a much denser frequency.

Part Two was creating a single sampled sine wave to then subsample via decimation at different factors. Using given values, a vector for sampling times, and a single sampled sine wave was created. A new vector was created for every 6th sample, this also created a new sampling frequency. This was repeated, plotted, and listened to for the 6th, 11th, 16th, 21st, 36th, and 31st samples. The densely sampled plot was layered on top.

Part Three was creating sine waves at increasing frequencies. Using given values, a vector for sampling times, and a single sampled sine wave was created. The densely sampled signal was overlaid. This was repeated with the sine frequency being multiplied by 2,3,4,5,6,7, and 8. Tone frequencies were then changed, and the part was conducted again.

Part Four was reading in a sound file. The sound file was downloaded from Canvas. Using the given code in the handout, the file was read in and the sampling rate was acquired. The file was listened to and then 1000 samples were plotted.

Part Five was to perform digital lowpass filtering. The filter coefficients were downloaded from Canvas. The filter was read into Num and the filter was applied using a built-in function. The convolution was then performed in an old-school way. With both methods performed, they were compared.

**Design/Engineering Work:**

**Part 1 Code:**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Brian Worts

%Lab 1

%ELC 433-L1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sigDuration = 2; %Seconds

samplingFreq = 8000; %Hz

samplingFreqDense = 80000;

toneFreq = 440; %Hz

samplingPeriod = 1/samplingFreq; %T=1/f - Sampling Freq be > 2\*toneFreq

samplingTimes = 0:samplingPeriod:sigDuration; %Vector between 0 and sigDuration by samplingPeriod

samplingPeriodDense = 1/samplingFreqDense;

samplingTimesDense = 0:samplingPeriodDense:sigDuration;

%sound(soundSignal) %Play sound

for plots=1:4:13 %Plots for 3&5, 3&5&7&9, 3&5&7&9&11&13

soundSignal = sin(2\*pi\*toneFreq\*samplingTimes); %Generate Sin wave, reset sound signal after iteration

soundSignalDense = sin(2\*pi\*toneFreq\*samplingTimesDense);

figure(plots) %Different figure each iteration

for i=3:2:plots %Harmonics 3 and 5, then 3 and 5 and 7 and 9, then...

soundSignal = soundSignal + sin(2\*pi\*toneFreq\*samplingTimes\*i)/i; %Add harmonic to signal

soundSignalDense = soundSignalDense + sin(2\*pi\*toneFreq\*samplingTimesDense\*i)/i;

%sound(soundSignal)

end

numSamples = 2\*samplingFreq/toneFreq;

plot(samplingTimesDense(1:numSamples\*10),soundSignalDense(1:numSamples\*10))

hold on;

stem(samplingTimes(1:numSamples), soundSignal(1:numSamples))

title(strcat(num2str(plots), ' is highest Harmonic in Signal'))

legend('Dense Sampling', 'Regular Sampling')

xlabel('Time')

ylabel('Signal Value')

hold off;

end

**Part 2 Code:**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Brian Worts

%Lab 1

%ELC 433-L1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

sigDuration = 2;

SamplingFreq = 48000;

Ts = 1/SamplingFreq; %Sampling Period

toneFreq = 1000;

samplingTimes = 0:Ts:sigDuration; %Sample from 0 to sigDuration by Ts

W = sin(2\*pi\*toneFreq\*samplingTimes); %Signal equation

plotMax = 1:500;

for sampleDivider = 1:5:31

plotSubMax = plotMax(1:sampleDivider:end);

newSignal = W(plotSubMax); %Signal at new sampling rate

figure(sampleDivider); %Names figure accordingly

plot(plotMax, W(plotMax)) %Original Signal

hold on

title(sampleDivider)

stem(plotSubMax,newSignal); %Stem plot sampled at division

legend(strcat('1/',num2str(sampleDivider), ' Sampling'),'Regular Sampling')

hold off

end

**Part 3 Code**

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Chris Jenson

%Lab 1

%ELC 433-L1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dur = 2; % Signal duration of 2 seconds

Fs = 8000; % Sampling frequency in Hz

FsDense = 80000; % The dense sampling frequency

Ts = 1.0/Fs; % Based off the sampling frequency

TsDense = 1/FsDense; % Dense version of Ts

num\_Samps = int32(dur \* Fs);

sampTimes = double(0:(num\_Samps-1)) \* Ts; % Vector of sampling times for the duration

sampTimesD = double(0:10\*num\_Samps-1) \* Ts/10; % Vector of dense sampling times

numPlot = 100;

for i = 1:8 % Part 3: multiplying the frequency of 995

Fo = 995 \* i;

W = sin(2\*pi\*Fo\*sampTimes); % Single sampled sine wave

WDense = sin(2\*pi\*Fo\*sampTimesD); % Dense single sampled sine wave

%sound(W) % Play the sine wave

figure(i)

plot(sampTimesD(1:numPlot\*10), WDense(1:numPlot\*10)) % Adding the dense plot

hold on

stem(sampTimes(1:numPlot), W(1:numPlot)); % Adding the single sample plot

hold off

xlim([sampTimes(1) sampTimes(numPlot)]);

ylim([-2 2]);

title( strcat('Fo of 995\* ', num2str(i)))

xlabel( 'Time(s)' );

end

for j = 1:8 % Part 4: multiplying the frequency of 1000

Fo = 1000 \* j;

W = sin(2\*pi\*sampTimes\*Fo); % Single sampled sine wave

WDense = sin(2\*pi\*sampTimesD\*Fo); % Dense single sampled sine wave

%sound(W) % Play the sine wave

figure(j)

plot(sampTimesD(1:numPlot\*10), WDense(1:numPlot\*10)) % Adding the dense plot

hold on

stem(sampTimes(1:numPlot), W(1:numPlot)); % Adding the single sample plot

hold off

xlim([sampTimes(1) sampTimes(numPlot)]);

ylim([-2 2]);

title( strcat('Fo of 1000\* ', num2str(j)))

xlabel( 'Time(s)' );

end

**Parts 4 & 5 Code**

% Part 4

[y, Fs] = audioread('good\_news.wav'); % Reading in the wav file

y = y(:,1)'; % Taking the left channel and transposing to row vector

y = y / max(y); % Normalizing to a max value of 1.0

%sound(y)

figure(1) % Creating the figure

title('good news plot') % Adding a title

hold on

plot(y(25000:26000)) % Adding the plot

hold off

% Part 5

load('filt\_3300\_48000\_coeffs.mat'); % Load in the lowpass filter

yf = conv(y, Num, 'same'); % 1kHz cutoff filter

yf = yf / max(y); % Normalize to max value of 1

% Tough convolution time

% Pad the waveform

impulse\_resp = Num;

impulse\_resp\_len = length(impulse\_resp); % Get the length

num\_padding = impulse\_resp\_len - 1; % Determine num of padding

left\_padding = (impulse\_resp\_len-1)/2; % Calc left side

right\_padding = num\_padding - left\_padding; % Calc right side

y\_pad = [zeros(1, left\_padding), y, zeros(1, right\_padding)]; % Pad both sides w/ 0's

num\_samples = length(y); % Get the number of samples

yf1 = zeros(1, num\_samples); % Will be the result, start with zeros

for n = 1:num\_samples

% Keep a running tally

sum = 0;

% For every value

for j = 0:num\_padding

% Add the product of the two vector to the current sum

sum = sum + impulse\_resp(impulse\_resp\_len - j) \* y\_pad(n + j);

end

yf1(n) = sum; % Store into the result

end

yf1 = yf1/max(yf1); % Normalizing to max val of 1

% Do the comparison

fprintf("Max Difference of yf1 vs conv(): %g\n", max(abs(yf-yf1)));

**Results:**

Part 1

* Questions
  + For each of the steps above, does the sampling satisfy the Nyquist criterion? Why or why not?
    - The Nyquist Criterion is met for the 3rd, 5th, 7th, and 9th harmonics since their frequencies are below 2\*sampling Rate. The 11th and 13th harmonics give the signal frequencies that are above twice the sampling rate.
    - The 80,000Hz sampling frequency causes all harmonics to satisfy the Nyquist criterion.
  + Were you able to hear the implications of aliasing?
    - Aliasing could be heard most strongly for the 11th and 13th harmonics where the frequency was clearly lower
  + How can you relate the sampling rates vs. frequency of wiggles in harmonic series with aliasing?
    - When the Nyquist criterion is not met for the frequencies of the wiggles than aliasing is most apparent

Part 2

* Questions
  + For each of the steps above, does the sampling satisfy the Nyquist criterion? Why or why not?
    - Nyquist Criterion is met for the initial sampling frequency, 1/6th, 1/11th, 1/16th, and 1/21st the initial sampling frequency. This is because these sampling frequencies remained above 2\*toneFreq. The opposite was true for 1/26th and 1/31st the sampling frequency.
  + Were you able to hear the implications of aliasing?
    - Aliasing was most obvious for 1/26th and 1/31st the original sampling frequencies where the sound was clearly a lower frequency
  + How can you relate the sampling rates vs. frequency of a sine wave with aliasing?
    - The lower the sampling rate compared to the frequency of the sampled wave, the more likely it is aliasing will occur

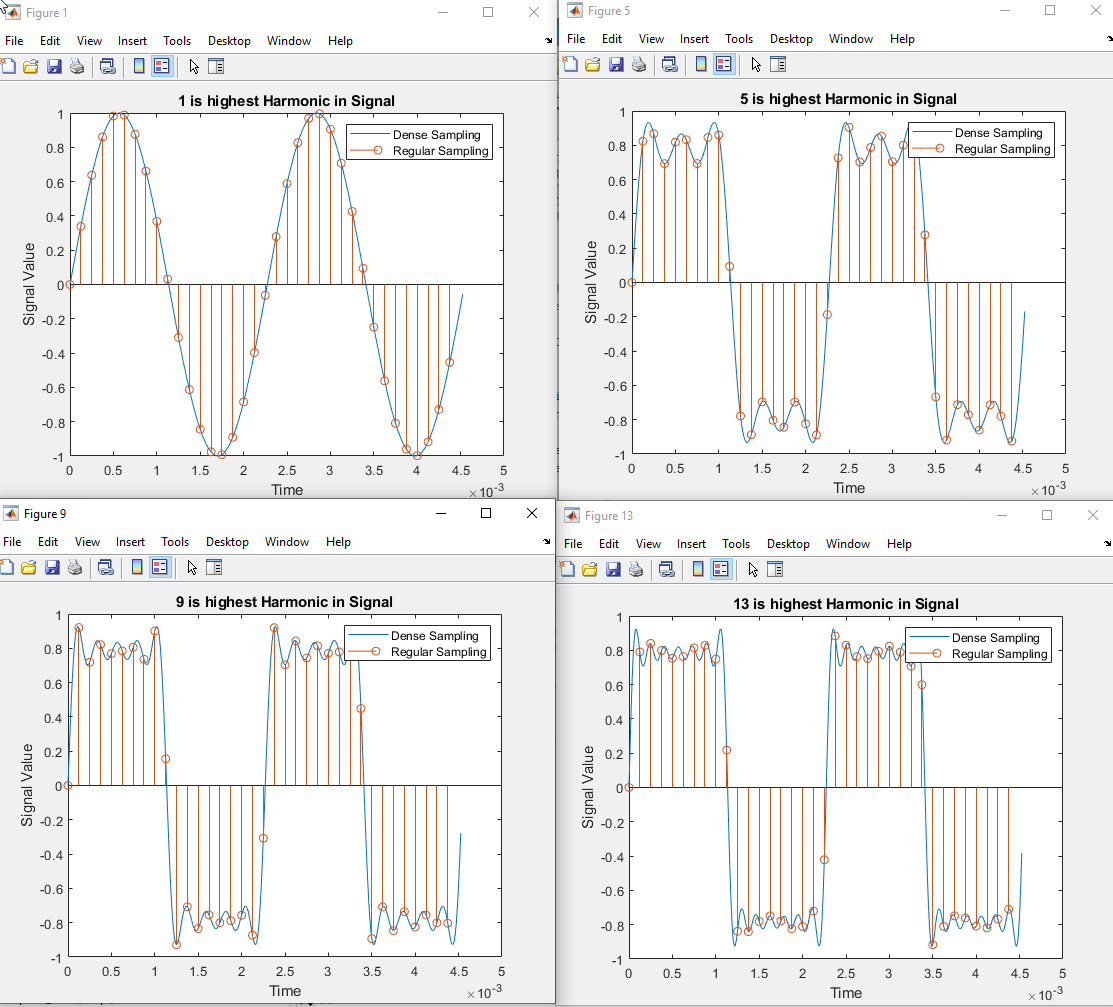
Part 3

* Questions
  + At what point did the sounds go from increasing frequency to Decreasing frequency, and why?
    - The sounds go to decreasing frequency about when the sine frequency reaches a multiplier of greater than 4. This is because the sampling rate is no longer sufficient to sample that frequency.
  + What major difference did you observe between 𝑓0 = 995 Hz and 𝑓0= 1000 Hz, and why?
    - The tone frequency impacts the sound. Once the tone frequency is double that of the fundamental frequency, it becomes the second harmonic which is an octave higher.
  + Why do you think that you couldn’t hear any sound when 𝑓0= 8\*995 Hz, even though the signal was nonzero?
    - Around that point, the sound has reached an octave higher than what humans can hear.

Part 4&5

* Questions
  + Explain, intuitively, the action of the convolution operator.  How does it combine two sequences, to produce a third sequence?  For your example you can make the first sequence a sampled sine wave, and the 2Nd sequence a short Tent sequence:
    - Convolution can be thought of similarly to listening to a voice through a wall. While listening, you hear what is being said as well as what was said moments ago. Meaning you are hearing a combination of the current output in addition to the previous. Using these iterations, it is possible to reconstruct the voice. Given a negative value, it would be hypothetical to hear before the speaking. Using the given example, the sine wave could be used to make the tent using the samples.

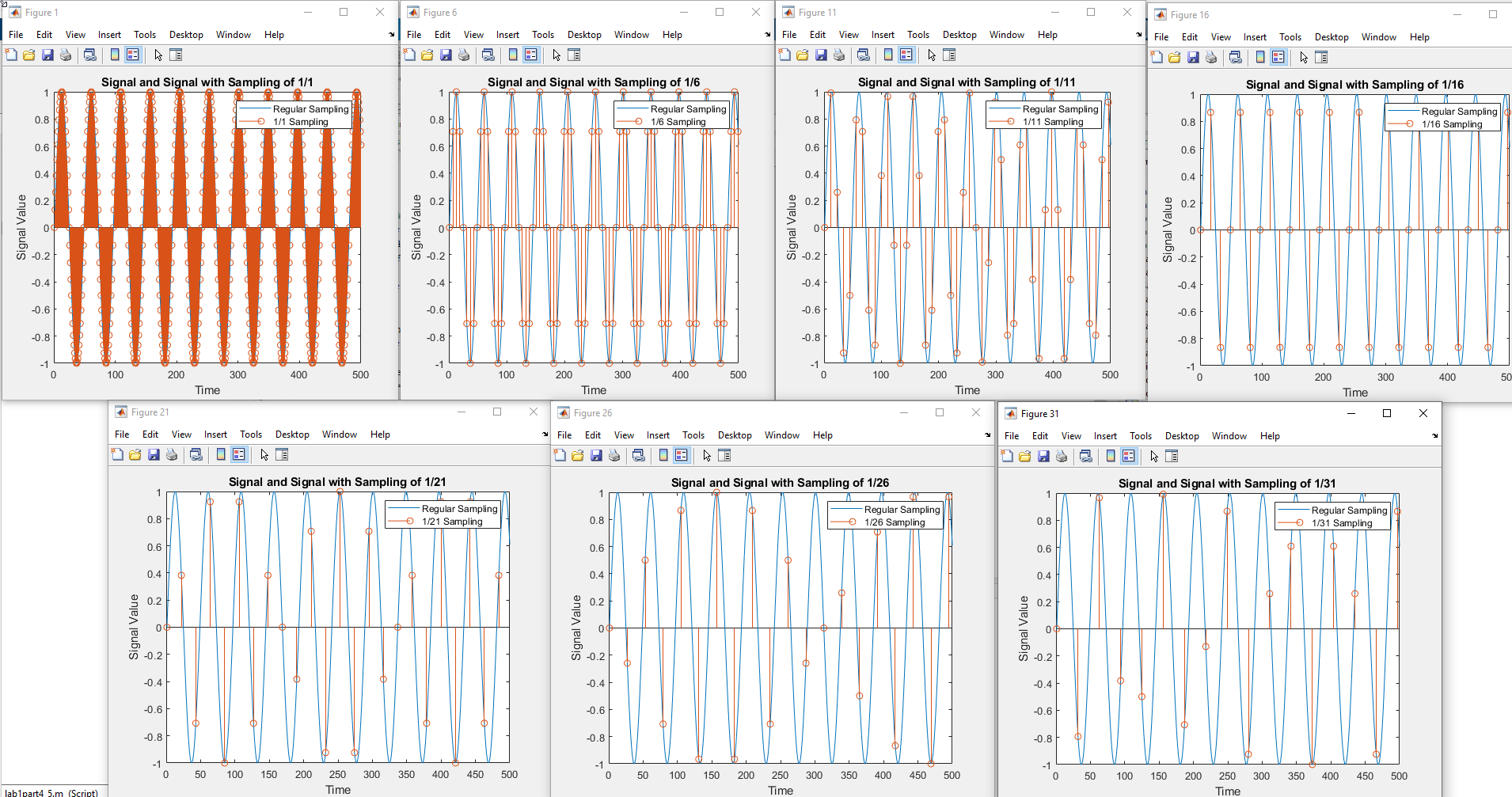
**Results** **(Part 1):**



Harmonic series representation of a square wave sampled at 8KHz(Regular) and 80KHz(Dense).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Description | Max Freq (Hz) | Sampling Rate | Nyquist Met? | Sound |
| 1st Harmonic | 440 | 8k | Yes | Normal |
| 3rd Harmonic | 1320 | 8k | Yes | Normal |
| 5th Harmonic | 2200 | 8k | Yes | Normal |
| 7th Harmonic | 3080 | 8k | Yes | Normal |
| 9th Harmonic | 3960 | 8k | Yes | Normal |
| 11th Harmonic | 4840 | 8k | No | Low |
| 13th Harmonic | 5720 | 8k | No | Low |

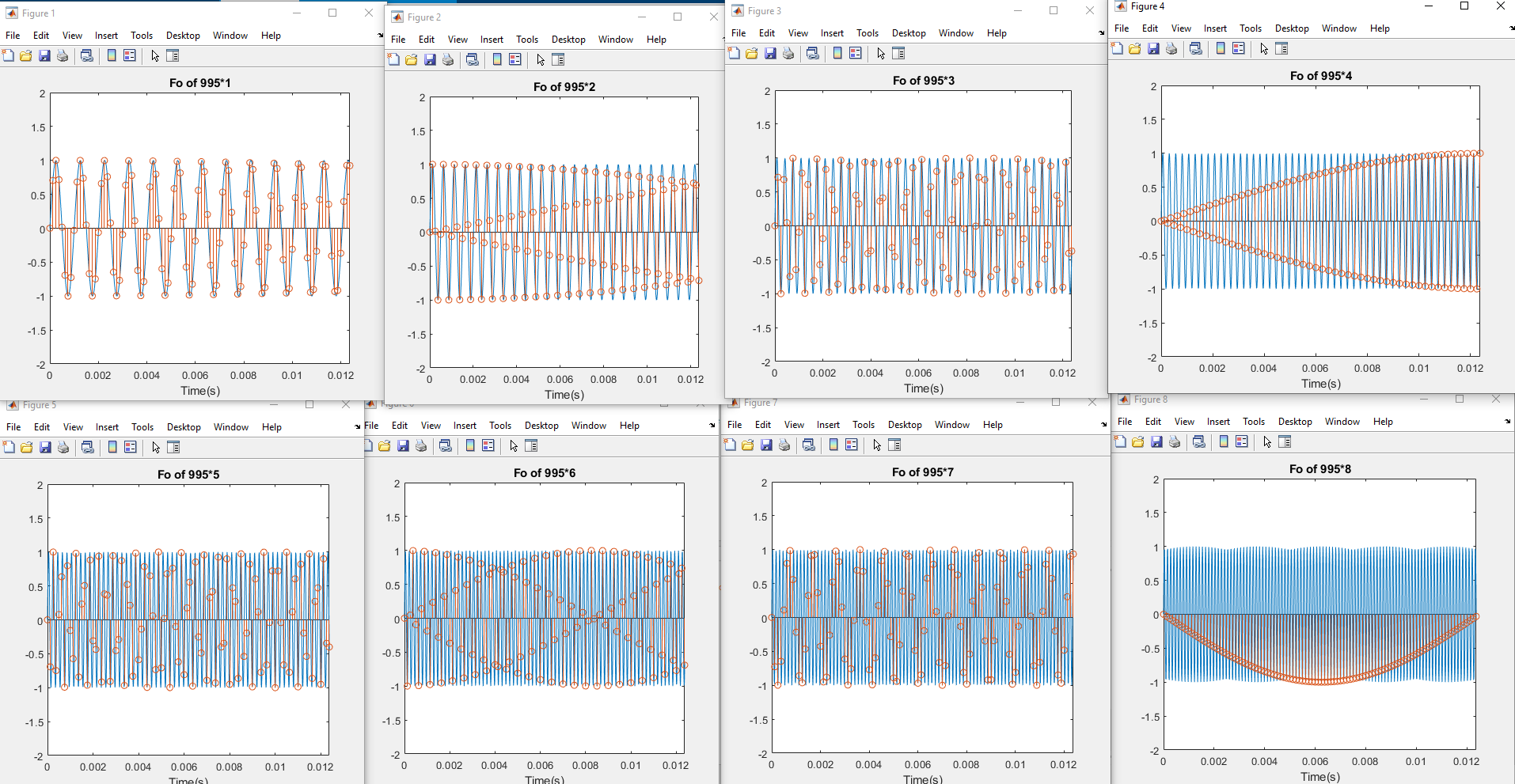
**Results (Part 2):**



Sine wave sampled at 1/1 Sampling (top left) to 1/31 Sampling (bottom right) compared to the nearly continuous signal (blue).

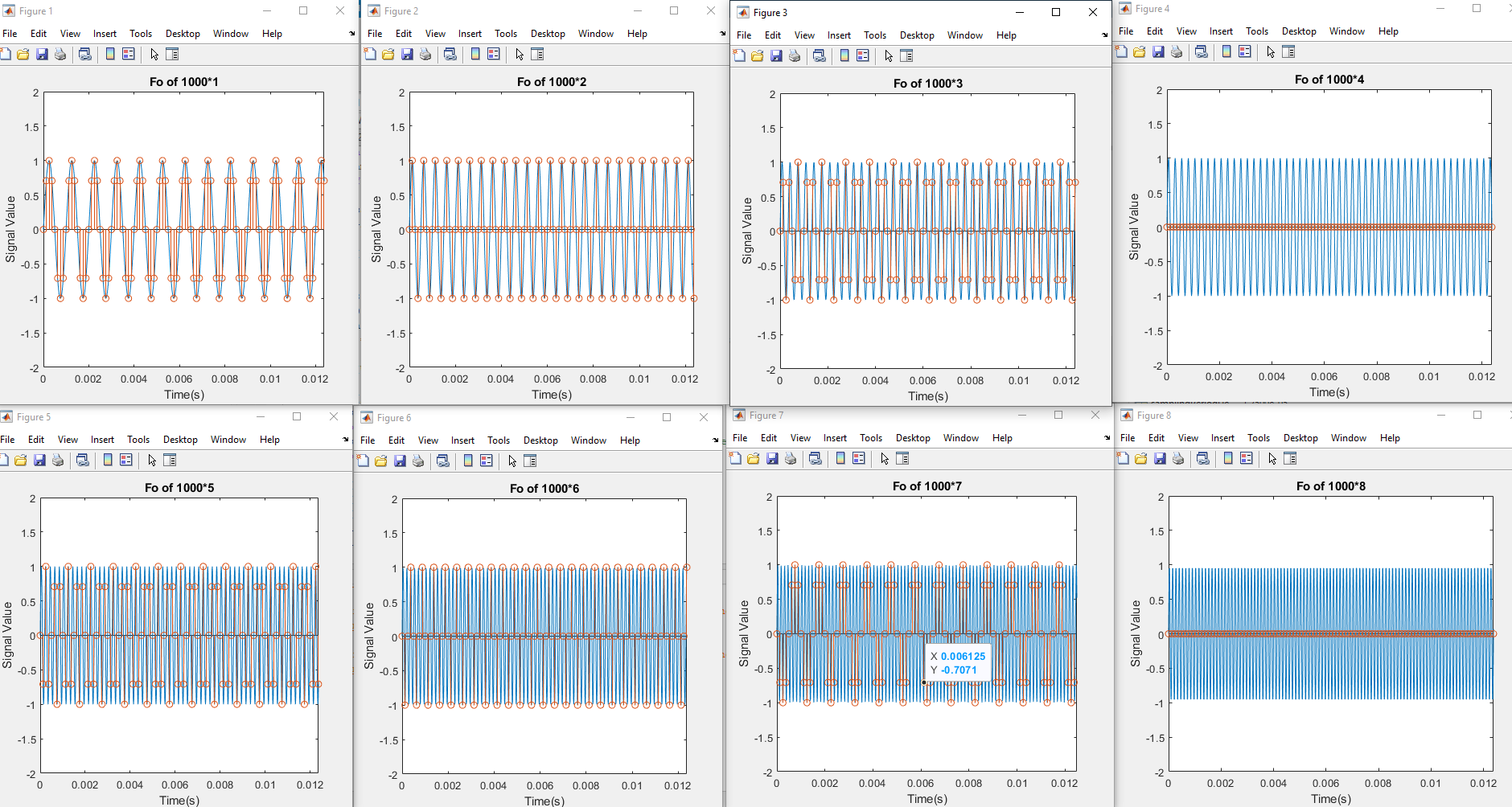
|  |  |  |  |
| --- | --- | --- | --- |
| Description | Effective Sampling Rate (Hz) | Nyquist Met? (Tone freq = 1000Hz) | Sound |
| Normal Sampling Rate | 48000 | Yes | Normal |
| 1/6 Sampling Rate | 8000 | Yes | Normal |
| 1/11 Sampling Rate | 4367 | Yes | Normal |
| 1/15 Sampling Rate | 3200 | Yes | Normal |
| 1/21 Sampling Rate | 2286 | Yes | Normal |
| 1/26 Sampling Rate | 1846 | No | Low |
| 1/31 Sampling Rate | 1548 | No | Low |

**Results (Part 3 Step 3):**

****

Sine wave of varying tone frequencies sampled at 8000Hz and 80000Hz.

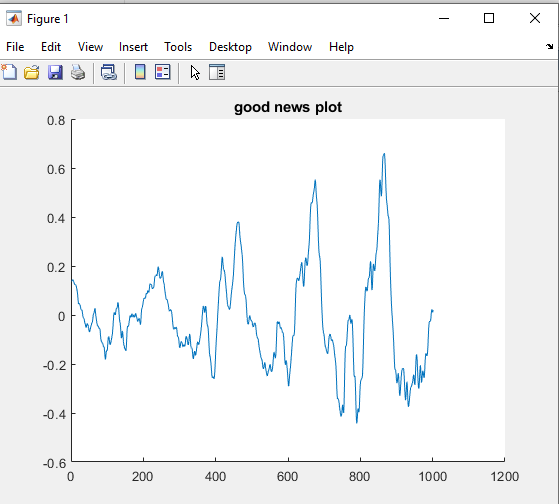
**Results (Part 3 Step 4):**



Sine wave of varying tone frequencies sampled at 8000Hz and 80000Hz. Note 4th graph is directly at Nyquist resulting in very poor representation of the continuous signal.

|  |  |  |  |
| --- | --- | --- | --- |
| Signal Frequency (Hz) | Sampling Rate (Hz) | Nyquist Met? | Sound |
| 995 | 8000 | Yes | Nearly the same |
| 1990 | 8000 | Yes | Nearly the same |
| 2985 | 8000 | Yes | Nearly the same |
| 3980 | 8000 | Yes | Nearly the same |
| 4975 | 8000 | No | Lower Freq |
| 5970 | 8000 | No | Lower Freq |
| 6965 | 8000 | No | Lower Freq |

**Results (Part 4):**



“Good News” sound signal graphed.

**Results (Part 5):**

Results of convolution

Max Difference of yf1 vs conv(): 0.0157655

**Knowledge Gained:**

The students both gained knowledge in using MATLAB as a tool to sample signals. The impact of sampling and tone frequency was heard and observed. The ability to use external files as input and as a filter was exercised. Skills like convolution were demonstrated using multiple methods. In addition to this, the students' existing understanding of various MATLAB functionalities was increased.

**Who Did What:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Student** | **Analysis** | **Development** | **Coding** | **Results** | **Writing** |
| Brian | 50 | 50 | 50 | 75 | 25 |
| Chris | 50 | 50 | 50 | 25 | 75 |