ECE1563: Lab Report
Lab 3: Fourier Analysis of Discrete-Time Signals and Systems

Nate Carnovale and Seth So
Lab Group: 1
TA: Kechen Shu

August 15, 2020

Introduction

In digital processing, computers can only process information in discretized, finite quantities. Therefore analog inputs must be converted to digital before any operators are applied. Theoretically this can be accomplished using the Z-transform to convert the mathematical representation of signal from the time domain to the discretized frequency domain. Further analysis makes use of the Fourier-transform, which is equivalent to the Z-transform evaluated at frequency components $z = e^{jw}$.

The purpose of Laboratory 3 was to understand the Fourier-transform mathematically as well as its practical application in discrete-time signals and systems. The practical example examined in the lab was voice recognition.

Our approach to the lab was to calculate the signal conversions mathematically by hand and then implement in MATLAB. Observations would be made after each controlled change of various parameters by looking at the frequency plots of the figures to see the effect of each parameter on the output. These observations were then compared with expectations from the manual calculations to help develop a better understanding of how theoretical changes are realized in practical systems.

*[BONUS] See line 349 of **carnovale_so_ece1563lab3.m** for self implemented voice recognition

Methodology

Experimental Design

The majority of Lab 3 was completed in MATLAB. Directions from the design problem document were followed to plot certain signals and examine particular characteristics of the Fourier-transform. For certain steps, the transform was solved analytically to confirm results and observe the properties from a mathematical perspective.

Important concepts explored were: the effects of up/down sampling (pts 1-4), effect of sampling rate and frequency resolution on Fourier plots (pts 5-13), the effects of zero padding for aperiodic signals (14-15), and the practical application of voice recognition (frequency for feature extraction, pts 16-17)

Protocol for Collecting Data

As mentioned above, signals were both plotted in MATLAB and calculated analytically. Observations were made from both the magnitude spectra and mathematical results to generate an understanding of the effects of changing various parameters (see last section). Laboratory instructions did not require any special procedures except self implementation of a voice classifier for extra credit.

For the voice recognition procedure, overall 30 voice recordings were collected: 10 "One"s and 10 "Two"s as training set, plus an additional 5 "One"s and 5 "Two"s as a test set. Recordings were taken over iOS13 memos app, manually trimmed, and exported to MATLAB as a .m4a audiofile. Results were tested two ways: 1. With the built in MATLAB functions fitcknn and predict, and 2. with a self-implemented version of KNN.

Data Analysis

Following are a list of expected characteristics taken from lecture notes: 1. Magnitude peaks in the magnitude spectra were expected to occur at 1/2 the amplitude of the time signal component, 2. Upsampling should result in frequency compression, while downsampling should result in frequency expansion, 3. Periodic signals result in an impulse at their frequencies in the frequency domain, 4. Imperfect (non-integer) fundamental frequency to frequency resolution ratio results in a reduction/obscured amplitude, 5. Zero padding should increase resolution as well as introduce noise into readings.

For the voice recognition classifier, proper functionality was verified by 1. Randomizing the order of the test set, and 2. Changing the number of features to confirm expected effects (see discussion (17) for detail).

Results

(1). Given the following signal, sampling at f$_s$ = 512 Hz for 2 seconds, the following amplitude spectra were observed:

$$x_a(t) = 3cos(10\pi t + \frac{\pi}{4})$$
$$x_a[n] = 3cos(\frac{10\pi n}{512} + \frac{\pi}{4})$$

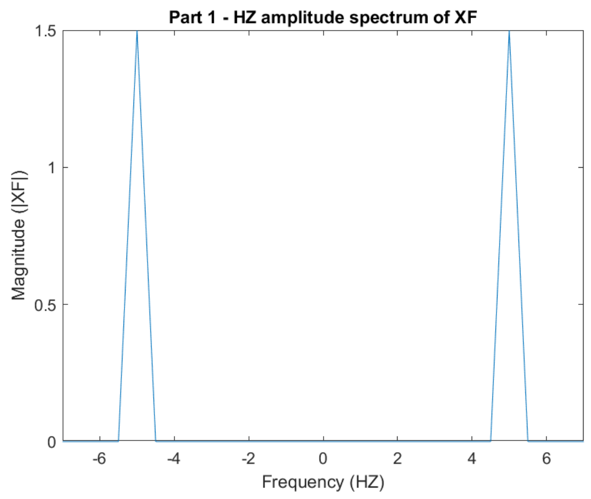


Figure 1: Plot of x$_a(t)$ amplitude spectrum in Hz

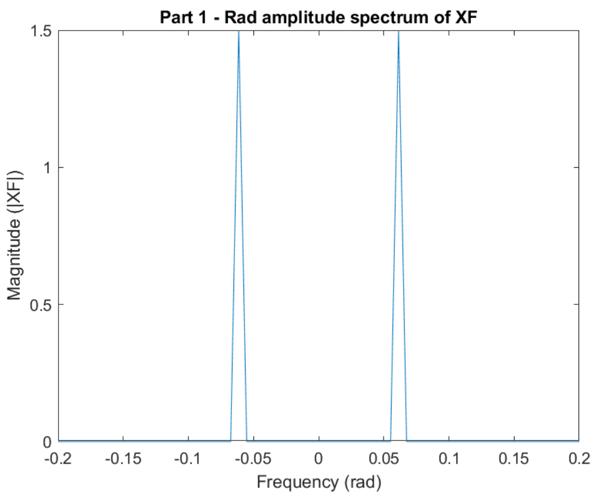


Figure 2: Plot of x$_a(t)$ amplitude spectrum in radians

(2). Given the following signal, downsampling at f$_s$ = $\frac{512}{2}$ Hz for 2 seconds, the following amplitude spectra were observed:

$$x_a(t) = 3cos(10\pi t + \frac{\pi}{4})$$
$$x_a[n] = 3cos(\frac{10\pi n}{512/2} + \frac{\pi}{4})$$

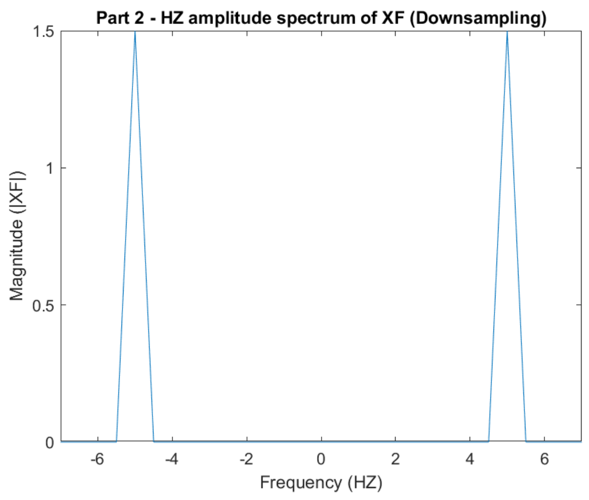


Figure 3: $x_a(t)$ amplitude spectrum in Hz (downsampling)

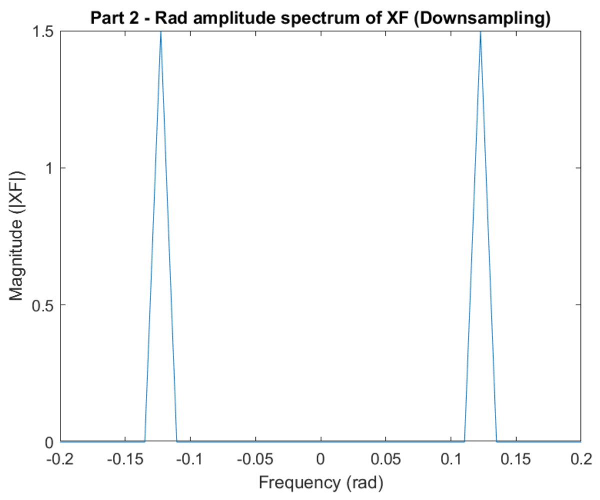


Figure 4: $x_a(t)$ amplitude spectrum in radians (downsampling)

(3). Given the following signal, upsampling at f_s = 512*3 Hz for 2 seconds, the following amplitude spectra were observed:

$$x_a(t) = 3cos(10\pi t + \frac{\pi}{4})$$
$$x_a[n] = 3cos(\frac{10\pi n}{512*3} + \frac{\pi}{4})$$

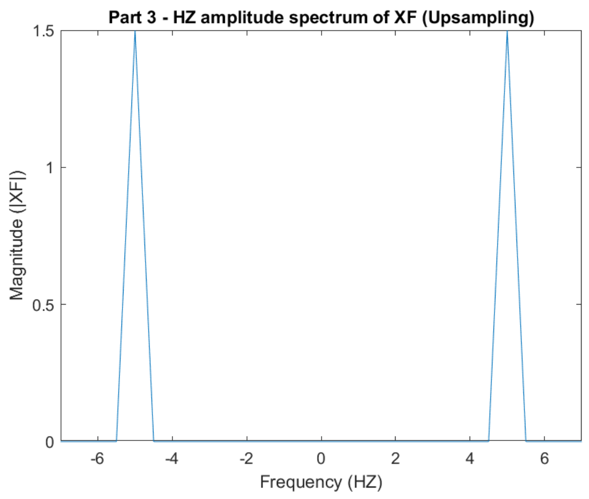


Figure 5: $x_a(t)$ amplitude spectrum in Hz (upsampling)

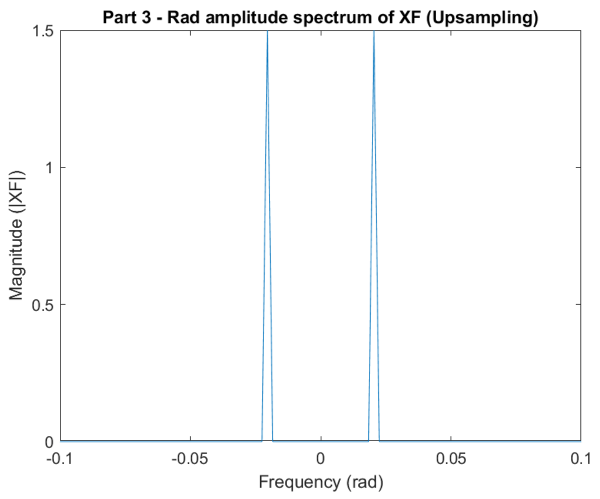


Figure 6: $x_a(t)$ amplitude spectrum in radians (upsampling)

(4). See discussion on upsampling and downsampling in the discussion section.
(5). The following signal was defined in MATLAB:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(121\pi nT_s/2) \qquad \text{for } 0 \leq n \leq N-1,\ T_s = 1/128 \text{ seconds}$$

(6). Derived DTFT of the signal x[n] analytically shown below:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(121\pi nT_s/2) \quad \text{for } 0 \leq \text{n} \leq \text{N-1, T}_s = 1/128 \text{ seconds}$$
$$x[n] = sin(\frac{20\pi n}{128}) + sin(\frac{11\pi n}{128}) + cos(\frac{121\pi n}{128 * 2})$$
$$X(e^{j\omega}) = -j\pi[\delta(\omega - \frac{20\pi}{128}) - \delta(\omega + \frac{20\pi}{128})] - j\pi[\delta(\omega - \frac{11\pi}{128}) - \delta(\omega + \frac{11\pi}{128})] + \pi[\delta(\omega - \frac{121\pi}{256}) - \delta(\omega + \frac{121\pi}{256})]$$
$$|X(e^{j\omega})| = \pi\delta(\omega + \frac{121\pi}{256}) + \pi\delta(\omega + \frac{20\pi}{128}) + \pi\delta(\omega + \frac{11\pi}{128}) + \pi\delta(\omega - \frac{11\pi}{128}) + \pi\delta(\omega - \frac{20\pi}{128}) + \pi\delta(\omega - \frac{121\pi}{256})$$
$$|X(e^{j\omega})| = \boxed{\pi\delta(\omega + 1.485) + \pi\delta(\omega + 0.491) + \pi\delta(\omega + 0.270) + \pi\delta(\omega - 0.270) + \pi\delta(\omega - 0.491) + \pi\delta(\omega - 1.485)}$$

(7). Given the following signal, sampling at f$_s$ = 128 Hz, let N = 128, the following amplitude spectrum was observed:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(\frac{121\pi nT_s}{2}), \quad \text{for } 0 \leq \text{n} \leq \text{N-1}$$

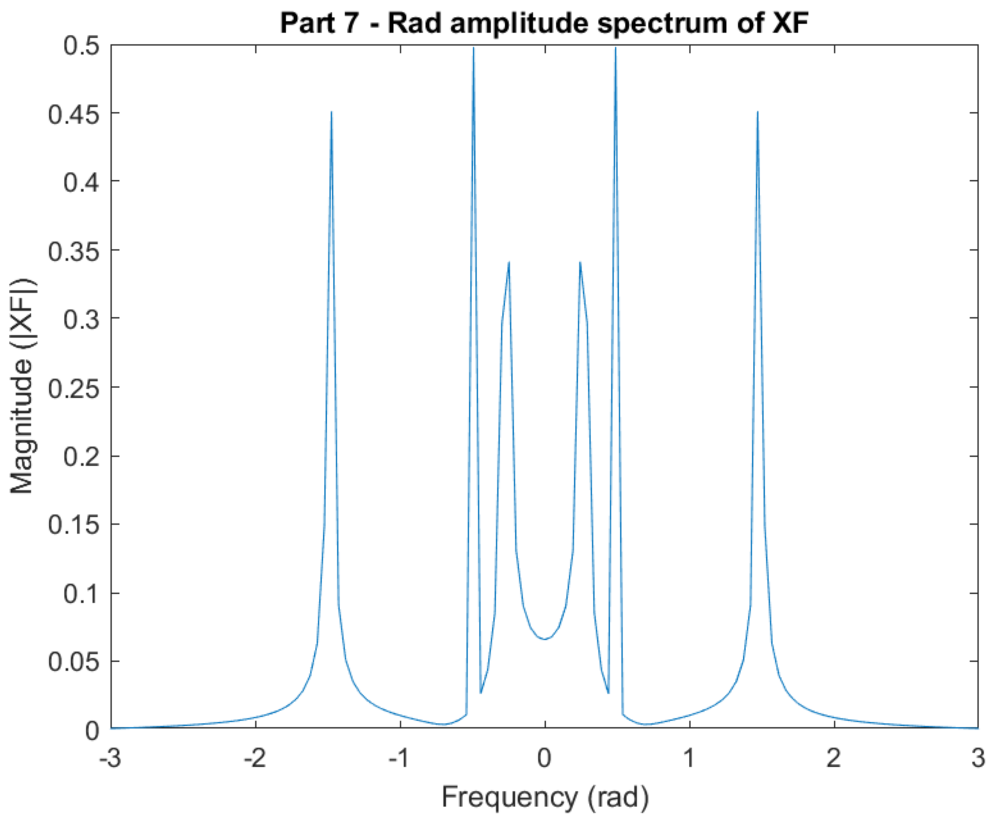


Figure 7: Plot of x[n] amplitude spectrum for f$_s$ = 128 Hz with N = 128

(8). Given the following signal, sampling at f$_s$ = 128 Hz, now choose N = 256 to accurately detect the frequency of the second component:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(\frac{121\pi nT_s}{2}), \quad \text{for } 0 \leq \text{n} \leq \text{N-1}$$

Determining the correct value of N:

From the signal we have the following signal frequencies:

$$f_1 = \frac{\omega}{2\pi} = \frac{20\pi}{2\pi} = 10 \text{ Hz}$$
$$f_2 = \frac{\omega}{2\pi} = \frac{11\pi}{2\pi} = 5.5 \text{ Hz}$$
$$f_3 = \frac{\omega}{2\pi} = \frac{121\pi/2}{2\pi} = 30.25 \text{ Hz}$$

To get the second component, we need to account for the half frequencies since $f_2 = 5.5$Hz. Therefore, multiply N by 2 to get $\boxed{N = 256}$.

The following amplitude spectrum was observed:

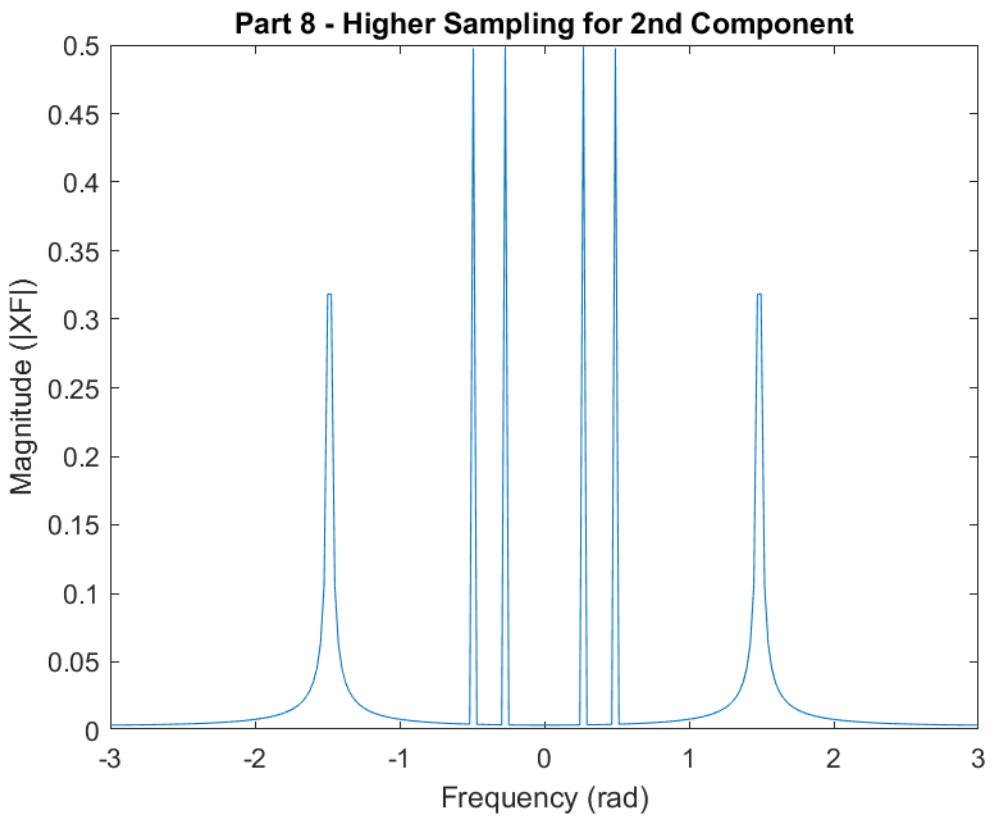


Figure 8: Plot of x[n] amplitude spectrum for f_s = 128 Hz with N = 256

(9). Given the following signal, sampling at f_s = 128 Hz, now choose N = 512 to accurately detect the frequency of all three components:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(\frac{121\pi nT_s}{2}), \quad \text{for } 0 \leq \text{n} \leq \text{N-1}$$

Determining the correct value of N:

From the signal we have the following signal frequencies:

$$f_1 = \frac{\omega}{2\pi} = \frac{20\pi}{2\pi} = 10 \text{ Hz}$$
$$f_2 = \frac{\omega}{2\pi} = \frac{11\pi}{2\pi} = 5.5 \text{ Hz}$$
$$f_3 = \frac{\omega}{2\pi} = \frac{121\pi/2}{2\pi} = 30.25 \text{ Hz}$$

To get the second component, we need to account for the quarter frequencies since $f_3 = 30.25$Hz. Therefore, multiply N by 4 to get $\boxed{N = 512}$.

The following amplitude spectrum was observed:

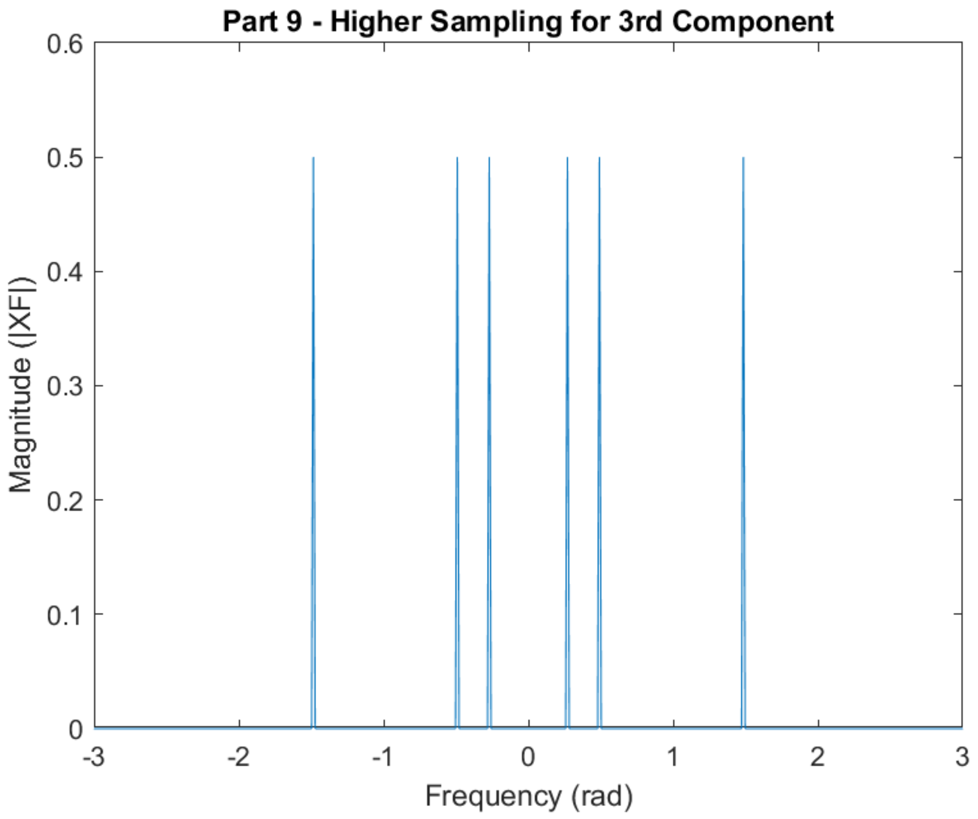


Figure 9: Plot of x[n] amplitude spectrum for f_s = 128 Hz with N = 512

(10). Given the following signal, sampling at f_s = 128 Hz, now assume N = 1024:

$$x[n] = sin(20\pi nT_s) + sin(11\pi nT_s) + cos(\frac{121\pi nT_s}{2}), \quad \text{for } 0 \leq n \leq N\text{-}1$$

Determine frequency resolution:

$$\begin{aligned} \text{resolution} &= \frac{2\pi}{N} \\ &= \frac{2\pi}{1024} \\ &= \boxed{0.006 \text{ rad}} \end{aligned}$$

The following amplitude spectrum was observed:

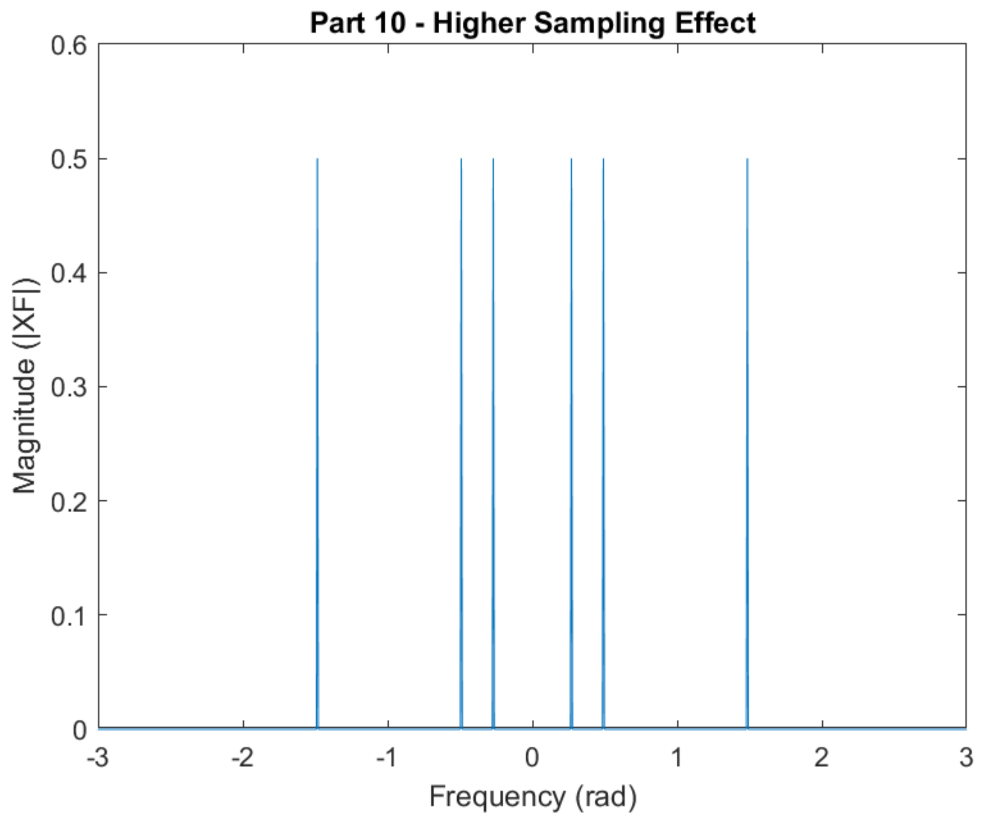


Figure 10: Plot of x[n] amplitude spectrum for f$_s$ = 128 Hz with N = 1024

(11). The following signal was defined in MATLAB:

$$x[n] = sin(30\pi n T_s) \quad \text{for } 0 \leq \text{n} \leq \text{N-1, T}_s = 1/128 \text{ seconds}$$

(12). Derived DTFT of the signal x[n] analytically shown below:

$$\begin{aligned}
x[n] &= sin(30\pi n T_s) \quad \text{for } 0 \leq \text{n} \leq \text{N-1, T}_s = 1/128 \text{ seconds} \\
x[n] &= sin(\frac{30\pi n}{128}) \\
X(e^{j\omega}) &= -j\pi[\delta(\omega - \frac{30\pi}{128}) - \delta(\omega + \frac{30\pi}{128})] \\
|X(e^{j\omega})| &= \pi\delta(\omega + \frac{30\pi}{128}) + \pi\delta(\omega - \frac{30\pi}{128}) \\
|X(e^{j\omega})| &= \boxed{\pi\delta(\omega + 0.763) + \pi\delta(\omega - 0.763)}
\end{aligned}$$

(13). Given the following signal, sampling at f$_s$ = 128 Hz, N = 128 and the amplitude spectrum was observed:

$$x[n] = sin(30\pi n T_s), \quad \text{for } 0 \leq \text{n} \leq \text{N-1}$$

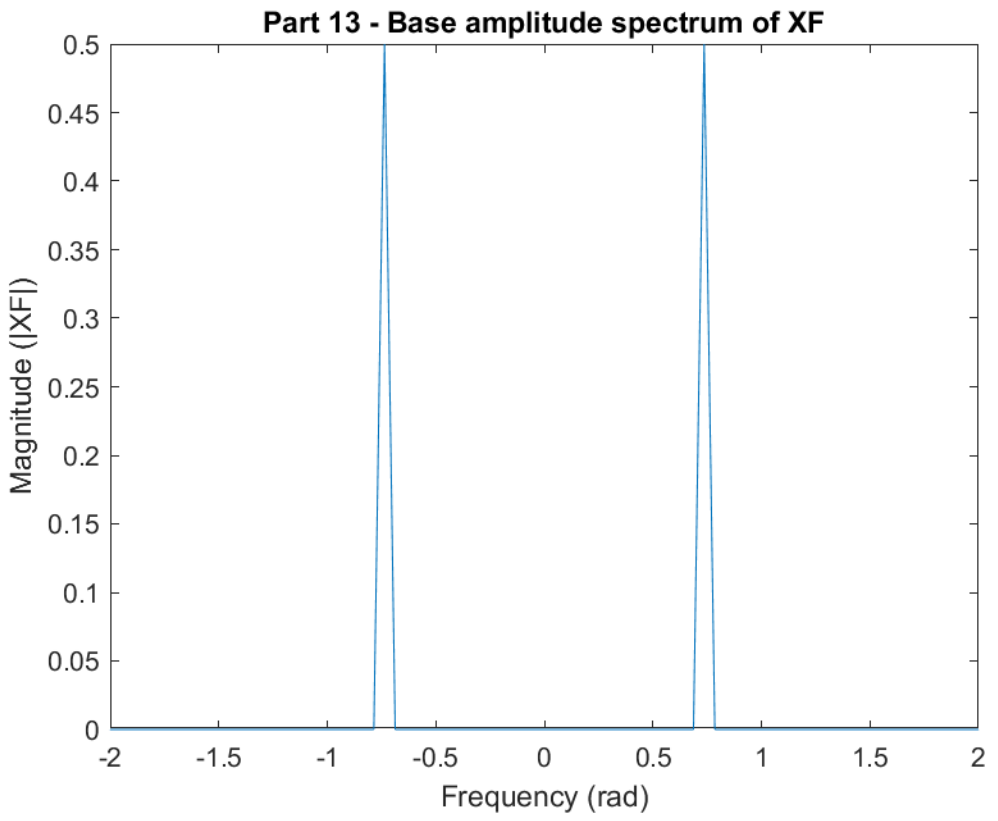


Figure 11: Plot of x[n] amplitude spectrum for f$_s$ = 128 Hz with N = 128

(14). The signal x[n] was padded with 20 zeros as follows:

$$x_1[n] = \begin{cases} sin(30\pi n T_s) & 0 \le n \le 127 \\ 0 & 128 \le n \le 147 \end{cases}$$

(15). Given the zero-padded signal above, the results of the amplitude spectrum and the determined frequency resolution are shown below:

Determine frequency resolution:

$$\begin{aligned} \text{resolution} &= \frac{2\pi}{N} \\ &= \frac{2\pi}{148} \\ &= \boxed{0.042 \text{ rad}} \end{aligned}$$

The following amplitude spectrum was observed:

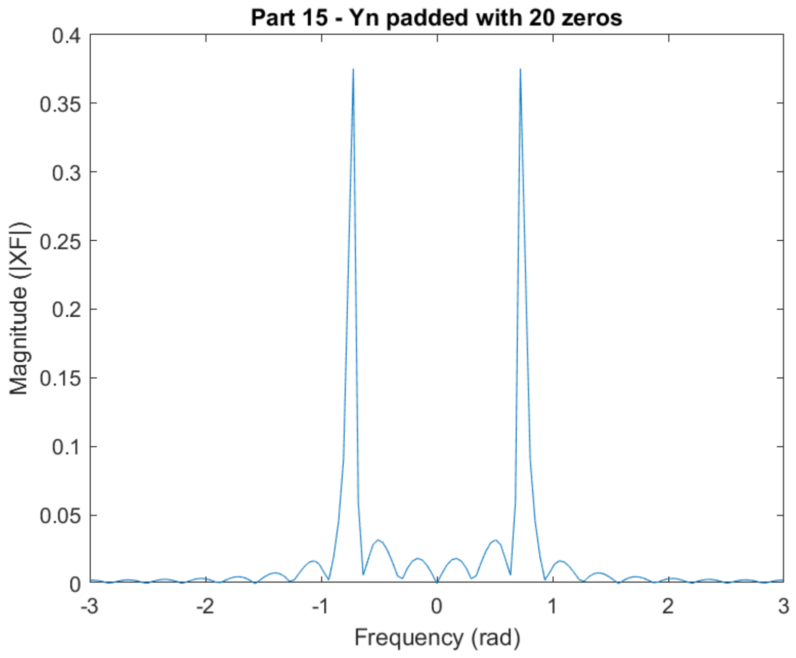


Figure 12: Plot of x[n] amplitude spectrum with zero-padding

(16). The following presents a plot of the amplitude spectrum of the guitar.wav sampled signal corresponding to a N = 176400 which achieves a frequency resolution of $\frac{2\pi}{176400} = 0.0000356$ rad. The sampling frequency used was 44.1kHz.

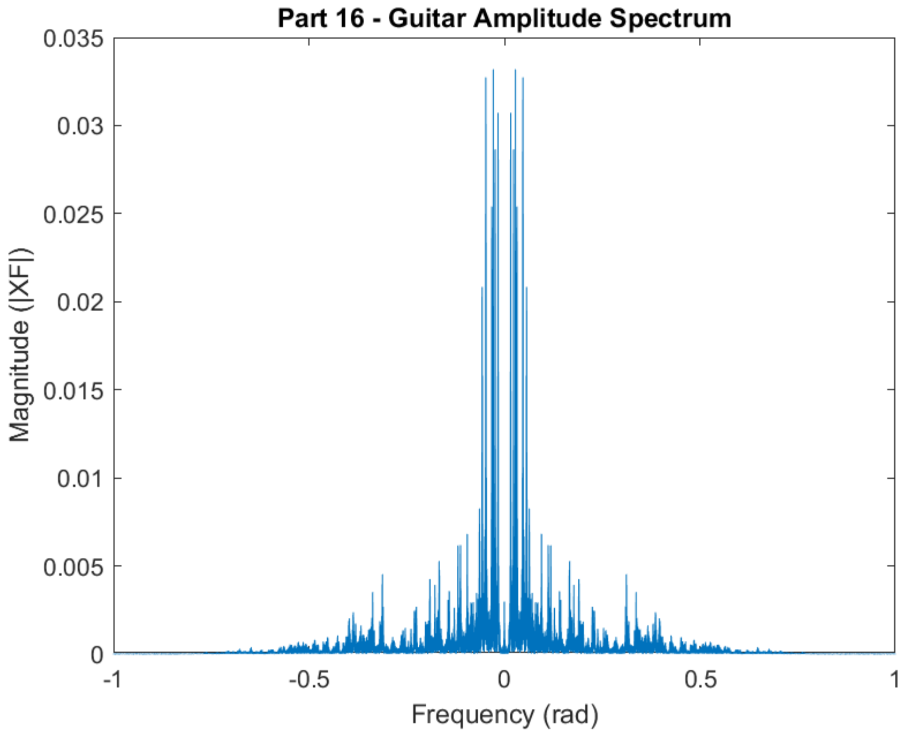
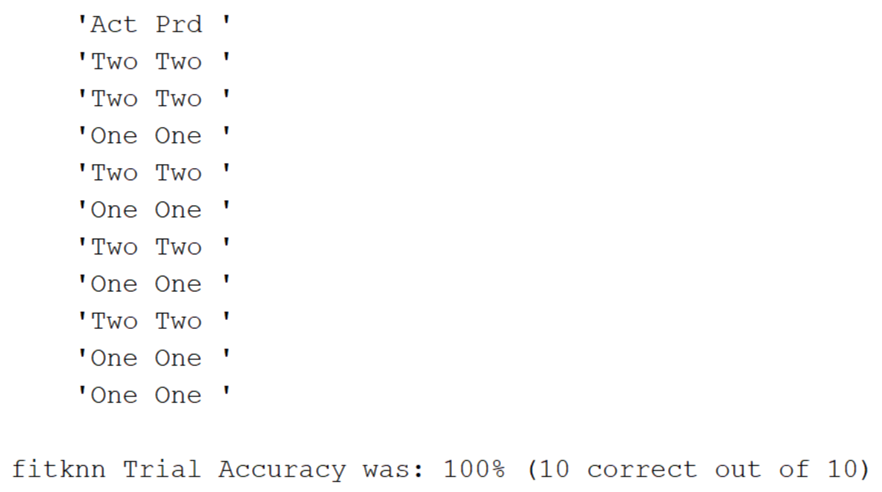


Figure 13: Plot of guitar.wav amplitude spectrum

(17). Below are the results for the K-Nearest Neighbor classification utilizing the MATLAB function, accuracy was evaluated through the program and results were taken directly from the command log. On the left column are the actual test points being fed in, and in the second colum are the predicted labels:

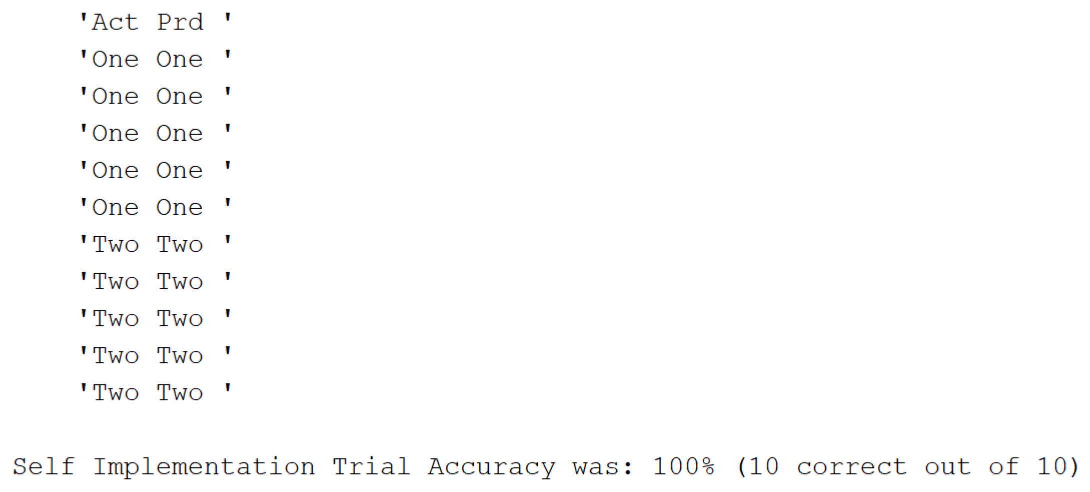


```
'Act Prd '
'Two Two '
'Two Two '
'One One '
'Two Two '
'One One '
'Two Two '
'One One '
'Two Two '
'One One '
'One One '

fitknn Trial Accuracy was: 100% (10 correct out of 10)
```

Figure 14: Results of MATLAB classification of ”One” and ”Two”

Following is the results from the same training and testing sets of the self implemented KNN:



```
'Act Prd '
'One One '
'One One '
'One One '
'One One '
'One One '
'Two Two '
'Two Two '
'Two Two '
'Two Two '
'Two Two '

Self Implementation Trial Accuracy was: 100% (10 correct out of 10)
```

Figure 15: Results of Self-implemented KNN classification of ”One” and ”Two”

Discussion

(1). As expected, the peaks both occur at 1.5, which is half of the amplitude of the time signal, in both Hz and radians. This is because the resolutions were defined such that the exact frequency of the time signal was included in the resolution set. In the frequency plot, the peaks occur at $\pm$5 Hz because it is defined as $\frac{\omega}{2\pi}$ with ω = 10*π as it is real time. The peaks in the radians plot occur at $\pm$0.0614 radians as the radian resolution is expected to be $\frac{\omega}{N}$ with ω = 10*π and N = 512, as it is discrete time.

(2). In the hertz plot, nothing changes because real time is being used - the peaks still occur at 5Hz. However since the radians plot is characterized discrete time, removing samples by downsampling, a compression in time, results in an expansion in frequency as proved in class. Therefore the peaks in the radian plot occurs around -0.125 radians, or double the amount in (1) as the sample was downsized by a factor of

2.

(3). Similar to (2), the hertz plot did not change as it plotted real time and changing the discrete time did not affect the signal. However for the radian plot, the opposite of (2) occurs - the expansion in time of the signal results in a compression in frequency by the same factor. Hence the peaks in the radians plot occurs at 1/3*0.0614, or 0.02, radians in discrete time.

(4). Expanding and contracting discrete signals have a dual effect in time and frequency. That is, expanding in time results in compression of frequency vice versa. This key relationship was explored in this lab through upsampling and downsampling (Results (1) - (3)), as discussed above. To summarize: downsampling results in removing samples which contracts the signal - this results in an expansion in the discrete frequency amplitude spectrum resulting in a shift outward of the two amplitude impulses; in the upsampling case, zeros are added to expand the signal in time - this results in a compression in the discrete frequency amplitude spectrum reflected in a shift inward of the two amplitude impulses.

(5). Signal defined in MATLAB - no corresponding discussion.

(6). See results for derivation. It is interesting to note that the DTFT results in an amplitude of π while the frequency of standard Fourier transform results in 0.5*A, A the amplitude of the signal. The MATLAB fft function executes the Fast Fourier Transform, an efficient algorithm for standard Fourier transform. Therefore signals plotted in the lab have resulted in frequency amplitudes of 0.5*A rather than π.

(7). See discussion part (6) for comments on amplitude differences. Otherwise, peaks occur where theoretically expected - each at $\frac{20\pi}{128}$, $\frac{11\pi}{128}$, $\frac{121\pi}{256}$ and radians. The following part will explain amplitudinal differences between the different components of the signal.

(8). As seen in (8) in the results section above, in order to determine the correct N necessary to accurately capture the second frequency component in the amplitude spectrum, it was necessary to double the original N. The was done because the fundamental continuous frequency of the second component was 5.5Hz. Therefore it was required to capture the half frequencies as whole numbers - hence, doubling N would accurately capture this frequency.

(9). Similarly to the statement in discussion (8) above, as seen in (9) in the results section above, in order to determine the correct N necessary to accurately capture all three frequency components in the amplitude spectrum, it was necessary to multiply the original N by four. The was done because the fundamental continuous frequency of the third component was 30.25Hz. Therefore it was required to capture the quarter frequencies as whole numbers - hence, quadrupling N would accurately capture this frequency.

(10). The higher sampling effect sharpens all of the peaks, making them closer to impulses since increasing N results in a finer frequency resolution. The amplitudes do not change any more because the fundamental frequencies of the signal are still captured as whole numbers with N = 1024.

(11). Signal defined in MATLAB - no corresponding discussion.

(12). See (12) in the results section for the derivation of the amplitude spectrum.

(13). Peaks occur where theoretically expected - at positive and negative $\frac{30\pi}{128} = 0.763$ radians. See discussion point (6) for comments on amplitude differences. While DTFT predicts amplitudes of π, fast Fourier transform results in amplitudes of A/2 which is what is seen in the plot in amplitudes of 0.5.

(14). Signal padded with 20 zeros in MATLAB - no corresponding discussion.

(15). The figure presented in part (15) of the results section shows the effect of padding a period sinusoidal signal with zeros. As seen in the figure, since the previous signal was already periodic in time, adding

K	Accuracy
3	100%
5	100%
9	90%
14	70%
20	50%

additional zeros actually negatively impacted the periodic nature of the signal. This resulted in the appearance of other unwanted harmonics in the signal noticed in the amplitude spectrum. The mini-peaks between -1 and 1 radians in the figure are the result of padding the periodic signal with zeros, therefore making not truly periodic with N and introducing new frequencies. See (15) in the results section for calculation of frequency resolution. Also important to note is that the expected peak amplitude of 0.5 is not the case in the plot. The peak amplitude now is about 0.37. This is the result of padding with zeros, because the amplitude now spreads across multiple frequencies, rather than just the single true frequency.

(16). The sampling frequency used and the frequency resolution achieved can be found in (16) of the results section. (16) of the results section also provides the amplitude spectrum of the signal. When the continuous frequencies were plotted in MATLAB (not shown), the dominant frequencies present in the sample lie in the 0 - 500Hz range, which are low frequency tones. This makes sense as mostly low guitar notes (Guitar low range is 52-800Hz) are audible in the audiofile.

(17). Both MATLAB functions and the self-implemented code were able to achieve 100% accuracy. This is likely in part due to the quality of training and testing sets obtained. Effort was put into having the subject speak clearly into the microphone with minimal ambient noise, and sound bytes were trimmed to only contain the sample in order to eliminate other extraneous noise. K was set to 3 Nearest Neighbors due to the small training set size. KNN was self implemented by using the sum of squares difference error function to calculate the three closest training points that the test point was close to and matching labels, similar to the MATLAB implementation.

In order to verify that the functions worked correctly and were not simply outputting duplicated labels, the input order was randomized. This did not affect the 100% accuracy. Additionally, the number of features extracted was altered to observe the effects of information on the system. As expected, lowering the amount of frequency features had a negative effect on classifier accuracy, dropping from 100% to a floor of 50% at 3 features. As final verification, different K's were also tested. Increasing K would help observe how much the two training labels overlapped in features. Below is a table of the resulting effects of lowering K:

This makes sense as values of K well below 10 seem to give 100% accuracy because the features of "One" and "Two" likely do not overlap much. However, as the number approached 10 and past, accuracy was reduced. at K = 9, the accuracy dropped, indicating that within the training set, there was an outlier among the Ones that made it characteristically unlike the others. at K = 14 the set was classification set was diluted again, and at K = 20 the model was based off of all training points without distinction, so the predictor was more or less just guessing. Generally speaking, it seems a smaller number of neighbors (relative to training set size) provides a more robust predictor but may be prone to extreme outliers. More neighbors can eliminate the effect of outliers but may increase risk of cross-classifications due to labels not being unique enough.

Conclusions

The laboratory was largely successful in helping us develop a better understanding of the Fourier transform and its practical effects in discrete time frequency analysis. Other noteworthy topics of exploration included upsampling, downsampling, frequency resolution, zero padding, and voice recognition accomplished by machine learning. All results were expected - hand calculations of amplitude spectra were correctly reflected in MATLAB amplitude spectra plots. Voice recognition was successfully implemented in a basic "one" vs. "two" comparison with K nearest neighbor built in functions and a custom implementation.

The major conclusion of this lab was that the Fourier Transform is extremely useful for analyzing both periodic and aperiodic signals in the frequency domain. It can provide unique insights into the properties of a time signal and make many features (i.e. frequency) readily available for extraction and use in a classifier applications. All in all, these conclusions form a key insight which defends the claim that it is often easier and more importantly, more useful, to analyze models in alternate domains like frequency because of the methods available for signal analysis. This has been a key theme of the course and has been relevant in the prior labs as well.

References

[1] Dr. Abdelhakim, Class Professor

[2] Lecture Part III: Frequency Analysis of Discrete Systems

[3] Kechen Shu, Lab TA