**Final Project Report**

Jidechukwu Obioha

Department of Electrical and Computer Engineering: Binghamton University

EECE 301: Signals and Systems

Professor Mark Fowler

November 25, 2024

**INTRODUCTION**

In this report, I explore an interesting use case of the Discrete Fourier Transform (DFT), a simple guitar tuner. Modern guitar tuners use more complicated algorithms to estimate the fundamental frequency of the guitar signal supplied to them. However, for this project, I use a more simplified signal processing technique, filtering, to estimate the fundamental frequency of a guitar string being plucked. The overall goal of this project is to use DFT processing to estimate the fundamental frequency of a guitar string to 5 cents of accuracy, a reasonable standard for guitar tuners.

The project begins with using the DFT as a tool to analyze a simple sinusoid, as well as the drawbacks that come with computing the DFT of a signal with constraints such as a limited number of samples and the extent to which zero-padding may be needed. I also discuss how windowing might be used to fix spectral leakage. This foundation of using the DFT for analyzing a simple sinusoid will be useful in the latter portions of the project where the DFT is used to analyze more complex audio signals.

As the project progresses, I load real audio files containing signals from a guitar string being plucked, compute the DFT of those audio signals, and analyze the plots and what they signify. I also create synthetic guitar signals using the Fourier Series model and use this signal as the basis for testing the DFT-based tuner through targeted filtering and DFT analysis.

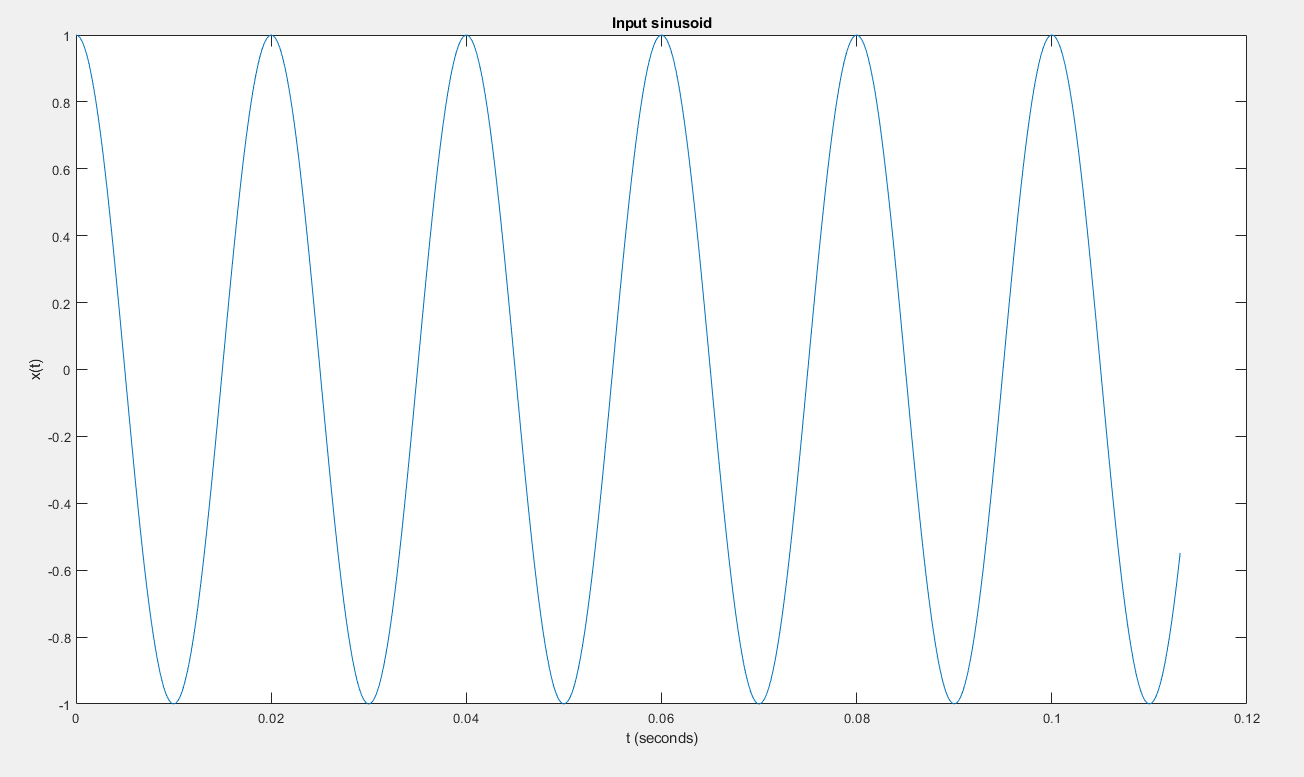
The final portion of this project involves designing bandpass filters, using them to approximate the frequency location of the input signal, and finally estimating the fundamental frequency of the guitar’s string through DFT analysis.

**TASK 1 – DFT of Single Sinusoid and the Use of the Hamming Window**

In this portion of the project, the DFT of a simple sinusoid is computed. Before computing the DFT of this signal, one can use theory to gain a good expectation of what the computed DFT will look like. Periodic signals can be broken down into simpler sinusoids via the Fourier Series, a subset of the more generalized Fourier Transform, specifically applicable to periodic signals. These simpler sinusoids make up the component frequencies that comprise the signal, with the lowest of these frequencies called the fundamental frequency; every other component frequency is an integer multiple of this fundamental frequency. The Fourier Series of a periodic signal in complex exponential form is summarized by the following equation:

|  | (1) |
| --- | --- |

From equation (1), one can see that a periodic signal, x(t) can be decomposed and represented as a sum of simpler sinusoids. The term ωo represents the fundamental frequency of the input signal, and the term, k is an integer. So with this view, it is easy to see that each term in the summation -each component sinusoid- has a frequency which is an integer multiple of the fundamental frequency. On a DFT plot, these harmonic frequencies show up as delta function ‘spikes’ on the DFT plot. These spikes occur because, for periodic signals, most of the signal's energy is concentrated at these frequencies. This means for a simple sinusoid, with only one frequency, we can expect to see only one ‘spike’ in its DFT plot.



*Figure 1 - Showing a simple sinusoid with a frequency of 50 Hz*

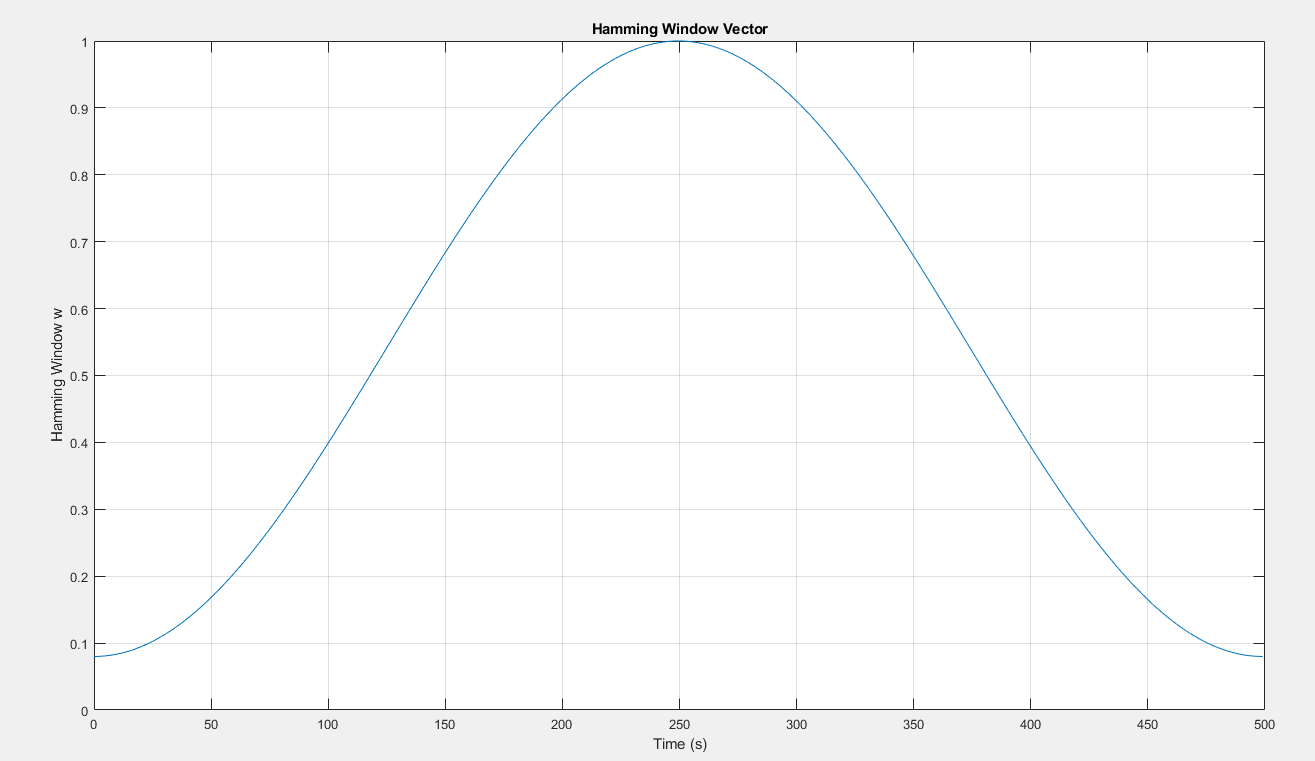
Figure 1 shows a sine waveform for a signal with a frequency of 50 Hz. Based on theory, the DFT of this plot should contain a delta function spike at 50 Hz. However, there are some constraints that may affect the accuracy of this idealized result in real-world applications.

One of which such factors is the sampling frequency: In computing the DFT of a signal, the signal must be sampled at a fast enough rate to ensure that DFT captures the signal’s frequency components. If the signal is not sampled, fast enough, this leads to aliasing, which affects the accuracy of the DFT computation. The minimum rate at which a signal can be sampled is called the Nyquist Rate (Fs / 2), and its relation to the maximum frequency or bandwidth, B, of a signal is given by the mathematical relation below:

|  | (2) |
| --- | --- |

For this project, the sampling frequency has been fixed to 4410 Hz. To avoid aliasing, we must choose the frequency of the input sinusoid to be less than or equal to the sampling frequency of 4410 Hz.

Another problem encountered when computing the DFT of a signal is smearing. Smearing occurs because a finite number of samples are collected from a signal. Consequently, the signal’s DFT gets ‘smeared’ up and may show all the information about the signal. One way to combat this is to take more samples. However, for any real system, a finite number of samples, no matter how many, can only ever be taken. As a result, there may always be some degree of smearing or ‘leakage’ on an input signal’s DFT. Oftentimes, a more advanced technique, windowing, is used to combat smearing.

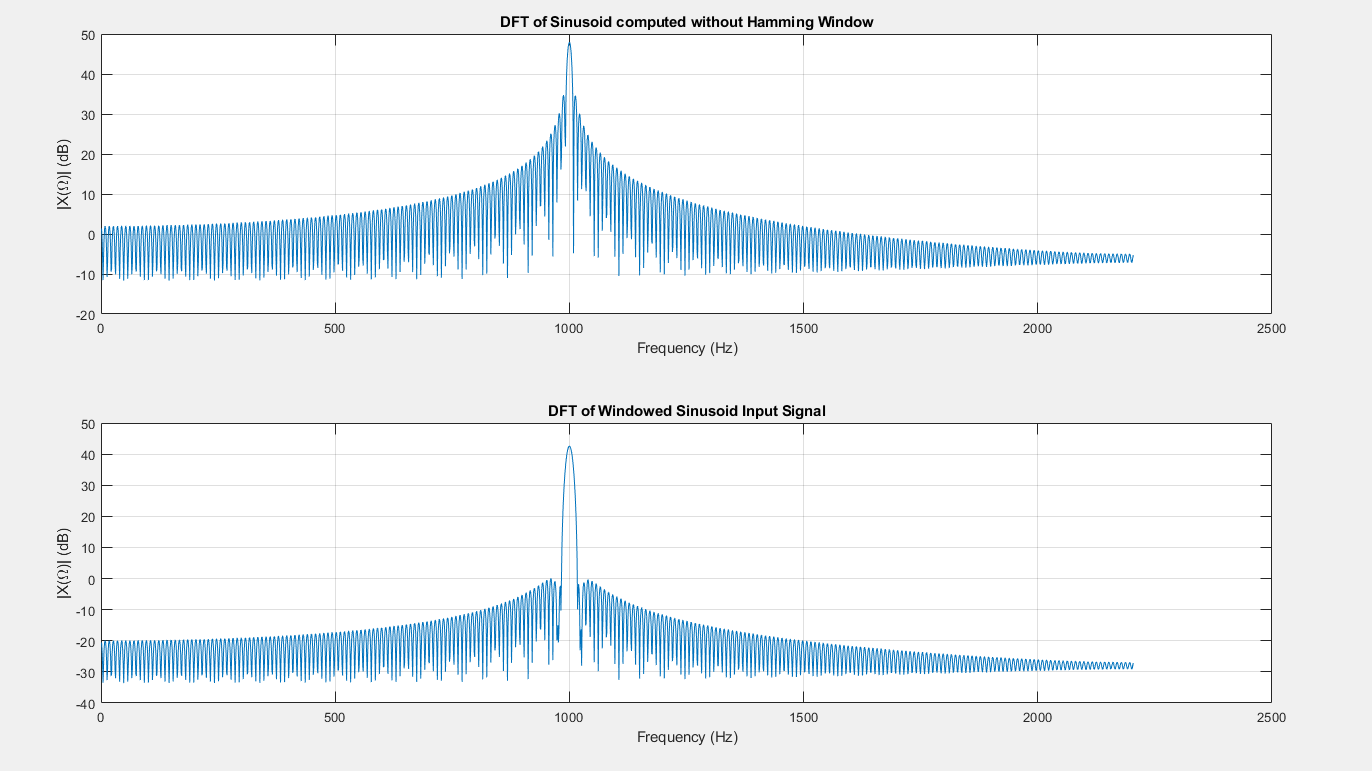


*Figure 2 – Showing the Hamming Window Vector, one of such vectors used when windowing a signal to reduce smearing*

Figure 2 above shows the Hamming Window Vector; a special vector applied to an input signal to reduce its leakage. This window tapers down the input signal. The shape of the plot shows that the Hamming window has little effect on the mid-range of the signal it is applied to, however, towards the ends of the signal, the Hamming Window ‘smooths’ out the input signal. Using the Hamming Window has the effect of reducing the spectral smearing of the signal, allowing for spikes within the DFT plot to be more clearly visible and not smeared into a lobe-like structure.

The last constraint that comes along with computing the DFT of a signal is the resolution of the plot of the signal’s DFT. In theory, most real-world signals inherently have a Continuous Time Fourier Transform. However, when those signals are put through an Analog-to-Digital converter, a finite number of samples are collected from which the signal’s truncated Discrete Time Fourier Transform. The DFT of a signal can be seen as points that lie on the signal’s truncated Discrete Time Fourier Transform. The question is, how many points does DFT actually take from the truncated Discrete Time Fourier Transform? If a small number of points are taken, the plot of the signal’s DFT will have a poor resolution. Conversely, if many points are taken, the plot of the signal’s DFT will have a good resolution. We can get a finer view of a signal’s truncated Discrete Time Fourier Transform through a technique known as zero-padding. Zero-padding simply means tacking on zeroes to the samples we have collected so that our DFT has a better resolution and by looking at it, we can get more insight as to what shape a signal’s truncated DTFT has. Usually, zero-padding is done to the extent that the total number of samples after zero-padding is around four to five times the number of original samples. It’s also worth mentioning that zero-padding doesn’t add more information to the input signal’s frequency spectrum; it only gives a better view of the information already contained in the signal’s truncated DTFT.

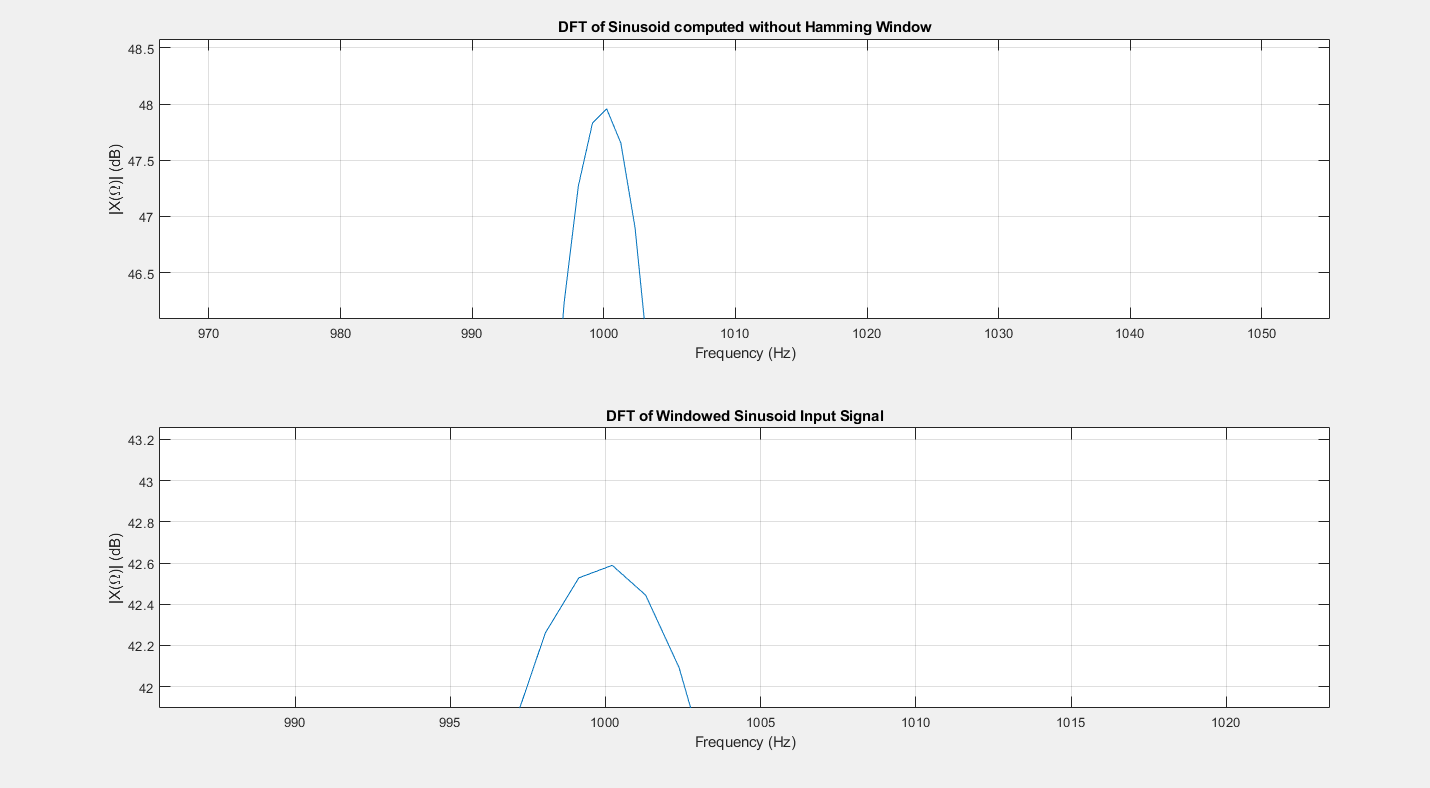
Now that the theories have been discussed, I can use a MATLAB script to compute the DFT of a simple sinusoid and compare theoretical expectations with what we see in the ‘real world’.



*Figure 3 – Showing the DFT of an input sinusoid with a frequency of 1000 Hz, with 500 samples collected, zero-padded up to 4096 samples*

Figure 3 above confirms the theories earlier discussed. A sinusoid with a frequency of 1000 Hz is passed into the function and the DFT of the sinusoid is computed. From the plots, we can see a delta function spike around 1000 Hz, signifying the fundamental frequency, really the only frequency, of the input sinusoid lies is at 1000 Hz. It‘s also worth noting that the magnitude of the DFT for a real-valued signal has even symmetry, so the plots above also have a spike at around -1000 Hz. However, since negative frequencies are fictitious, for practical purposes, the negative side of the spectrum is often omitted.

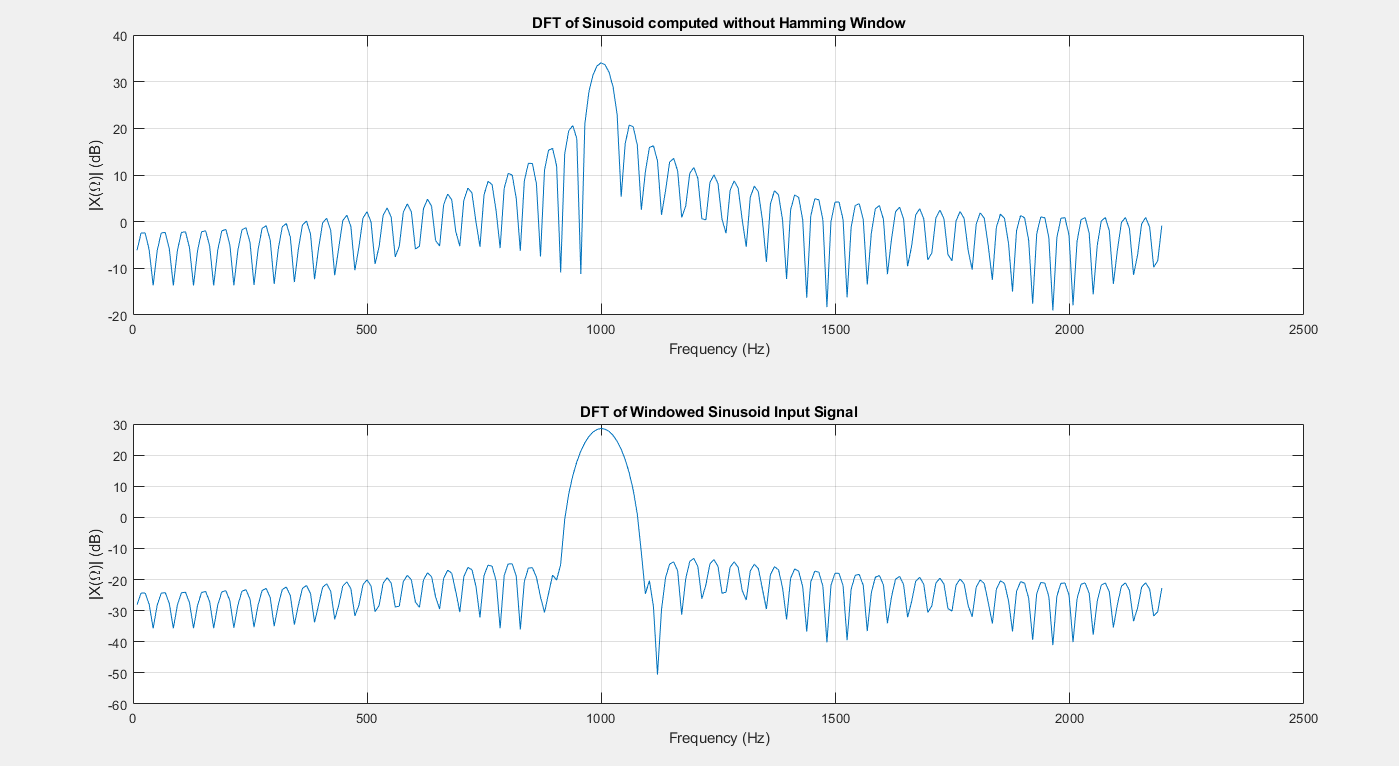
Comparing the two DFT plots, one can easily observe the effect using a window has on the DFT, it emphasizes the peak, giving the main lobe a narrow profile with de-emphasized side lobes, resulting in less spectral leakage, and a better view of the plot. However, there may be some changes in the amplitude, or height, of the DFT plot when windowing is used.



*Figure 4-Showing the same plot from Figure 3 but zoomed in*

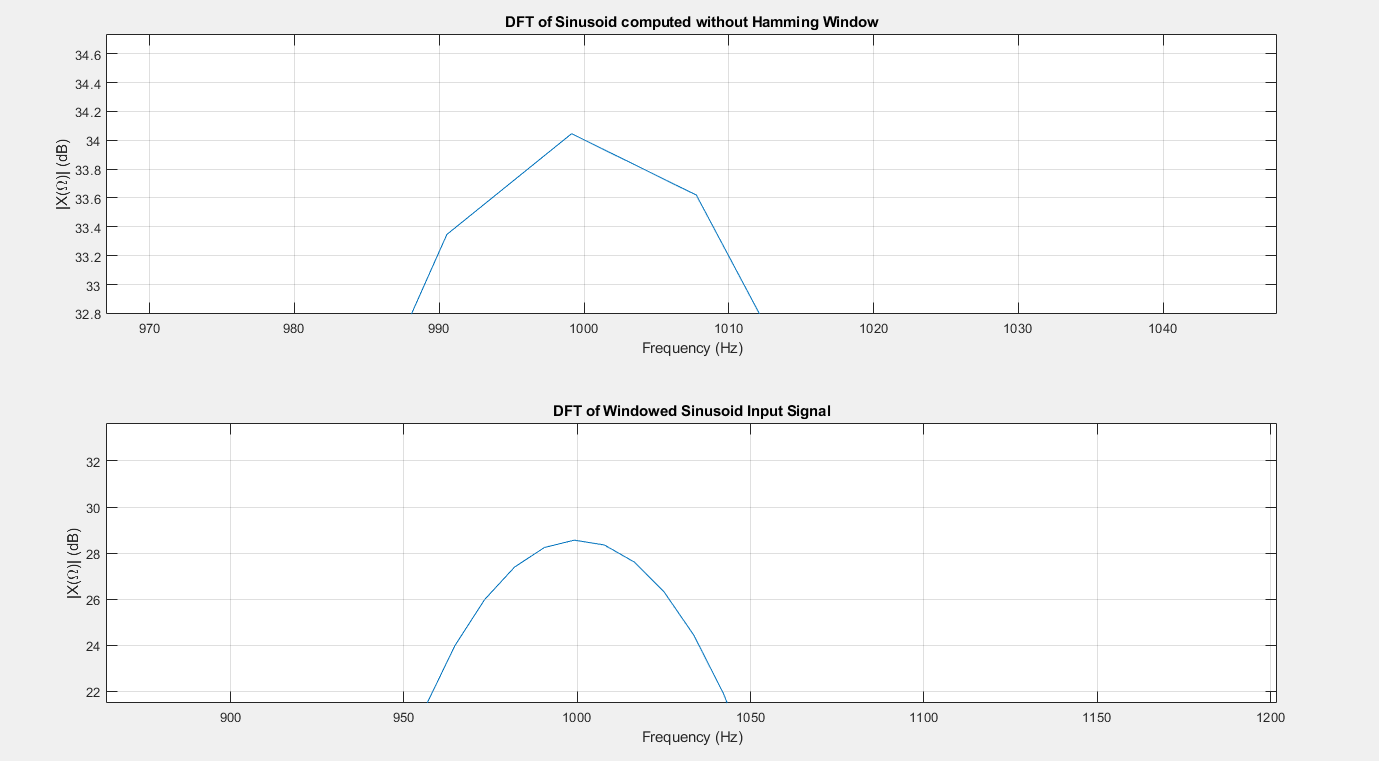
Figure 4 shows that applying a window may have the effect of modifying the amplitude of the spike in the DFT plot. The plot produced by the unwindowed signal had an amplitude of about 48dB while the plot produced from the windowed signal had an amplitude of around 42.6 dB, approximately 60% difference in power. So with windowing signals, there’s a tradeoff between getting a more emphasized main lobe, making it easier to see what frequency gives the spike in the plot, but there’s a less accurate measure power of the signal at that frequency.

Having examined the effect windowing has on the DFT of a plot, let’s go into how the number of samples collected also has an impact on the DFT.



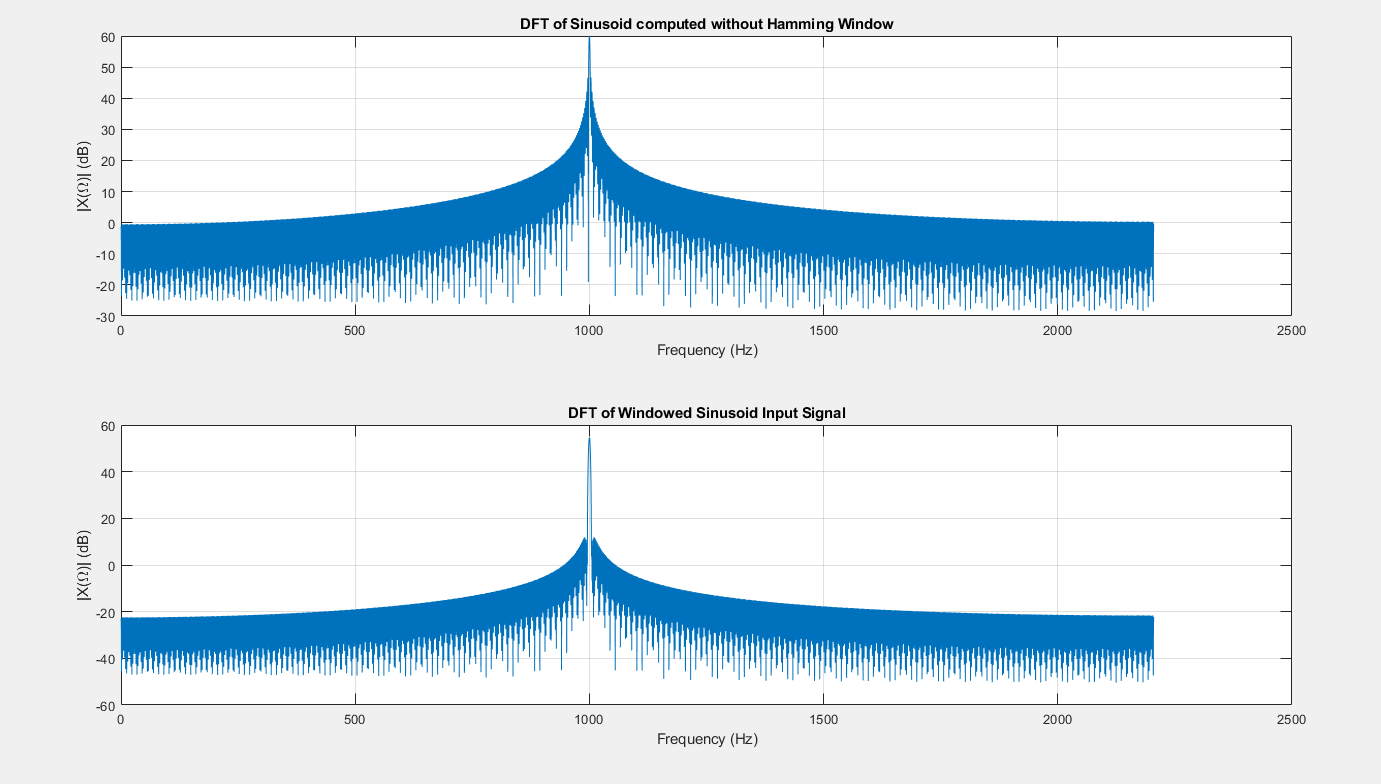
*Figure 5 - Showing the DFT plots of a sinusoid with a frequency of 1000 Hz but with only 100 samples, 512 samples with zero-padding*

Just by looking at the plots, it is easy to see the loss of information and lack of detail compared to the plots with more samples taken. Although zero-padding gave a clearer view of the DFT, the overall DFT suffers from smearing, with the frequency components at the main lobe smearing onto adjacent frequencies. Even with windowing, although there is a more emphasized main lobe from pushing frequency components at adjacent frequencies towards the fundamental frequency, there still isn’t a clear view of the delta function spike we expect from a periodic function due to the extreme degree of smearing from a small number of samples taken.

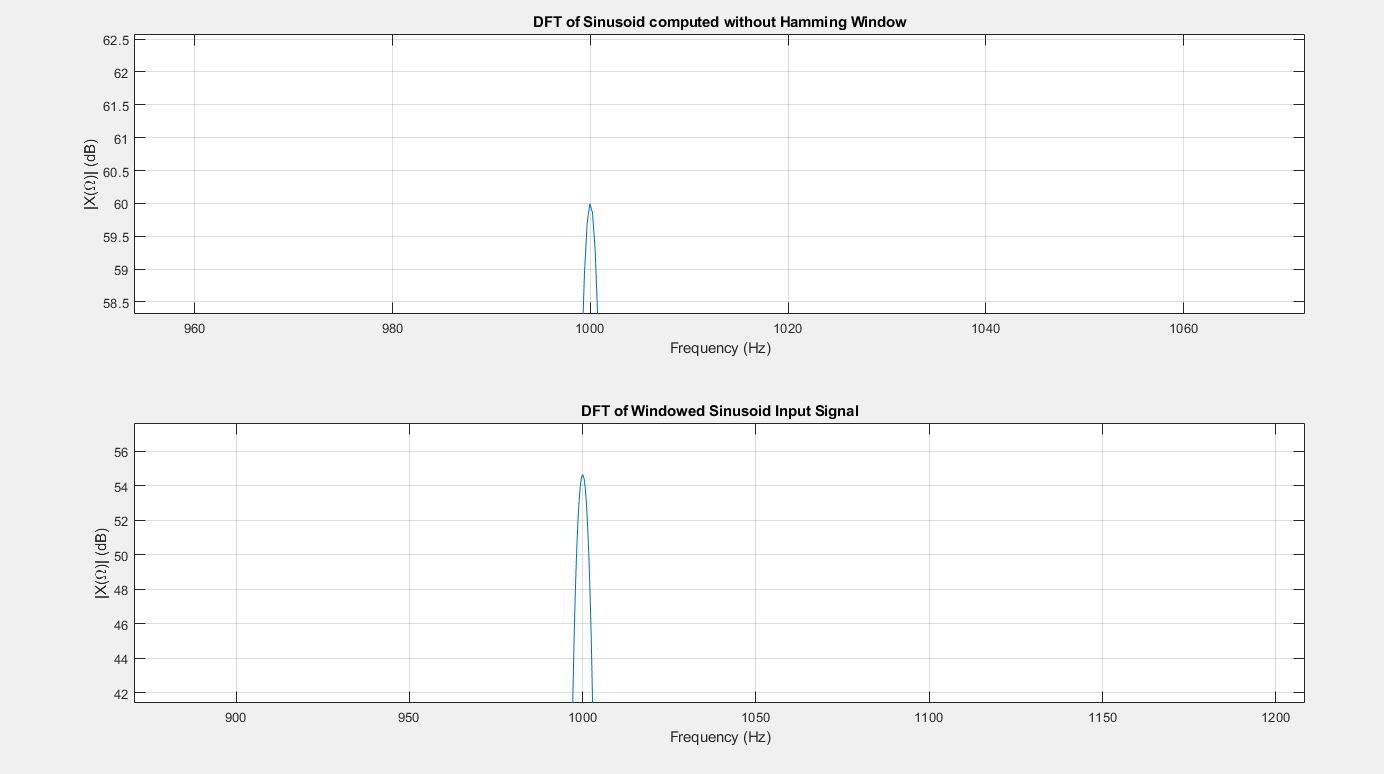


*Figure 6 -Zoomed-in version of Figure 5*

Comparing Figure 6 with Figure 4, the effect of a low number of samples on the DFT plot. Visually, it seems like the low number of samples taken reduces the information on the DFT plot, by flattening, or smearing, the DFT. Conversely, we get an even more detailed DFT, when we take even more samples than the 500 samples that were taken in Figure 4.

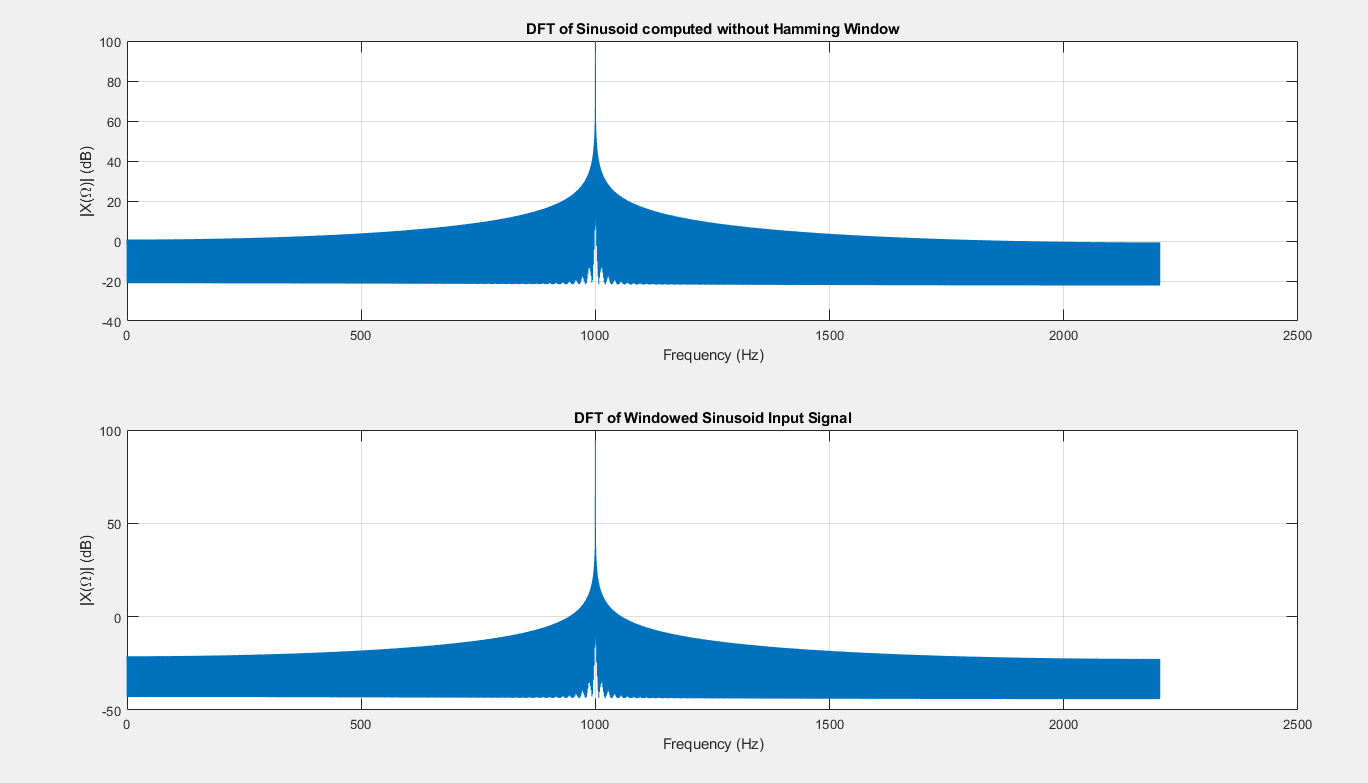


*Figure 7 - Showing the DFT of a sinusoid with a frequency of 1000 Hz, with 2000 samples taken, and 16384 samples with zero-padding*

****

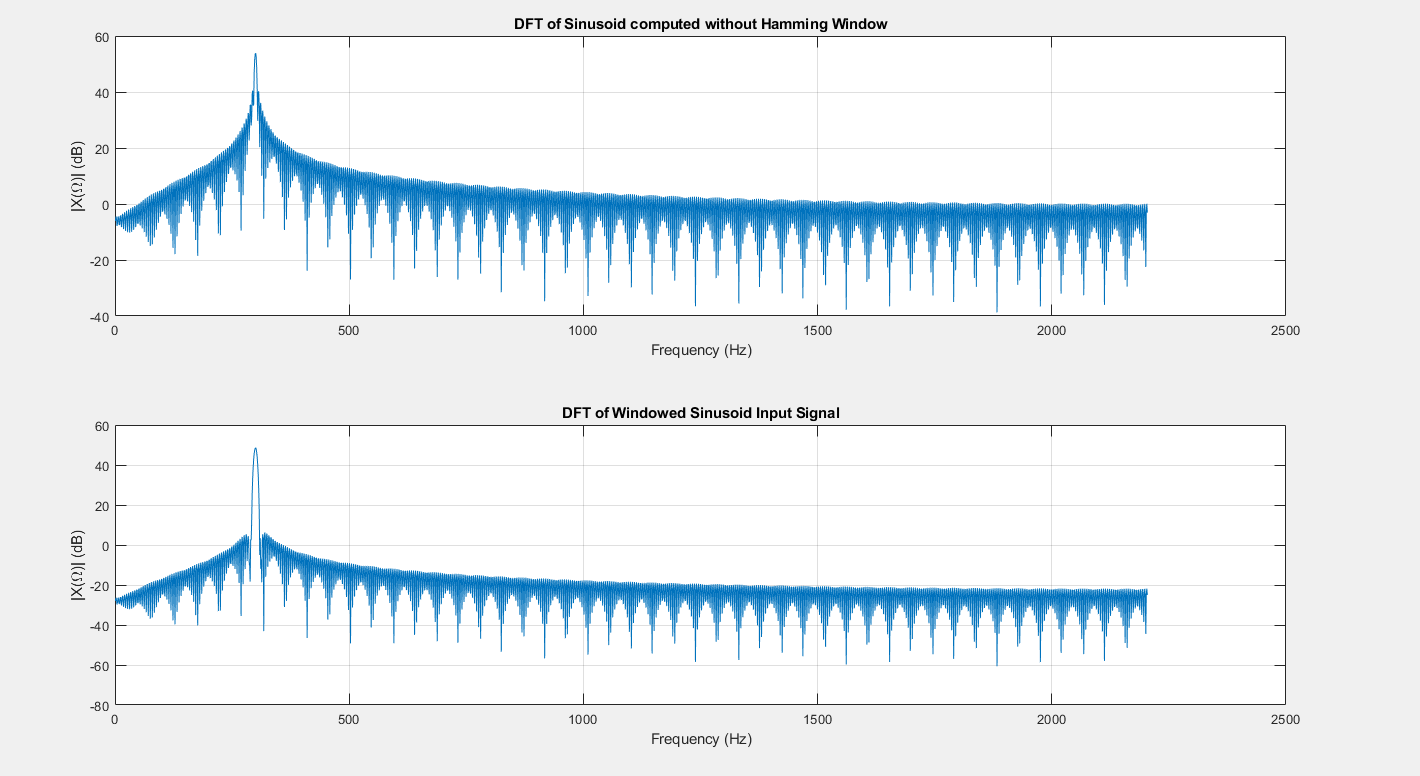
*Figure 8 - Zoomed in Version of Figure 7*

Comparing Figure 8 with Figures 7, 6, and 4, it becomes increasingly clear that as we increase the number of samples, the delta function spike that we expect looks more and more like a delta function (gets ‘spikier’) and occurs at the frequency we expect (the frequency of the sinusoid). Increasing the number of samples further will yield similar results. In real-world applications, however, samples are collected to the degree of accuracy needed for the plot, as there is a cost incurred on a computer as more samples are collected – all computers have a finite amount of memory, and the number of samples one can collect is limited by that memory.



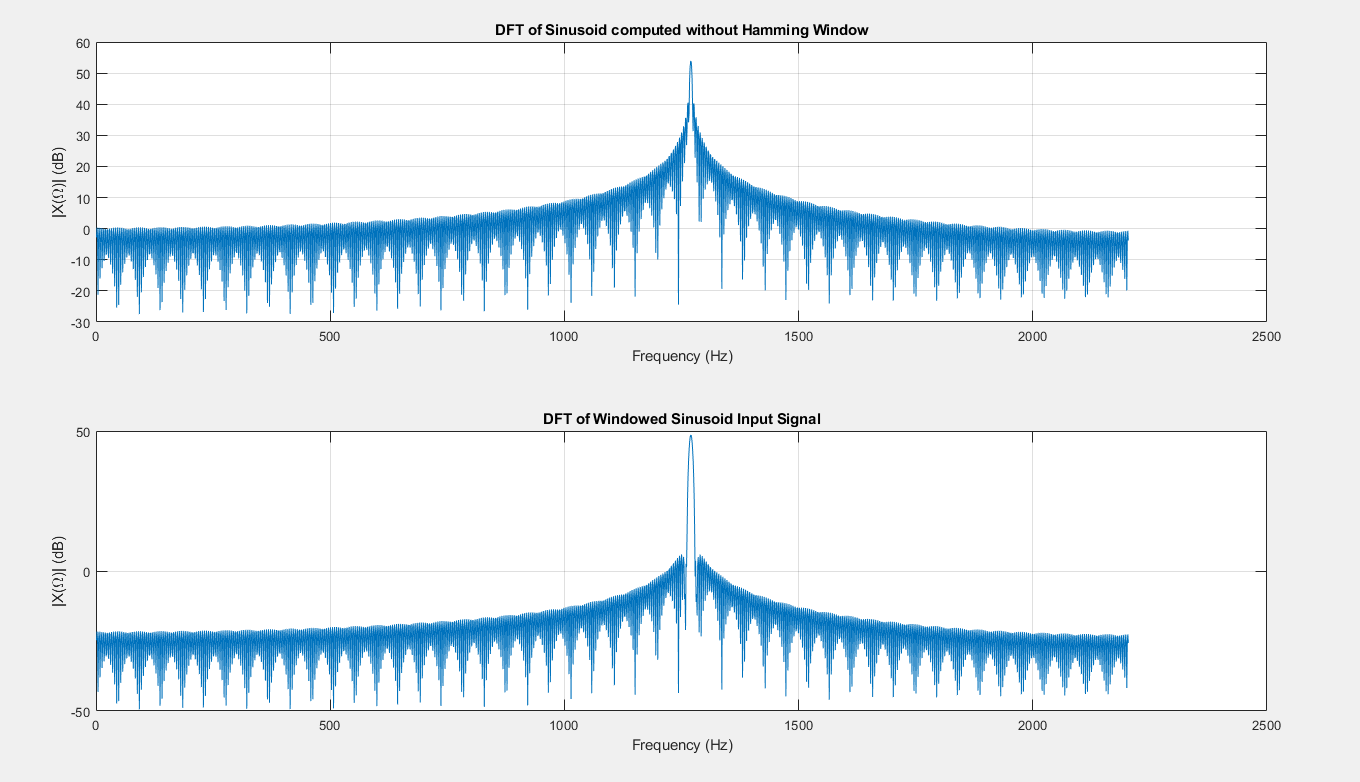
*Figure 9 - DFT of 1000Hz Sinusoid, 200,000 samples collected, 1048576 samples with zero-padding. Illustrating how a true delta function is realized at the frequency of the sinusoid as the number of samples collected increased*

Theory suggests that we would obtain similar results for different frequencies, that is to say, our delta function spike would occur at whatever frequency the sinusoid is.



*Figure 10 - Showing the DFT of a sinusoid with a frequency of 300 Hz, with 1000 samples taken and 4096 samples with zero padding. Our theory suggests we see a spike in the frequency of the sinusoid, and the plot above confirms this*

However, this only holds if our input signal is sampled fast enough. Equation (1) shows us that for proper sampling, the sampling frequency should be at least twice the maximum frequency of the signal or the signal’s band. Proper sampling allows for spectral replicas of the DFT to be adequately spaced apart so that they don’t interfere with each other. When this requirement is not met, aliasing occurs. Aliasing refers to when a signal’s DFT appears to have a different frequency than it should have. This happens because of the overlap in spectral replicas. Consequently, in our frequency range of interest – from 0 Hz to (Fs / 2 ) Hz, we may get the delta spike of a spectral replica, not of our actual DFT. The delta spike of a spectral replica usually is at a different frequency from our input signal and can be misleading. The simple solution for addressing aliasing is sampling faster.



*Figure 11 - Showing the DFT of a sinusoid with a frequency, of 7550 Hz, with 1000 samples collected, and 4096 samples with zero padding. Our theory says that for sinusoids we should expect a spike at 7550 Hz for this signal. However, we get a spike at approximately 1250 Hz. This is due to aliasing; a sampling rate of 4410 Hz is below the Nyquist Frequency and can’t sample a sinusoid with a frequency of 7550 Hz fast enough to gain an accurate DFT plot. The DFT seen in the plot arises from a spectral replica of the input signal.*

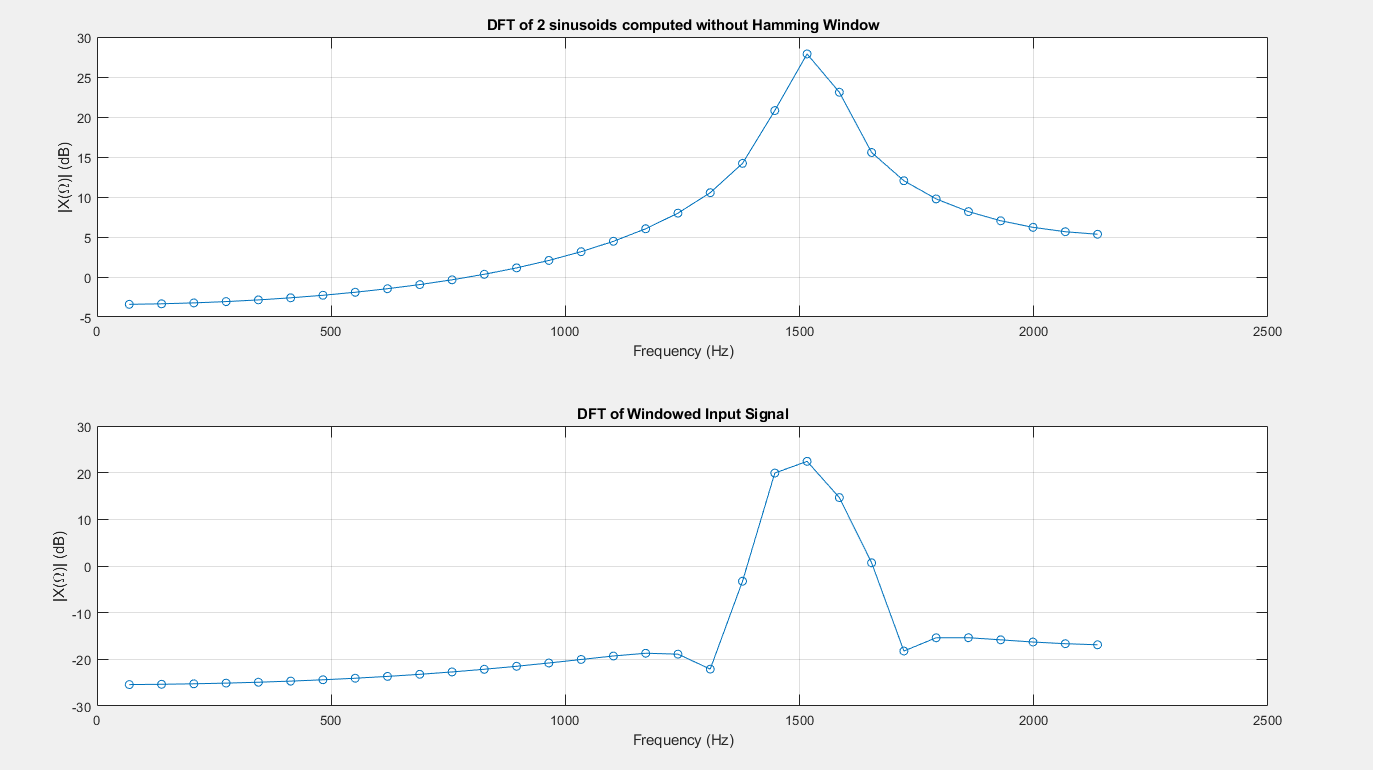
Now that I’ve discussed the effects of windowing, zero-padding, sampling frequency and the number of samples that affect the DFT of a simple sinusoid, I will explore the effect these constraints have on a signal containing two sinusoids or two frequencies.

**TASK 2 – DFT of Two Sinusoids**

In this portion of the project, I add another sinusoid to the input signal. Doing so introduces a new component into the DFT analysis. When a signal contains two frequencies, it becomes critical to be able to discern those two frequencies, and the proper sampling or the use of a window to combat smearing becomes more important. Based on theory, if a signal contains two sinusoids with frequencies that are not close to each other, we can expect to see two distinct spikes on the DFT plot. However, can we get a clear view of these spikes when the frequencies of the component sinusoids are very close in proximity? Consider a voltage signal consisting of two sinusoids, both having frequencies of 1500 Hz and 1550 Hz, and the second sinusoid having an amplitude of *a*.

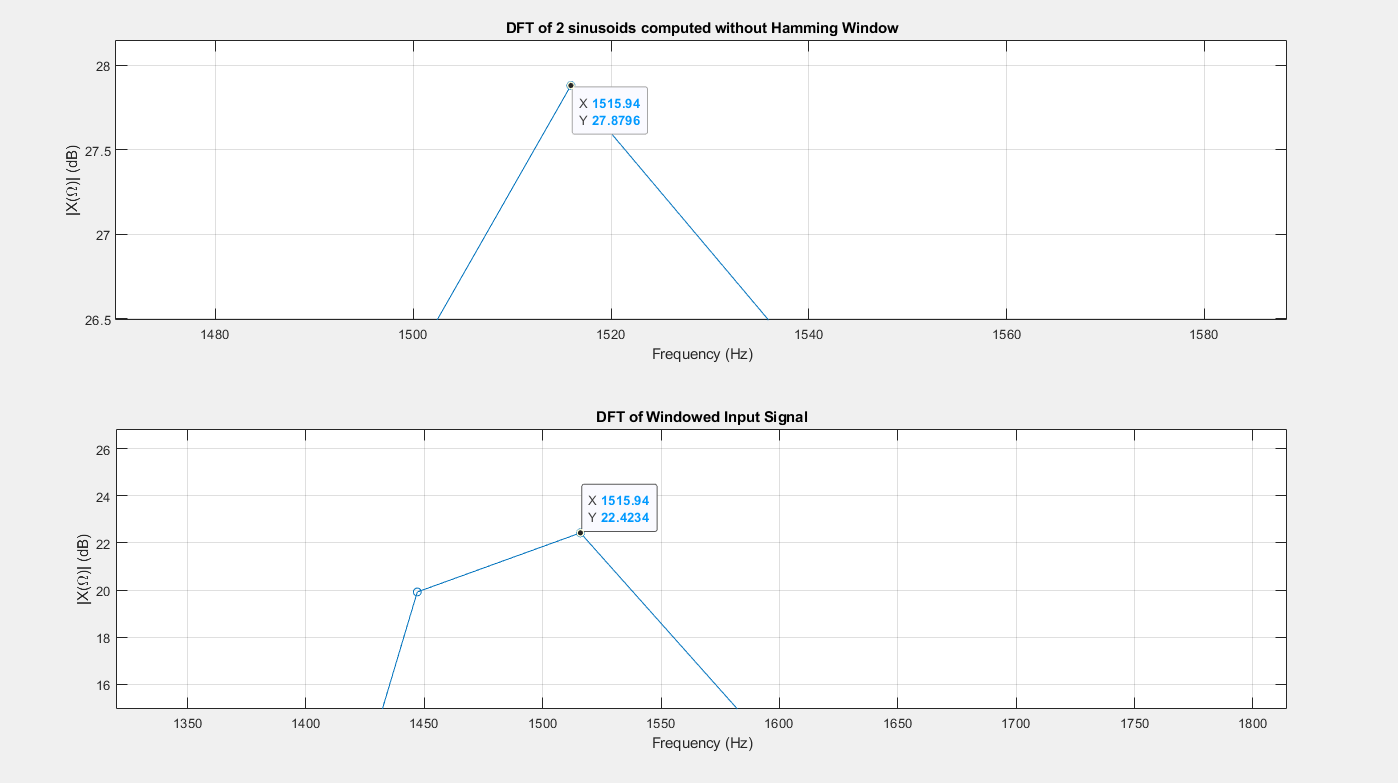
|  | (3) |
| --- | --- |

Our theory tells us that the plot of the DFT of this signal will have two delta function spikes corresponding to the frequency components of the signal. However, we don’t have an infinite number of samples, the computer sampling such a signal has finite memory, and as a result, can sample the input signal a finite number of times. As previously discussed, the number of samples collected has a direct influence on the shape the DFT plot takes on.



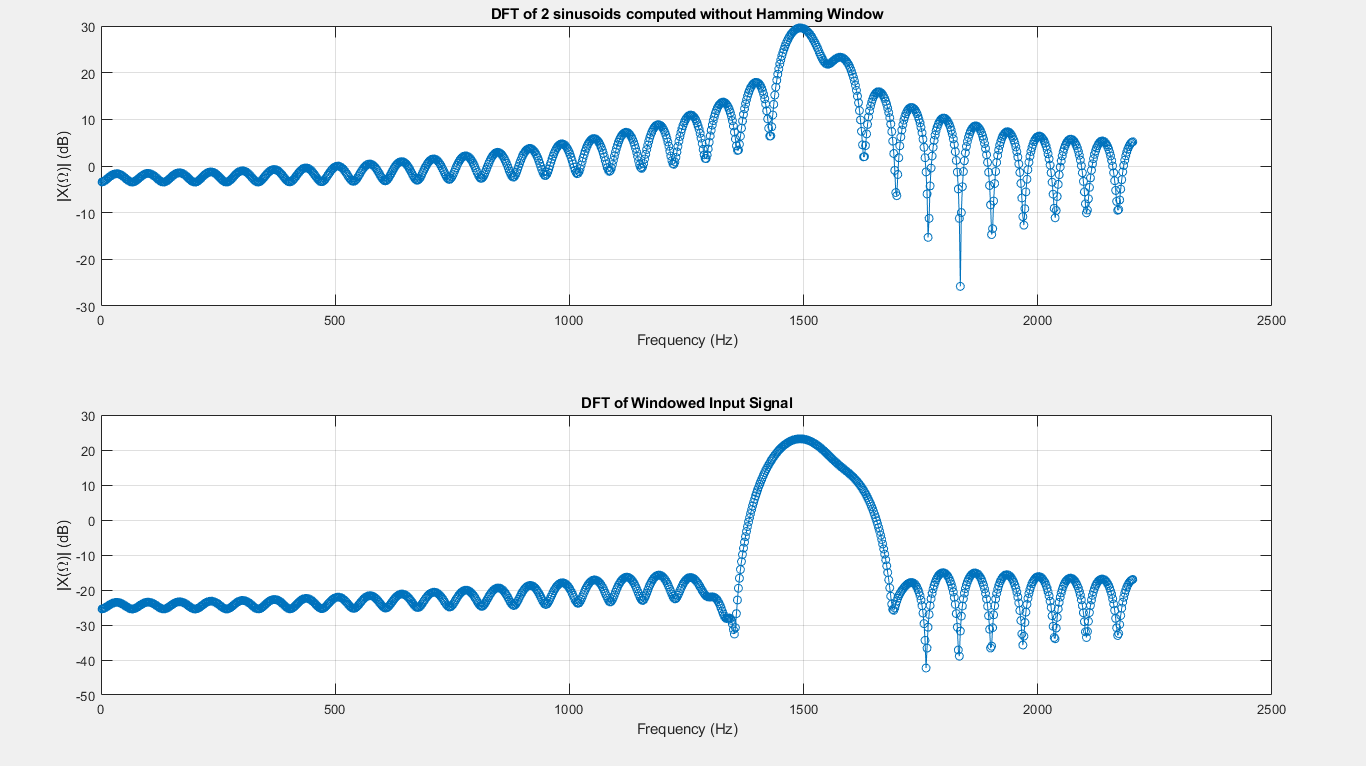
*Figure 12 - Results for 64 samples of the signal with no zero-padding*

Based on theoretical expectations, we expect to see two distinct spikes on our DFT plot. Yet, the plots in Figure 12 fail to depict this, as we have a very small amount of samples taken and no zero padding. The lack of these significantly affects what we can see in a DFT plot. Unlike in our previous plots, where we got lobe structures at component frequencies, in the plots above, we only get the DFT point at the top of the lobes. We’re also expecting two distinct spikes, however, figure 12 plots barely have 1 spike. This is because we have a small number, N of samples collected. The smearing that comes from such a small number of samples has smeared the two positive frequency spikes into a single spike. Zooming in on Figure 12’s plot, it becomes even more clear that we don’t see the expected two spikes at 1500 Hz and 1550 Hz.



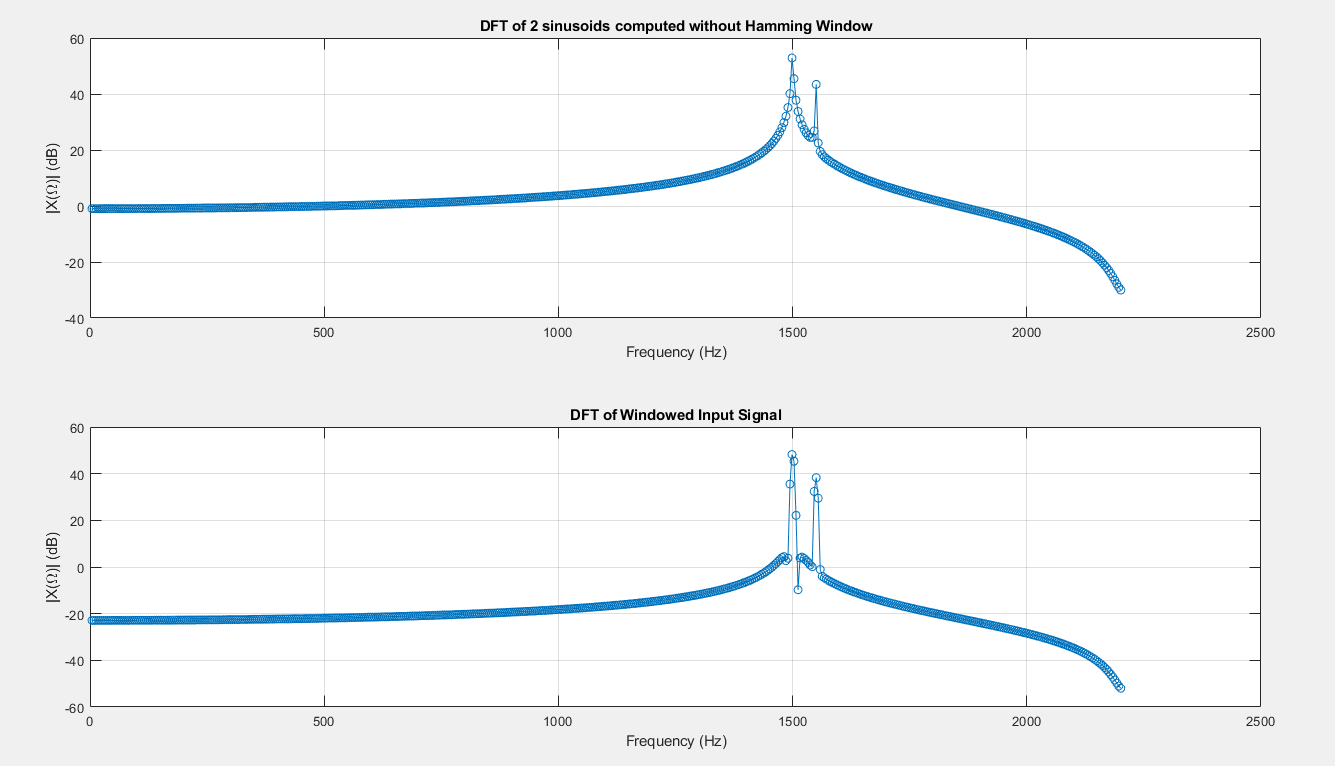
*Figure 13 - Zoomed-in version of figure 12 – demonstrating that we don’t see the delta function spikes where theory tells us they would occur*

Can zero padding fix this problem? From earlier discussion, it is the expectation that applying zero-padding will only give a better view of the DFT – it won’t fundamentally change the shape of the DFT plot in Figure 12.



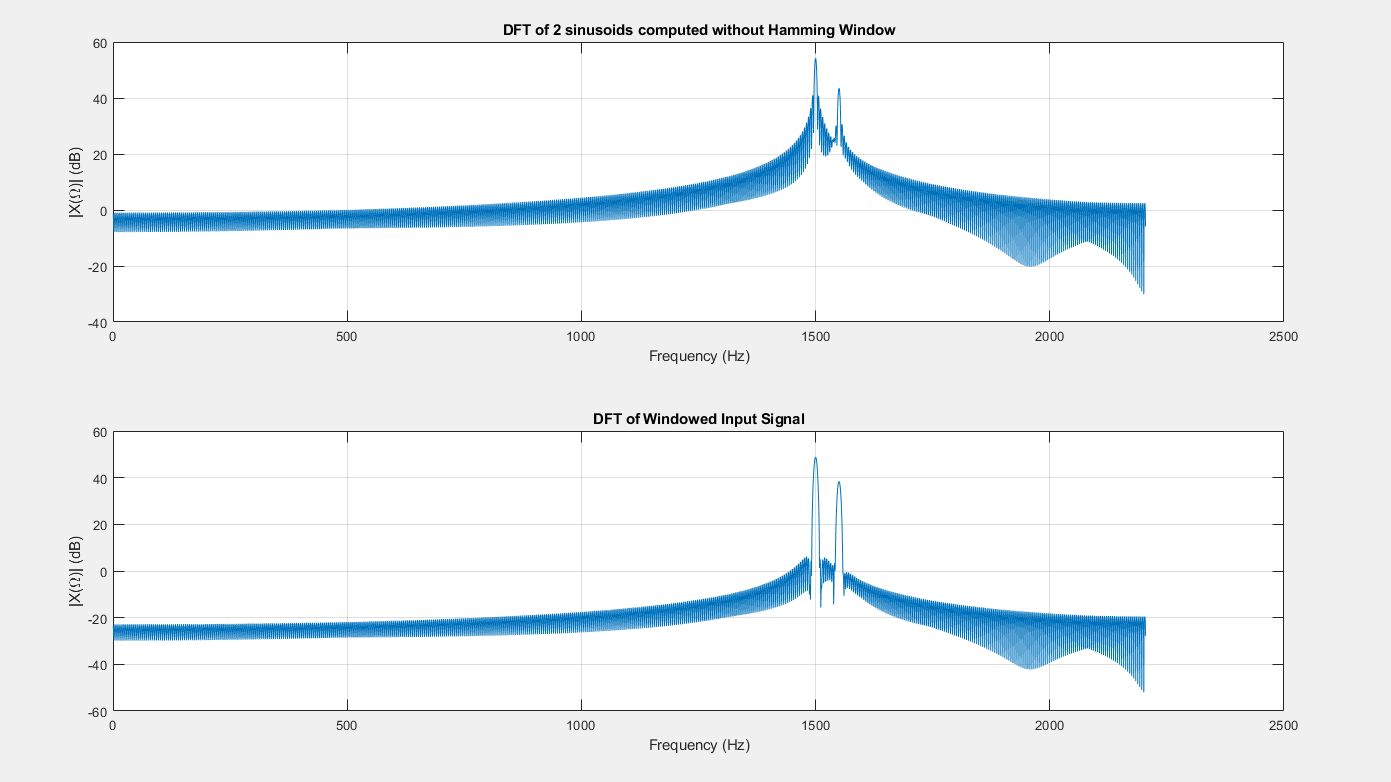
*Figure 14 - Results for 64 samples of the signal with zero-padding out to 2048 DFT points total.*

Figure 14 shows DFT plots with a lot more close points and the up-and-down “lobes” just as expected. One thing worth noting is the plots in Figure 12 and Figure 14 have the same points from the DTFT of the 64-point signal. The only difference is Figure 14 gives a better view of the underlying DTFT because we used zero-padding. Additionally, both plots suffer from the same degree of smearing because they both use the same amount of actual data points (64 samples). So, zero-padding didn’t fix the underlying problem – the smearing. Because of the spectral leakage, the two positive-frequency spikes get smeared into a single spike, side-lobe smearing. Zooming in on Figure 14, it’s observed that there are no spikes that occur at 1500 Hz or 1550 Hz, as theory suggests, even with zero-padding, another indication that zero-padding doesn’t fix the smearing; it gives a better view of the smeared DTFT, with more closely-spaced points on it. To fix smearing, more samples of the input signal are collected. To get an even better plot of the DFT, zero-padding can be combined with collecting more samples.



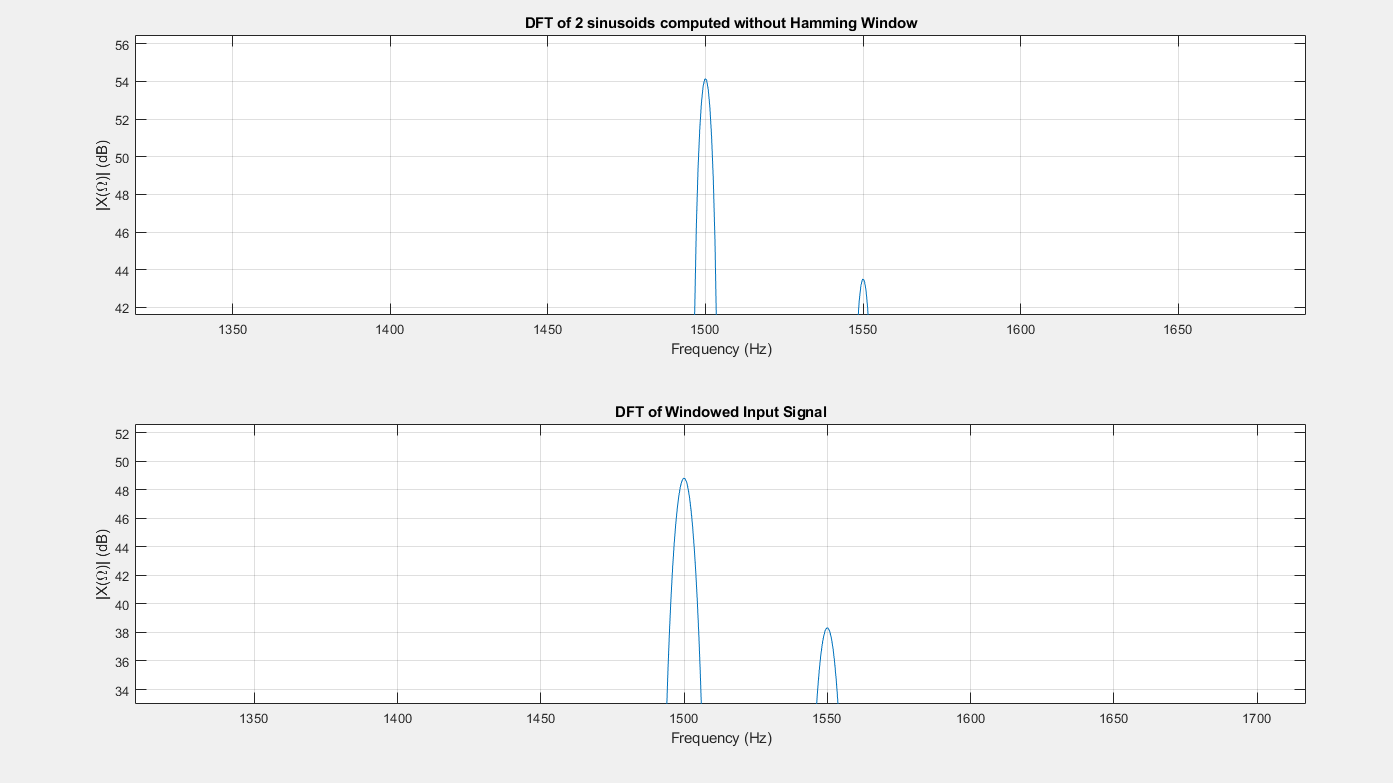
*Figure 15 - Results for 1024 samples of the signal with no zero-padding*

Figure 15 shows a better view of what the infinite-duration-of-data plot would show. We even get a coarse view of the input signal’s two frequency component spikes, although they appear jagged here. However, this can be attributed to the fact that there aren’t many points on this DFT plot. Using zero-padding can give an even finer view of this DTFT for 1024 samples of data.



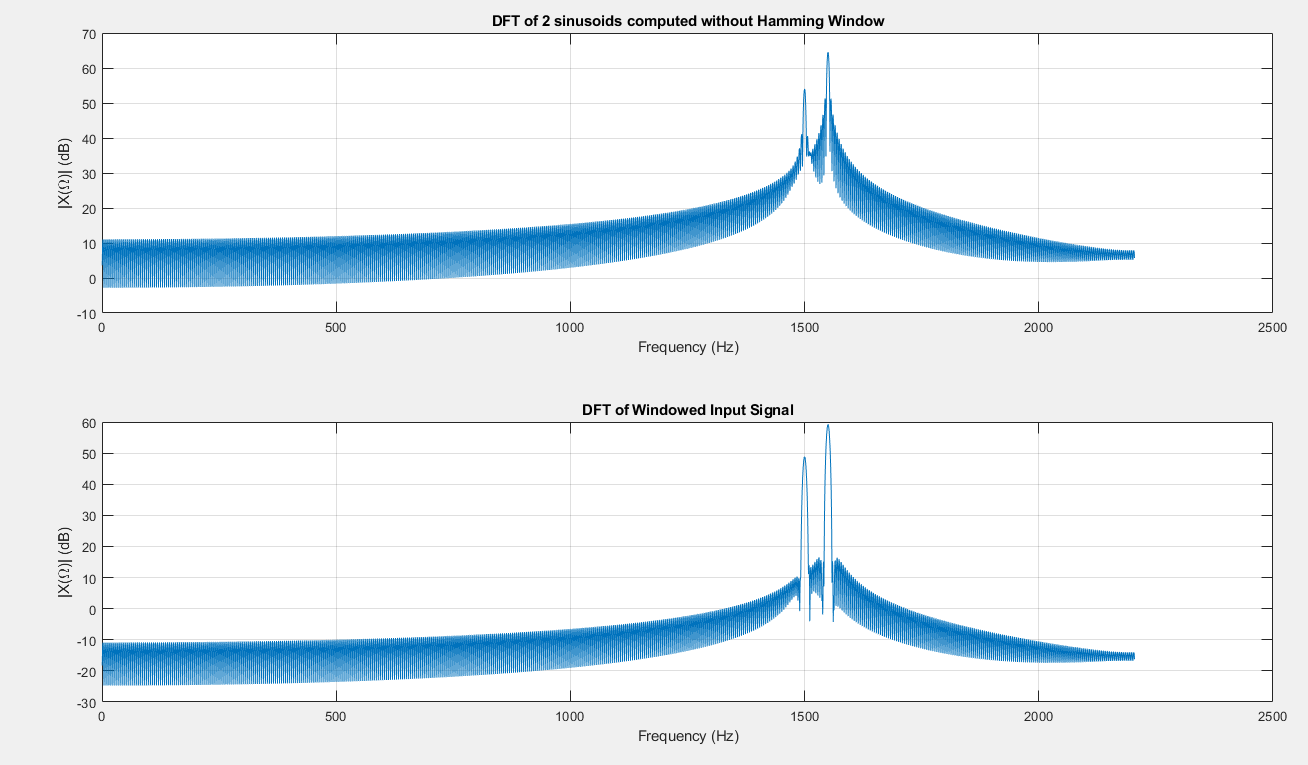
*Figure 16 - Results for 1024 samples of the signal with zero-padding out to 8192 DFT points total. Plotted with just a line rather than with circles so the shape of the plots is easier to see.*

Figure 16 combines everything so far discussed. By collecting a large number of samples and zero-padding, we get a good view of a signal’s frequency components, even when those components are close together. Similar to part 1, using a window on the signal emphasizes the spike at the component frequencies. Zooming in, we can see that the spikes occur at 1500 Hz and 1550 Hz, as our theory predicts.



*Figure 17 - Zoomed in Version of Figure 16 – Showing that the spikes occur exactly where theory predicts.*

One thing worth noting is the amplitude of the second spike is smaller than the amplitude of the first. This is because the amplitude of the sinusoid corresponding to the second spike was a smaller amplitude than that of the first sinusoid. Conversely, if the amplitude of the second sinusoid in the input signal was larger than the amplitude of the first, the DFT plot would reflect this as well.

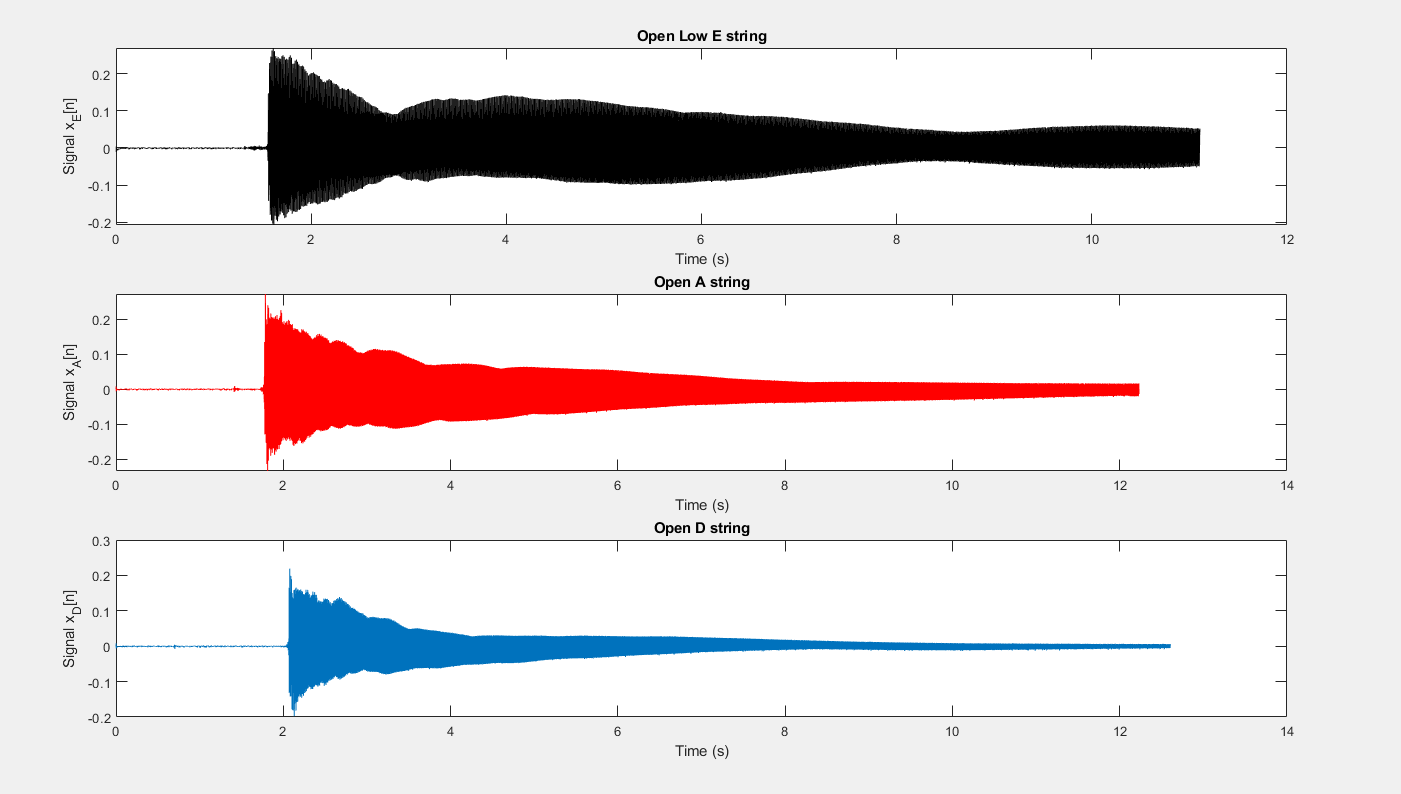


*Figure 18 - Showing the DFT of a signal consisting of two sinusoids. The amplitude of the second sinusoid is larger than the amplitude of the first. Consequently, on the DFT plot, the amplitude of the second spike is higher compared to the first.*

Now that we’ve explored how the DFT works with signals with either 1 or 2 frequency components, we can go into how the DFT can be used to analyze signals with many frequency components, specifically, the signals that come from a real guitar string being plucked.

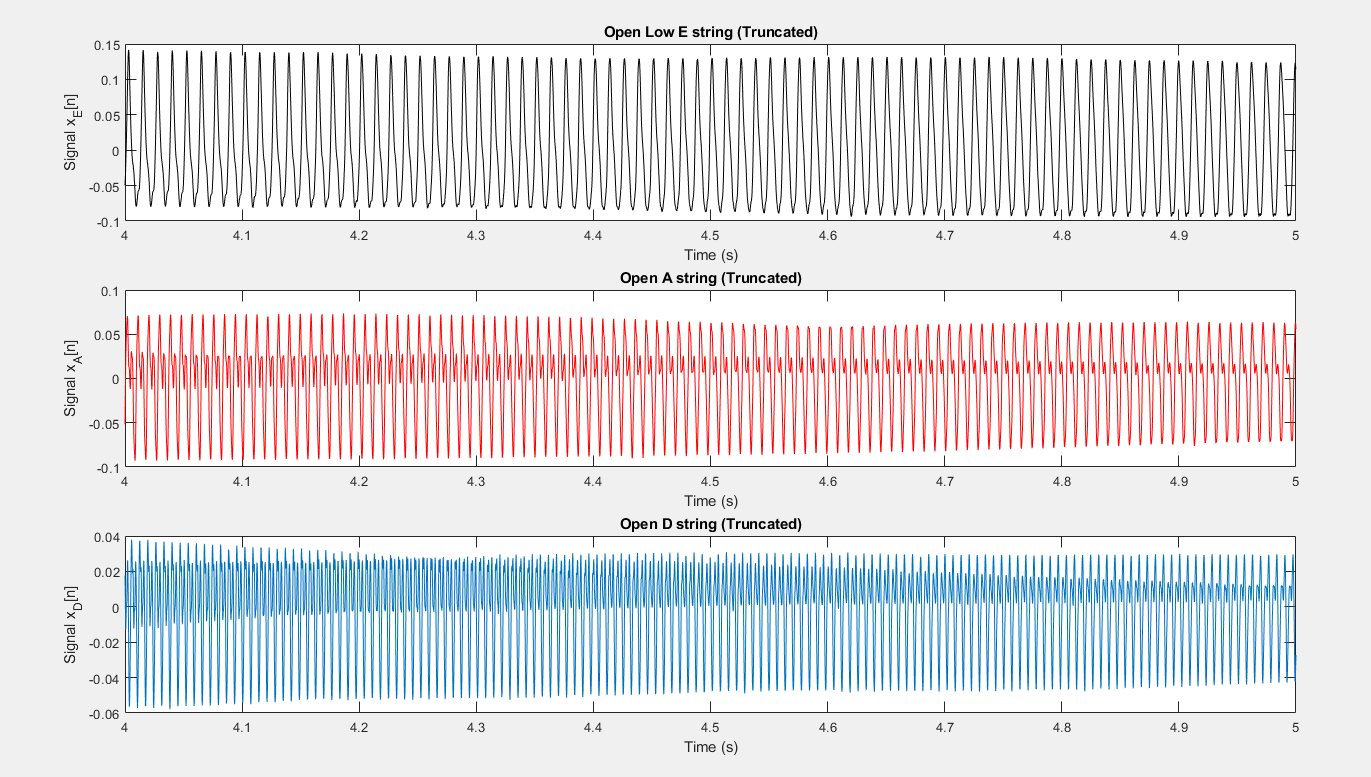
**TASK 3 – DFT of Individual Real Guitar Notes**

For this portion of the project, the DFT of real guitar open strings, low E, A, and D. Since, these signals are based on the vibration of a guitar signal, they are inherently periodic and oscillatory. In other words, they are periodic functions.



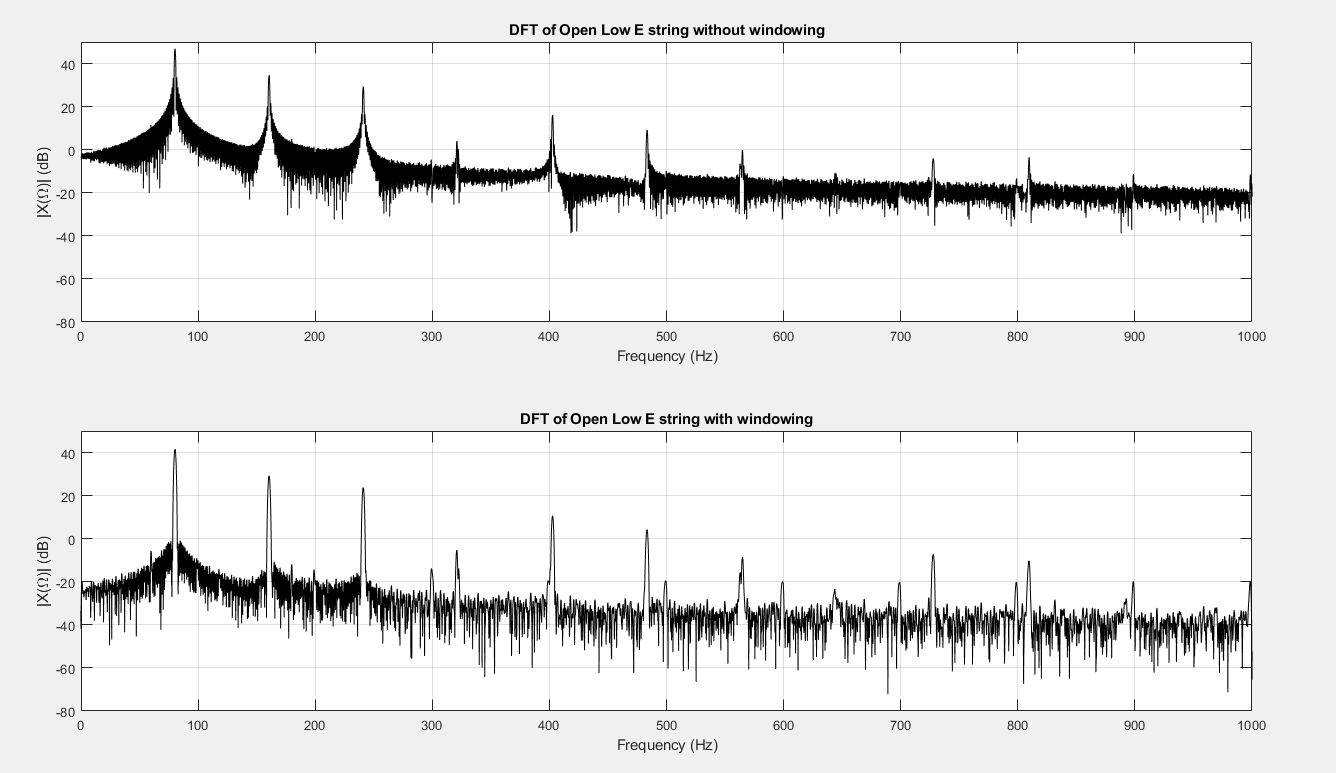
*Figure 19 - Signal from plucked guitar string*

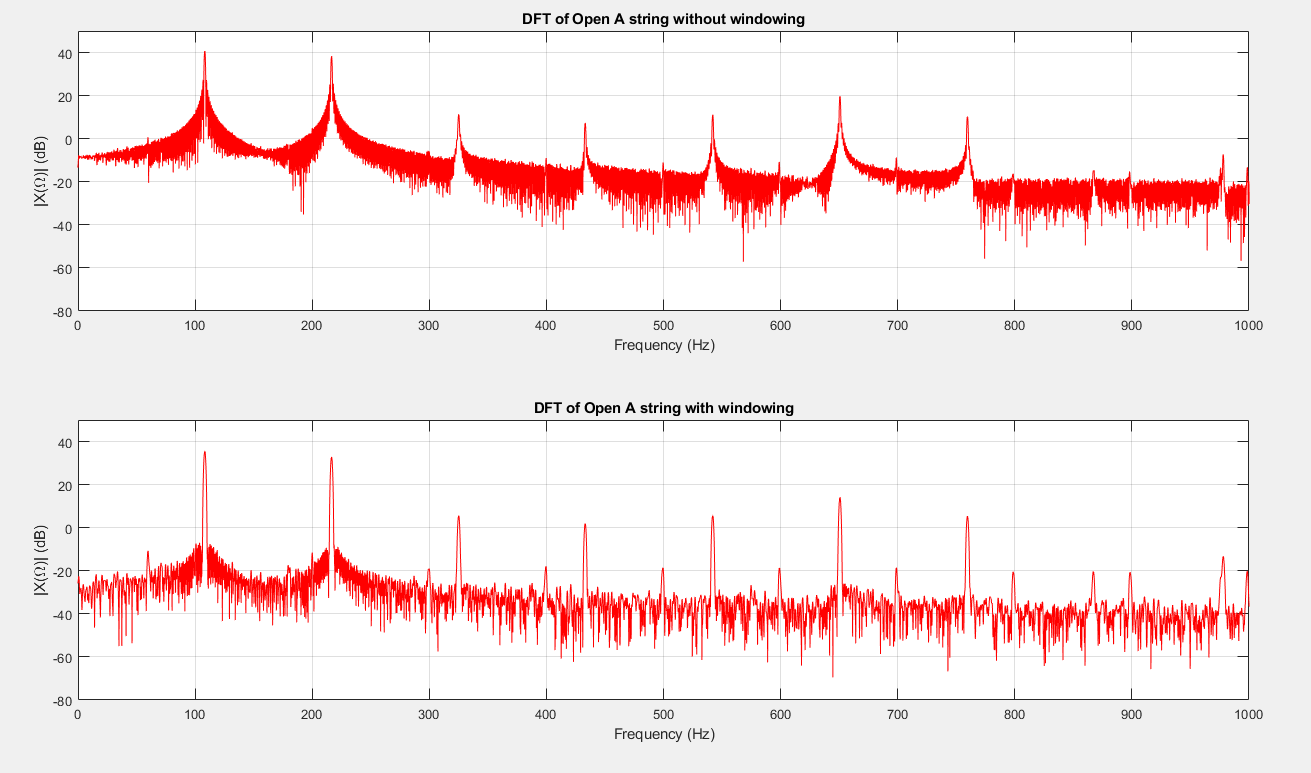
While Figure 19 doesn’t say much about the signals, zooming in on their waveforms for a 1-second duration, the oscillatory nature of the signal is accentuated. One thing to notice though, is while the waveforms are oscillatory, they are also decaying. The decay signifies how the guitar signal dies out with time.

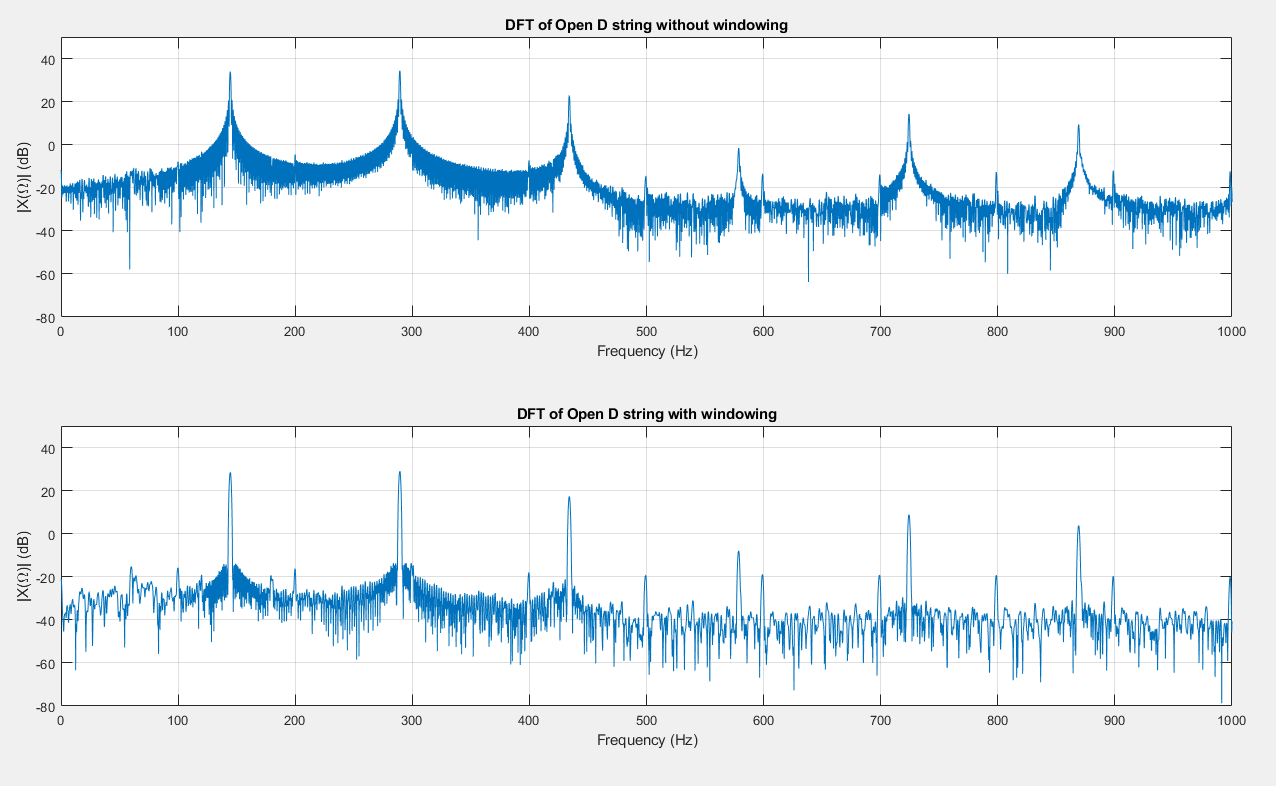


*Figure 20 - Showing the oscillatory nature of the guitar strings in the 5th second of their oscillation*

Figure 20 shows the sinusoidal waveform of the guitar signals, giving more insight about the guitar signals than the plots in Figure 19. Since the oscillation from a guitar’s strings is periodic, they have a Fourier series. Equation 1 shows how the mathematics behind how a periodic signal can be decomposed into simpler sinusoids. In the frequency domain, each of these sinusoids will have a delta function spike at their corresponding frequencies. For a guitar string, these spikes represent the harmonics of the string with the frequency of the first spike being its fundamental frequency. Based on the theories discussed earlier, we’re expecting to see spikes at the fundamental frequency and at integer multiples of the fundamental frequency of the guitar string. The guitar string produces harmonics at frequencies that are multiples of its fundamental (spikes in the DFT plot) because as a guitar string vibrates, it does so in multiple modes. The lowest of such modes is the string’s fundamental frequency and the other spikes we expect to see in the DFT plot are the natural harmonics that fundamental frequency produces. It is important to note that the DFT plots in subsequent figures are plotted up until a frequency value of Fs / 2. So this procedure will fail to pick up any harmonic components occurring after that point.

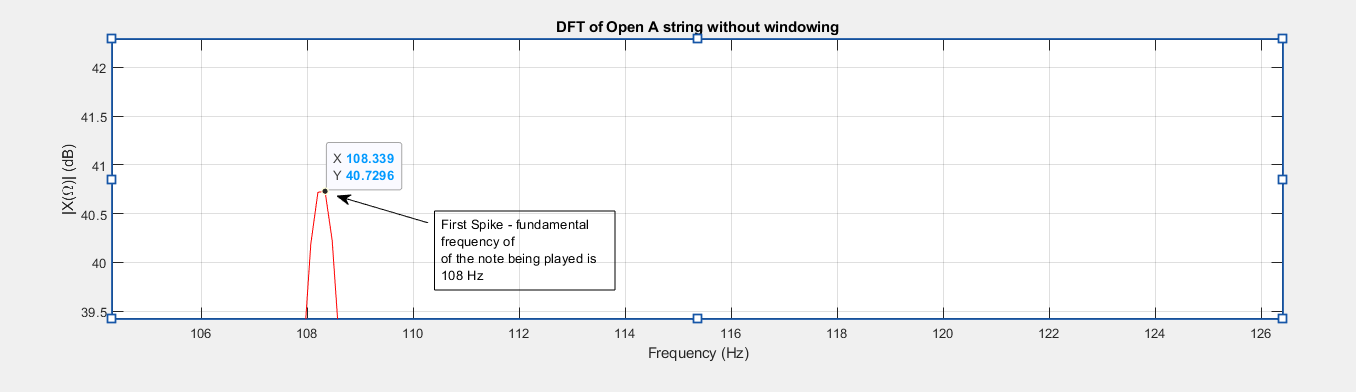




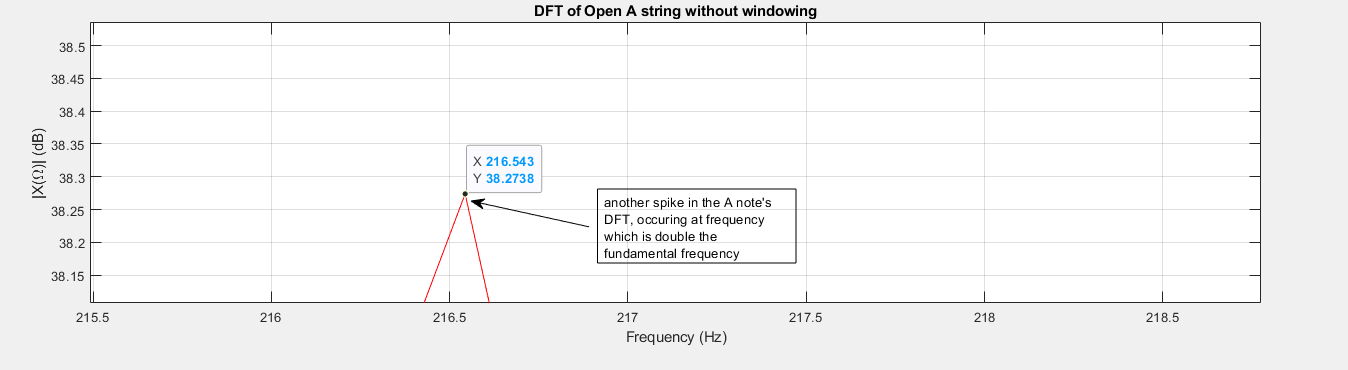


*Figures 21-23 - Showing the DFT of the guitar strings*

Figures 21-23 confirm the theories I’ve been building off of. Just from observing the DFT plots, I see the expected structure: spikes that occur at what seems to be integer multiples of the frequency of the first spike – the fundamental frequency of the note being played.



*Figure 24 - Showing the Fundamental frequency of the A string*



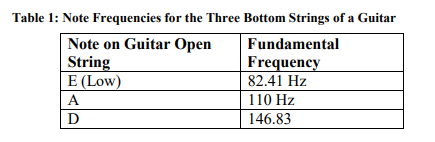
*Figure 24 - Showing a harmonic frequency of the A string*

Figures 23 and 24 provide a zoomed-in view of Figure 22, allowing us to see and reasonably estimate that the spikes in a guitar string’s spectral plot occur at frequencies that are multiples of the string’s fundamental frequency.

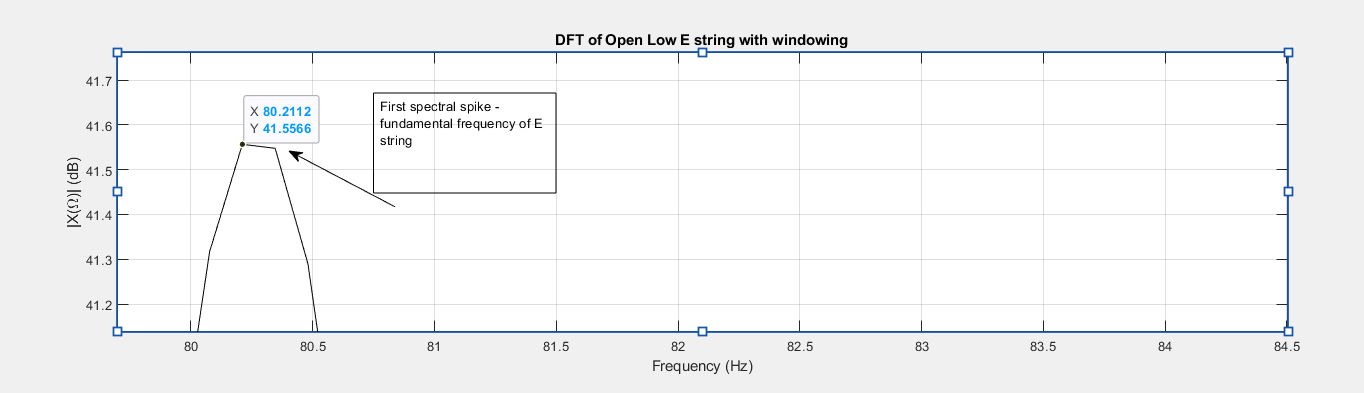
Similar to the previous parts of this project, the hamming window helps emphasize the lobe-like structure at harmonic frequencies. We’ve seen that the hamming window is great for signals that have frequency components close to each other, as it helps ‘separate’ their spikes. However, in this application, all harmonic frequencies are guaranteed to be sufficiently spaced from each other. But this doesn’t necessarily mean that using a window in this application isn’t practical. Guitar strings produce signals with many harmonics each, differing in amplitude. It is possible that spectral leakage causes stronger amplitudes to leak out to adjacent frequencies, distorting the view of the weaker harmonics. Windowing helps combat this by reducing spectral leakage, so each of the spectral spikes remains intact.

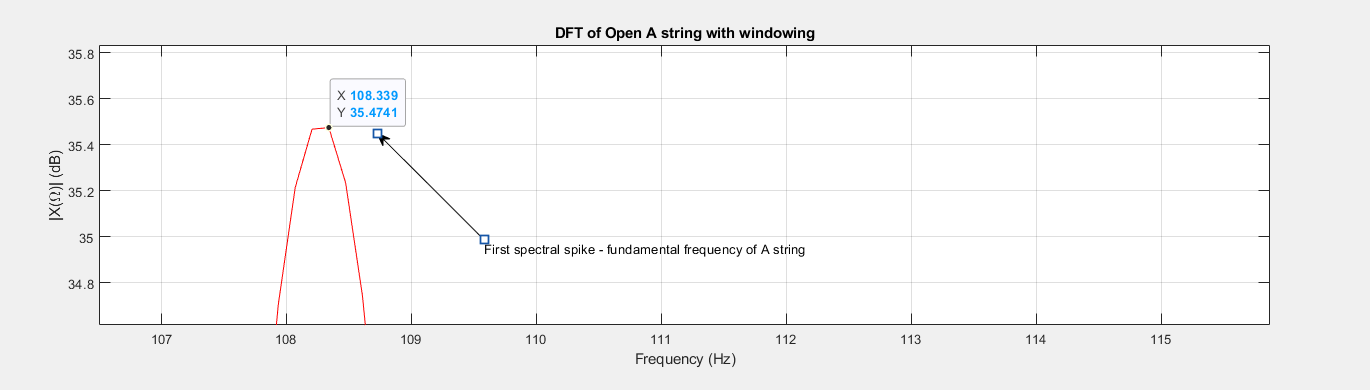
Comparing the fundamental frequency of the guitar signals with the standard fundamental frequencies of the notes the signals are emulating, we can easily see how in-tune or out-of-tune the guitar strings are, by comparing the frequency of the guitar string, f2, and the target frequency f1, and obtaining the accuracy, in cents, of the guitar’s string. This mathematical relationship is described below:

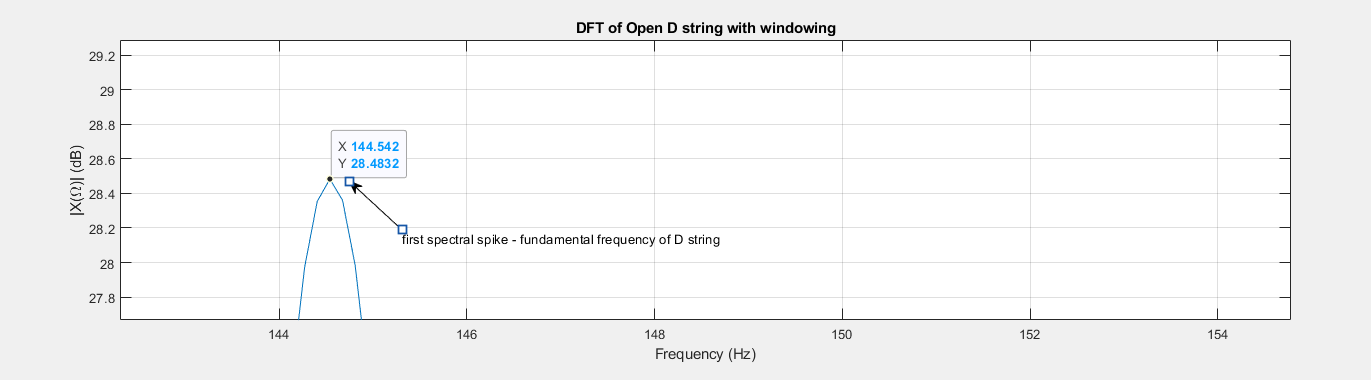
|  | (4) |
| --- | --- |



By simply comparing the fundamentals of the guitar strings from the plots with the values in the DFT plot with the values in Table 1, we can easily tell if a note is perfectly tuned or not.







*Figures 25-27 - Showing the fundamental frequencies of the guitar strings*

Figures 25-27 show the fundamental frequency of the guitar strings (x - value). By inspection, it is easy to see that the guitar string’s fundamental frequencies are slightly out of tune, by roughly 2 Hz or -46 cents, a significant value by most tuning standards.

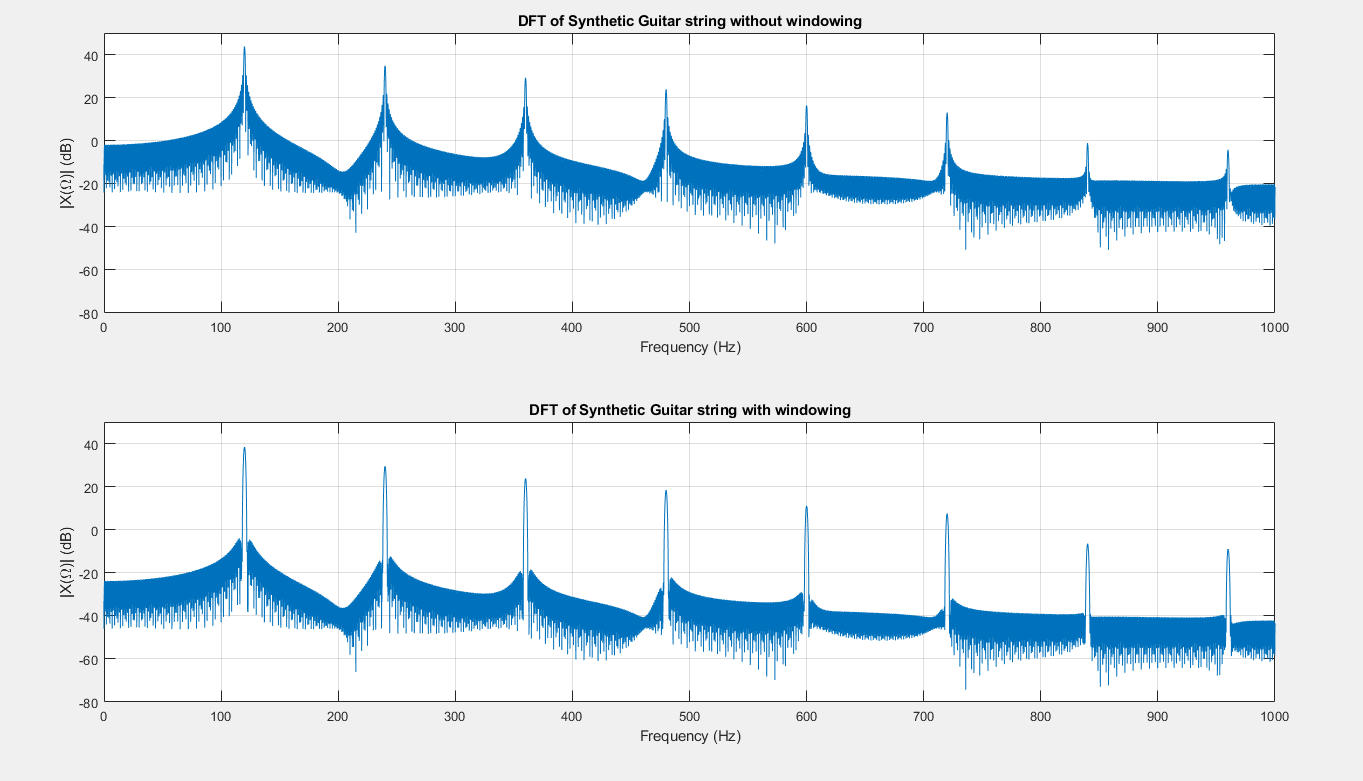
To further test how the DFT can be used to analyze notes from a guitar, we’ll need a whole variety of guitar signals, which aren’t always readily available. To get around this, a synthetic guitar signal can be made by modeling the oscillatory motion of a guitar string when plucked by a periodic function. By doing this, we can create a guitar signal of any frequency of choice.

**TASK 4 – DFT of Individual Synthetic Guitar Notes**

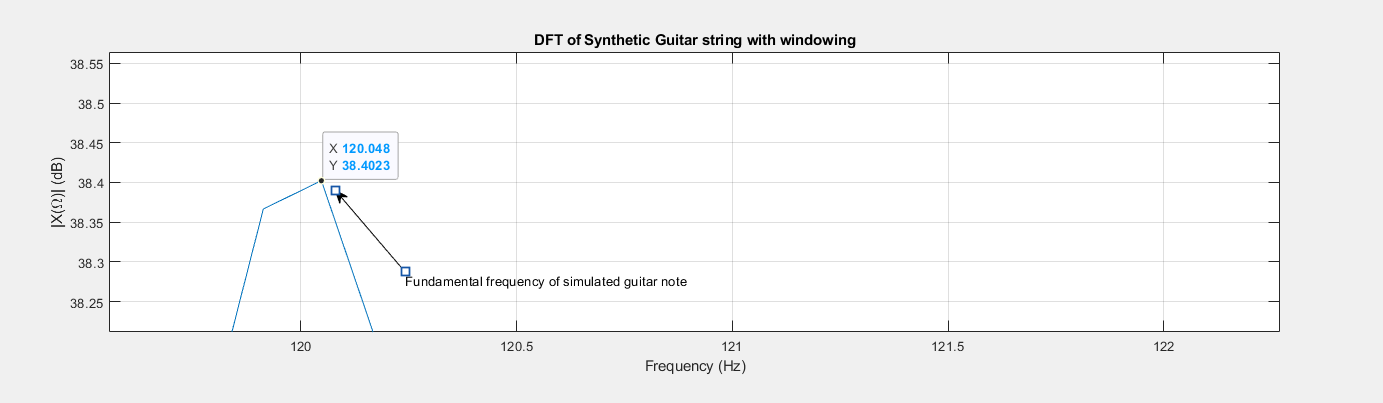
Having analyzed real guitar signals, we can create a synthetic guitar signal for which we can continue DFT analysis. With a synthetic signal, we can choose the fundamental frequency to be whatever we like, and we aren’t limited to the fundamentals of 3 strings on a guitar. To construct this synthetic signal, I use the Amplitude Phase model of the Fourier Series, which expresses a periodic signal as a sum of its component cosines with amplitude Ak and phase shift, Φk.

|  | (5) |
| --- | --- |

With the constants given in the project description, a guitar string of any fundamental frequency can be simulated.

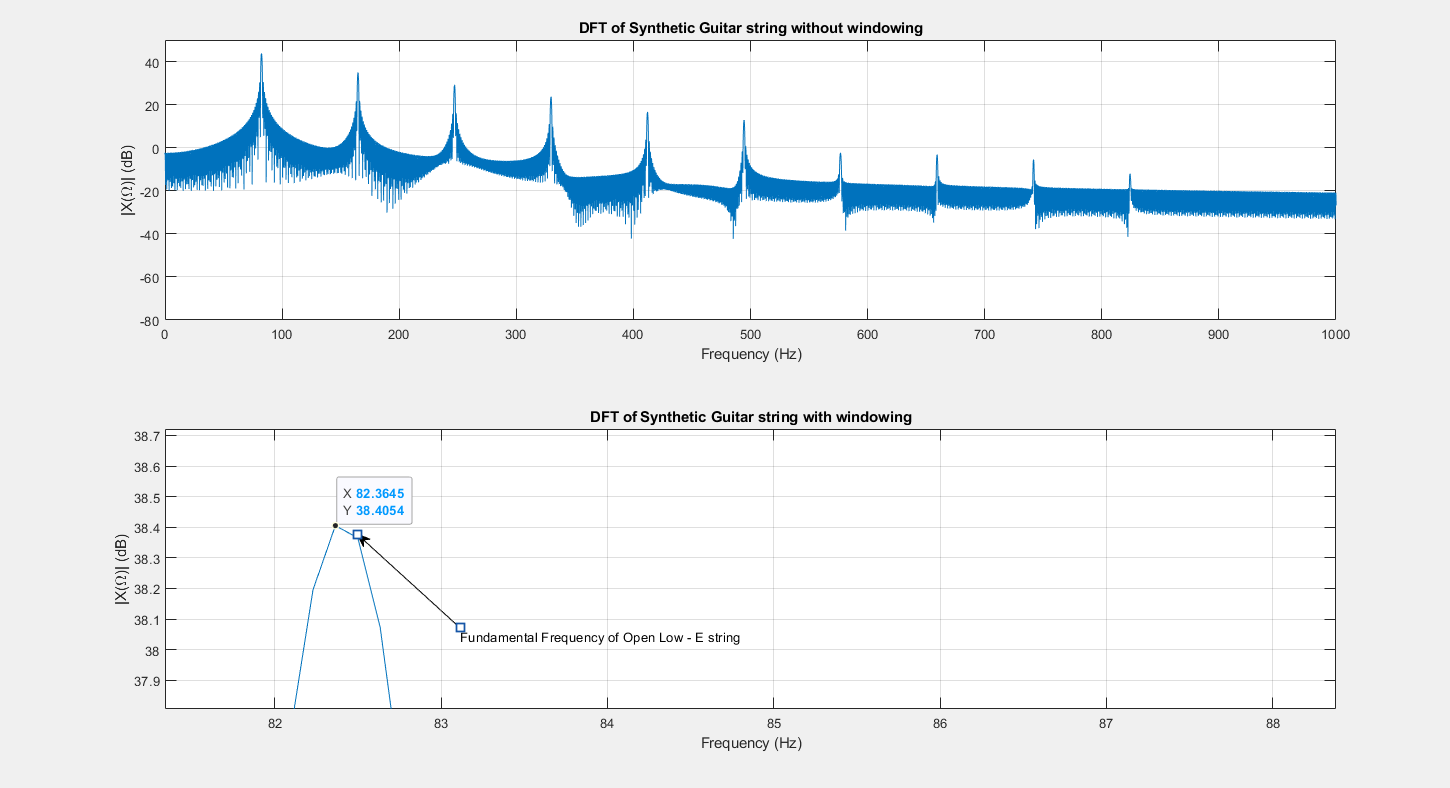


*Figure 28 - DFT of a synthetic guitar string with a fundamental of 120Hz. The advantage of using a synthetic signal is notes of any frequency can be simulated*

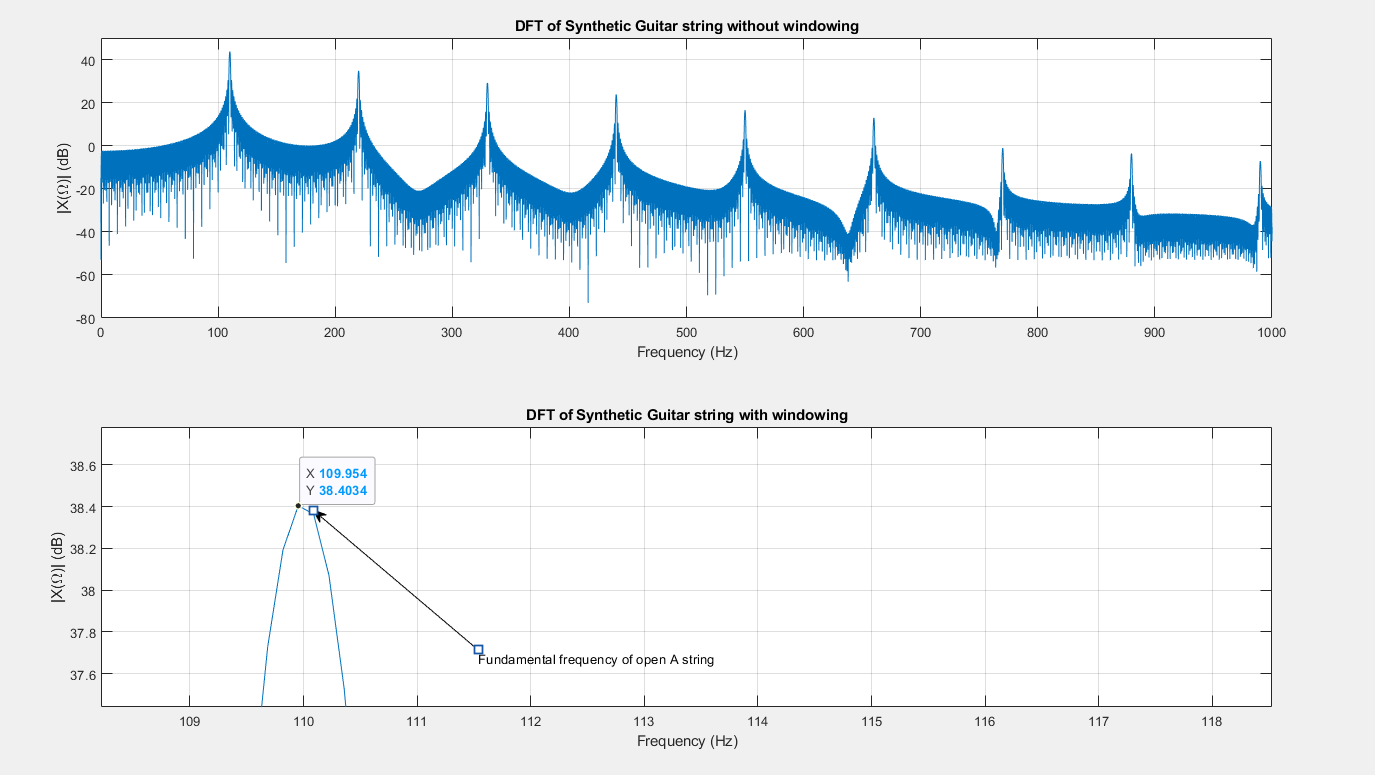
****

*Figure 29 - Zoomed-in version of Figure 28*

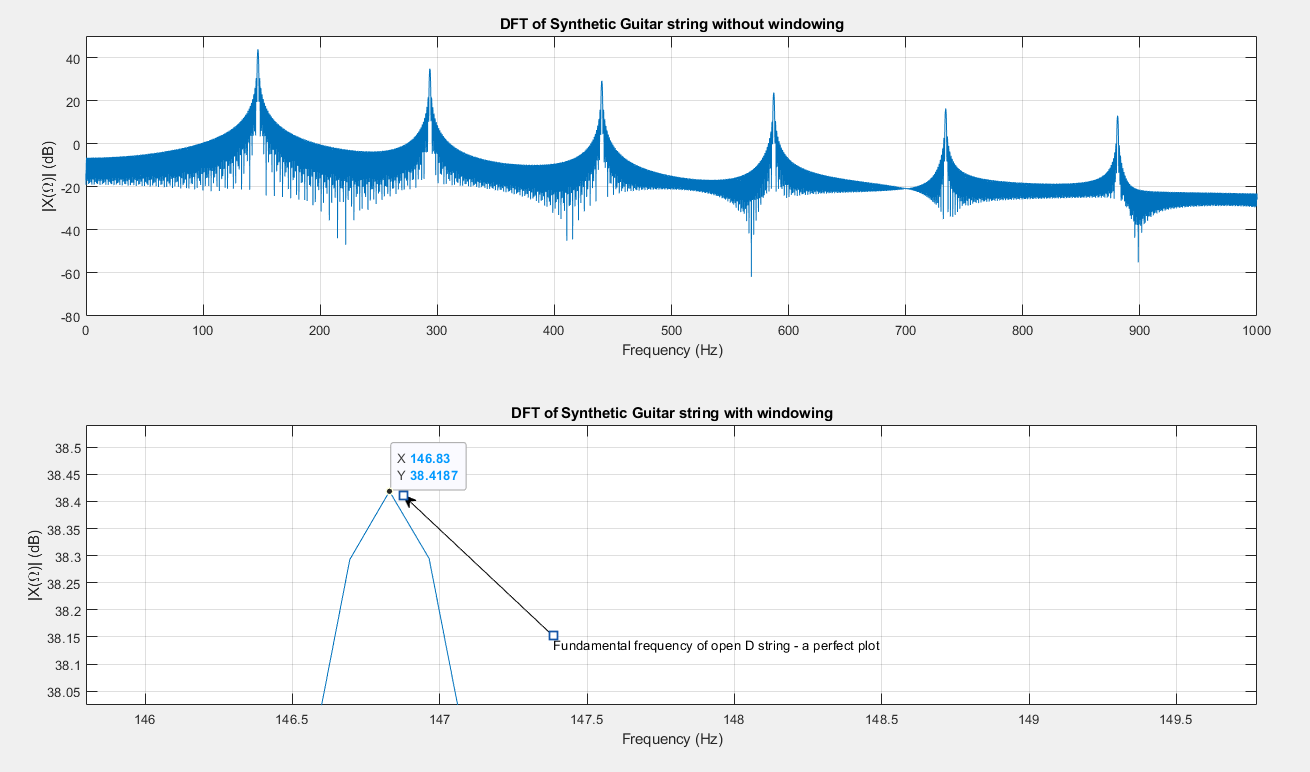
Figure 29 shows the fundamentals of the synthetic guitar note in Figure 28. The plot in figure 29 accurately depicts the fundamental frequency of the note being played as 120.048Hz which deviates from the true 120 Hz by only 0.70 cents, a negligible deviation. With this degree of precision, the E, A, and D strings from Part 3 can be simulated and the plots produced will more accurately show the corresponding fundamental frequencies.



*Figure 30 - Showing the Fundamental Frequency of a simulated Open Low E string*



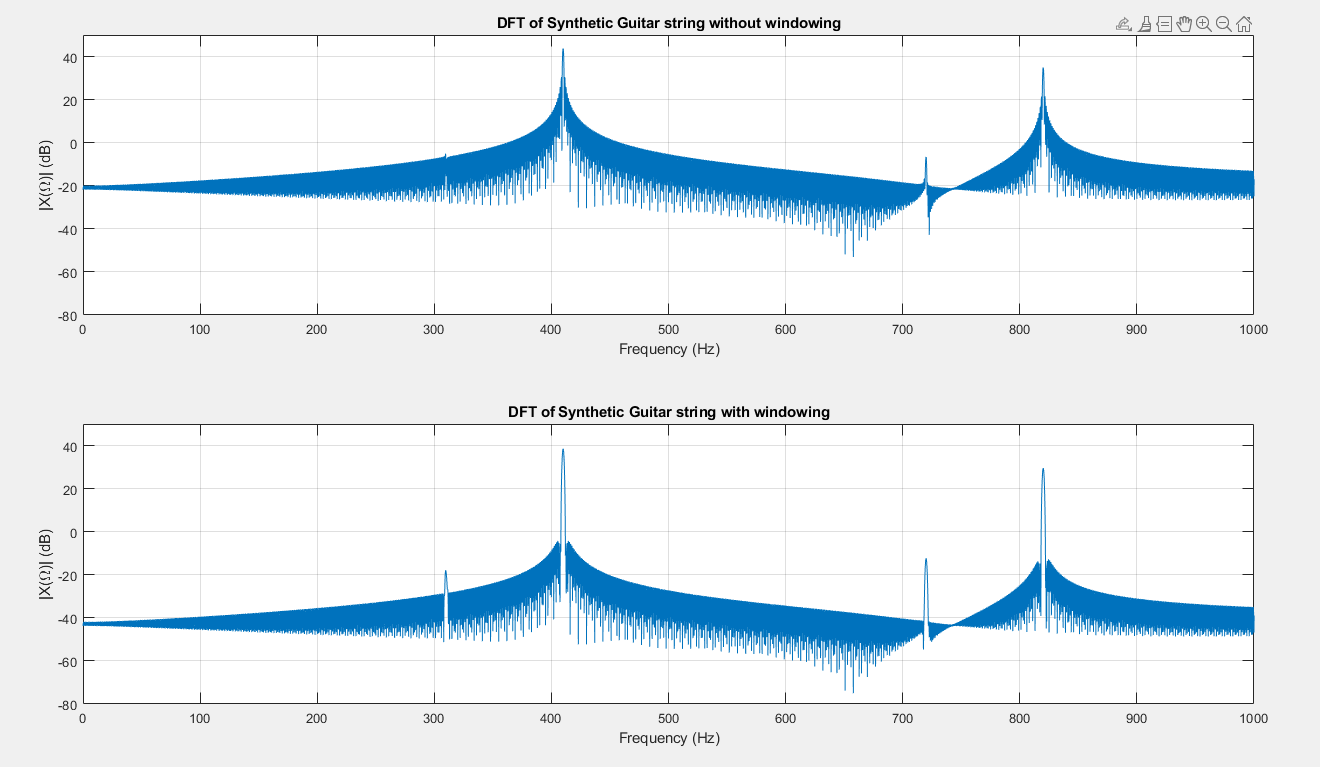
*Figure 31 - Showing the Fundamental Frequency of a simulated Open A string*



*Figure 32 - Showing the Fundamental Frequency of a simulated Open D string*

The above figures show that with a high degree of accuracy, a synthetic signal can be simulated given a specific fundamental frequency. The DFT plots for the synthesized guitar signal are also cleaner than the DFT plots for the real signals, with the magnitudes at every other frequency being approximately zero, except at harmonic frequencies where there is a delta-function spike.

As previously mentioned, to avoid aliasing, signals must be sampled at the Nyquist Rate or faster. In the context of frequency components of a musical instrument, a guitar in this case, the fundamental frequency of the instrument must be the Nyquist frequency (Fs / 2) at most. Any note with a higher fundamental will be improperly sampled and aliasing will occur, making it impossible to measure anything from such a signal’s DFT plot. So the functionality of the synthesizer is limited by the sampling frequency.



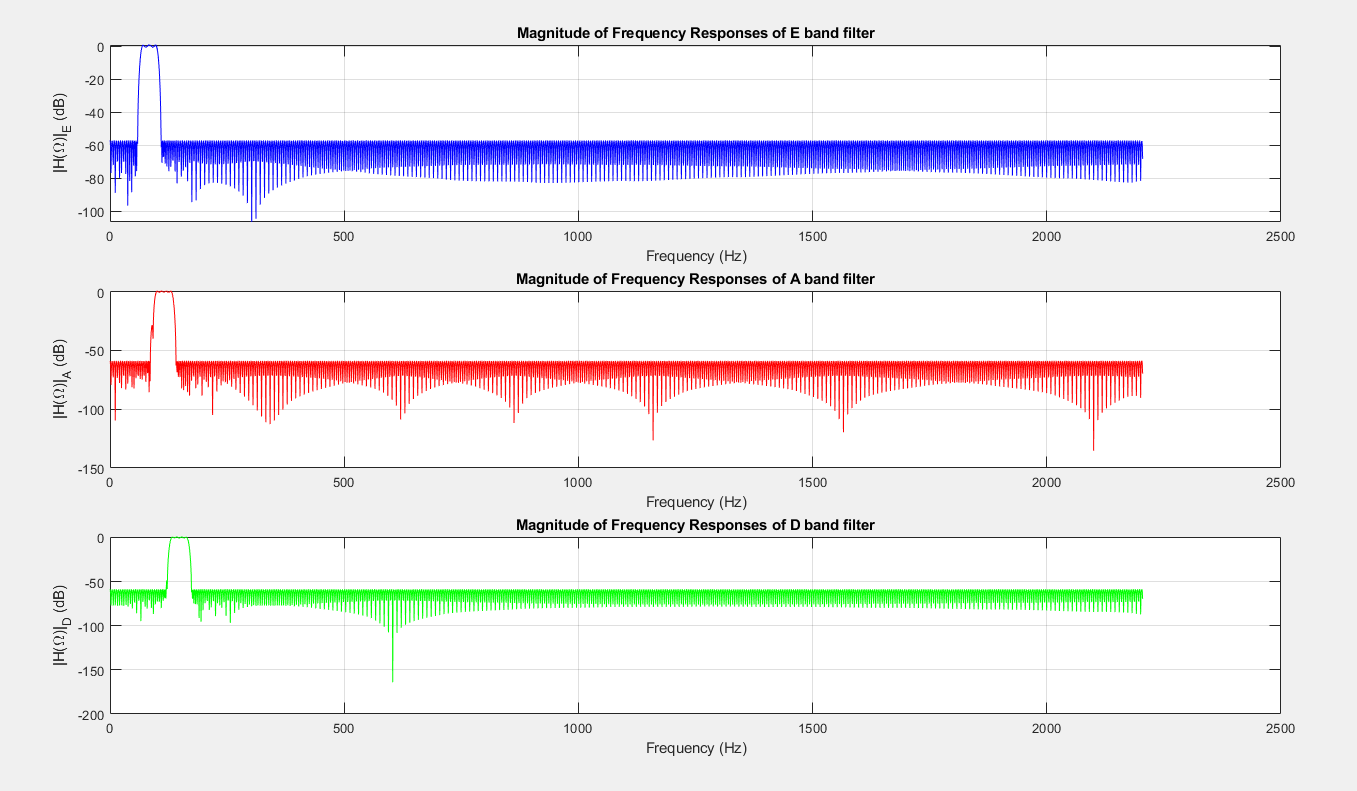
*Figure 33 - Results for a synthetic signal with a fundamental frequency of 4000Hz. Such an input frequency violates the sampling rule, and this leads to aliasing. Very little information can be extracted from the above plots due to aliasing.*

With a function capable of synthesizing the sound of a plucked guitar string, a filter for detecting what note an out-of-tune guitar is closest to can be designed, and the synthetic guitar signal can be used to test the functionality of the filter.

**TASK 5 – DFT-based Guitar Tuner**

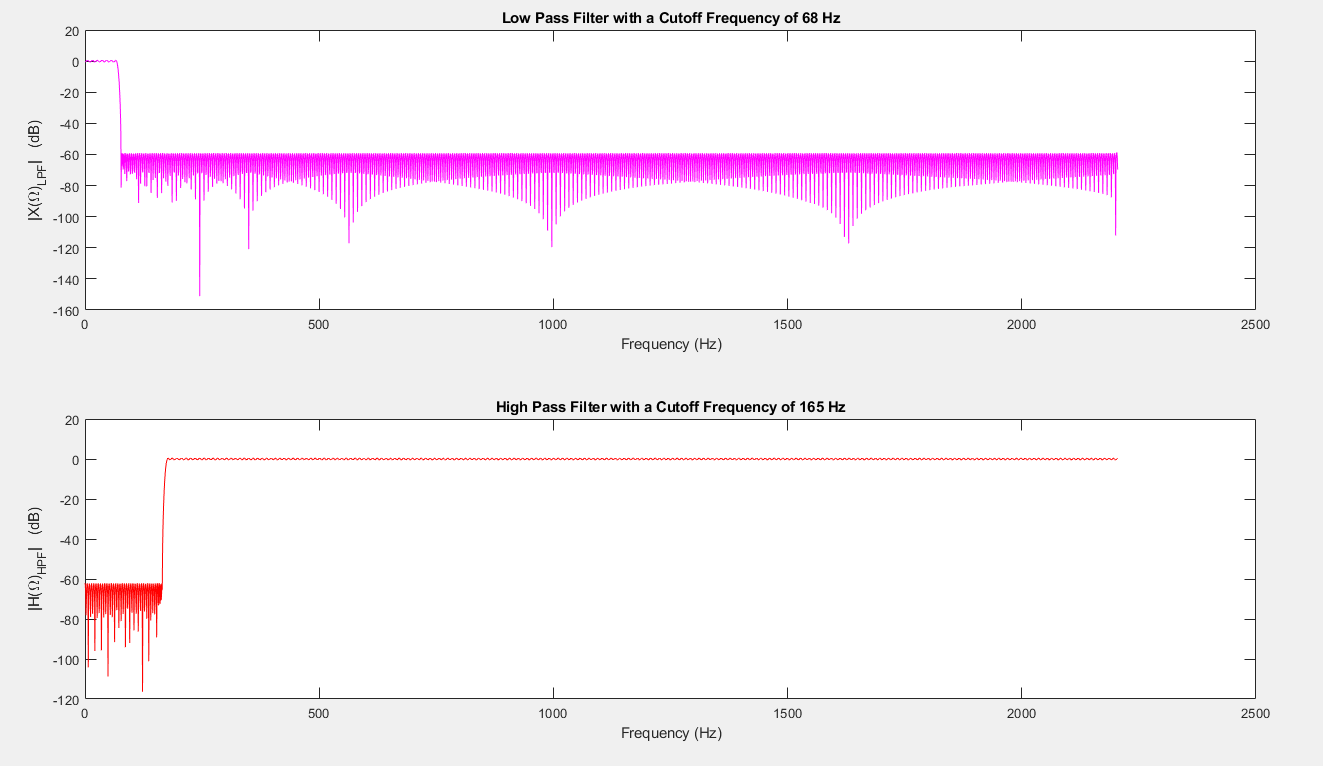
Building on the insight gained from the DFT analysis of a synthesized guitar signal, a more practical application of DFT analysis, designing a guitar tuner, can be explored. We can use a frequency analysis technique such as filtering, to detect what band of frequencies the input signal lies in. This is done by computing the power of the signal after filtering it. From the insight, we can determine if the note is within a certain band of frequencies. For the purpose of this project, the DFT tuner being designed will only work to identify if the frequency of the guitar string being played is close to the frequency of open E, A, or D strings. All other frequencies outside this frequency window will be considered out of our range of interest. We use the same procedure of filtering and finding the power of the filtered signal to determine if the guitar string being played is out of the frequency range of interest.

The first part of designing the tuner is creating band-pass filters that detect if the guitar string is within a specific frequency range. Since our tuner detects nearby frequencies for the notes, E, A, and D, three bandpass filters are created, each corresponding to the frequency band of one of the three notes the tuner can detect. The filters are designed using the Parks-McClellan routine and the FIR filter coefficient vectors are then passed as outputs to the function



*Figure 34 - showing the bandpass filters for showing notes in E-band, A-band, or D-band*

Figure 34 shows that the filters work by passing the frequencies within their band and attenuating all other frequencies with about 60dB of attenuation, tanking a signal’s amplitude to below the threshold of hearing. To detect if the fundamental of the guitar string is above or below the frequency range of interest (68 Hz – 165 Hz), I created a low pass and high pass filter using the same procedures described above.



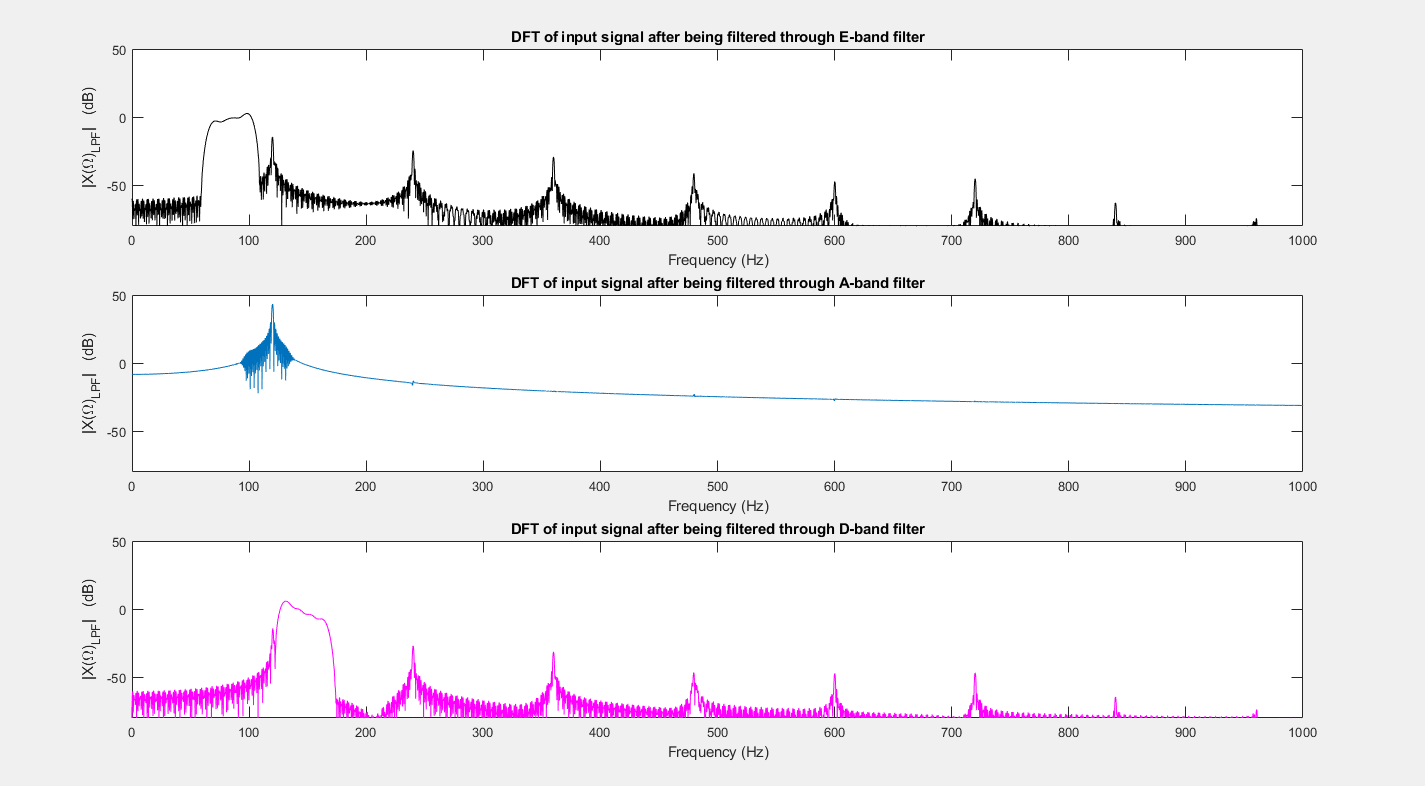
*Figure 35 - Low Pass and High Pass Filters*

After designing the filters, the input signal is then passed through all the filters, leaving us with 5 versions of the input signal, each passed through one of the filters created. By analyzing the power in the different versions of the filtered input signal, we can determine what band the signal is in. To compute the power in a filtered input signal, we square each term in the filtered signal, add them all up, and divide by the number of terms. Performing this computation gives the average power in the signal. This works because squaring each term in the signal captures both the negative and positive parts of the filtered signal, and dividing by the length of the signal gives the average. For periodic signals, this relationship is described as follows:

|  | (6) |
| --- | --- |

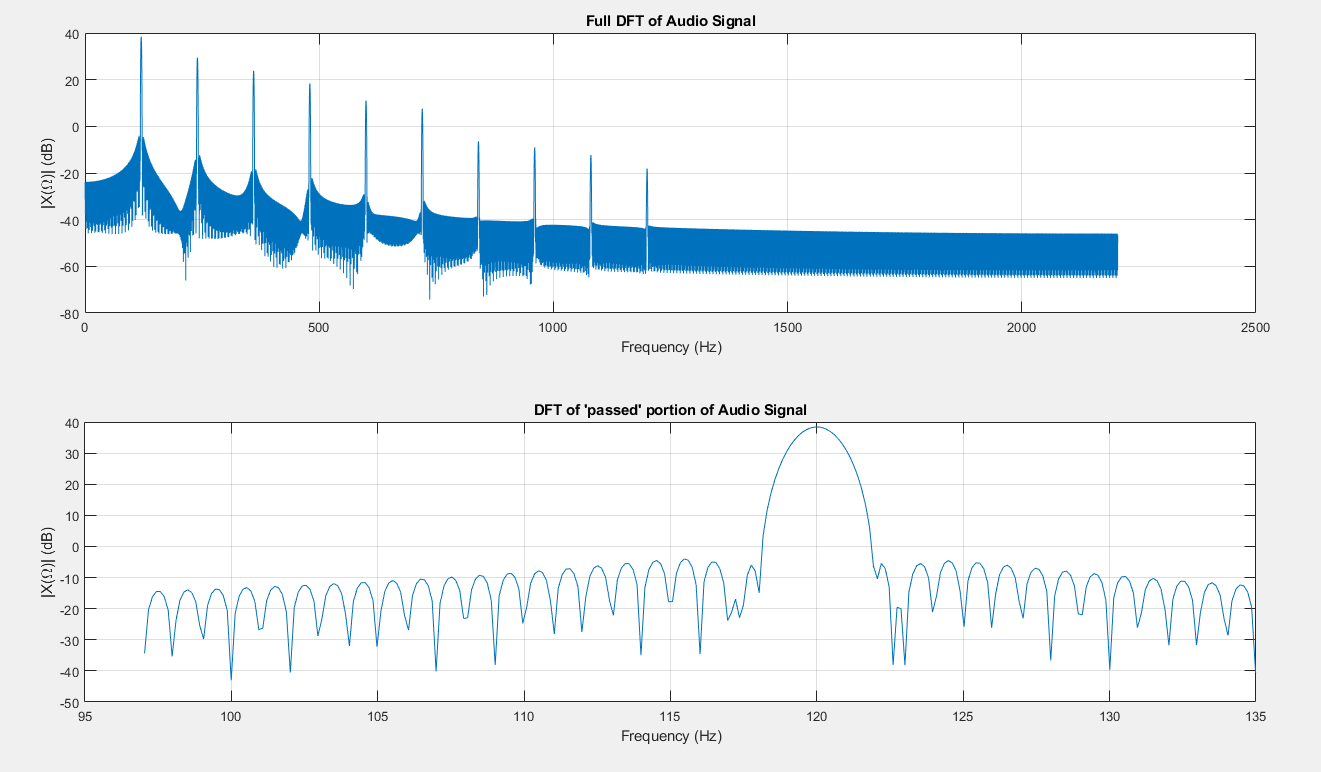
The input signal with the most power coming out of the filter will have its strongest frequency components over the band that the filter passed. Consequently, by computing the power of the filtered input signal, we get an indication of where the fundamental frequency of the signal is.

Consider the guitar signal in Figure 30 of a note being played with a fundamental frequency of 120 Hz. To determine the fundamental frequency of the signal, the signal is passed through the filters,



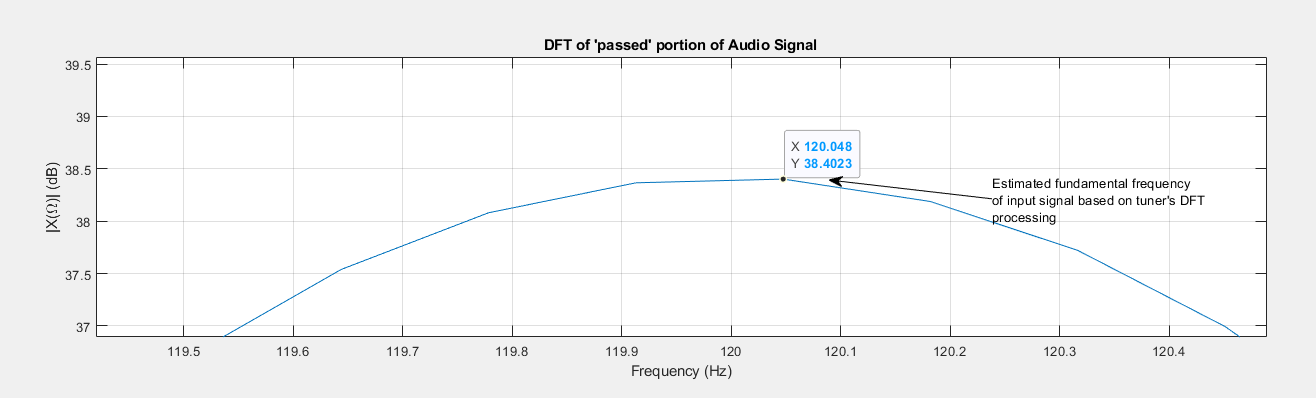
*Figure 36 - Showing the DFT of the input signal after being passed through each of the filters*

Figure 36 shows how the DFT of the plot of the input signal after being passed through each of the filters looks. Just by observing the plots, it’s clear to see the output from the A-band filter has the greatest magnitude, indicating that the signal’s most powerful frequency components align most closely with this band. This gives a good indication that the note being played is an off-tune version of the A string. In a more complicated guitar tuner, this information allows us to determine what direction the string should be tuned the string in order to achieve the correct frequency. However, in this simplistic approach, this analysis allows us to identify which note, a guitar string is closest to playing.



*Figure 37 - A closer view of the DFT of the input note and its frequency band that is passed from the A-band filter*

By zooming in on the bottom plot in Figure 37, we can determine what fundamental frequency the tuner has estimated for the input signal.



*Figure 38 - Showing a zoomed-in version of the bottom subplot in Figure 37*

Figure 38 shows that the fundamental frequency of the guitar string was estimated to be 120.048 Hz, which is a superb estimation. Using Equation 4, we calculate the accuracy of the estimated fundamental to be 0.69 cents, indicating the difference between the estimated frequency and true frequency is negligible. This confirms the proper functionality of the DFT tuner.

It is reasonable to expect that for guitar signals with fundamental frequencies within the range of 68-165Hz, our tuner will accurately estimate the frequency of the note being played as well as indicate what the desired note is. Additionally, by applying the low-pass and high-pass filters in the same way the bandpass filters were applied, our DFT tuner can determine whether the fundamental frequency of the input signal falls outside the frequency window of interest.

It’s worth investigating the tuner’s handling of edge cases. For signals at the edges of the frequency window, the tuner’s performance varied depending on which edge was being tested. I observed that the tuner was able to correctly note that signals with a fundamental frequency greater than 165 Hz were outside our range of interest. This held true even for aliased signals.

However, the tuner fell short when handling frequencies at the lower boundary. Specifically, the DFT Tuner incorrectly asserted that a note with a fundamental frequency of 68Hz was below our frequency range of interest, rather than identifying that the note was in the E-band as one would expect. This demonstrates a potential drawback in how the DFT tuner handles frequencies at the lower boundary of our frequency range of interest.

On the other hand, when I tested the Tuner with a note with a fundamental frequency of 165 Hz, the tuner correctly estimated the note’s frequency and correctly identified that the note was in the D-band. This shows the Tuner handles frequency at the upper boundary better than it handles frequencies at the lower boundary.

Perhaps, this behavior could be due to the filter design, or due to the resolution of the DFT plot. In either case, rigorous testing with frequencies near the lower boundary of the frequency range of interest is needed to determine whether the problem lies in the filter’s design or in the resolution of the DFT plots.

**CONCLUSION**

This project explores the DTFT and challenges if the theories discussed in class agree with what we observe for both real-world audio and synthesized audio signals.

In Part 1 of the project, we confirmed that the DFT of the sinusoids are delta-function spikes, and also went into detail on how various parameters such as the sampling frequency, the number of samples collected, the extent to which zero-padding is used, and the effect of windowing can affect our view of the DFT of a signal. The experiments performed here demonstrate how these factors affect the representation of a signal’s frequency content.

Part 2 of the project expands on the analysis of Part 1, exploring how the DFT handles signals with 2 component frequencies. It was seen that windowing is a good technique to reduce spectral smearing. For signals with closely spaced component frequencies, windowing helped give two distinct delta-function spikes, by reducing the smearing and emphasizing the lobe-like structure of the spikes.

Part 3 explores real-world signals with many frequency components. It was seen how the DFT can be used to pick out the component frequencies of these kinds of signals. This portion of the project also confirmed the Fourier Series theory, by showing periodic signals are comprised of simpler sinusoids. In accordance with this theory, the DFT plots contained delta-function spikes only at frequencies that were multiples of the frequency of the first spike, the fundamental frequency of the signal.

Part 4 of this project is similar to Part 3, however, this portion involved the creation of synthetic audio signals using the Amplitude Phase form of the Fourier Series model. The DFT of the synthetic signals appeared a lot cleaner than their real-world counterparts, with the values of the magnitude of the DFT effectively negligible except for at component frequencies where there would be a delta-function spike. This portion of the project nicely stages Part 5, enabling us to generate audio signals of specifiable fundamental frequencies, which can be used for testing the tuner.

Part 5 is the culmination of everything done in this project, from analyzing how the DFT works, to using DFT-analysis to solve a real-world problem, tuning instruments. Filtering was used as a technique to isolate unimportant portions of a signal from the signal’s strongest frequency component, allowing for the estimation of the fundamental frequency of the audio signal to 5 cents of accuracy. However, there were some drawbacks to the usability of the tuner. Particularly, the tuner improperly handled signals with a fundamental frequency of 68Hz, misidentifying the signal as outside the range of interest rather than close to the E string. In this application of the DFT, windowing is just as essential as in portion 2. Even though harmonic frequencies aren’t close together, windowing helped emphasize the spikes at these frequencies, by reducing leakage onto adjacent frequencies, which could be useful when analyzing noisy signals.