To test the various spectral estimators implemented as part of this project some test signals should be defined.

**Complex Exponentials:**

A signal containing multiple tones at specific frequencies can be defined by Equation 1. The amplitude (A), frequency (*f*) and phase (θ) are constants defined for each tone desired in the combined signal.

$$x[n] = \sum_{i=1}^{N\ Signals} A_i exp(j2\pi f_i n + \theta_i) \quad (1)$$

As an example *x[n]* is computed by summing three exponentials with frequencies, 5, 10, and -25Hz, with no phase offset, and equal amplitude. For this realization the sample rate is 100Hz and the signal duration is 20 seconds. The computed time and frequency domain representations of *x[n]* is shown in Figure 1.

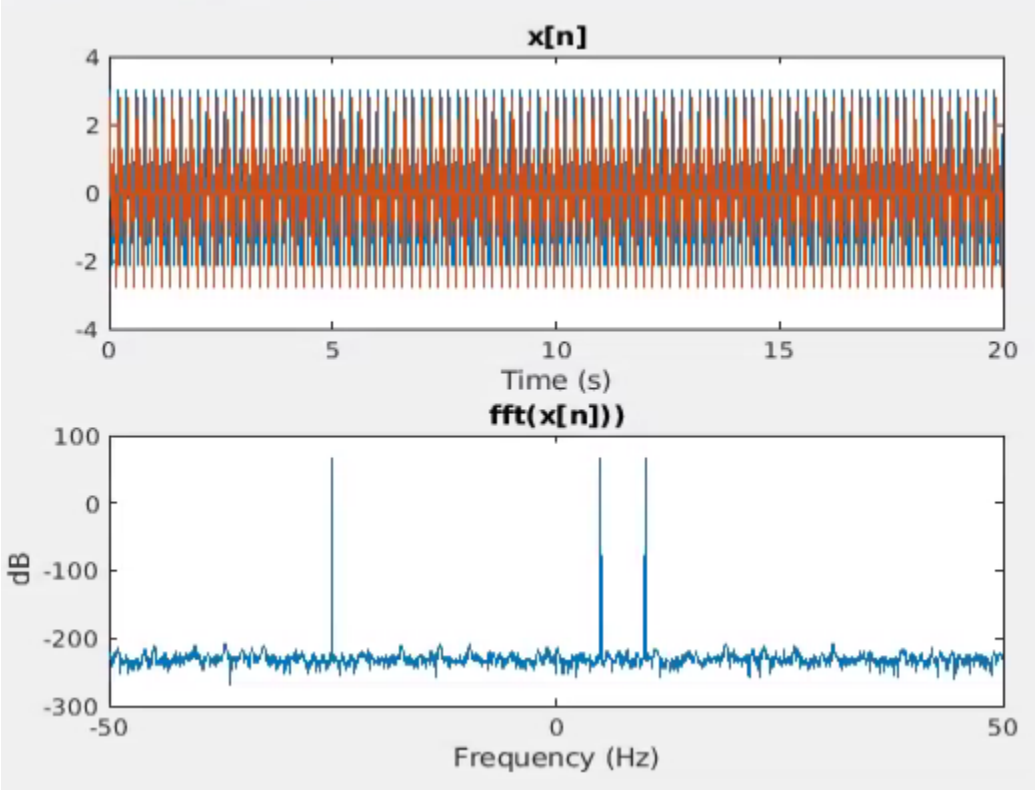


**Figure 1.** Example output of 3 noiseless complex exponentials

**Complex White Noise:**

For practical applications of spectral estimation signals will never be pure sinusoids like those shown in Figure 1. All mediums, and equipment will impart some noise onto the signal. White noise, that is, noise whose power is equally distributed across all frequencies can be added to simulate these non-ideal signals. To realize a white noise signal the *randn* function in Matlab can be utilized. The *randn* function generates a sequence of values drawn from a normal distribution. Each sample is independent from all other samples and drawn from the same distribution therefore it can be said that the noise is independent and identically distributed (IID). The normal distribution has zero mean, and a variance of

one. To simulate environments with various signal to noise ratios (SNR) it will be useful to be able to scale the variance noise signal. When dealing with complex noise this can be done using Equation 2, where $\sigma_n^2$ is the noise variance.

$$noise[n] = \sqrt{\frac{\sigma_n^2}{2}} randn(N,1) + j\sqrt{\frac{\sigma_n^2}{2}} randn(N,1) \quad (2)$$

A realization of our noise signal with a variance of 10 is shown in the time and frequency domains in Figure 2. Despite some negative spikes induced by not having an infinite length signal to transform it can be seen that the frequency domain spectrum is relatively flat.

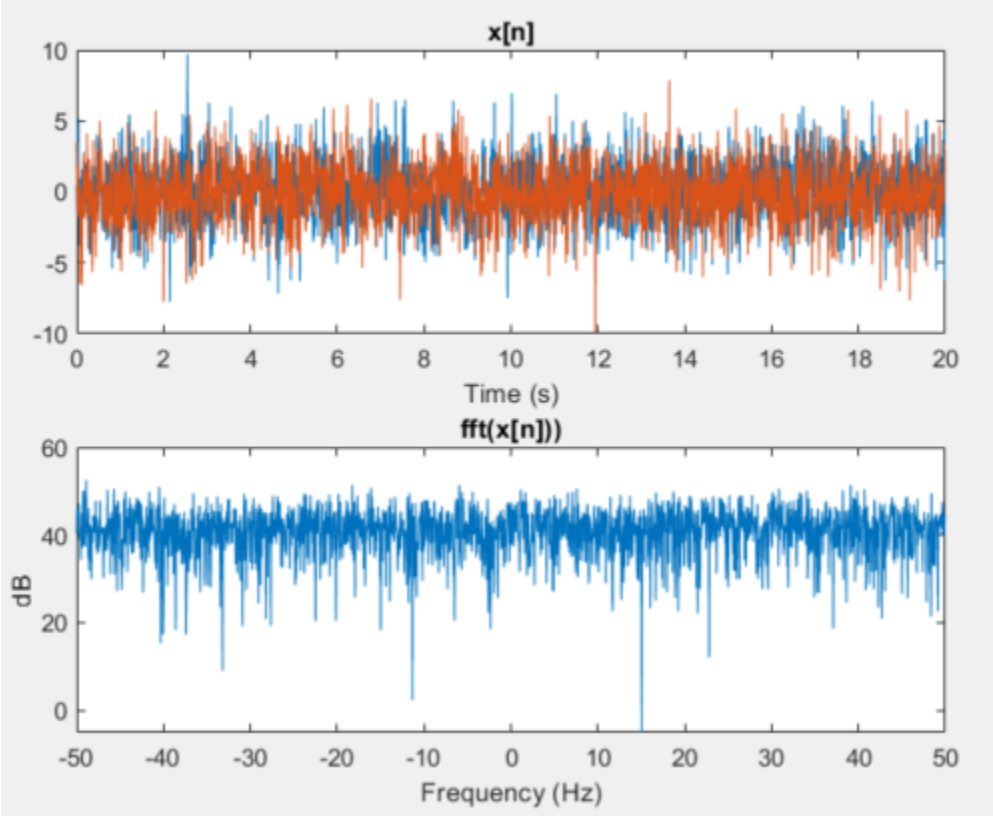


**Figure 2.** Example output of complex white noise

Figure 3 shows the same signal generated in Figure 1 with the noise from Figure 2 added. In the time domain the easily visible periodicity of the signal is obscured by the newly added noise. In the frequency domain we again see the three tones at the expected frequencies but the difference between the peaks and the noise floor is significantly reduced. If the noise variance were to increase this difference would shrink further until at some point the peaks become indistinguishable.

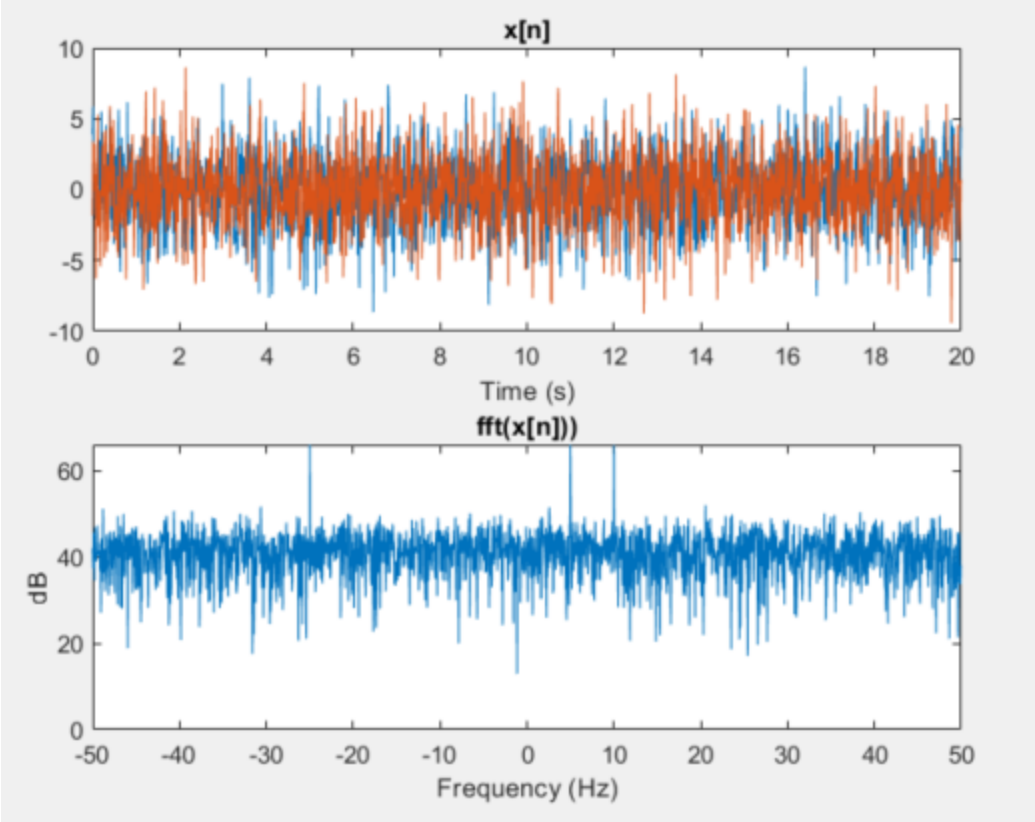


**Figure 3.** Example output of 3 complex exponentials with noise

Now that our simulated example inputs are defined we can discuss estimating the power spectral density (PSD) of a signal.

**Peroidigram:**

The periodogram is an estimate of the PSD. It is implemented by squaring the absolute value of the FFT of the entire signal. The equation for computing the periodogram is defined in equation 3, where ω is the discrete frequency variable defined as $\frac{k}{NT}$ where *N* is the total number of samples and T is one over the sample rate. This will be referred to as the frequency bin for the remainder of this report.

$$PSD_{periodogram}[\,\omega\,] = \sum_{n=0}^{N-1} |x[n]\, exp(-\, j2\pi n\omega)|^2 \tag{3}$$

Because the sequence *x[n]* will always in practice be a finite number of samples the PSD estimate from the periodogram will have a high variance. The spectral resolution of the periodogram increases with the number of samples (N) used in the FFT computation. The number of bins can be increased by zero padding the signal for the FFT computation, but this will not increase the real resolution of the estimate.

Applying the periodogram two our previously used example case of three noisy tones, we generate the result shown in Figure 4. As expected from the definition of the periodogram the output looks very

similar to the result from Figure 3. Since Figure 3 shows only the magnitude of the frequency domain the periodogram is simply a squared version of $|fft(x[n])|$. Any phase information contained in the FFT is lost by computing the PSD estimate. Also for consistency across the PSD estimators the output is normalized by dividing by $N^2$.

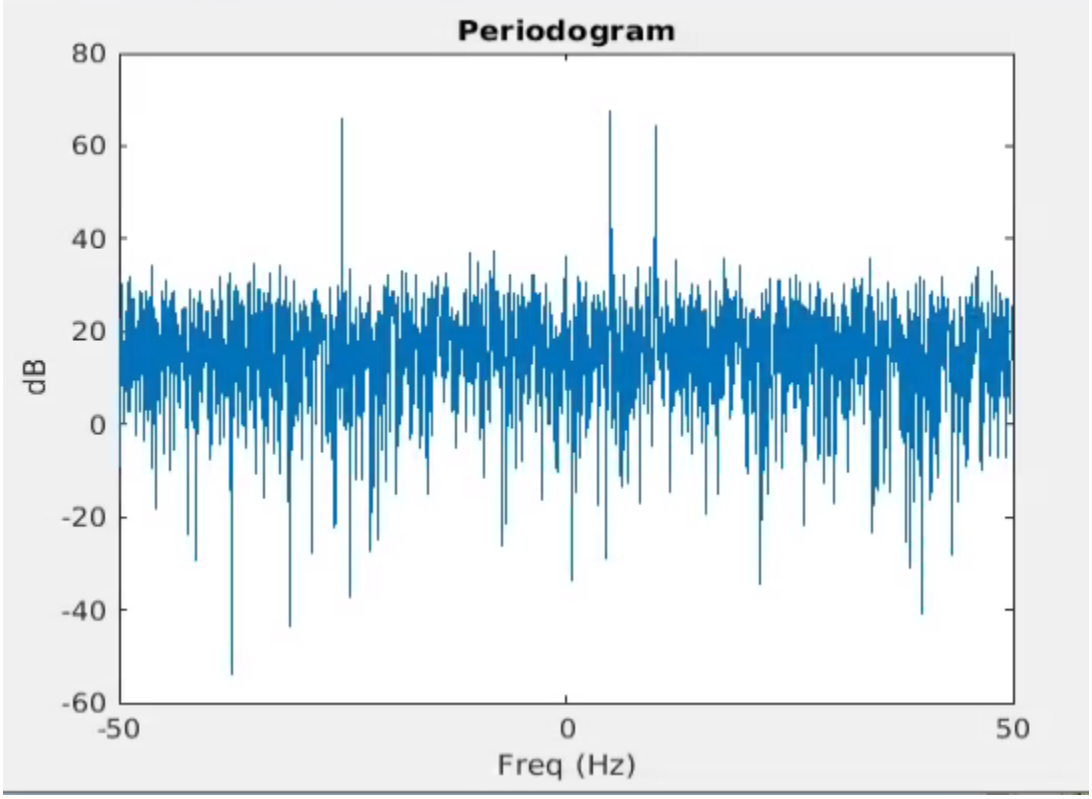


**Figure 4.** Periodogram of three tone case

We know from the definition of the test signal, with an infinite number of samples our true PSD should be three spikes, with a flat noise floor. Clearly in Figure 4 the noise floor has a high variance, the trade off for this high variance is the periodogram has better resolution with equal data when compared to the other PSD estimates to illustrate this consider *x[n]* to be two tones at 10 and 10.1Hz. From the result shown in Figure 5 it is clear that the two tones are visually separable. This resolution will later be contrasted with other PSD estimators.

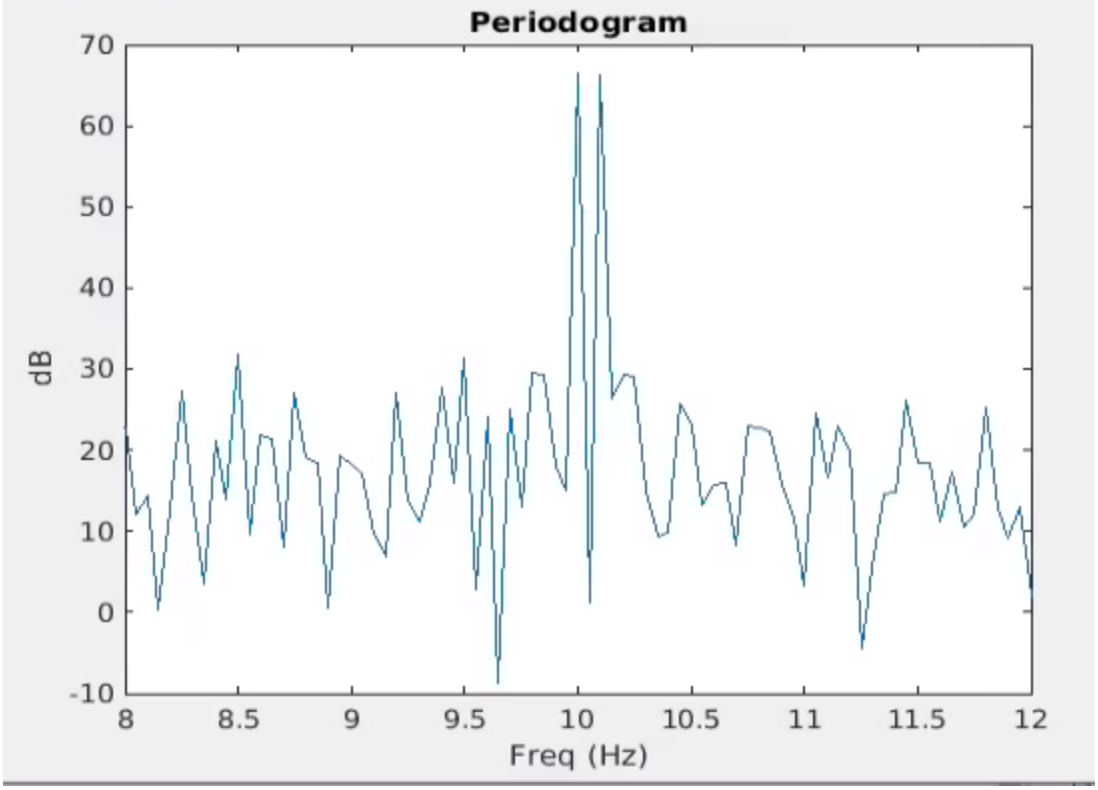


**Figure 5.** Periodogram of two tones space by 0.1Hz

**Welch-Bartlett:**

The Welch-Bartlett PSD estimator attempts to address the issue of the high variance in the periodogram. If we assume the noise present in our signal is zero mean, then we would expect that averaging the energy at a particular bin with sufficient measurements would approach some deterministic value. Since we know the ideal PSD of white noise is a constant with the value of the noise variance in the frequency domain. We would then expect the energy of our signal PSD in bins where no signal is present to approach this noise constant with sufficient averaging.

The equation for computing the Welch-Bartlett PSD is defined in Equation 4. The computation works by segmenting *x[n]* into *M* segments of size *N,* if desired, applying a window (*w[n])* to *x[n],* computing the periodogram of *w[n]x[n]* then averaging the *M* periodograms together.

$$PSD_{welch-bartlett}[\,\omega\,] = \frac{1}{M}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}\left|w_m[n]x_m[n]\,exp(-\,j2\pi n\omega)\right|^2 \quad (4)$$

Applying the Welch-Bartlett PSD estimator to the three tone example case we've been using produces the results shown in Figure 6, the estimate was produced with 128 samples per segment (*N*). In one application a rectangular window was used, in this case no overlap is needed between consecutive segments to *M* is 16. Since *x[n]* has 2000 total samples. In the case where the hanning window is applied a 50% overlap was used, therefore *M* is 31.

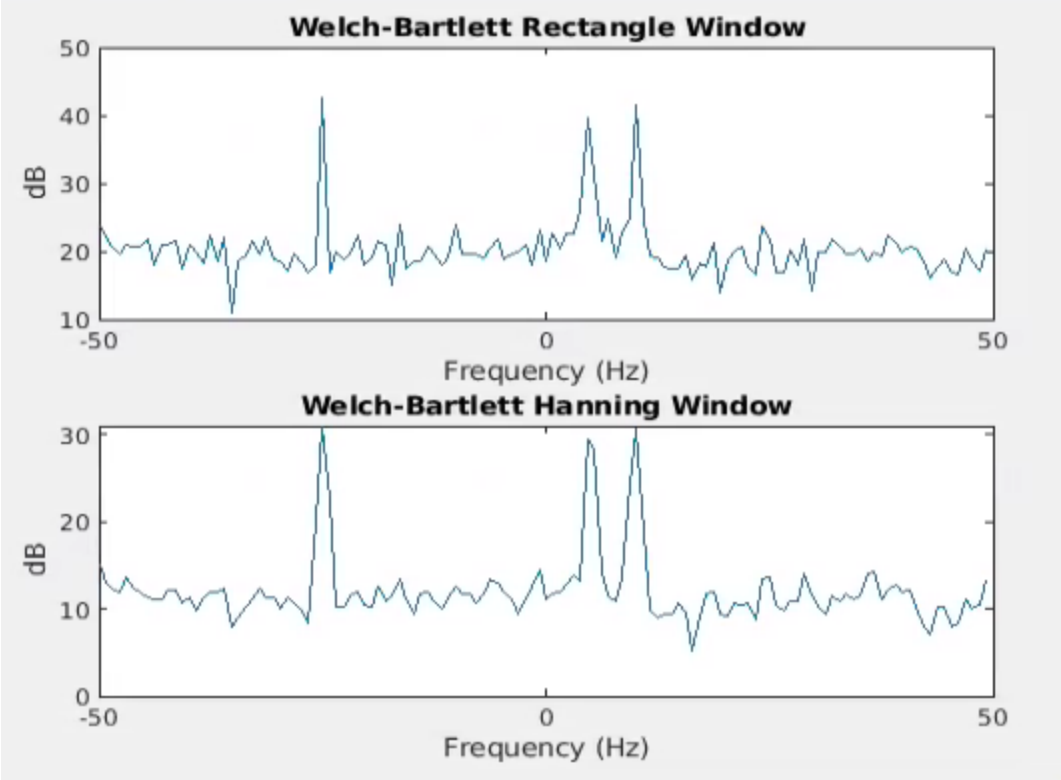


**Figure 6.** Welch-Bartlett of three tone case

Comparing Figure 6 to Figure 4 it is clear that the Welch-Bartlett estimate has a much lower variance. But since each periodogram being averaged is only 128 samples the frequency resolution is substantially worse. This poor resolution is only exacerbated once the hanning window is applied since applying a window mathematically means convolving with a sinc like function in the frequency domain, and the hanning window has a larger main lobe than the rectangle window. In Figure 6 it can be seen that the effect of the hanning window produces wider peaks, but reduces the variance of the spikes in the noise floor.

Applying the Welch-Bartlett estimate to the 0.1Hz spaced two tone example case used for Figure 5. We see that in Figure 7 the two tones become indistinguishable, where as when the periodogram was used two distinct tone could clearly be seen.

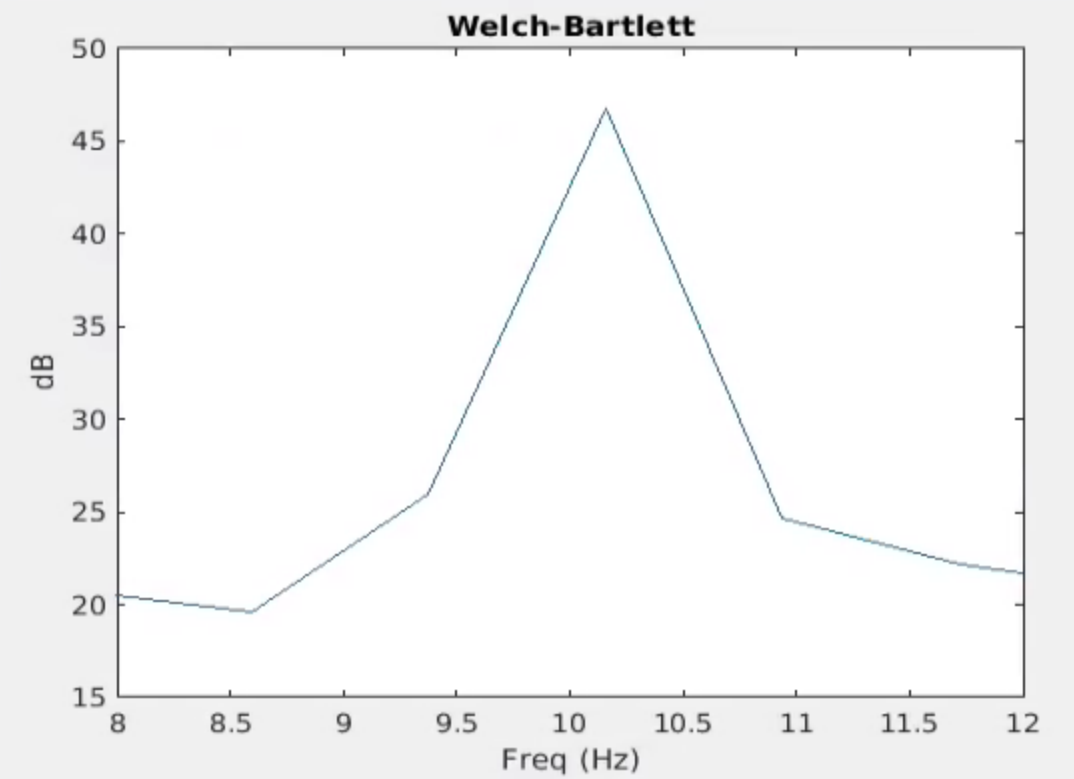


**Figure 7.** Welch-Bartlett of two tones space by 0.1Hz

Consider the case where *x[n]* consists of solely white noise. If we sample at 100Hz and take 2000 seconds worth of data and compare the PSDs generated by the periodogram and Welch-Bartlett estimators we get the results shown in Figure 8. Clearly even with a large amount of data the periodogram still has an extremely large variance whereas the Welch-Bartlett estimate is converging towards a constant value. In this case the white noise was generated with a variance of 1, so as expected the Welch-Bartlett psd is converting towards 0dB.

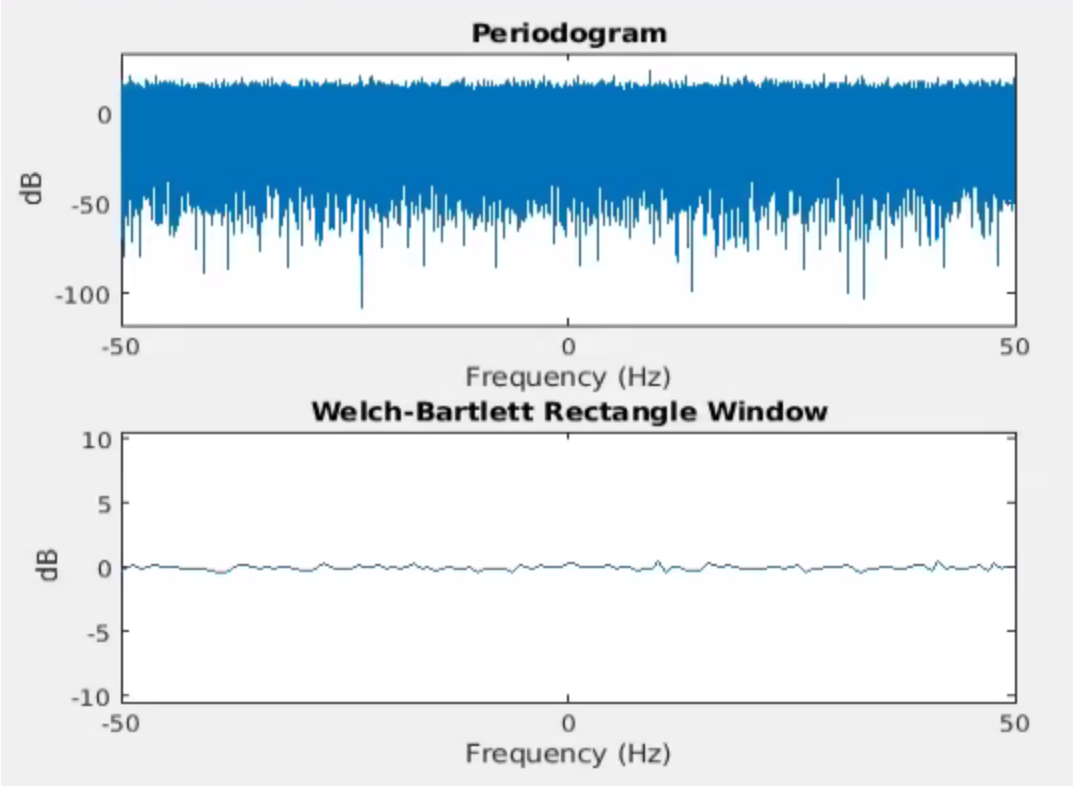


**Figure 8.** Periodogram and Welch-Bartlett of white noise

**Blackman-Tukey:**

The Blackman-Tukey approach for estimating the PSD is done by computing the sample autocorrelation sequence *r[l]* defined by Equation 5.

$$r\ [\ l\ ]\ = x[n]\ *\ x[-\ n]^{*} \tag{5}$$

Then, since $l$ is the lag between samples, for small values of $l$ there are lots of points used in the computation of *r[l]*. For large values of $l$ which are created by, when convoling, summing the tail ends of $x[n]$ and $x[n]^{*}$ the value of *r[l]* is only computed using a small number of samples. In this case *r[l]* will likely not be a good estimate of the true autocorrelation vector. To account for this a window with tapering ends, such as a hanning window is applied to *r[l]*. Once windowed the PSD can be computed by taking an FFT of *r[l]*, as shown in Equation 6. Unlike the previous two methods there is no need to square the result since *r[l]* is already computed by effectively summing squared values of *x[n]*.

$$PSD_{Blackman-Tukey}[\ \omega\ ]\ = \sum_{l\ =\ -N-1}^{N-1} w[l]x[l]\ exp(-\ j2\pi n\omega) \tag{6}$$

Applying the Blackman-Tukey window to the three tone test case produces the result shown in Figure 9.

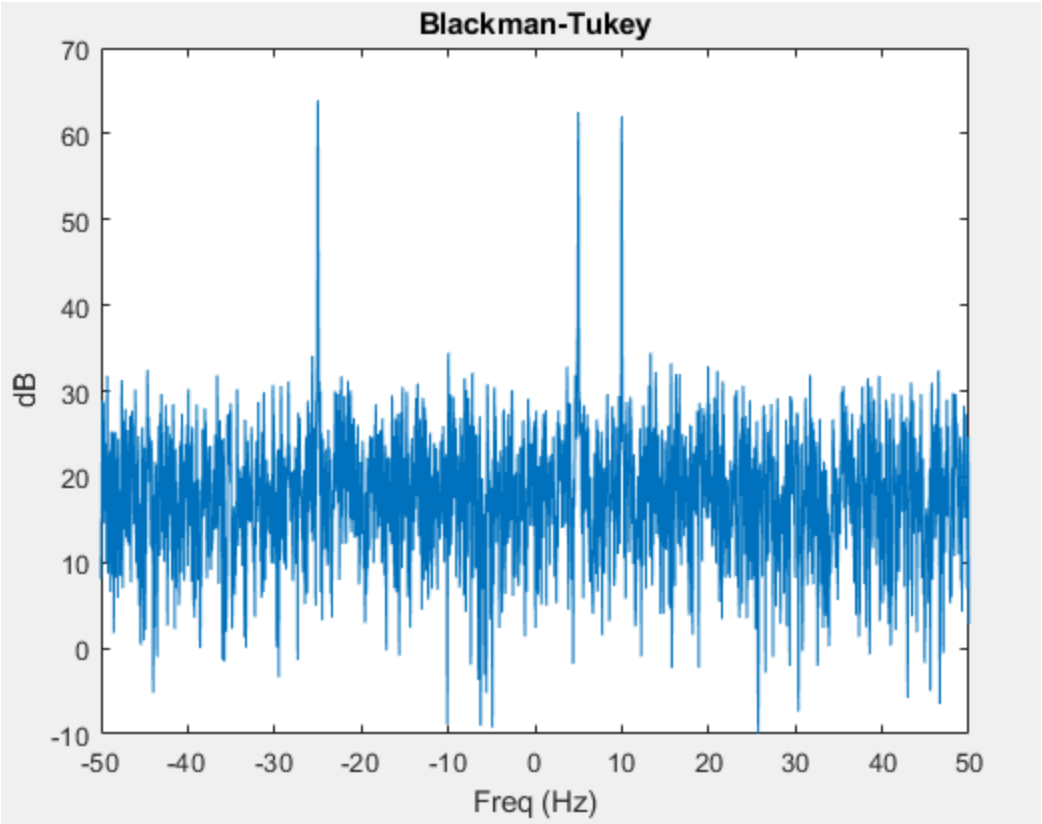


**Figure 9.** Blackman-Tukey of three tone case

Like the periodogram since no averaging is taking place the signal has a high variance. However since a window is being applied in the case some additional smoothing compared to the periodogram is taking place.

**Capon (mvdr):**

The capon method for estimating the PSD works by computing a vector of weights *w*, for a specific frequency that when applied using Equation 7 the output power is minimized. Adding a bounding condition enforces that *w* is not simply 0. By minimizing the output power we are effectively reducing the effect of out of band noise at that specific frequency.

$$PSD_{capon}[\,\omega\,] = w_{\omega}^{H} R_x w_{\omega} \tag{7}$$

Where $R_x$ is the sample autocorrelation computed as,

$$R_x = \frac{1}{N_{segments}} \sum_{n=1}^{N_{segments}} x[n]x[n]^H \tag{8}$$

By choosing a bounding condition such that $w_{\omega}^{H} v(w) = C$ where *v(ω)* is the frequency vector at frequency ω and *C* is a normalizing constant. Then we can show that the PSD can be computed at a certain frequency using Equation 9.

$$PSD_{capon}[\,\omega\,] = \frac{M}{v^{H}(w) R_{x}^{-1} v\ (w)} \tag{9}$$

When implementing the capon estimation $R_x$ is required to be a full rank matrix. For that to happen the number of segments must be greater than the number of samples per segment. Applying the capon estimate to the three tone case we've been previously using with 2000 samples means we would need at minimum to satisfy the full rank autocorrelation matrix 44 segments of 44 samples. If done using the bare minimum number of samples to satisfy full rank the result is shown in the upper plot of Figure 10. Clearly it is not obvious where, or how many tones are present, or if they even are any. To increase the fidelity of the peaks including more segments in the computation of $R_x$ will help. If we choose the segment size to be 25, 80 segments can now be used in the autocorrelation computation. The results of which are shown in the lower plot of Figure 10. We can now clearly see two peaks, but we would expect three to be present. Furthermore the real frequency resolution of the estimate is decreased by decreasing the number of samples per segment.

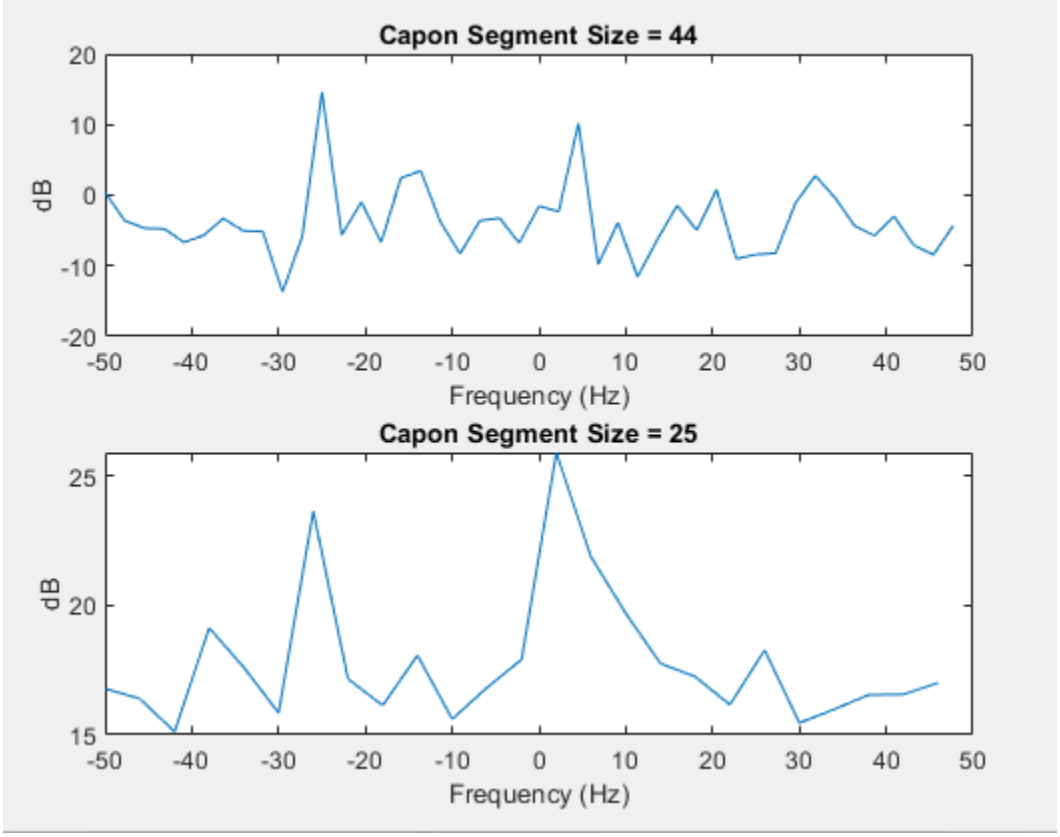


**Figure 10.** Capon of three tone case

To resolve the issue illustrated in Figure 10 the Capon estimator needs substantially more data then the previously discussed estimators. If we increase the signal duration to 100E3 samples. And use a segment size of 100, we will have 1000 segments used in the computation of $R_x$. The results of such a case are shown in Figure 11, clearly all three tones are distinct from the noise.

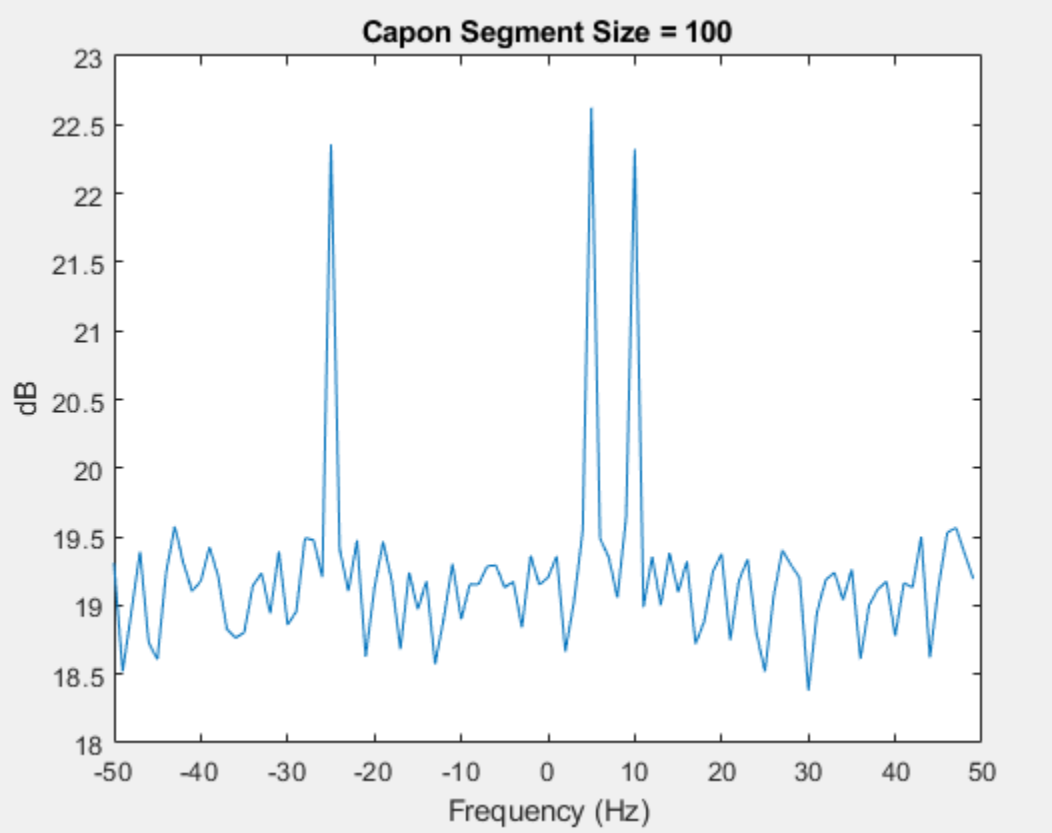


**Figure 11.** Capon of three tone case

The computation of $R_x$ is akin to averaging like in the Welch-Bartlett estimator. As a result the Capon estimator will also have a lower variance when compared to the periodogram. This can be demonstrated by using the white noise signal as an input. Figure 12 shows the PSD estimate, the estimate is converging towards the expected constant value of 0dB.

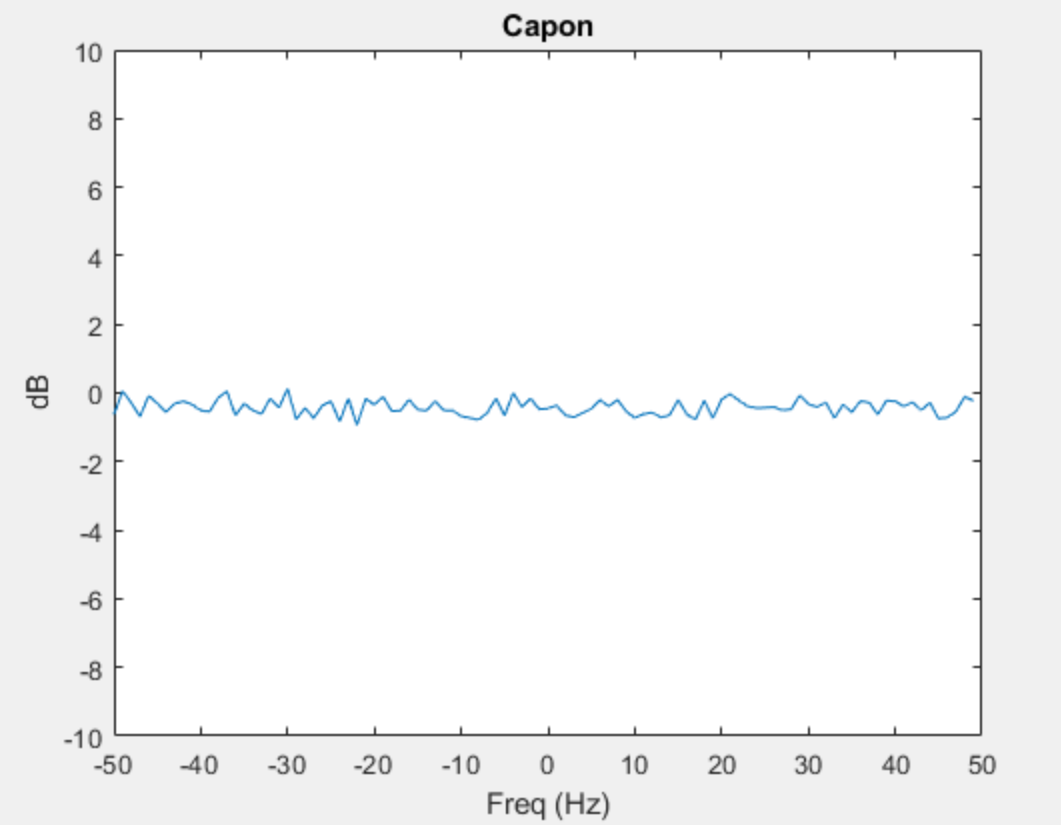


**Figure 12.** Capon of a white noise input

**MUSIC:**

Unlike the previous estimators the MUSIC approach doesn't attempt to estimate the PSD. Instead the MUSIC algorithm is used to determine if tones are present. The MUSIC algorithm works by computing the eigen decomposition of the *MxM* autocorrelation matrix. Eigen decomposition of $R_x$ computed with *P* signals will have *P* eigen values larger than the remaining *M-P* eigen values. These large eigen values correspond to the eigen vectors that comprise the signal subspace $U_{signal}$. The remaining eigen vectors comprise the noise subspace $U_{noise}$. If we define a vector *e* that consists of a complex exponential at a specific frequency then when projected into each of the directions of the eigen vectors because they are all orthogonal, all but one of the eigen vectors should have a post-projection magnitude of near 0. If we instead project our vector *e* into exclusively the eigen vectors which comprise the noise subspace if the particular frequency ω was present in the original signal then it will be orthogonal to the noise subspace and produce a projection magnitude near zeros, likewise if the ω used to compute *e* wasn't present in the signal then it should be somewhere in the noise subspace. Therefore at least one of the projections into the noise subspace at a particular ω should give a non zero magnitude.

To visualize the results since we have a value approaching zero whenever *e* with ω corresponding to a signal in the dataset is present, it is useful to present the results as 1 over the projection magnitude so that approaching zero turns into a spike. The MUSIC algorithm can be implemented using Equation 10.

$$MUSIC[\,\omega\,] = \frac{1}{e^{H}(w)U_{noise}U^{H}_{noise}e\ (w)} \tag{10}$$

The music algorithm, like the capon estimator requires computing $R_x$ to be full rank. Which means again when compared to the periodogram, Welch-Bartlett, or Blackman-Tukey methods it will require more data. Applying the MUSIC algorithm to the three tone case we've been using and choosing *M* to be 227 we produce the results shown in Figure 13. Clearly there are three peaks present in the frequency domain. And when looking at the eigen values, as expected three are substantially larger than the rest. In an ideal case the remaining *M-P* eigen values would be equal but since we only have a finite amount of data the added noise can never be perfectly white.

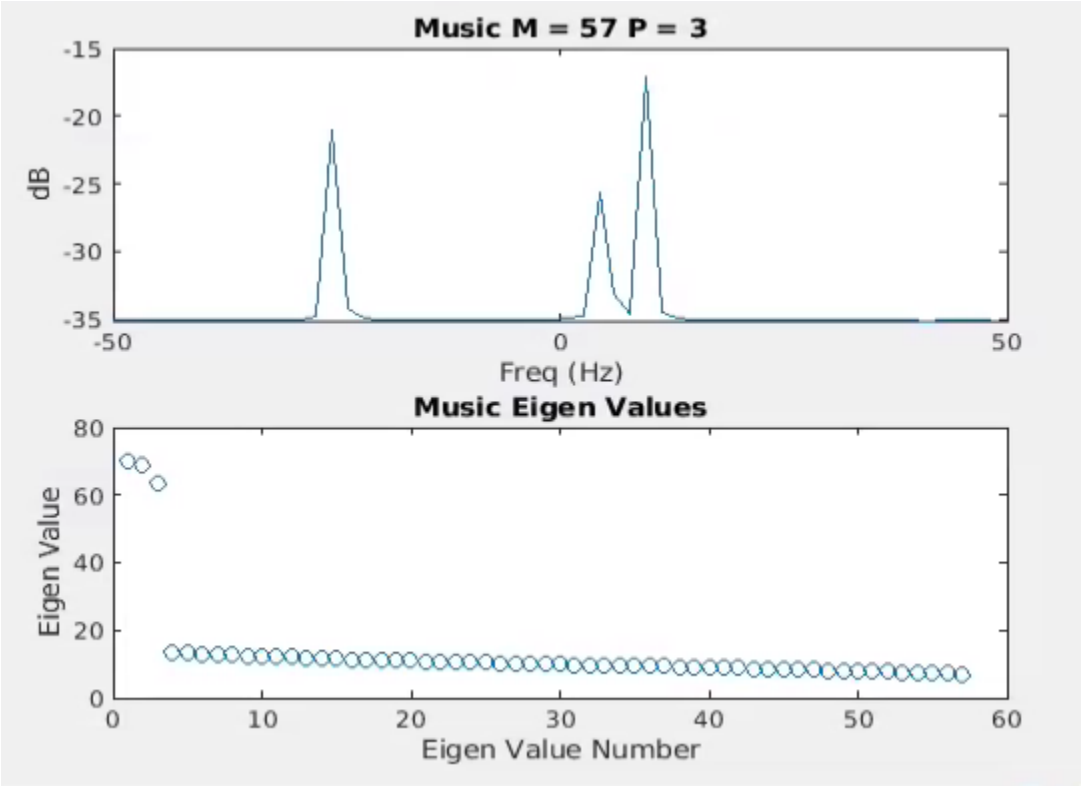


**Figure. 13** MUSIC algorithm on the three tone case

If we were to apply white noise to the MUSIC algorithm since the signal would be uniformly distributed across all frequencies it would also be uniformly distributed across all eigen vectors, meaning there wouldn't be a clear large eigen value so it wouldn't be clear which eigen vector correspond to the signal subspace and which to the noise subspace.

**Real World Data:**

To apply these estimators to a real world signal I took an audio recording of a plucking a guitar at three different strings, E, d, and b. The goal is to derive the frequency at which the string vibrates. For a quick first look at the data a periodogram was applied to all three signals, the results of which are shown in Figure 14. It can be seen that there are clear tones and we can see as the notes get higher the peaks are

spaced further apart, which is expected since each string should produce harmonics of the fundamental vibration frequency. The higher the pitch of the note, the wider the space between the harmonics. But for further analysis since we only have tones and their harmonics are present, i.e. we don't need a very high frequency resolution we are better off using a Welch-Bartlett estimate to try and reduce the noise variance. Before leaving the periodogram we can make note that even though the phone microphone samples at 48KHz some sort of filter is applied around 35KHz causing the sharp drop in power at +-17.5KHz.

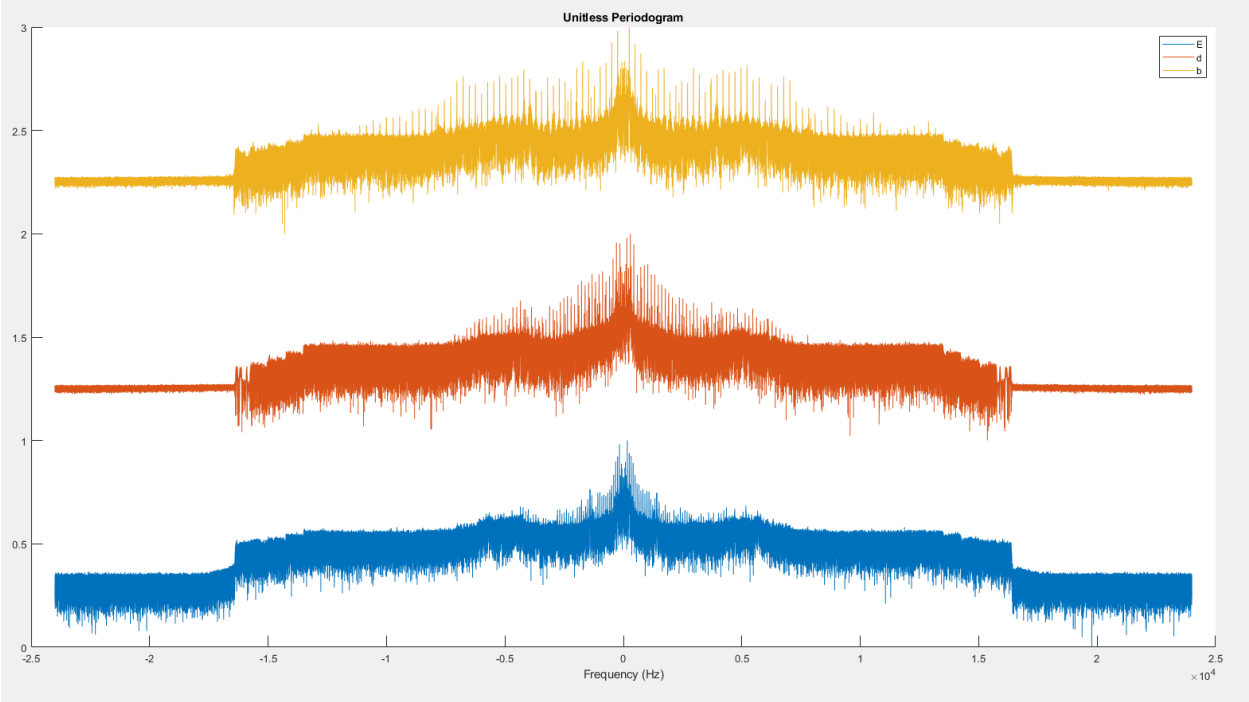


**Figure 14.** Unitless periodogram of three different guitar strings

Applying the Welch-Bartlett window produces Figure 15. Again we see the same tone spacing and as expected the noise variance is significantly reduced. From this Welch-Bartlett plot we can compute the fundamental frequencies of the 3 strings. Figure 16 shows the first 1KHz of data, ignoring the negative frequencies since a sound signal should be symmetric.

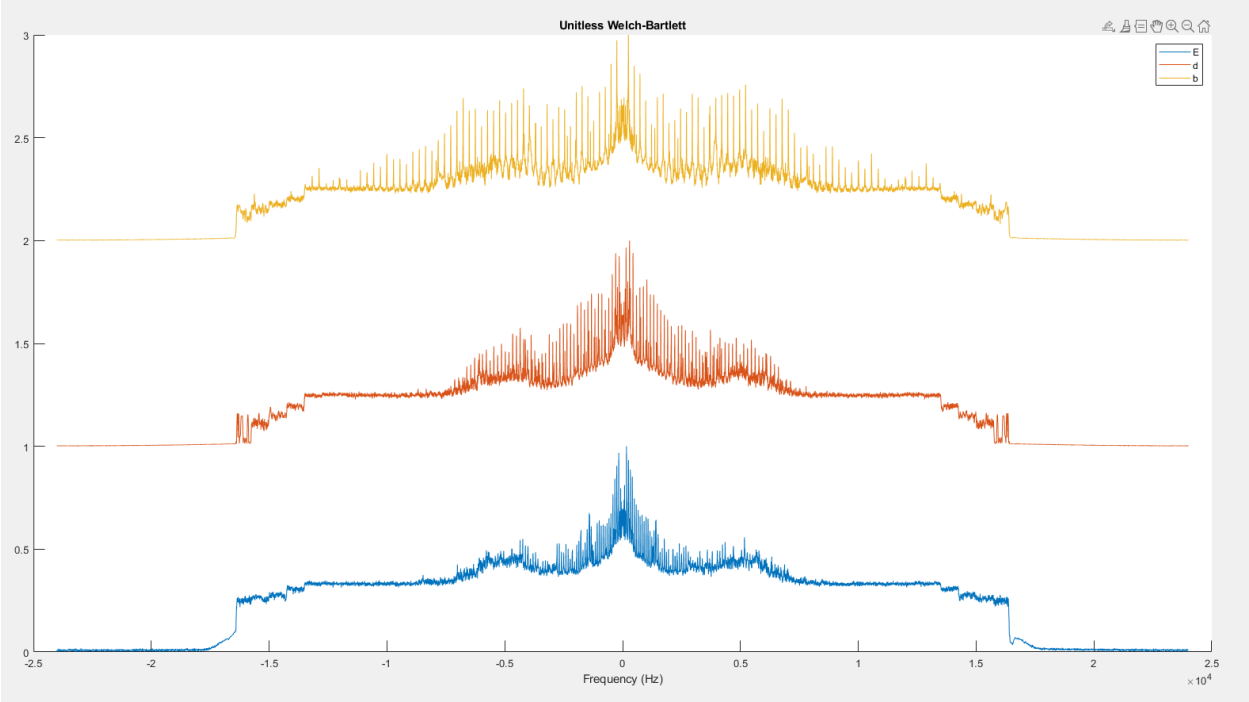


**Figure 15.** Unitless Welch-Bartlett of three different guitar strings

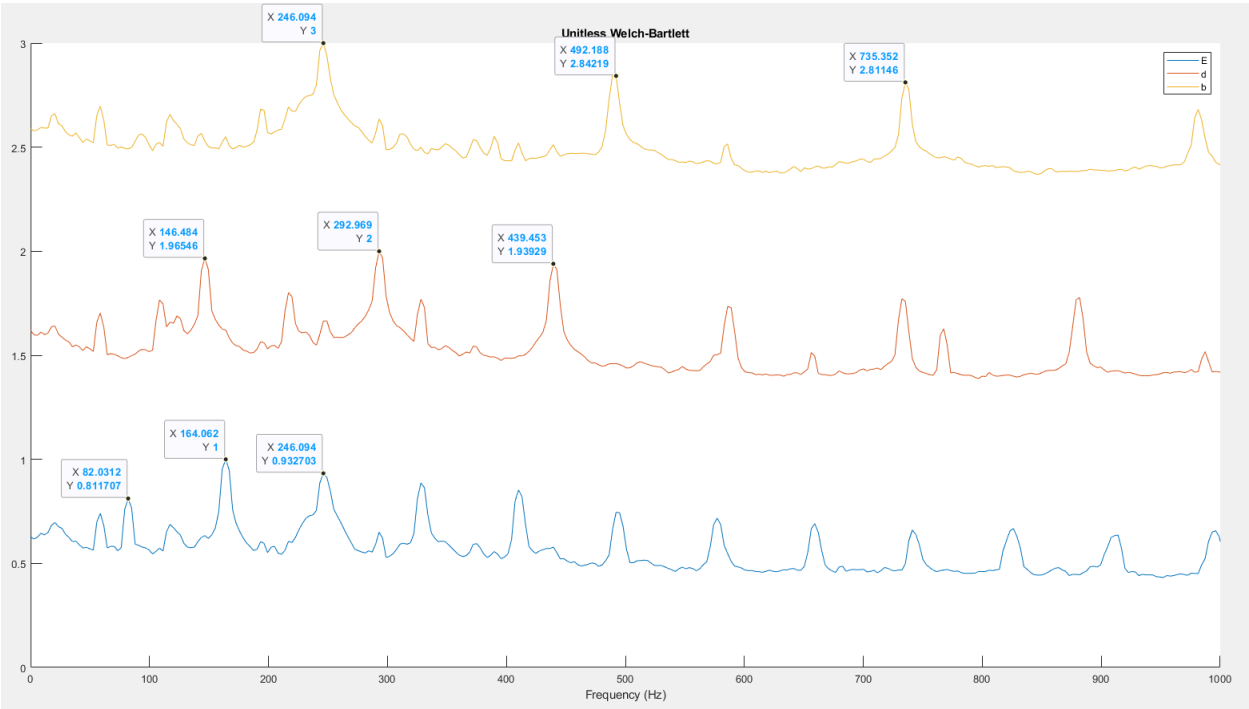


**Figure 16.** Labeled unitless Welch-Bartlett of three different guitar strings

From the markers on Figure 16 we can conclude that the E string has a fundamental frequency of 82.03Hz, the d string has a frequency of 146.48Hz, and the b string has a fundamental frequency of

246.09Hz. When compared with the true frequencies we can conclude two things. A. That I'm really good at tuning a guitar, and B. that the Welch-Bartlett estimation was sufficient for analyzing our signal.

| String | Note | Frequency (Hz) | Rounded up |
|---|---|---|---|
| 6 | E | 82.407 | 82 |
| 5 | A | 110.000 | 110 |
| 4 | D | 146.832 | 147 |
| 3 | G | 195.998 | 196 |
| 2 | B | 246.942 | 247 |
| 1 | E | 329.628 | 330 |

**Figure 17.** True guitar string frequencies
https://www.fuelrocks.com/how-to-choose-the-right-guitar-strings-for-your-sound/

Interestingly in Figure 16, the fundamental frequency of the E string 82.03Hz was lower power than the subsequent harmonics which wasn't true for the other strings. Since we know this shouldn't be the case I can only assume my phone microphone has worse gain at very lower frequencies.

The Blackman-Tukey estimate shown in Figure 18 provides an alternative to the Welch-Bartlett method which also gives clear peaks and a relatively low variance. And because Capon approach isn't averaging segments like the Welch-Bartlett it has better resolution.

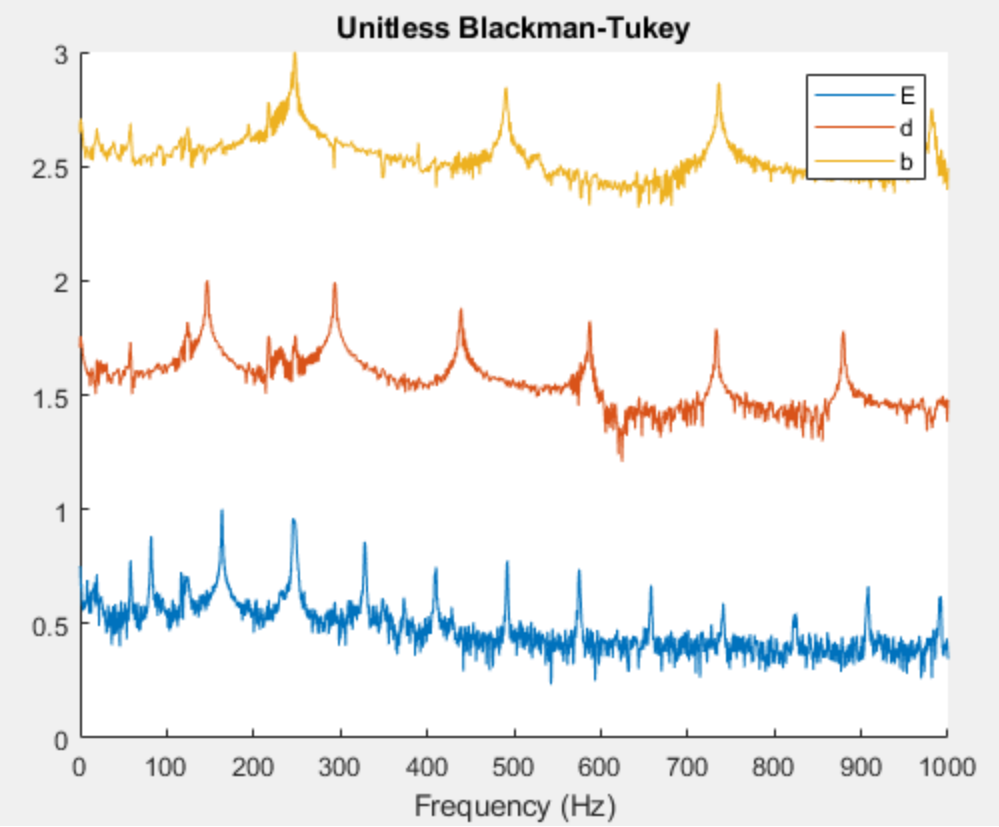


**Figure 18.** Unitless Blackman-Tukey of three different guitar strings

For the capon estimate and music algorithm as previously discussed they are significantly more data hungry than the periodogram, Welch-Bartlett, and Blackman-Tukey estimators. I recorded 20 seconds worth of data for each string, which was about 1 million samples per collection. At that data volume my laptop was struggling to processes since both algorithms are noticeably less efficient than the earlier estimators.

For the capon estimate shown in Figure 19 even though it looks roughly like the output shapes of the previous estimators because of insufficient data the resolution wasn't good enough to distinguish the distance between peaks.

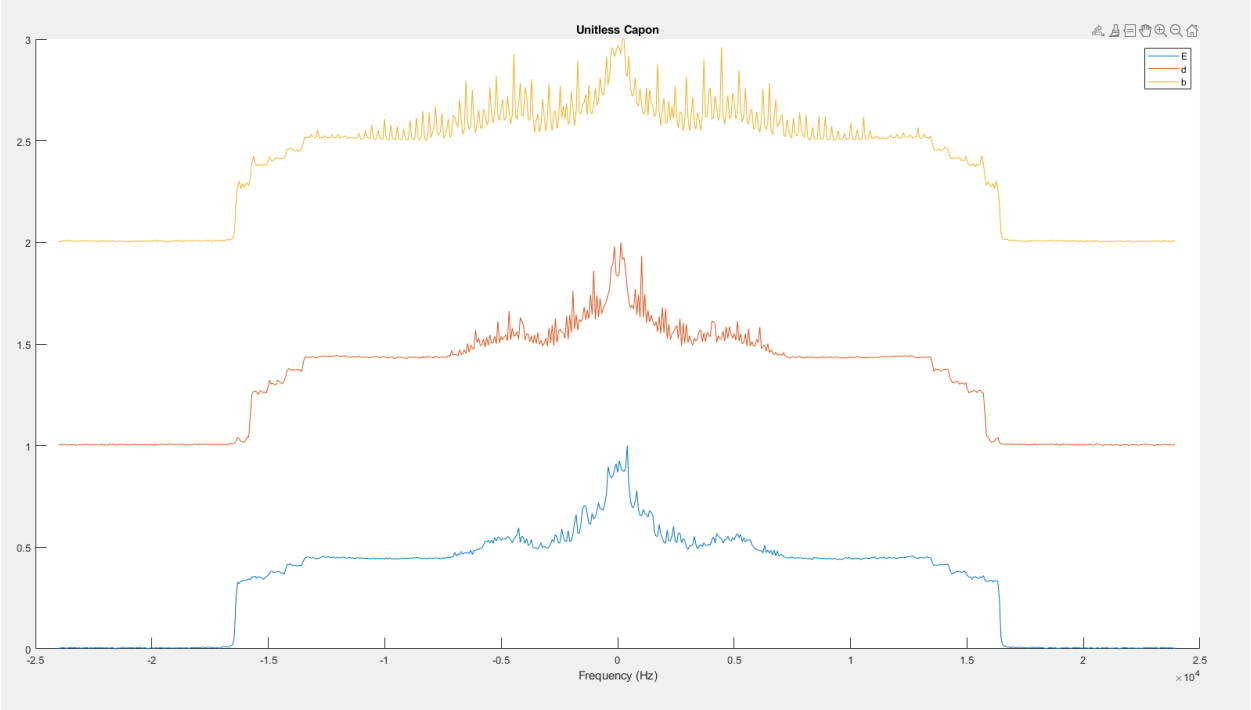


**Figure 19.** Uniteless Capon of three different guitar strings

For the application of the MUSIC algorithm on the three strings. Since as seen by the Welch-Bartlett estimate of the signals there are lots of peaks caused by the harmonics, MUSIC probably wouldn't be a preferred algorithm to throw at this dataset, at least without implementing some auto tone number estimator.

To attempt to get something to work I applied three bandpass filters to the b string signal, since it had the furthest harmonic spacing. The center frequency of the three filters was 246Hz, 1.719KHz, and 2.956KHz. The Welch-Bartlett of the signal post filter is shown in Figure 20. The hope would be that the filters would provide enough spacing between tones that even though the frequency resolution would be terrible we would hopefully be able to detect some signals. The results of the MUSIC algorithm on the filtered dataset are shown in Figure 21. Using $P$ of 10 for the 10 highest tones in the filtered dataset the algorithm was able to detect a few of the tones. Which given how data starved this approach was I'm fairly happy with.

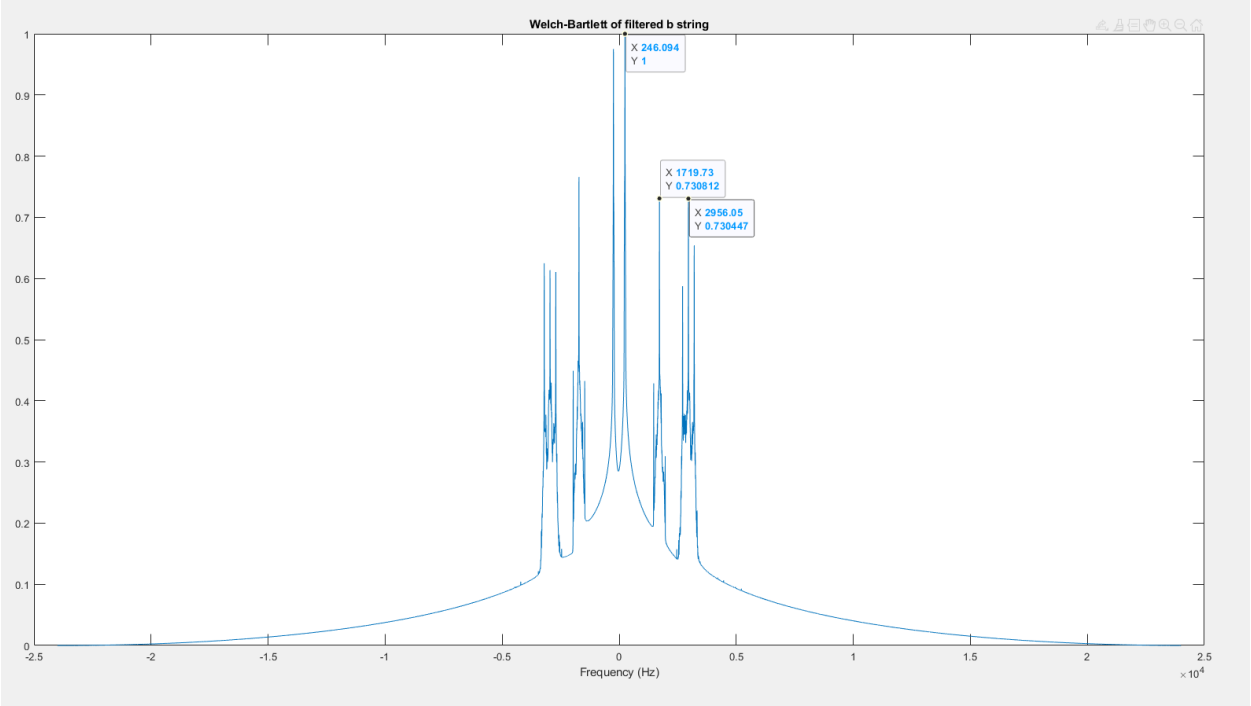


**Figure 20.** Filtered Welch-Bartlett PSD of the b string

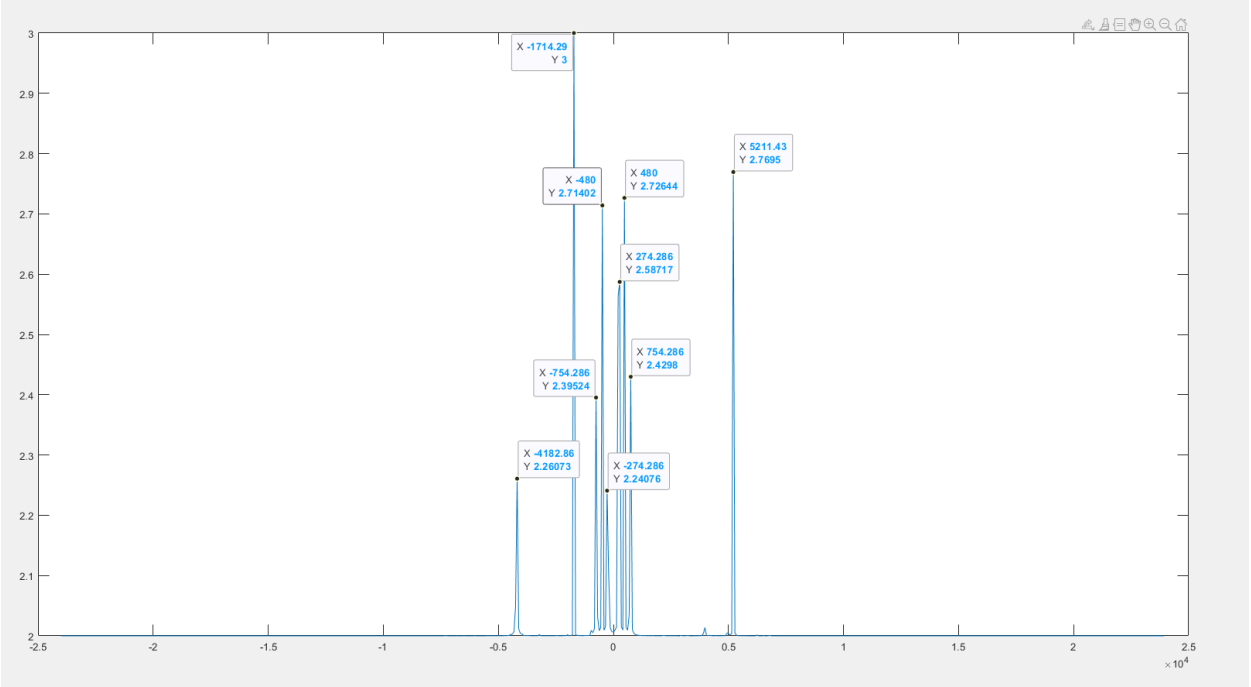


**Figure 21.** Uniteless MUSIC of three different filtered guitar strings

**Project Discussion:**

**What was the hardest part?:**

The hardest part was getting the Capon and MUSIC algorithms to work correctly. I routinely use the periodogram and Welch-Bartlett estimators as part of my normal job requirements. The Blackman-Tukey method was new but conceptually the implementation was straight forward. The Capon estimator gave me trouble when figuring out how to define *v(ω)* for a specific *ω*. Once I figured out *v* for the Capon estimator the same approach could be applied to computing *v(ω)* in the MUSIC algorithm. However knowing how to compute *e* ment I could correctly compute the output of MUSIC but I'm still a little unclear about how the subspace defined by the eigen vectors of $R_x$ map to the vectors of *e* which are defined by a frequency coordinate system. The final project paper I choose is focused on the Pisarenko algorithm which is a special case of MUSIC so I expect to have a better understanding after completion of that project.

**What was the most important concept you learned?:**

I think the most important concept I learned was about tailoring the estimator method to the problem. As discussed in the real world data section when analyzing the guitar strings both the Capon and MUSIC algorithms were unable to provide meaningful results, mainly due to a lack of data. Although I think more generically it was because the spacing of the tones with respect to the sample rate of the collect didn't mesh well with needing to compute a full rank matrix. Attempting to analyze the signal again using MUSIC, knowing what I know now I would've applied a low pass filter before the microphone and sampled at a much lower rate. The same number of samples at a lower sample rate would allow me to use smaller segments of data, and therefore use more segments in the autocorrelation matrix calculation. Even with smaller segments of data because the sample rate would be lower the frequency resolution would change at some ratio of sample rate to segment size but designing it in such a way that the resolution is able to differentiate between the observed minimum spacing of ~82Hz I think could've been achieved.

Through discussion in class I can see the usefulness of an algorithm like MUSIC. This project gave me some background knowledge to better understand how to setup a test where I expect to use MUSIC somewhere in the processing chain.

**What improvements could you make to your toolbox?:**

Of the bat the implementation of my algorithms aren't taking advantage of the efficiency of the FFT. To help understand the theory better I implemented the frequency vectors *v* (for Capon) and *e* (for music) as complex exponential vectors. We know from the textbook and class discussions that in both cases their application to $R_x^{-1}$ and $U_{noise}$ respectively can be computed as an FFT reducing the computational complexity of the implementations.

Since the MUSIC algorithm is dependent on having some foreknowledge of the number of tones present before application, like I did in Figure 20, adding one of the number of tone estimators would be useful. Since the MUSIC algorithm is already computing the eigen values before the number of signals is relevant there is an opportunity to add some processing to eigen values shown in the lower plot of Figure 13. In our simulated test cases the eigen values pertaining to an input signal could easily be processed with a rough linear threshold, but some of the more complex detectors discussed in could come in handy in worse SNR environments.