Raymond Fleming

Digital Signals and Systems

Final Project

I)Project Overview

The goal of this project is to demonstrate the ability to use a relatively simple discrete Fourier transform to differentiate between two words. This can easily be used for simple commands such as starting an appliance like a coffee maker or a light. While its not as capable as full speech recognition, the implementation is much simpler and more resource efficient. This allows the system to be used in equipment and situations where a full speech recognition solution would not be feasible as well as for assist technologies where full speech recognition may not be effective or necessary such as a medical environment. Additionally, the same process could be used to differentiate between multiple words (likely with a loss to accuracy) as long as the words are different enough in the frequency spectrum.

II) Approach

In considering how to approach this problem, the words used were first considered. Between the words 'yes' and 'no' the primary difference will likely be the 's' phoneme in 'yes'. This seems likely to have a relatively high frequency component, which will be entirely lacking in 'no'. Additionally this phoneme will be noisy but should be at a high enough signal to distinguish from recording noise. This will allow detection of 'yes', and any other entry will be classed as 'no'. To implement this, first the signal is recorded directly through Matlab to allow the user to enter the word directly instead of using recording software to do so. In the first implementation, Audacity was used for the recording to make it faster to develop the rules. This proved to be a poor choice and made the rule creation more difficult. Once the audio is recorded it is cleaned up by removing any present DC offset, then the data is normalized in the time domain. The normalization makes comparisons easier in the case that there is some filter on the microphone, or two different microphones are used. The discrete Fourier transform is then calculated using Matlab's fast Fourier transform function, then renormalized in the frequency domain to make comparisons more accurate for the reasons stated above.

Following the cleanup of the data, a loop is used to cycle through all samples in the data stream. Two separate conditional statements are used to extract the important sections of data. The first of these conditionals removes all data lower than the Freq_Threshold (this is set at 300Hz after tuning of the algorithm). The lower frequencies will be present in both 'yes' and 'no' signals and so only serve as noise for the comparison itself. The frequency threshold is a variable which can be changed as needed, since each user's voice will likely be in a slightly different frequency range. The second conditional is intended to remove noise in the data, with any value below the Power_Threshold (set to .01 after

tuning) to be zeroed. This removes background and most microphone noise from the data to be compared.

Once the data is fully prepared for comparison it is plotted, this plot can then be used to tune the algorithm for the best accuracy and for recording purposes but is otherwise unnecessary for usage of the program.

The comparison itself is performed by using the findpeaks() function in Matlab to locate the maxima in the input stream. The number of peaks is then determined using the numel() function. The comparison itself is a simple comparison between the number of peaks and the Peaks_Threshold variable, which is another tunable variable (set to 225 peaks after tuning). If the number of peaks is greater than the Peaks_Threshold value the recorded input is 'yes', otherwise it is 'no'.

III)Results

After tuning the program was tested a total of 20 times, all 'yes' inputs were correctly identified (10 out of 10). 80% of the 'no' inputs were correctly identified with 8 out of 10. It appears that any additional noise aside from the word 'no' will skew the results, and making certain to be silent aside from stating the word increases accuracy. The total accuracy for the 20 trails is 90%.

Three additional words were used to verify that the 's' phoneme is the portion being used in this case to decide between 'yes' and 'no'. The three additional words tested were 'yesno', 'test', and 'home'. 'yesno' and 'test' both contain the 's' phoneme and both resulted in 'yes' outputs. 'home' does not contain this phoneme, and returned 'no'. This appears to support my hypothesis on the basis of distinguishing 'yes' and 'no'. In addition, the power spectrum diagram of the cleaned up data for 'yesno' appears to contain the elements of both 'yes' and 'no' as is expected. The power spectrum diagrams for each test can be viewed in the Appendix section as can the test inputs and outputs.

IV)Conclusion

The results indicate that using a discrete Fourier transform to distinguish between two words is an effective method. While it has not been tested, it is also likely to be much more resource efficient than the methods used for full speech recognition such as Siri, Alexa, and Google Home. While these technologies have a definite place in the modern world, in some cases a simple yes or no (or On/Off) will suffice and can be developed for a much lower cost, and much less expensive hardware.

Appendix

Figure 1

Trial	Test Input	Test Output
1	Yes	Yes
2	Yes	Yes
3	Yes	Yes
4	Yes	Yes
5	Yes	Yes
6	No	No
7	No	No
8	No	Yes
9	No	No
10	No	No
11	Yes	Yes
12	Yes	Yes
13	Yes	Yes
14	Yes	Yes
15	Yes	Yes
16	No	No
17	No	Yes
18	No	No
19	No	No
20	No	No
21	Yesno	Yes
22	Test	Yes
23	Home	Home

Figure 1. Inputs and results from testing.

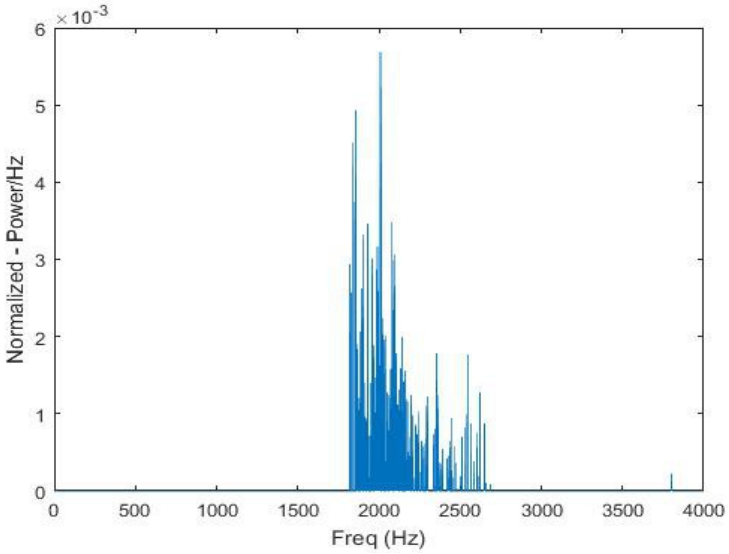

Figure 2. Power Spectrum Diagram for Test Case 1 'yes'

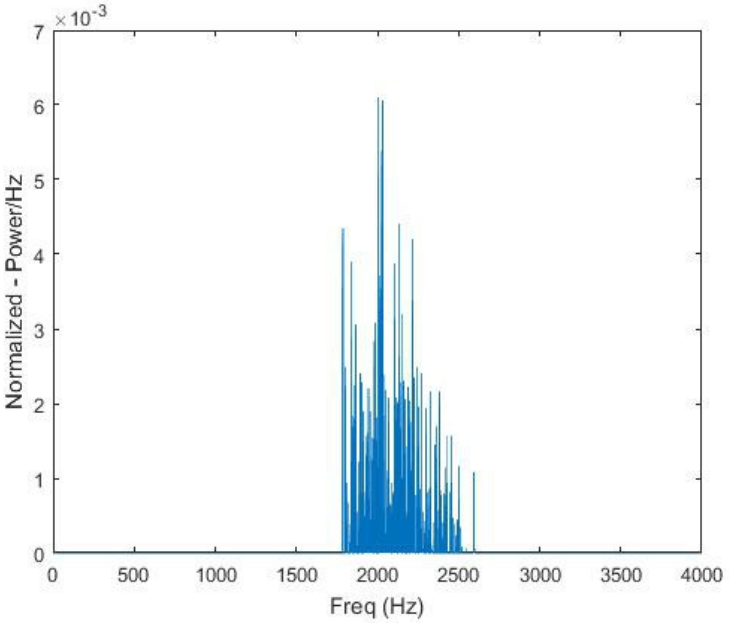

Figure 3. Power Spectrum Diagram for Test Case 2 'yes'

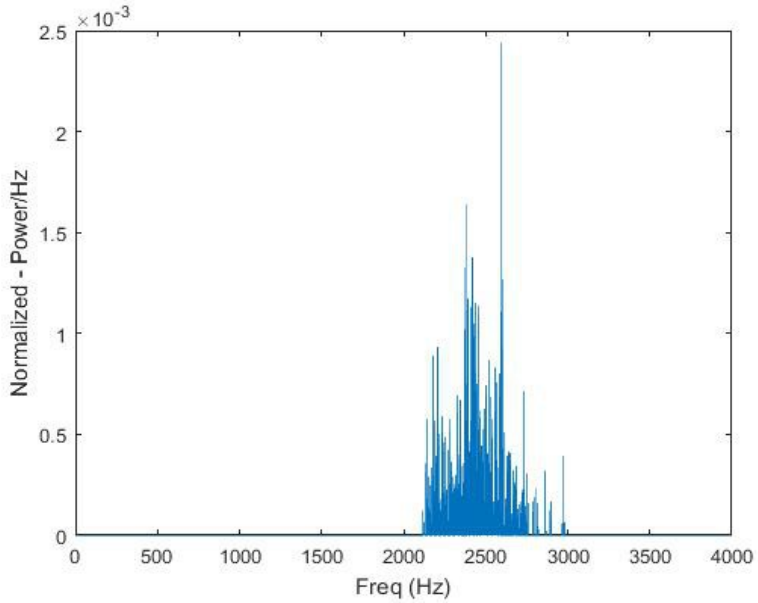

Figure 4. Power Spectrum Diagram for Test Case 3 'yes'

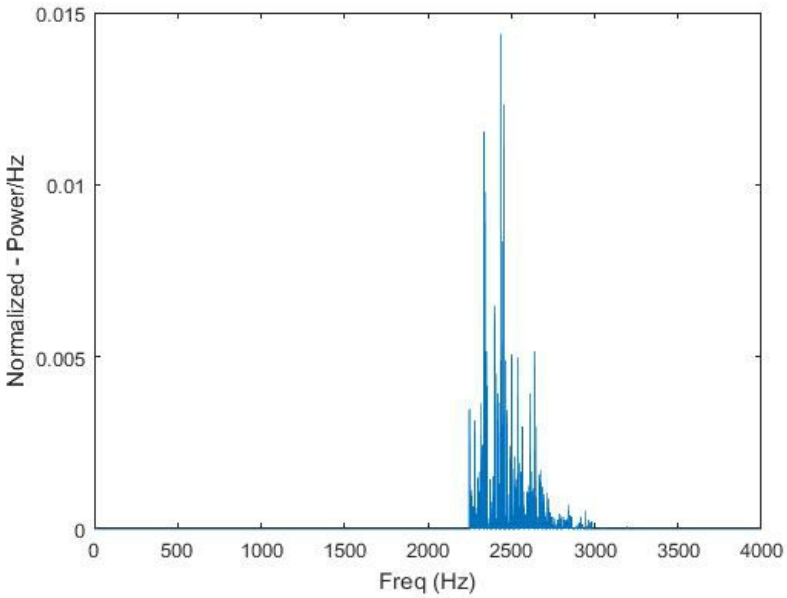

Figure 5. Power Spectrum Diagram for Test Case 4 'yes'

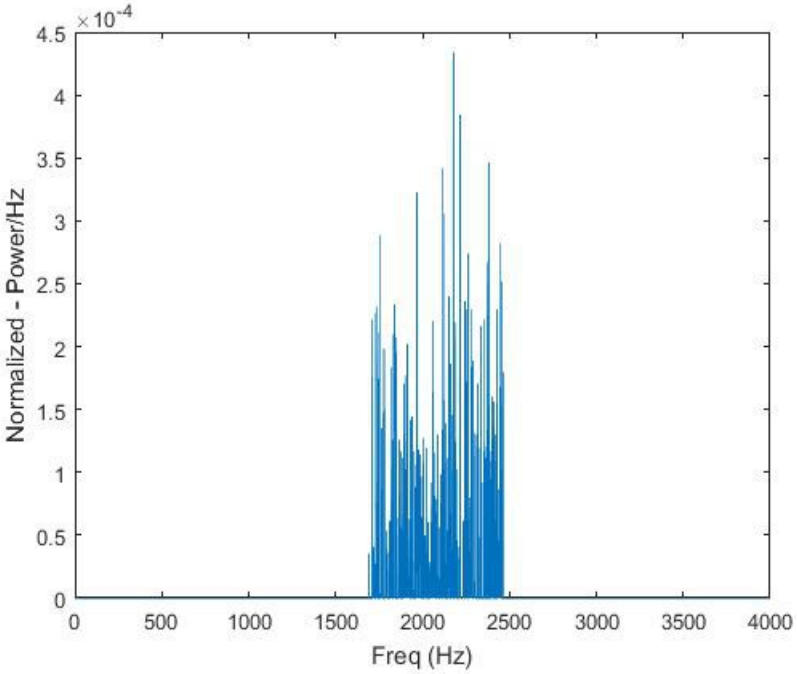

Figure 6. Power Spectrum Diagram for Test Case 5 'yes'

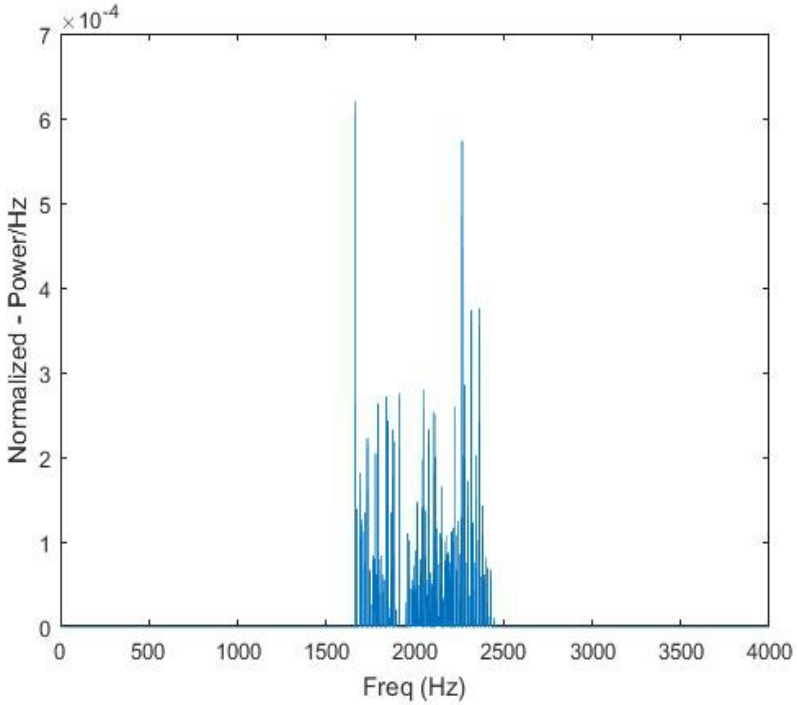

Figure 7. Power Spectrum Diagram for Test Case 6 'no'

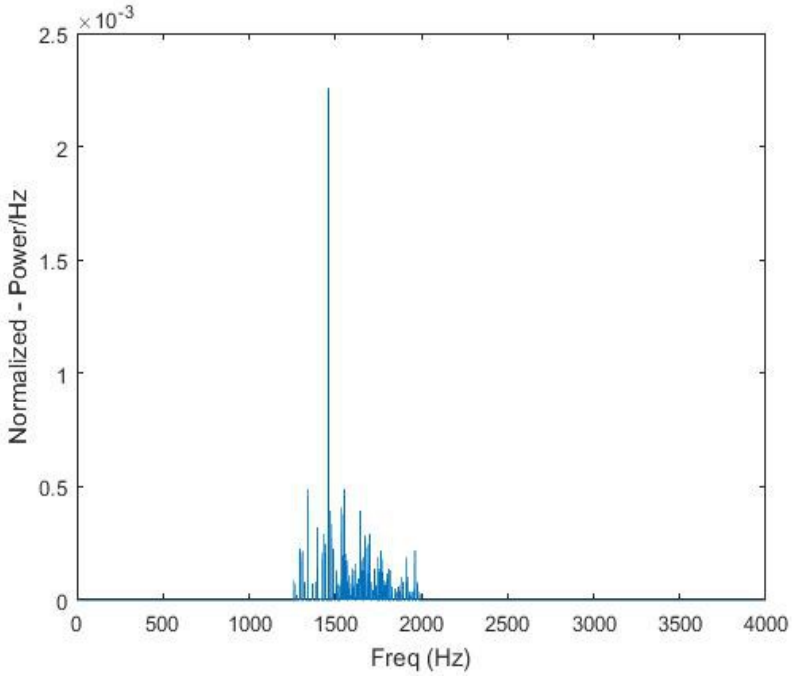

Figure 8. Power Spectrum Diagram for Test Case 7 'no'

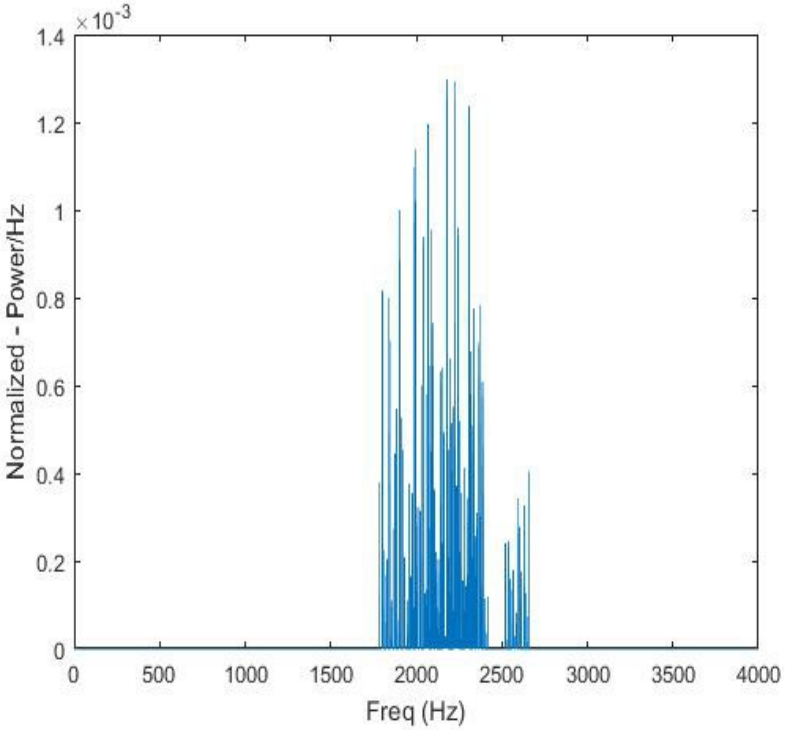

Figure 9. Power Spectrum Diagram for Test Case 8 'no'

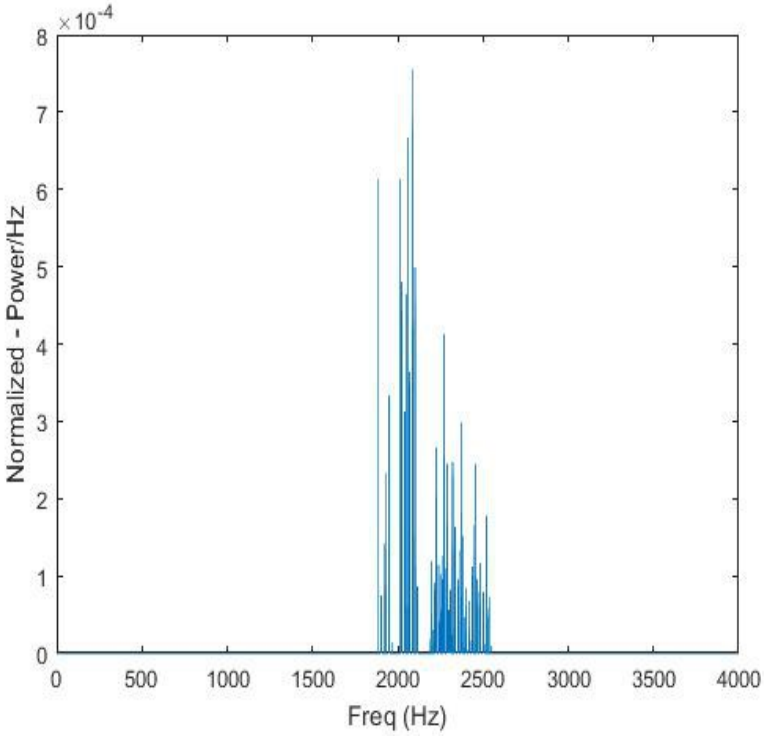

Figure 10. Power Spectrum Diagram for Test Case 9 'no'

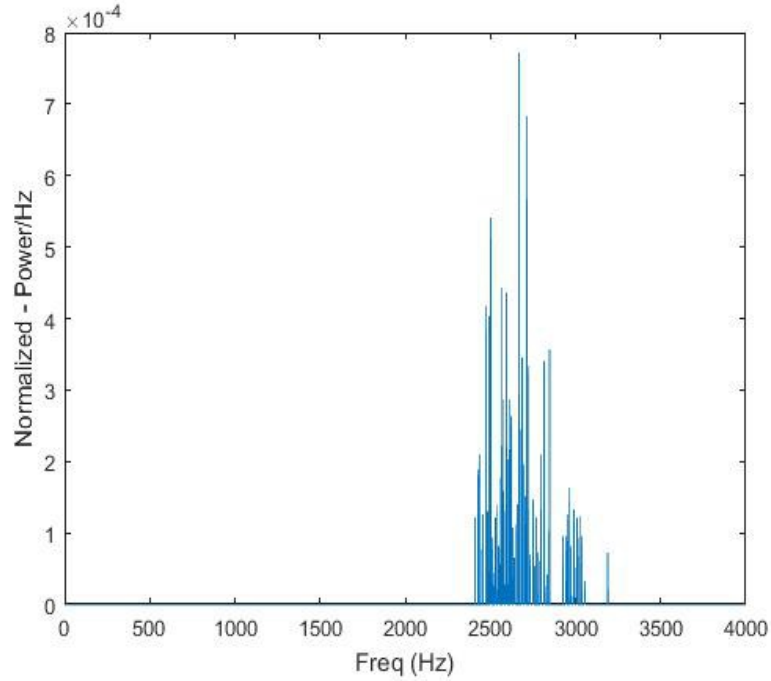

Figure 11. Power Spectrum Diagram for Test Case 10 'no'

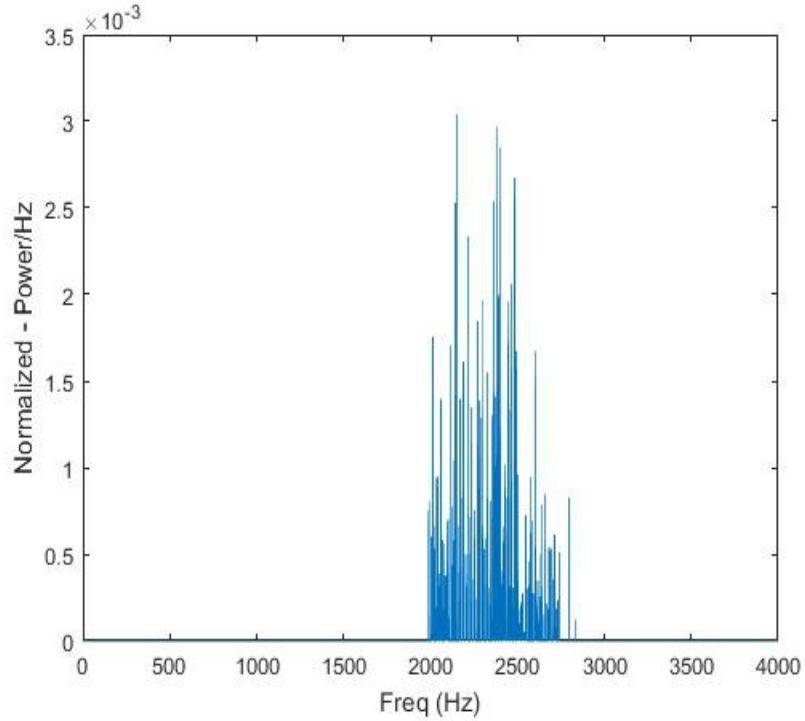

Figure 12. Power Spectrum Diagram for Test Case 11 'yes'

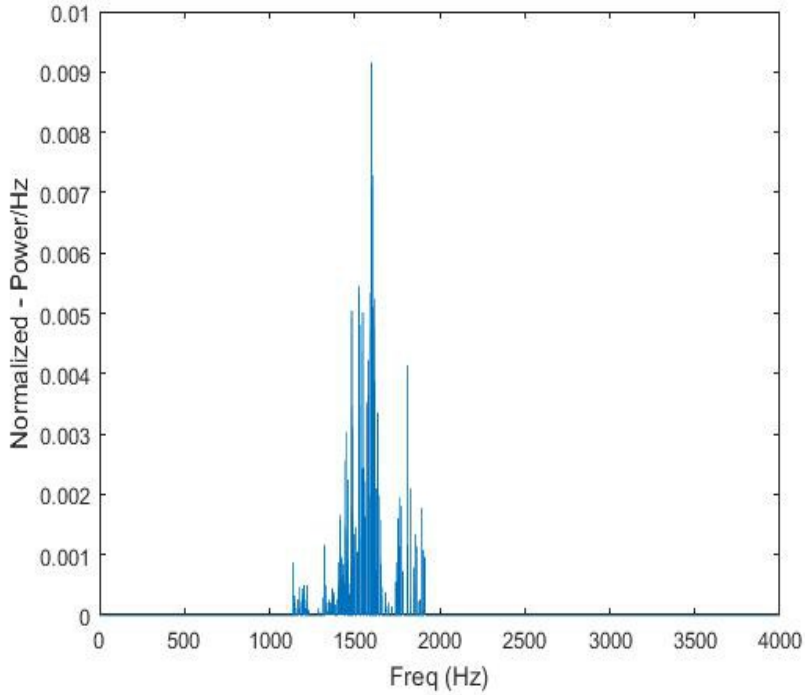

Figure 13. Power Spectrum Diagram for Test Case 12 'yes'

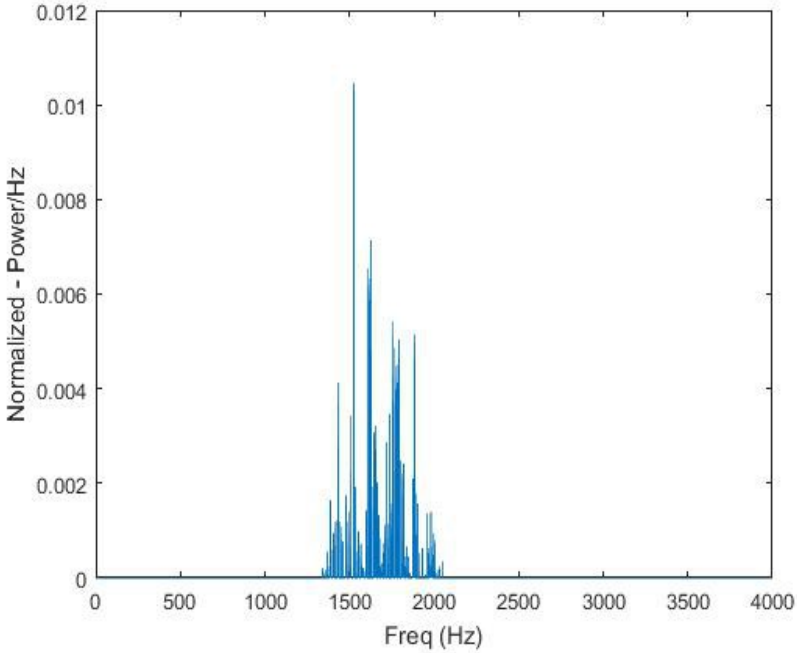

Figure 14. Power Spectrum Diagram for Test Case 13 'yes'

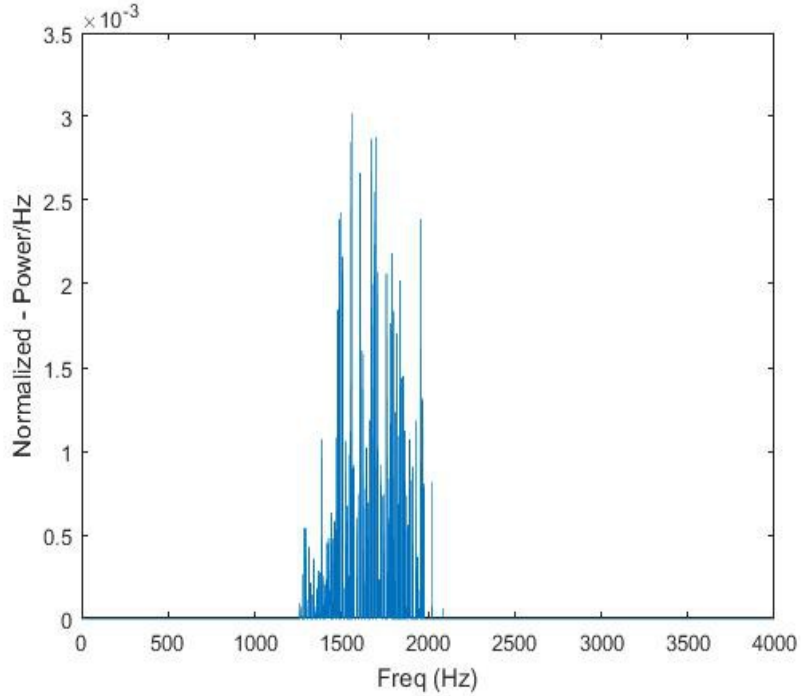

Figure 15. Power Spectrum Diagram for Test Case 14 'yes'

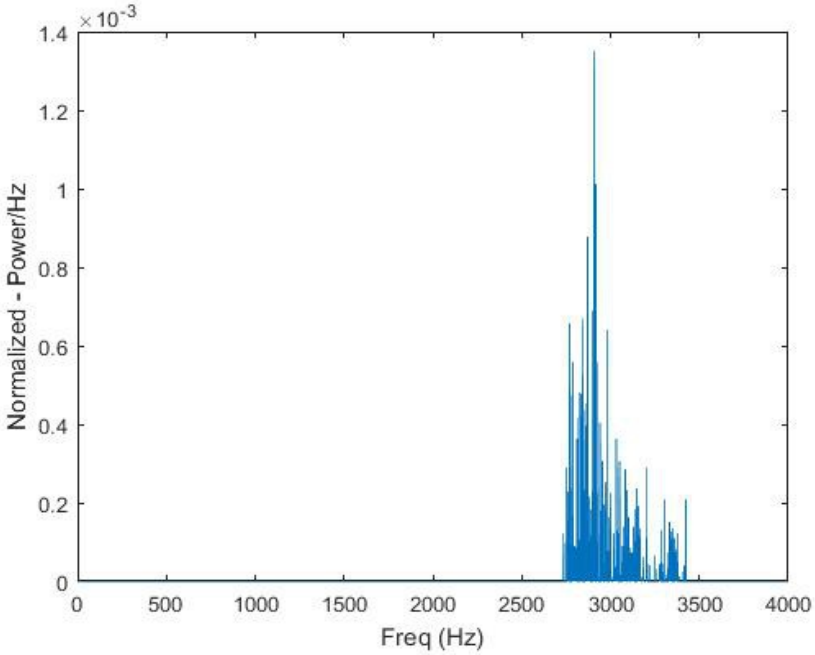

Figure 16. Power Spectrum Diagram for Test Case 15 'yes'

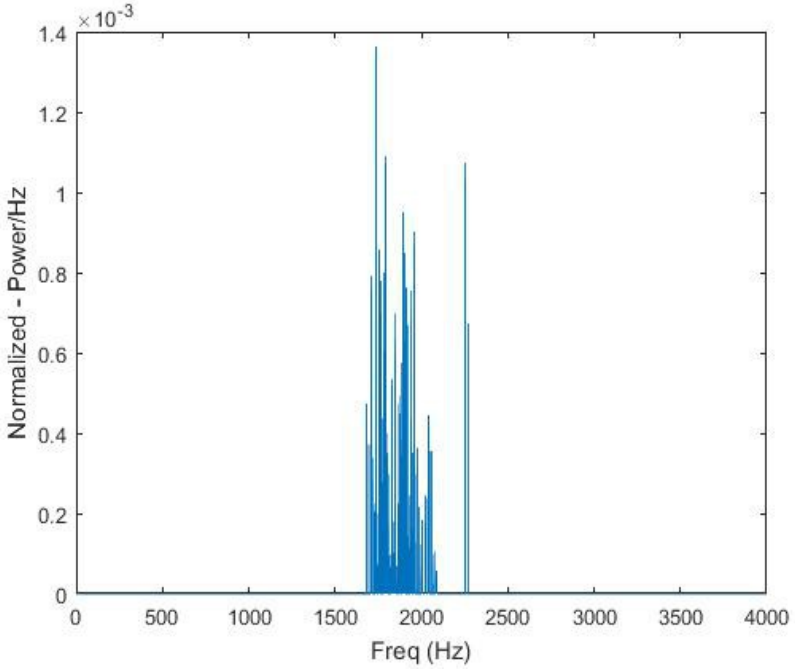

Figure 17. Power Spectrum Diagram for Test Case 16 'no'

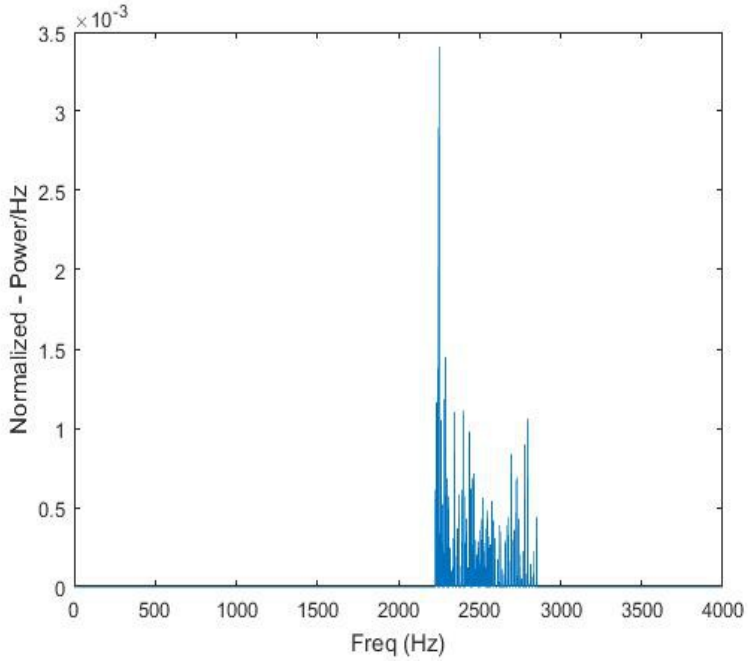

Figure 18. Power Spectrum Diagram for Test Case 17 'no'

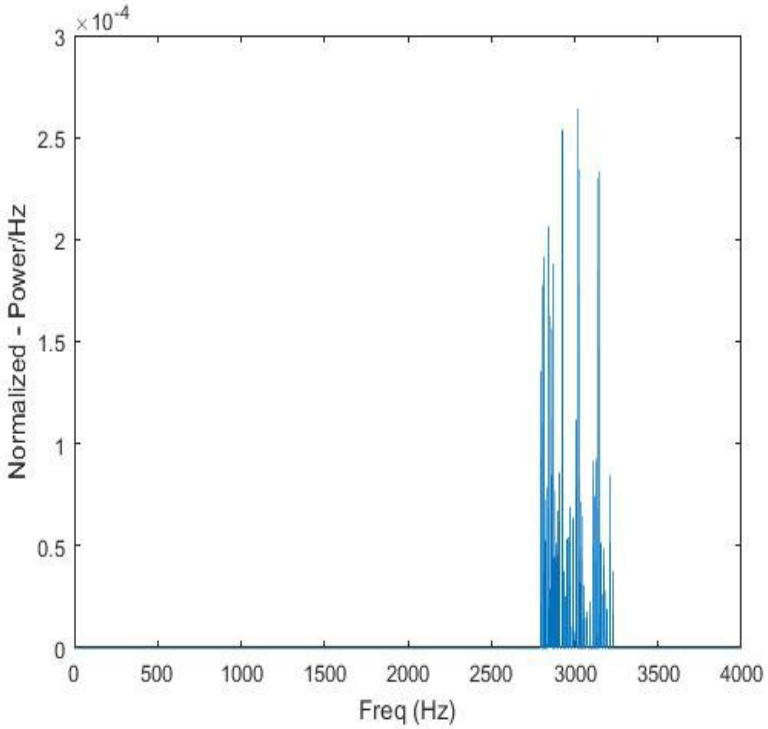

Figure 19. Power Spectrum Diagram for Test Case 18 'no'

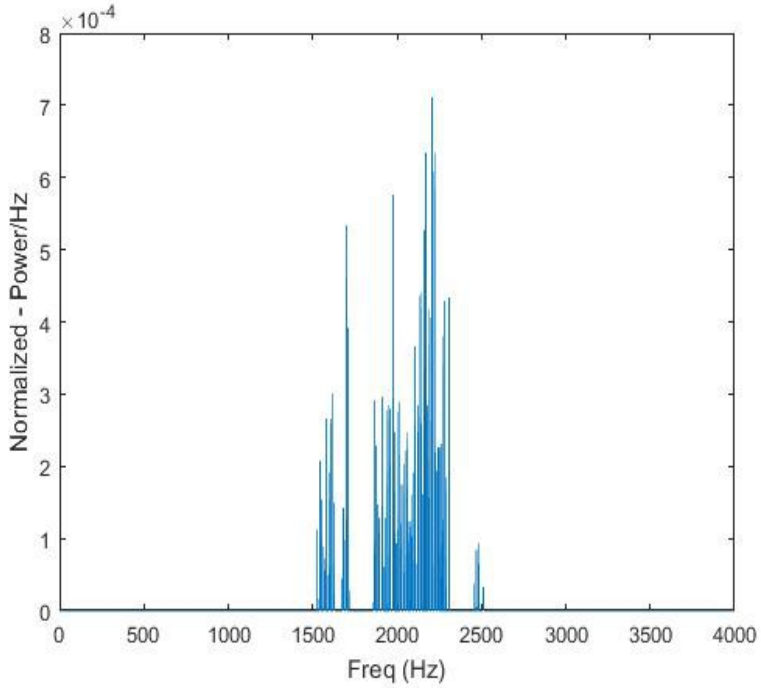

Figure 20. Power Spectrum Diagram for Test Case 19 'no'

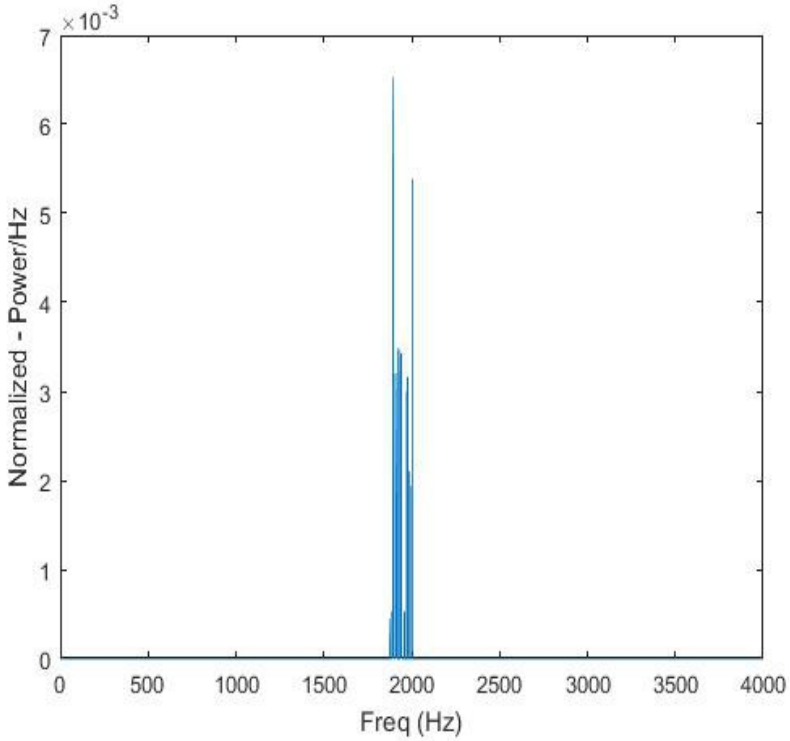

Figure 21. Power Spectrum Diagram for Test Case 20 'no'

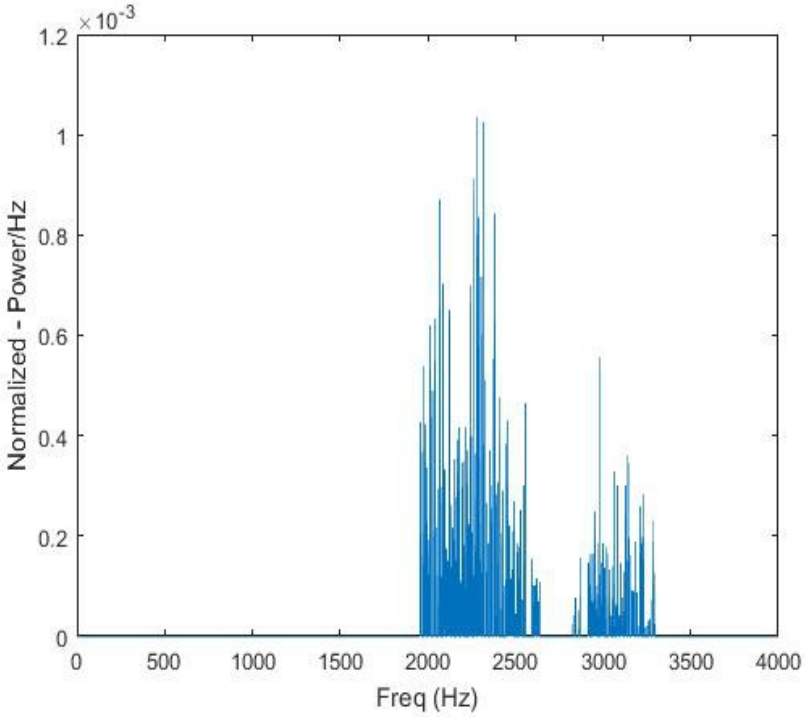

Figure 22. Power Spectrum Diagram for Test Case 21 'yesno'

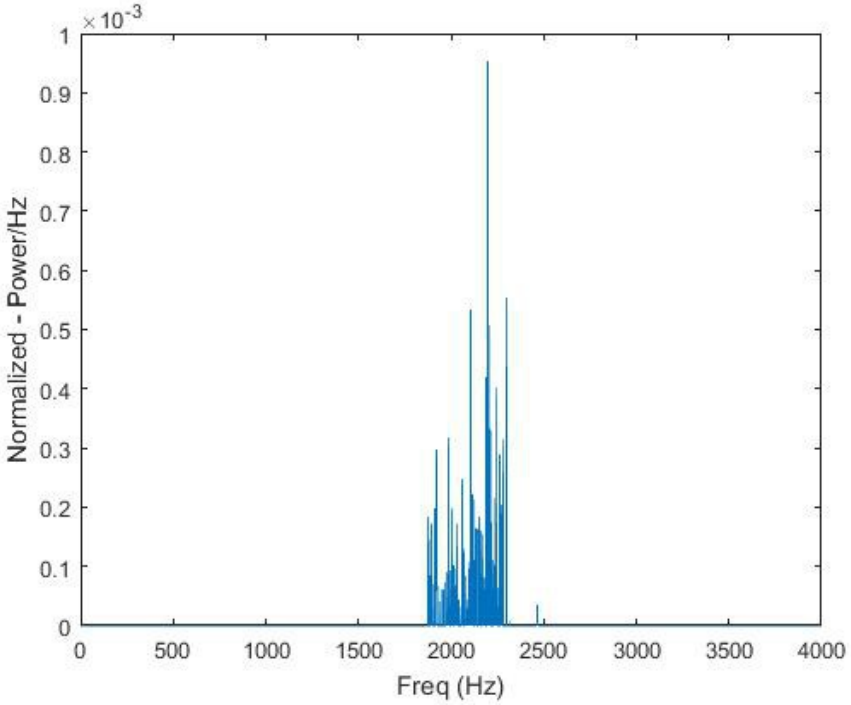

Figure 23. Power Spectrum Diagram for Test Case 22 'test'

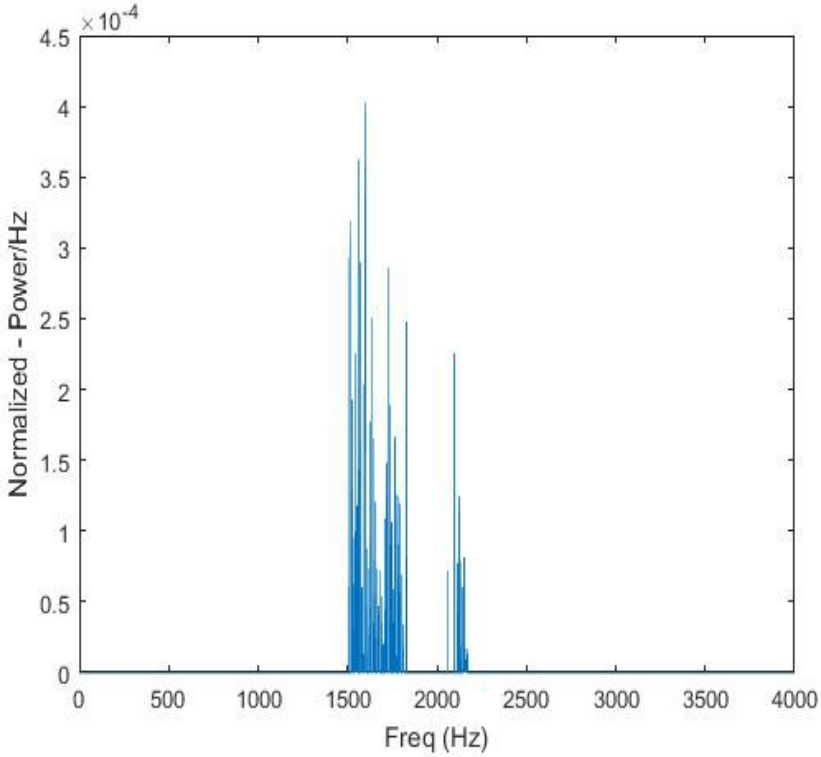

Figure 24. Power Spectrum Diagram for Test Case 23 'home'