ENCE688D/ENSE689E

Mini Project

NOVEMBER 12, 2019

KAMESH ARVIND SARANGAN
115805354

A spectrogram is a visual representation of the Short-Time Fourier Transform. Think of this as taking chunks of an input signal and applying a local Fourier Transform on each chunk. Each chunk has a specified width and we apply a Fourier Transform to this chunk. We should take note that each chunk has an associated frequency distribution. For each chunk that is centered at a specific time point in the time signal, we get a bunch of frequency components. The collection of all of these frequency components at each chunk and plotted all together is what is essentially a spectrogram.

```
spectrogram(x, window, noverlap, nfft, fs)
```

- x - This is the input time-domain signal you wish to find the spectrogram of.
- window - If we recall, we decompose the image into chunks, and each chunk has a specified width. window defines the width of each chunk in terms of **samples**.
- noverlap - Another way to ensure good frequency localization is that the chunks are **overlapping**. A proper spectrogram ensures that each chunk has a certain number of samples that are overlapping for each chunk and noverlap defines how many samples are overlapped in each window. The default is 50% of the width of each chunk.
- nfft - tells us how many FFT points are desired to be computed per chunk. The default number of points is the largest of either 256, or floor(log2(N)) where N is the length of the signal. nfft also gives a measure of how fine-grained the frequency resolution will be. A higher number of FFT points would give higher frequency resolution and thus showing fine-grained details along the frequency axis of the spectrogram if visualized.
- fs - The sampling frequency of your signal.

Windowing gives us an advantage of examining segments of larger spectrum of data for transient events and time averaging of frequency spectrum. Windows gives an advantage of retrieving the smaller signals that could be lost while analyzing the wider range of spectrum. Truncating the signals can be done without maximum loss in data. By using windowing functions, you can further enhance the ability of an FFT to extract spectral data from signals. Windowing functions act on raw data to reduce the effects of the leakage that occurs during an FFT of the data. Leakage amounts to spectral information from an FFT showing up at the wrong frequencies. The people who first studied the effect thought of the spectral information as "leaking" into adjacent frequency values. Here random window functions are tested across data. Hamming window, triangular window and Kaiser window are tested as samples here

Hamming Window :

The Hamming window is defined as

w(n)=0.54−0.46cos(2πnM−1) $0 \le n \le M-1$ w(n) = 0.54−0.46cos(2πnM−1) $0 \le n \le M-1$

It was recommended for smoothing the truncated autocovariance function in the time domain. Most references to the Hamming window come from the signal processing literature, where it is used as one of many windowing functions for smoothing values. It is also known as an apodization (which means “removing the foot”, i.e. smoothing discontinuities at the beginning and end of the sampled signal) or tapering function. The Hamming window is a taper formed by using a raised cosine with non-zero endpoints, optimized to minimize the nearest side lobe.

Hamming length = 25, overlap = 10 resolution = 30000 -

- **Window -**

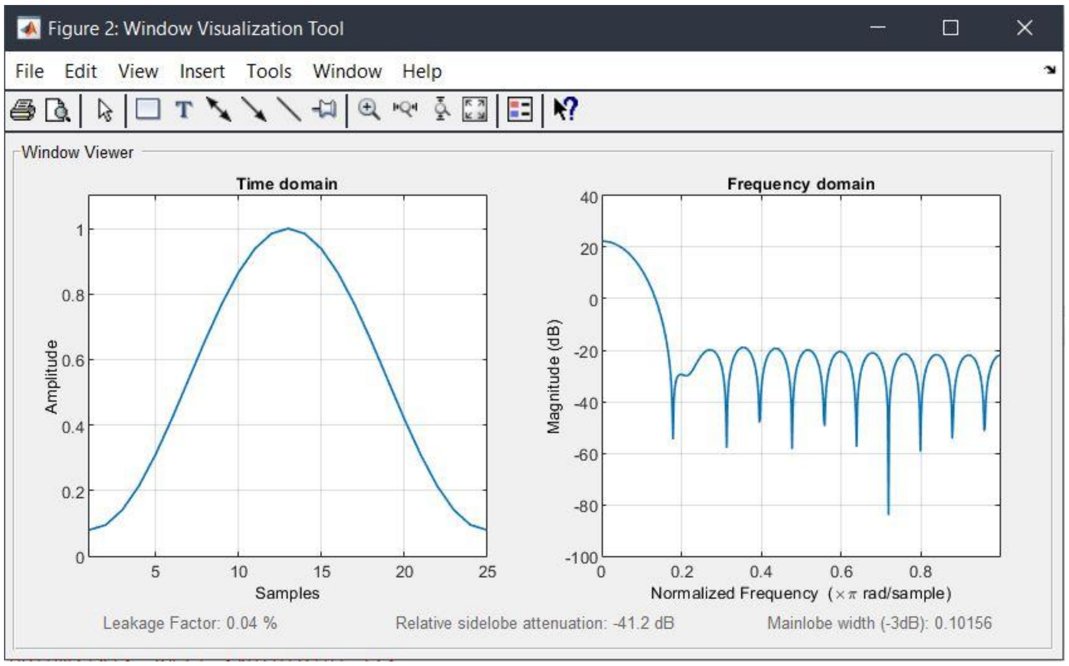


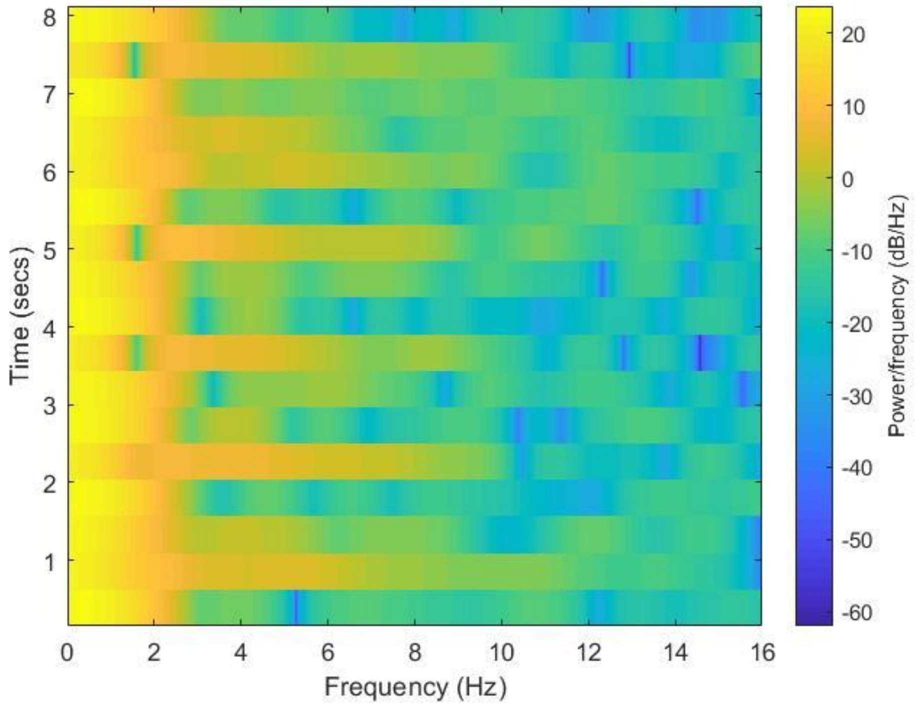
Figure 2: Window Visualization Tool
File Edit View Insert Tools Window Help
Window Viewer
Time domain
Amplitude
Samples
Frequency domain
Magnitude (dB)
Normalized Frequency (×π rad/sample)

Leakage Factor: 0.04 %

Relative sidelobe attenuation: -41.2 dB

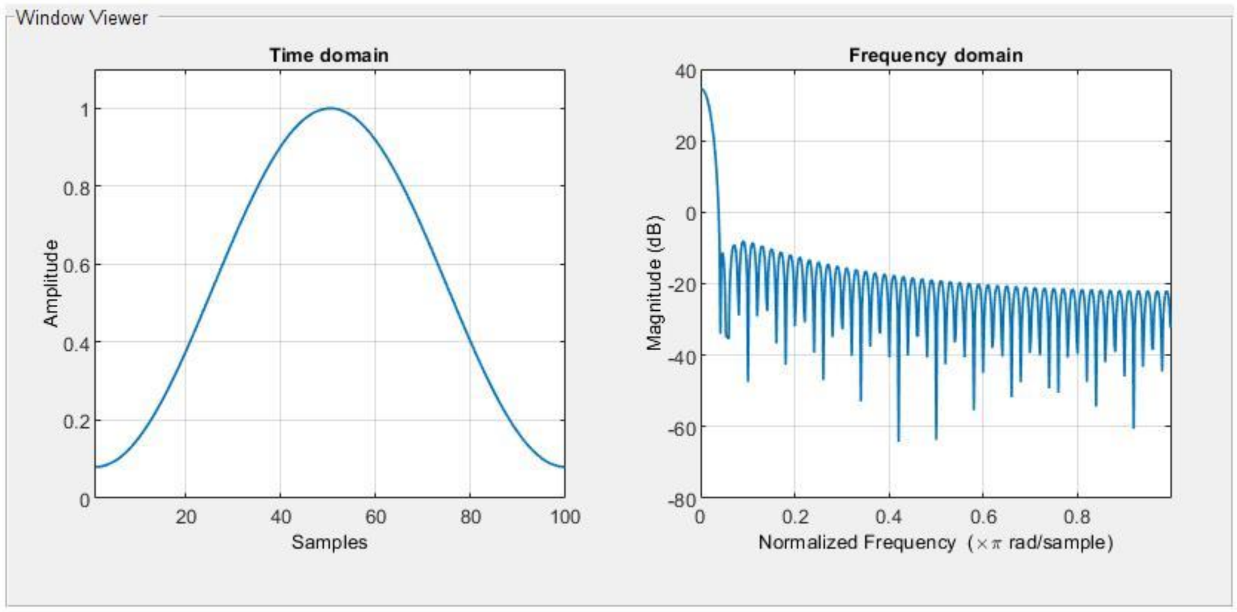
Mainlobe width (-3dB): 0.10156

- **Spectrogram –**

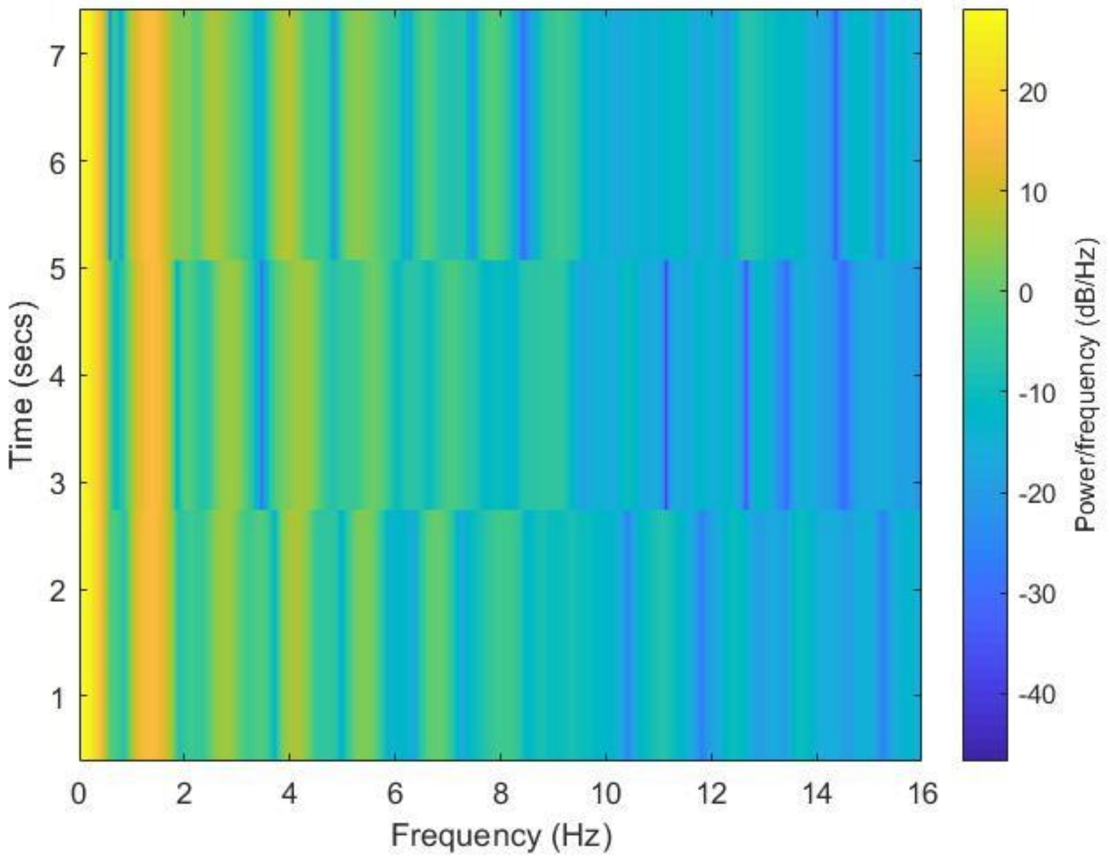


Hamming length = 100, overlap = 25 resolution = 30000 –

- **Window**

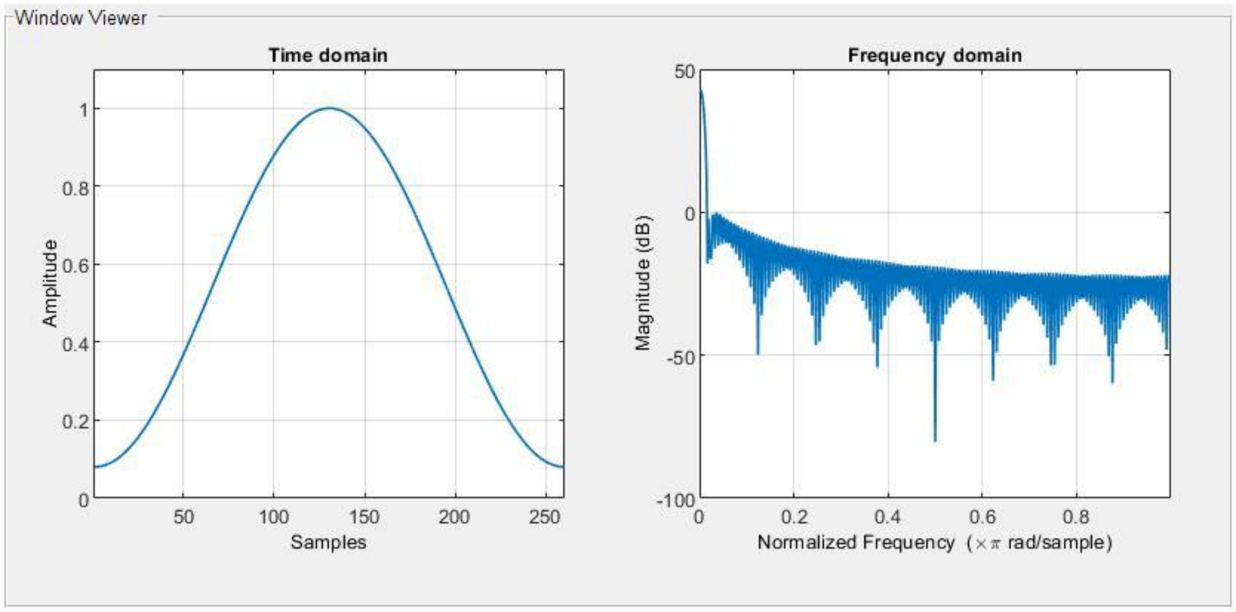


- **Spectrogram –**

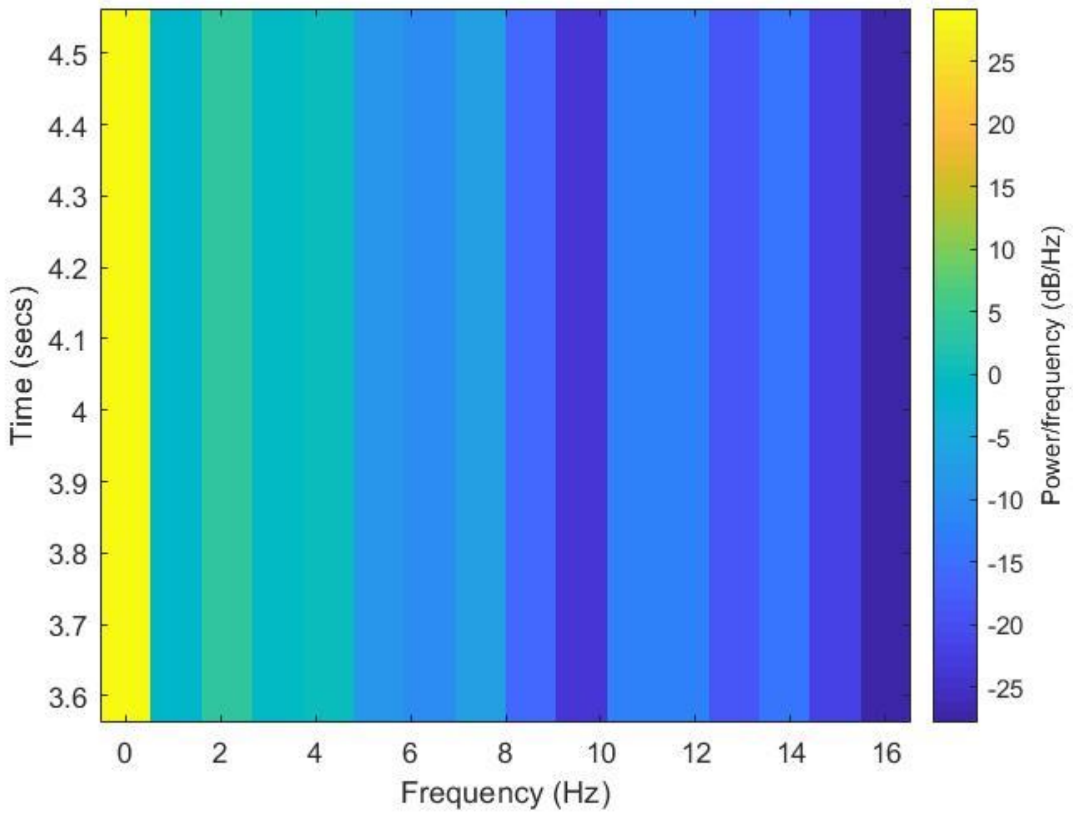


Hamming length = 260, overlap = 150, resolution = 30 –

- **Window –**



- **Spectrogram –**



The above figures illustrates the spectrogram and windows for hamming window, and the difference in hamming length, overlap values & nfft costing the lower resolution of data, as the smaller values are overlapped even with windowing.

Ideally you would only see that frequency as a narrow peak but because of the finite-length window you also get that other artificial crud: Windowing equals time domain multiplication by the window function (Hamming). Time domain multiplication equals frequency domain convolution: All frequencies will be replaced by the Fourier transform of the window function. The hamming length segments the signal to desired viewpoints as mentioned by hamming length. Lower the hamming, length the resolution of the windowed spectrogram is lower as the smaller activities over time is hard to differentiate. Different hamming length are tried to show how the windowed data looks and differs for varying hamming length.

Triangular Window:

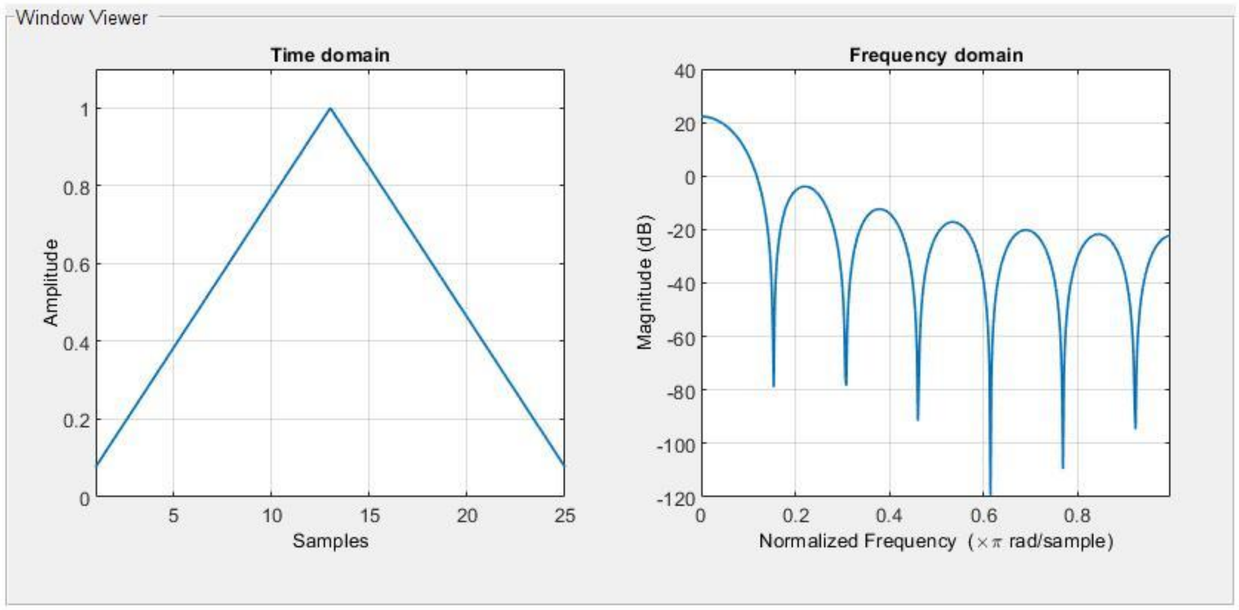
The triangular coefficients are given by the following formula

$$a(k) = \frac{2}{N-1}\left(\frac{N-1}{2} - \left|k - \frac{N-1}{2}\right|\right) = 1 - \left|\frac{k - \frac{N-1}{2}}{\frac{N-1}{2}}\right|$$

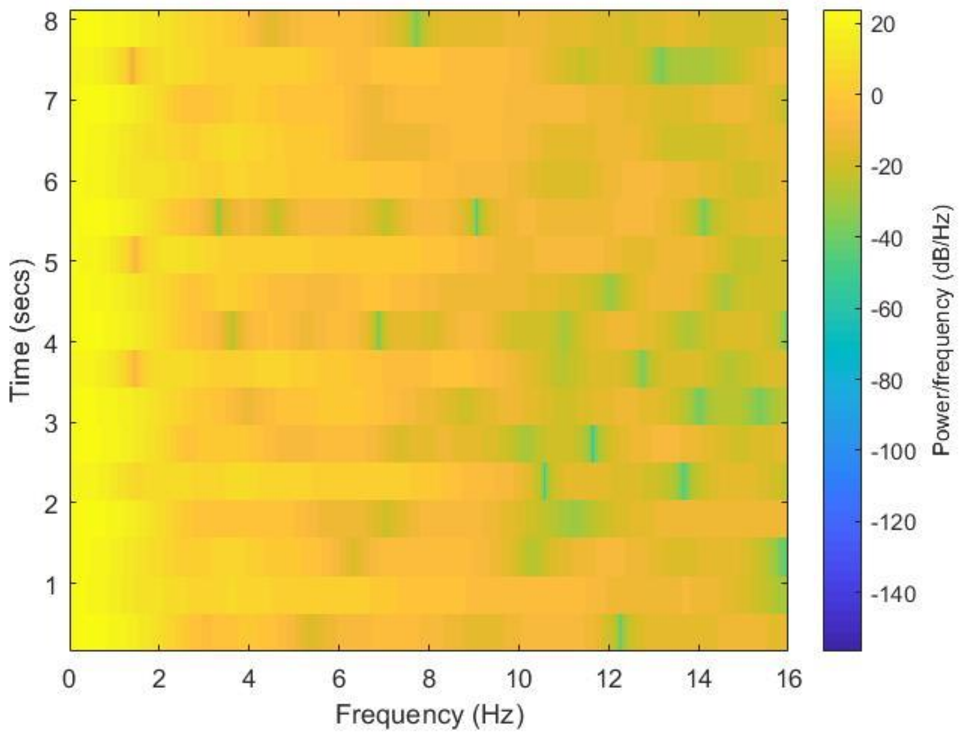
where N is the length of the filter and k = 0, 1, …, N – 1. Different definitions may use N / 2 or (N + 1) / 2 in the denominator of the last expression. When (N – 1) /2 is used, as in the formula above, the window is also known as the Bartlett window, or the triangular window with zero end points.

Triangular length = 25, overlap = 10, resolution = 30000 –

- **Window –**

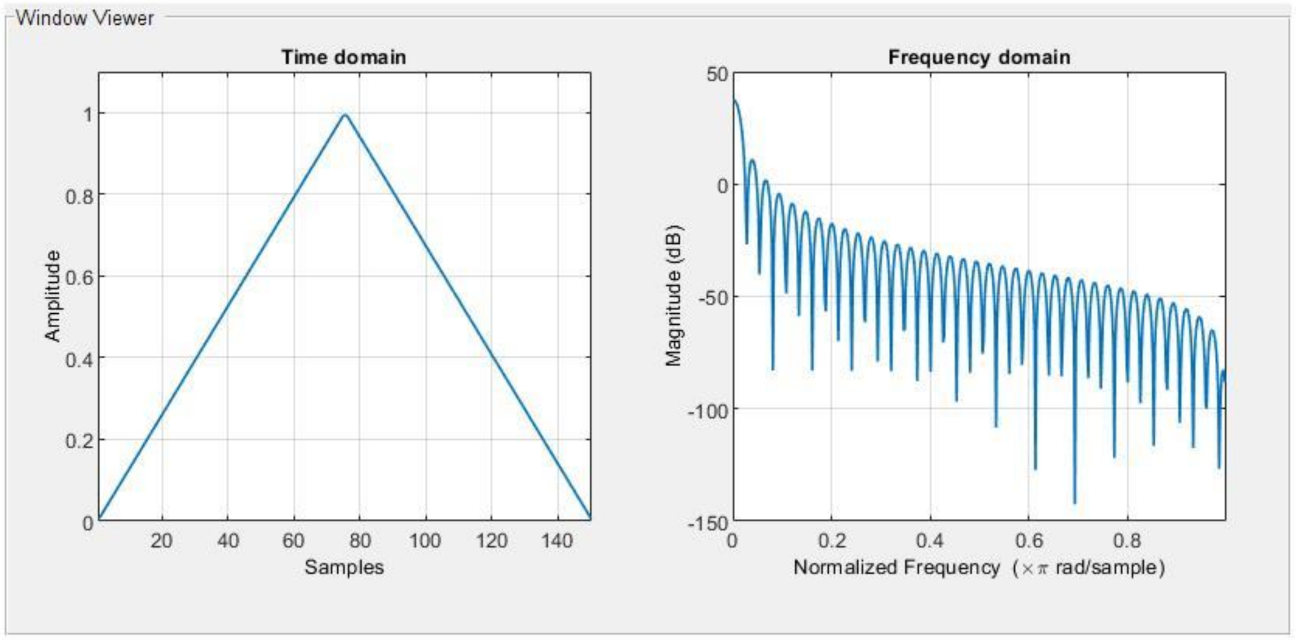


- **Spectrogram –**

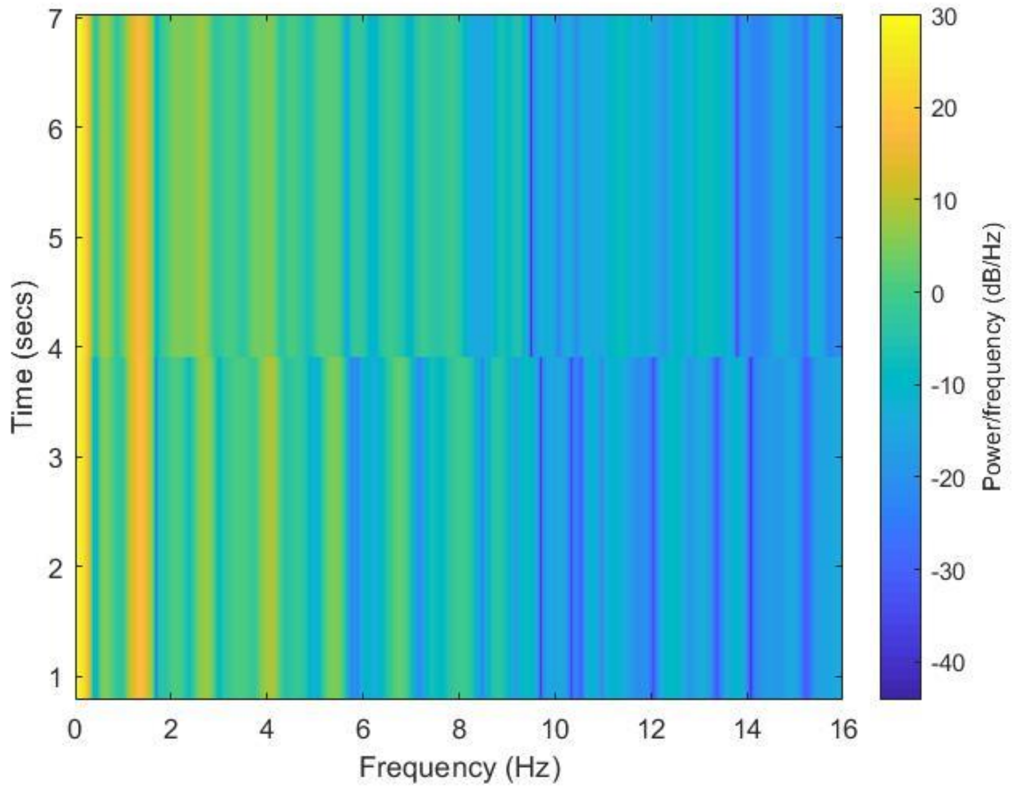


Triangular length = 150, overlap = 50, resolution = 30000 –

- **Window –**

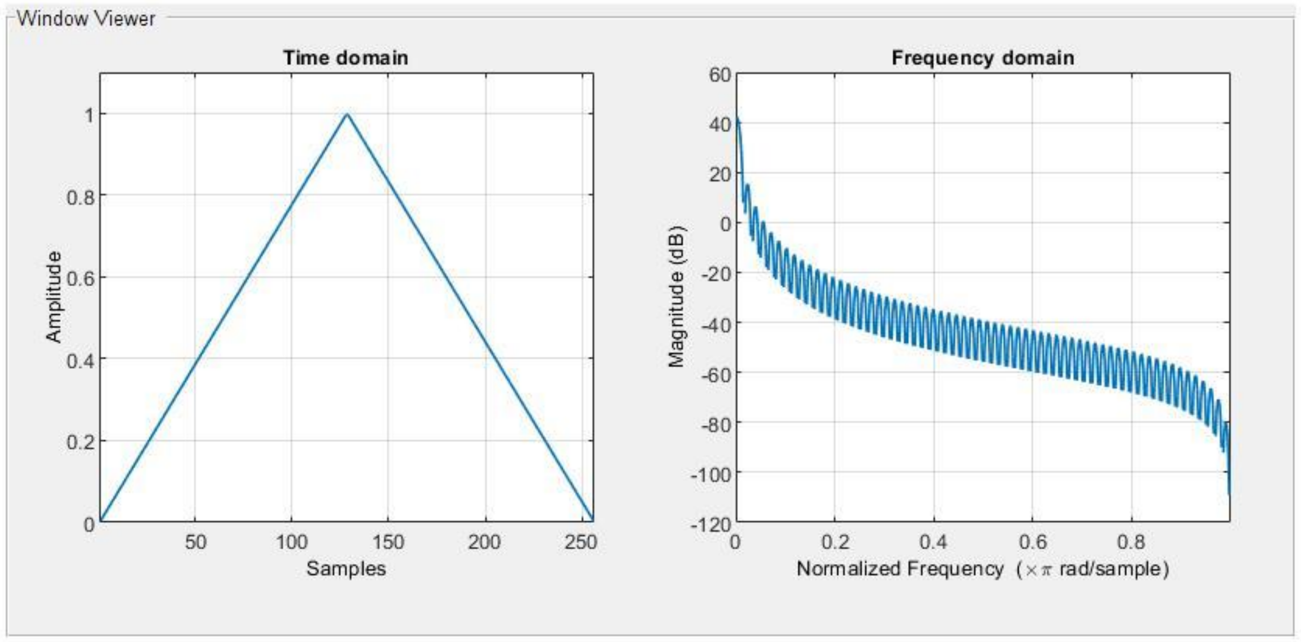


- **Spectrogram –**

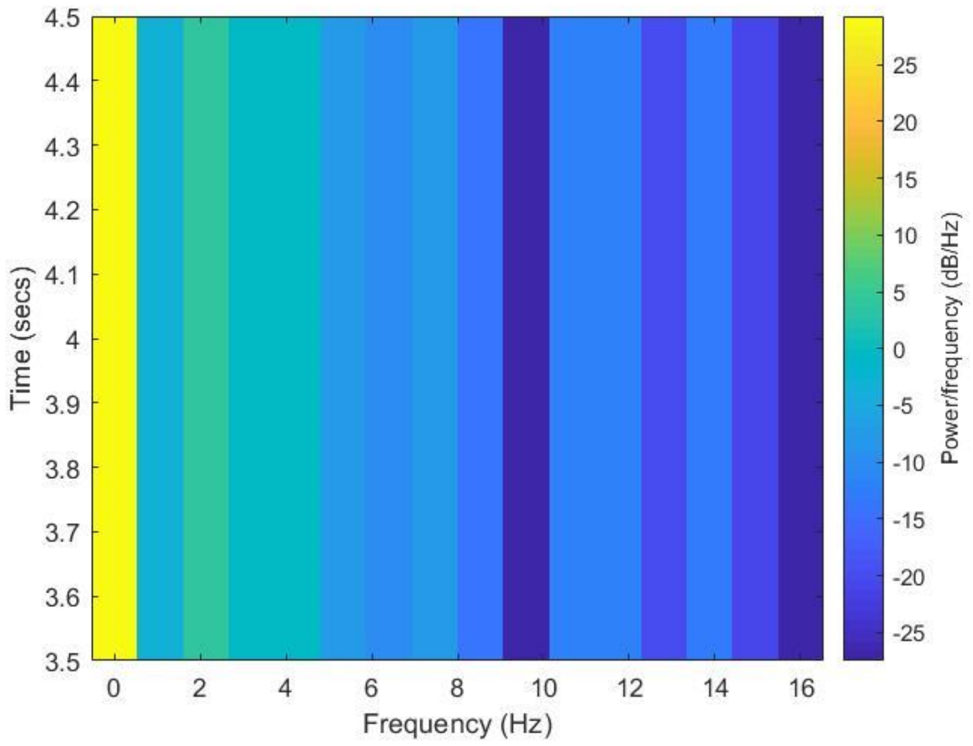


Triangular length = 260, overlap = 150, resolution = 30 –

- **Window –**



- **Spectrogram –**



Triangular Smoothing refers to using a triangular weighting of the data points in the moving window which generates the smoothed values. With a triangular window the points in the middle of the window matter more, and the weighting decreases to zero going out towards the edges of the window. This reduces the effects of aliasing and is often a good replacement for a traditional moving average (using a rectangular window where each point has equal weighting).

The above figures illustrates the spectrogram and windows for triangular window, and the difference in triangle length, overlap values & nfft costing the lower resolution of data, as the smaller values are overlapped even with windowing.

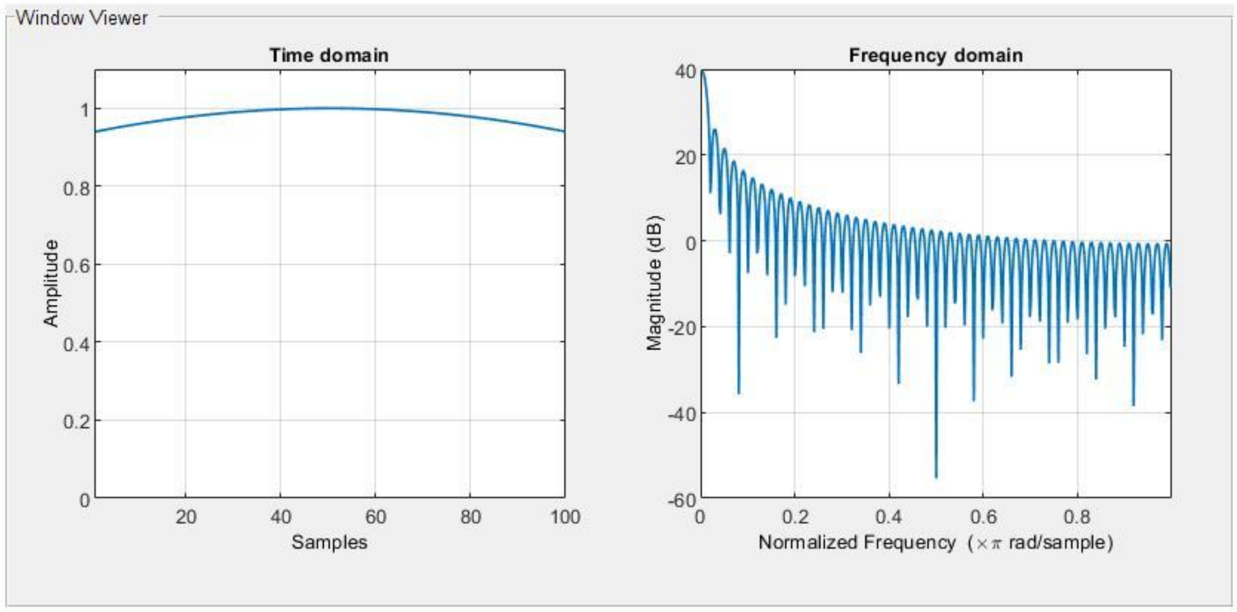
Kaiser Window:

The Kaiser window approximates the prolate spheroidal window, for which the ratio of the main lobe energy to the sidelobe energy is maximized. For a Kaiser window of a length, the parameter β controls the relative sidelobe attenuation. For a given β, the relative sidelobe attenuation is fixed with respect to window length. The statement kaiser(n, beta) computes a length n Kaiser window with parameter beta.

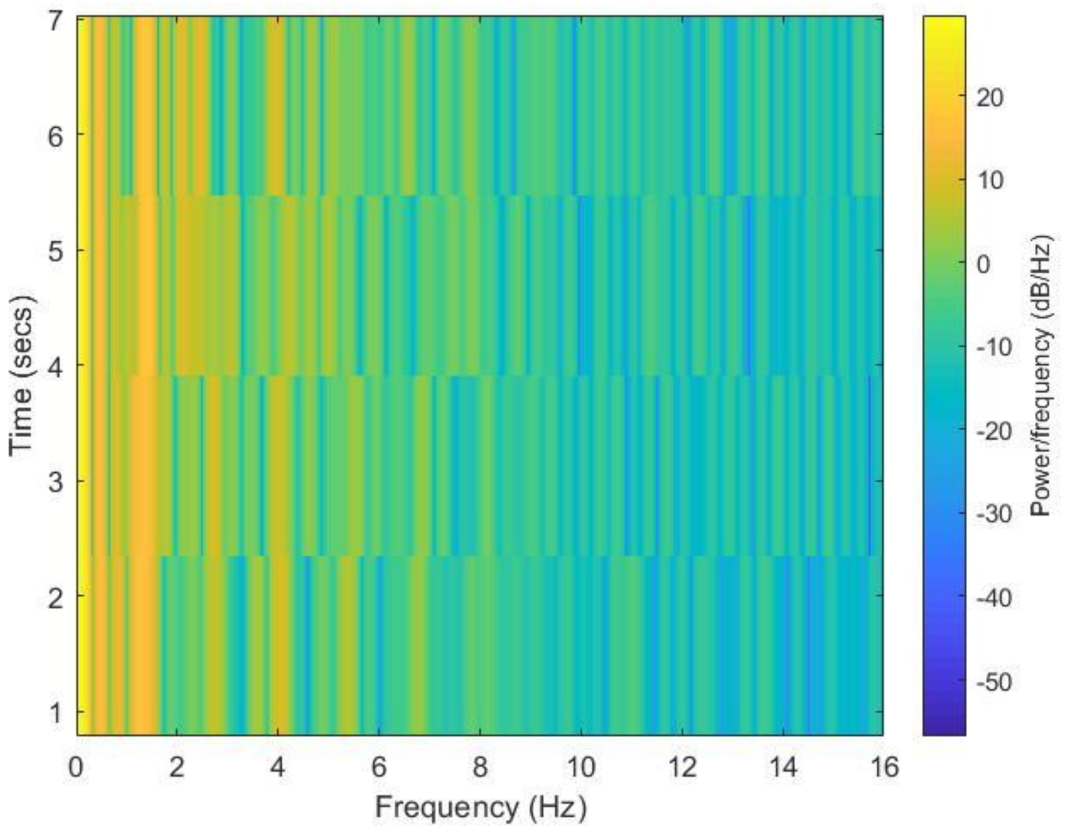
As β increases, the relative sidelobe attenuation decreases and the main lobe width increases.

Kaiser length = 100, overlap = 50, beta = 2.5, resolution = 30000 –

- **Window –**

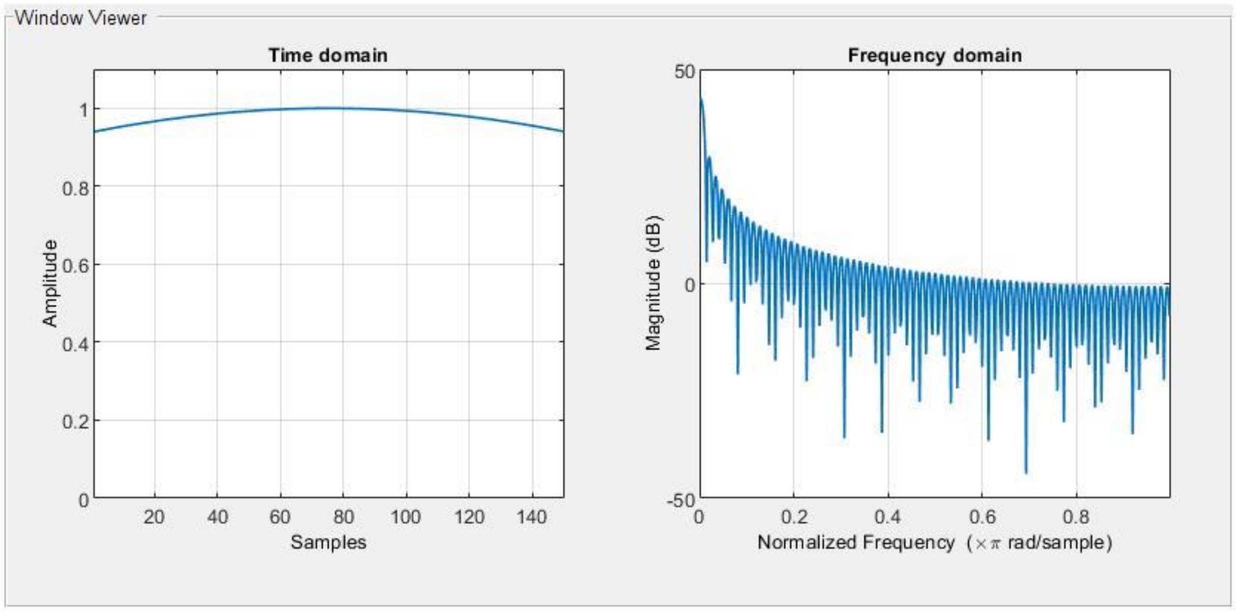


- **Spectrogram –**

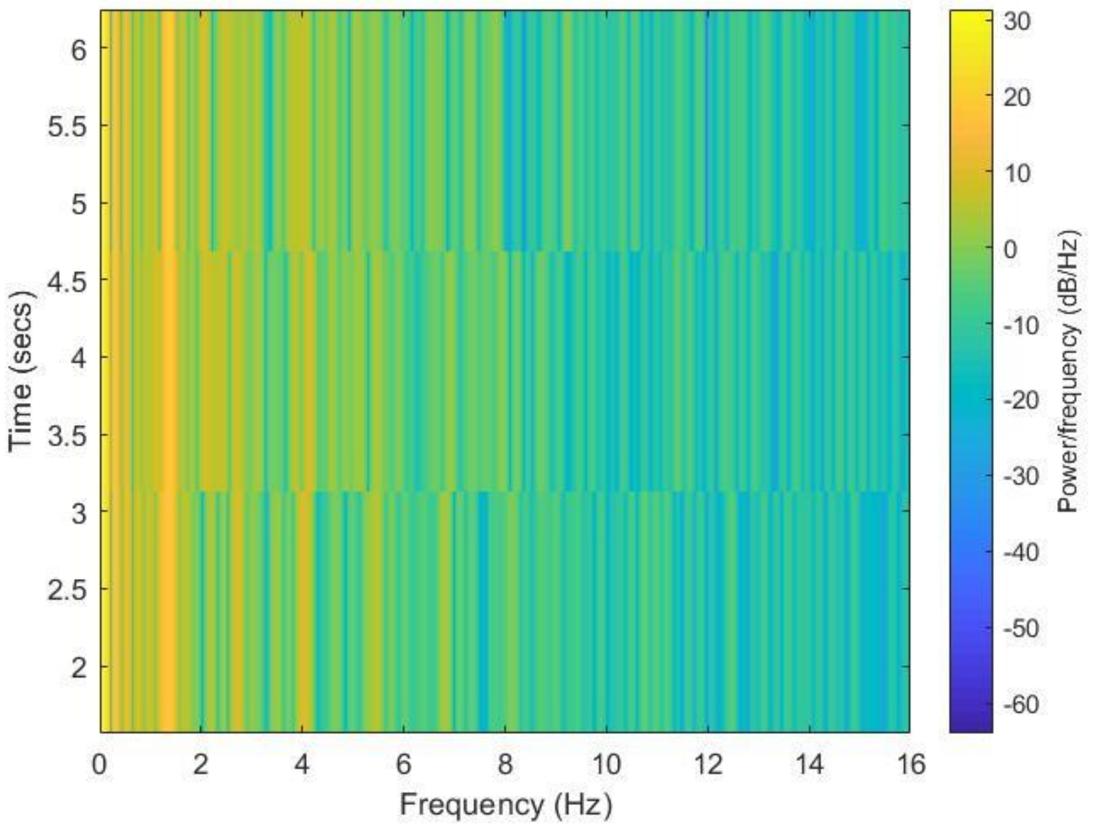


Kaiser length = 150, overlap = 50, beta = 0.5, resolution = 30000 –

- **Window –**

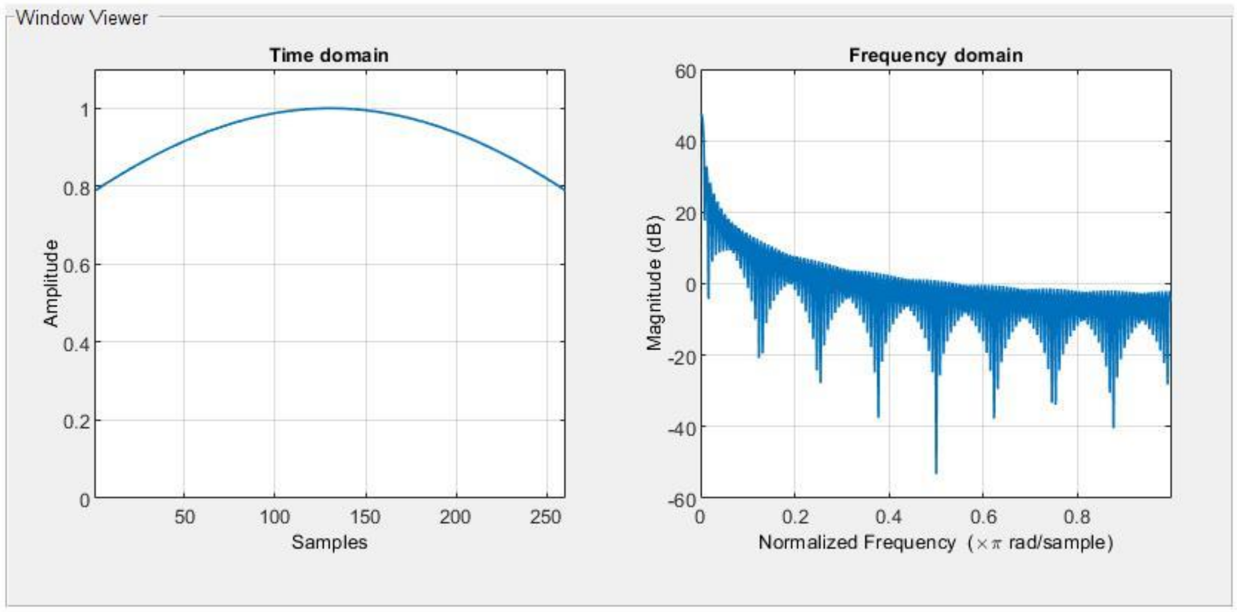


- **Spectrogram –**

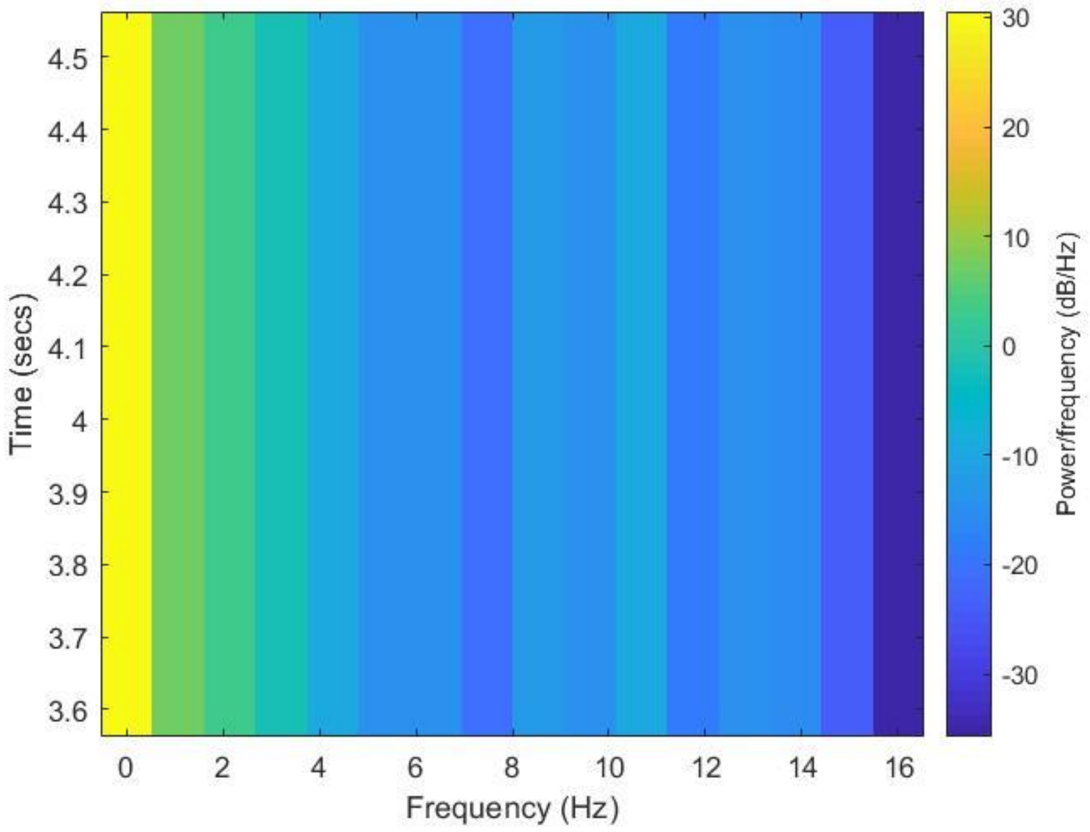


Kaiser length = 260, overlap = 150, beta = 1.0, resolution = 30 –

- **Window –**

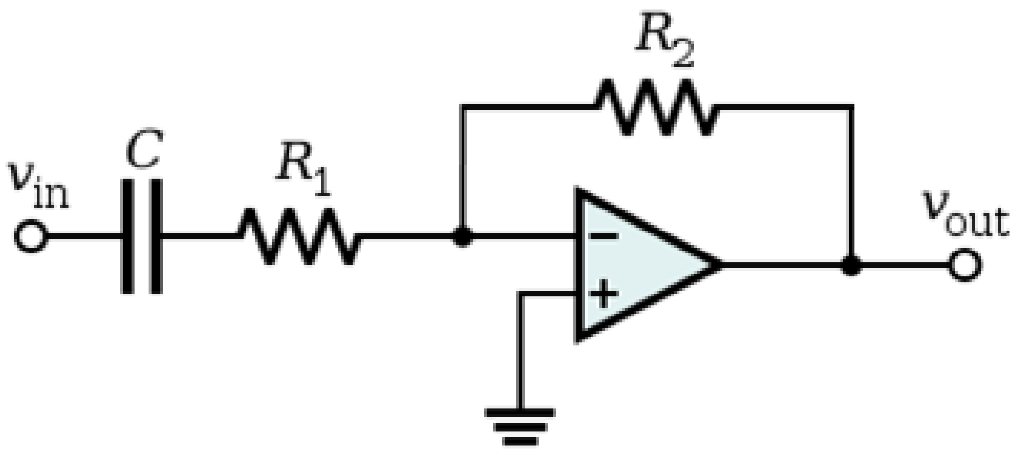


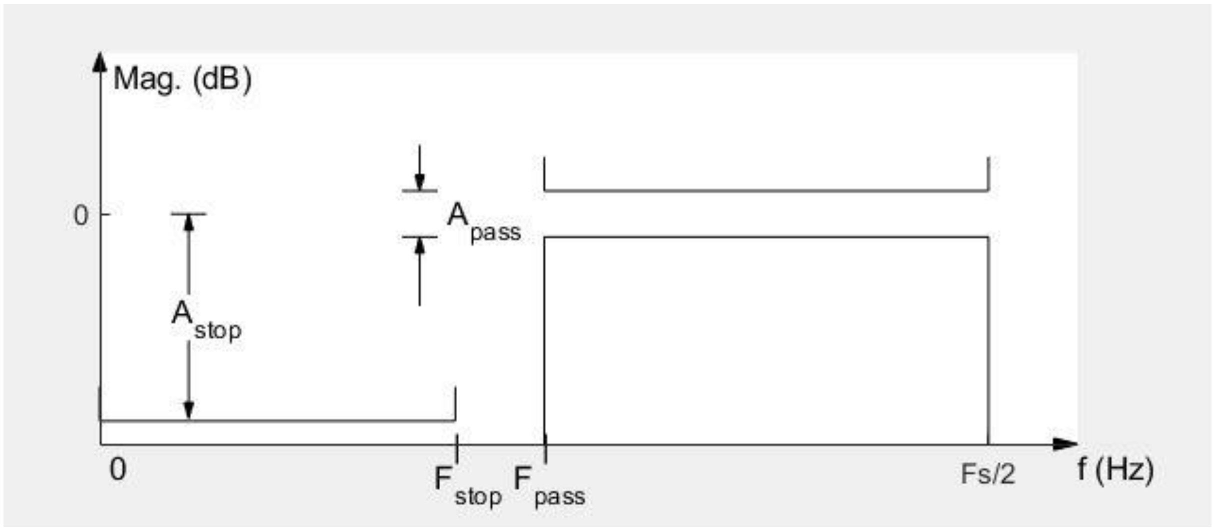
- **Spectrogram –**

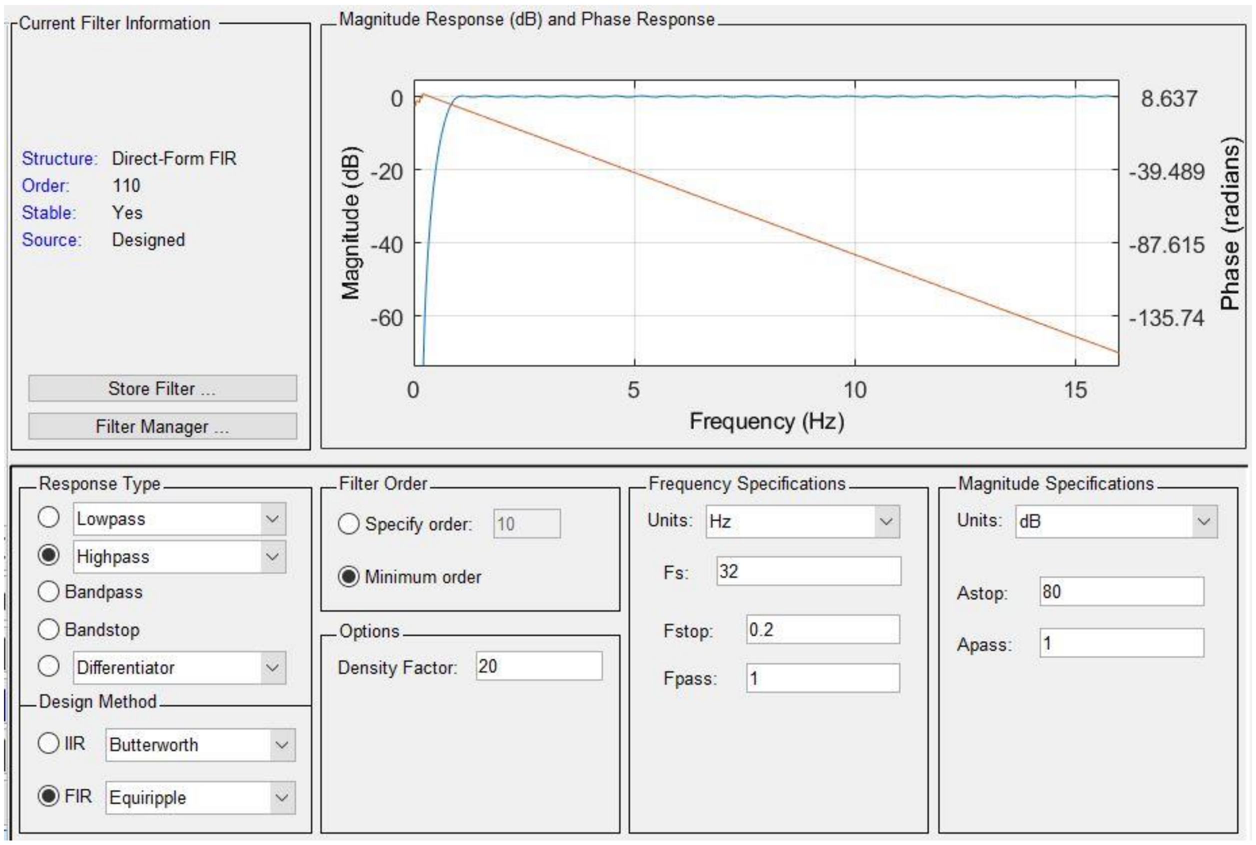


Filter Design:

- **Highpass Filter –** A high-pass filter (HPF) is an electronic filter that passes signals with a frequency higher than a certain cutoff frequency and attenuates signals with frequencies lower than the cutoff frequency. The amount of attenuation for each frequency depends on the filter design. A high-pass filter is usually modeled as a linear time-invariant system. It is sometimes called a low-cut filter or bass-cut filter. High-pass filters have many uses, such as blocking DC from circuitry sensitive to non-zero average voltages or radio frequency devices.
  Filter Design, Magnitude/Phase vs frequency –







Current Filter Information

Structure: Direct-Form FIR
Order: 110
Stable: Yes
Source: Designed

Response Type

Lowpass [✓ Highpass] Bandpass Bandstop Differentiator

Design Method

IIR Butterworth [✓ FIR Equiripple]

Filter Order

Specify order: 10 [✓ Minimum order]

Options

Density Factor: 20

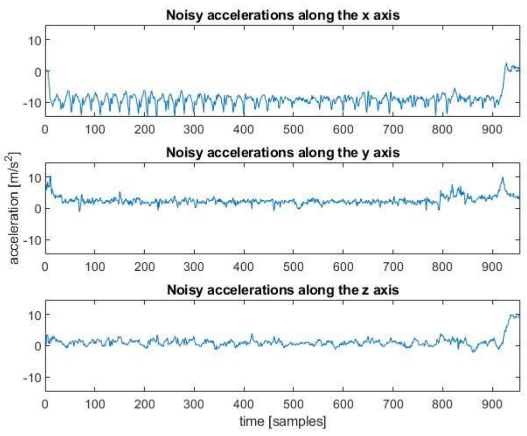
Frequency Specifications

Units: Hz
Fs: 32
Fstop: 0.2
Fpass: 1

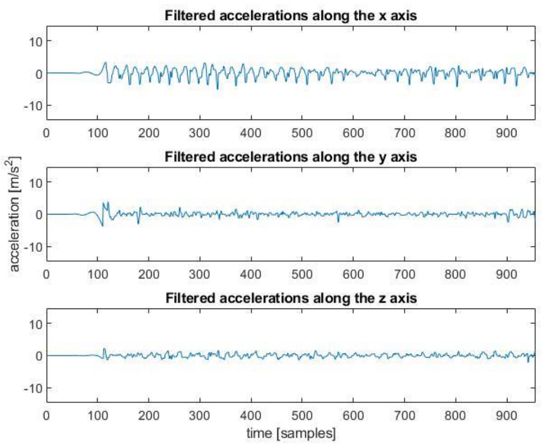
Magnitude Specifications

Units: dB
Astop: 80
Apass: 1

The filter is designed with the corresponding observed values Fstop = 0.2, Fpass = 1. The graph is plotted with noise data vs Highpass data.



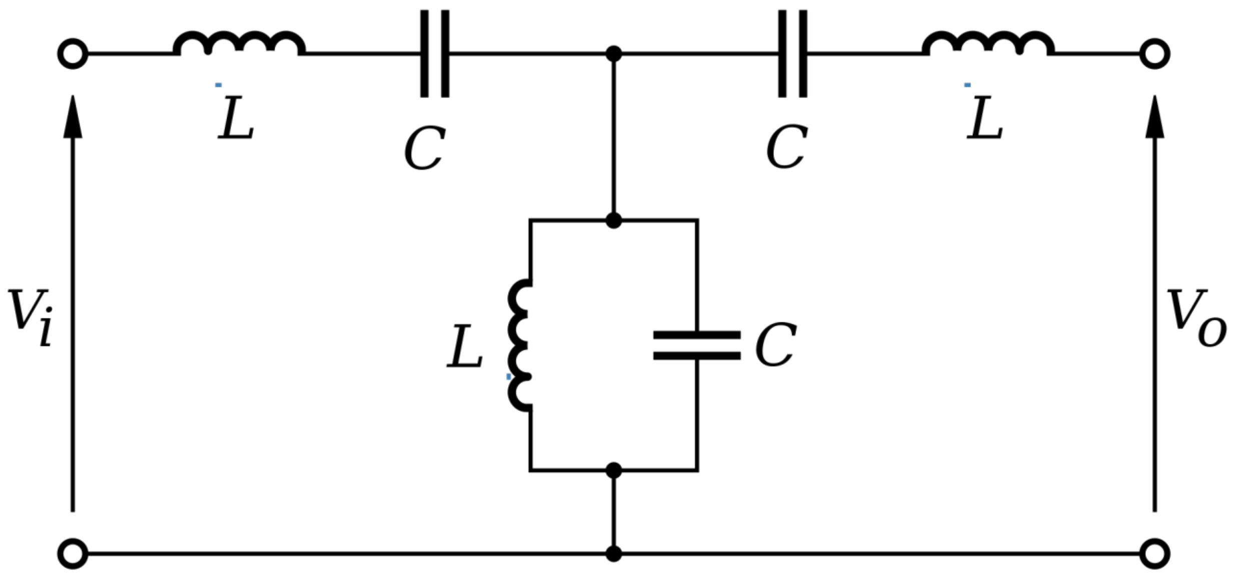
RAW DATA

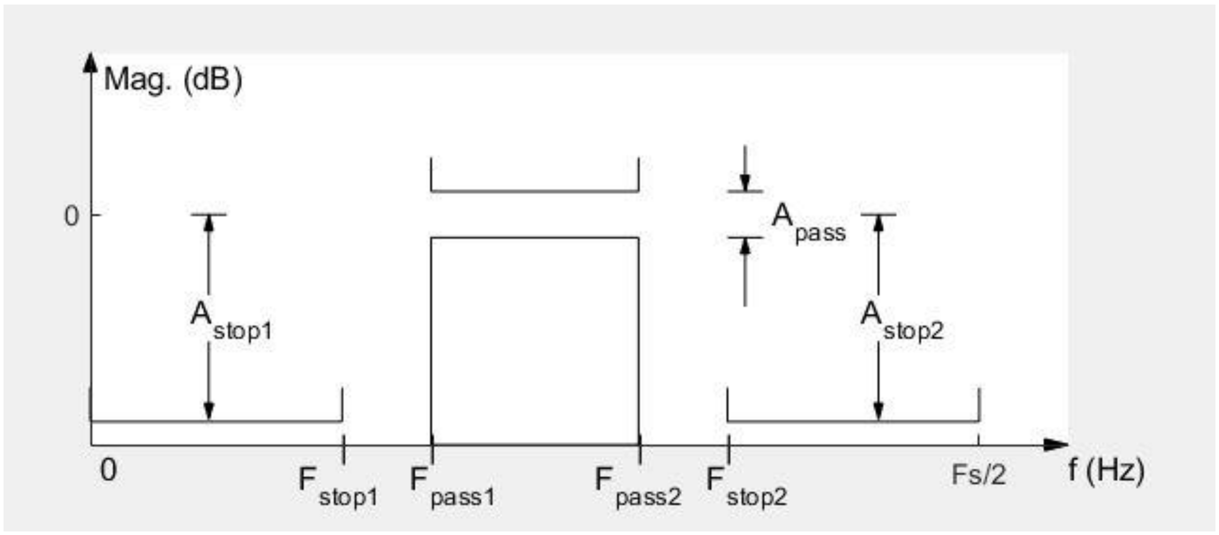


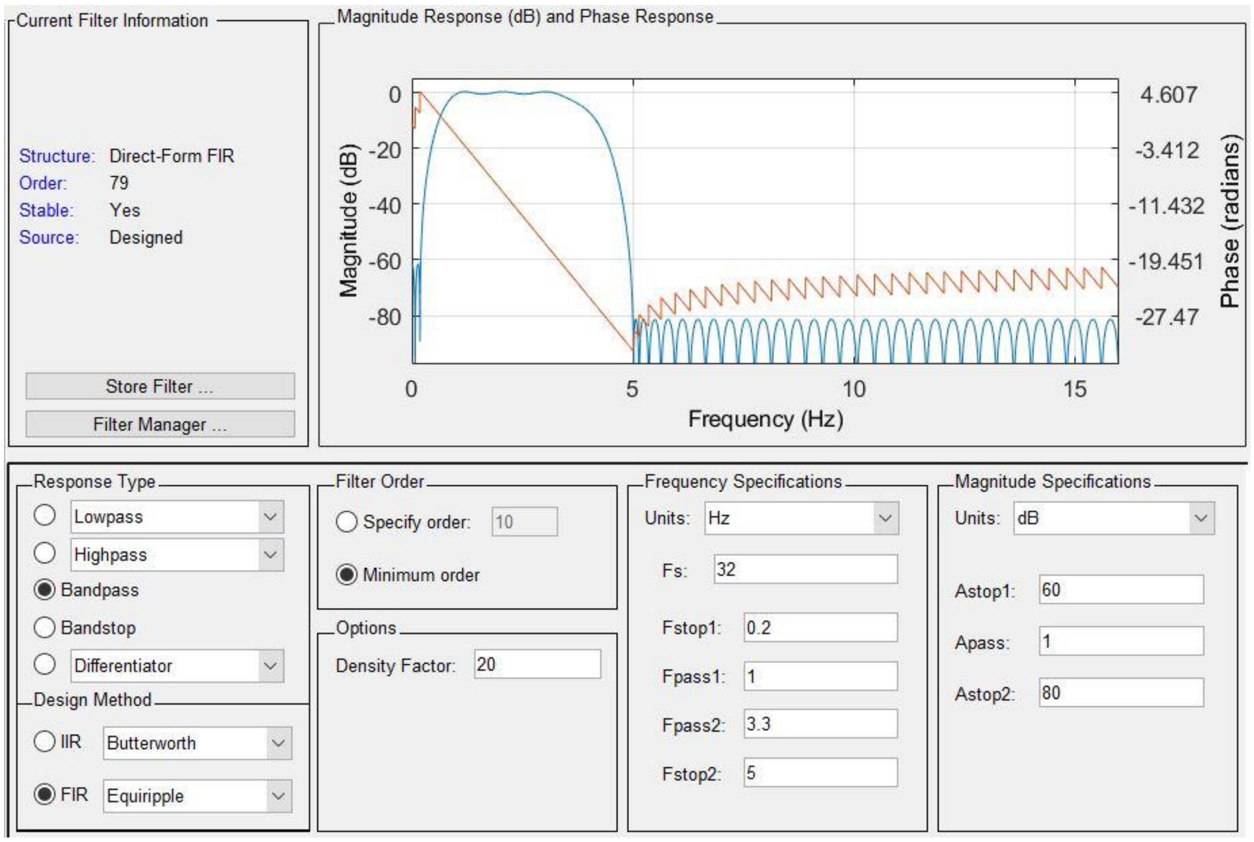
HIGHPASS DATA

- **Bandpass Filter -** Bandpass is an adjective that describes a type of filter or filtering process; it is to be distinguished from passband, which refers to the actual portion of affected spectrum. Hence, one might say "A dual bandpass filter has two passbands." A bandpass signal is a signal containing a band of frequencies not adjacent to zero frequency, such as a signal that comes out of a bandpass filter. Bandpass filters are widely used in wireless transmitters and receivers. The main function of such a filter in a transmitter is to limit the bandwidth of the output signal to the band allocated for the transmission. This prevents the transmitter from interfering with other stations. In a receiver, a bandpass filter allows signals within a selected range of frequencies to be heard or decoded, while preventing signals at unwanted frequencies from getting through. A bandpass filter also optimizes the signal-to-noise ratio and sensitivity of a receiver.

  Filter Design, Magnitude/Phase vs frequency –







Current Filter Information

Structure: Direct-Form FIR
Order: 79
Stable: Yes
Source: Designed

Response Type

- ○ Lowpass
- ○ Highpass
- ● Bandpass
- ○ Bandstop
- ○ Differentiator

Design Method

- ○ IIR Butterworth
- ● FIR Equiripple

Filter Order

- ○ Specify order: 10
- ● Minimum order

Options

Density Factor: 20

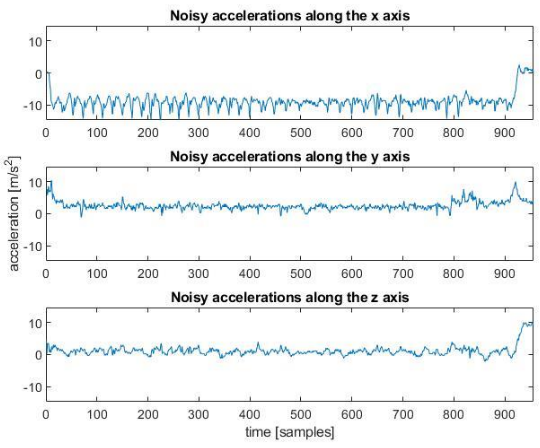
Frequency Specifications

Units: Hz
Fs: 32
Fstop1: 0.2
Fpass1: 1
Fpass2: 3.3
Fstop2: 5

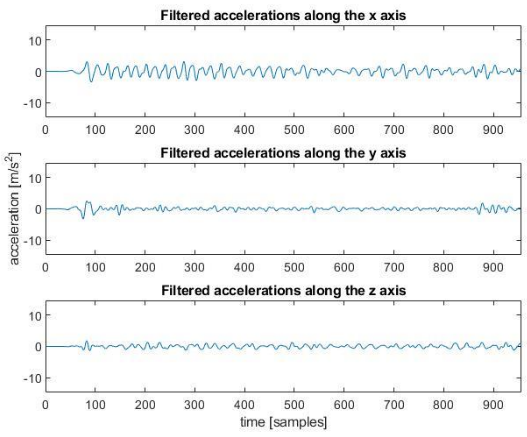
Magnitude Specifications

Units: dB
Astop1: 60
Apass: 1
Astop2: 80

The filter is designed with the corresponding observed values Fstop1 = 0.2, Fpass1 = 1, Fpass2 = 3.3, Fstop2 = 5. The graph is plotted with noise data vs bandpass data.

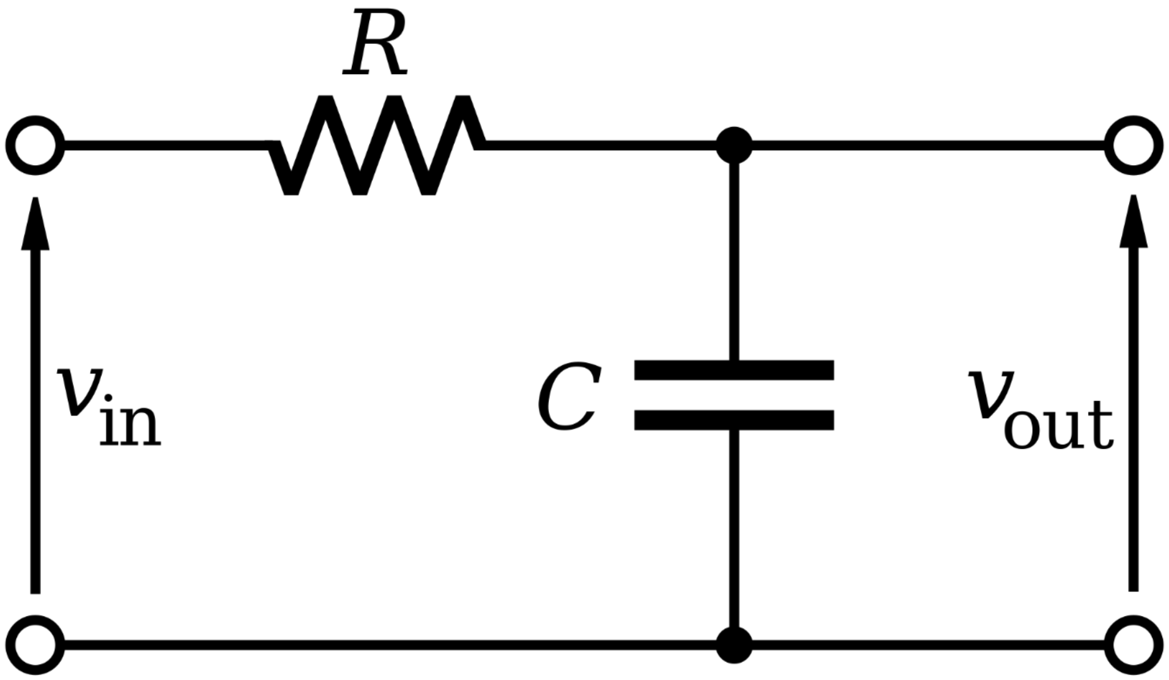


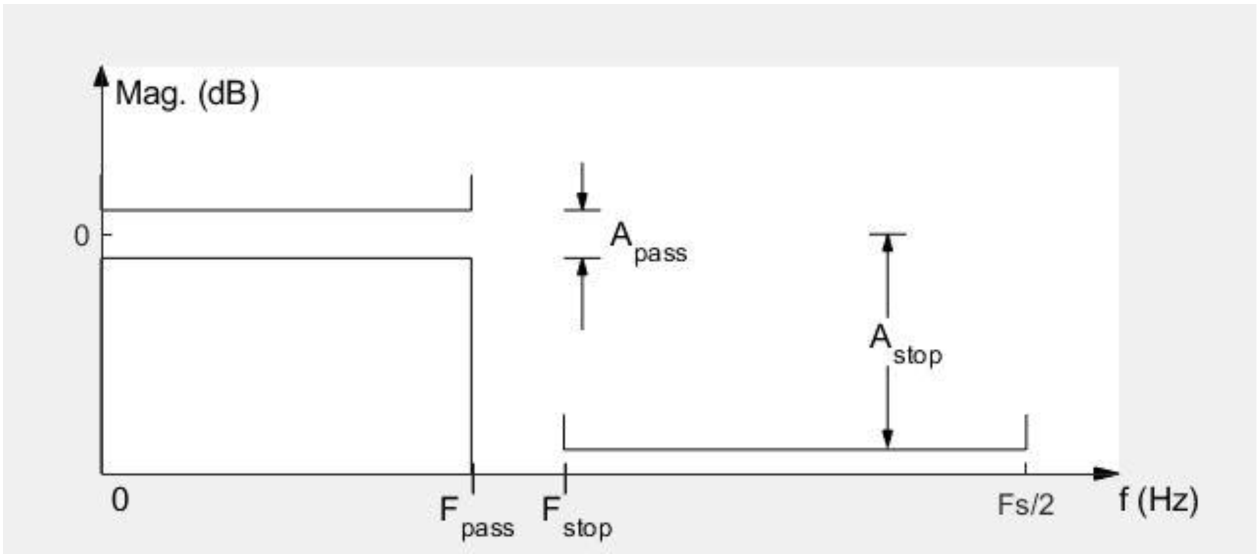
RAW DATA

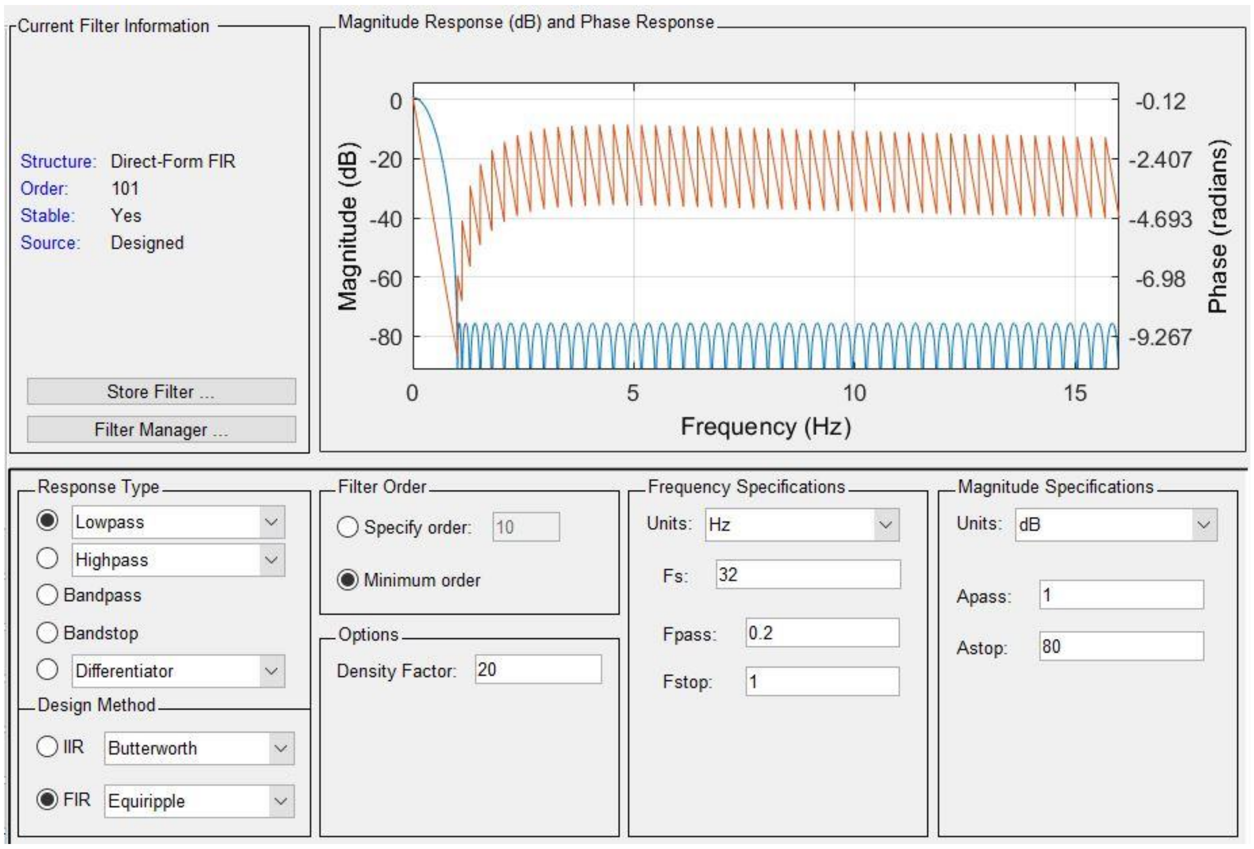


BANDPASS DATA

- **Lowpass Filter -** A low-pass filter (LPF) is a filter that passes signals with a frequency lower than a selected cutoff frequency and attenuates signals with frequencies higher than the cutoff frequency. The exact frequency response of the filter depends on the filter design. The filter is sometimes called a high-cut filter, or treble-cut filter in audio applications. A low-pass filter is the complement of a high-pass filter.
  Filter Design, Magnitude/Phase vs frequency –







Current Filter Information

Structure: Direct-Form FIR
Order: 101
Stable: Yes
Source: Designed

Response Type

[✓ Lowpass] Highpass Bandpass Bandstop Differentiator

Design Method

IIR Butterworth [✓ FIR Equiripple]

Filter Order

Specify order: 10 [✓ Minimum order]

Options

Density Factor: 20

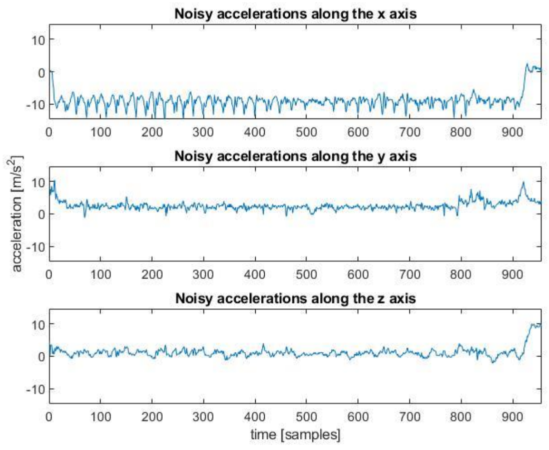
Frequency Specifications

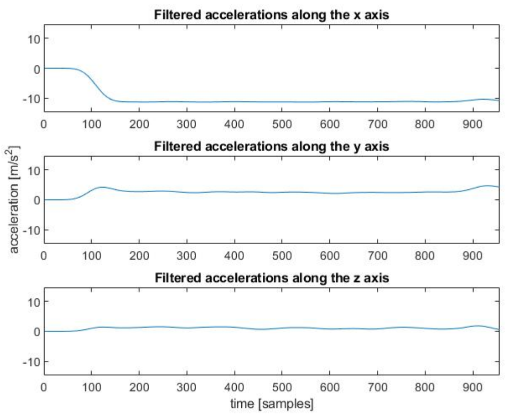
Units: Hz
Fs: 32
Fpass: 0.2
Fstop: 1

Magnitude Specifications

Units: dB
Apass: 1
Astop: 80

The filter is designed with the corresponding observed values Fstop = 0.2, Fpass = 1. The graph is plotted with noise data vs Lowpass data.



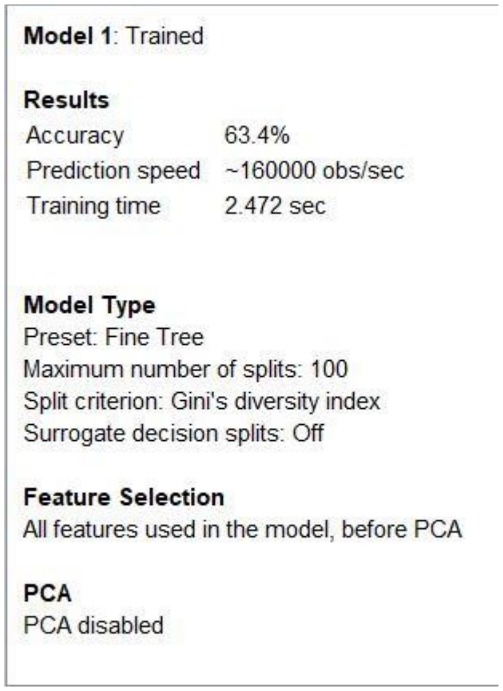


RAW DATA **LOWPASS DATA**

Classifier Design, Analysis & Performance:

Six text data files for each activity is chosen as a sample for each activity. The files with closer weighted average values are chosen and placed in a separate folder for training. Data from each folder is taken as an array, these arrays are concocted into an array that contains X,Y,Z components as three columns of three activities namely climb.txt, descend.txt and walk.txt. For ease of use, these arrays is fed into excel file which is given as input to the classifier app. After importing the dataset, the desired classifier is chosen for training. For this project, Fine tree classifier and Coarse KNN is chosen as classifier and the data set is trained with each classifier. The accuracy of the trained data is visualized with a scatter plot and confusion matrix.

- **Fine Tree Classifier** – A fine tree with many leaves is usually highly accurate on the training data. However, the tree might not show comparable accuracy on an independent test set. A leafy tree tends to over train, and its validation accuracy is often far lower than its training (or resubstituting) accuracy. The feature importance is clear and relations can be viewed easily. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree.



Model 1: Trained

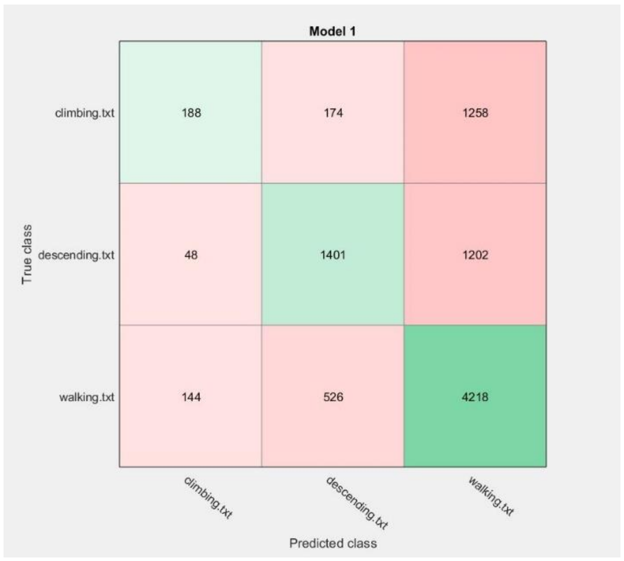
Results

Accuracy	63.4%
Prediction speed	~160000 obs/sec
Training time	2.472 sec

Model Type
Preset: Fine Tree
Maximum number of splits: 100
Split criterion: Gini's diversity index
Surrogate decision splits: Off

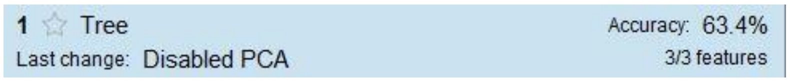
Feature Selection
All features used in the model, before PCA

PCA
PCA disabled



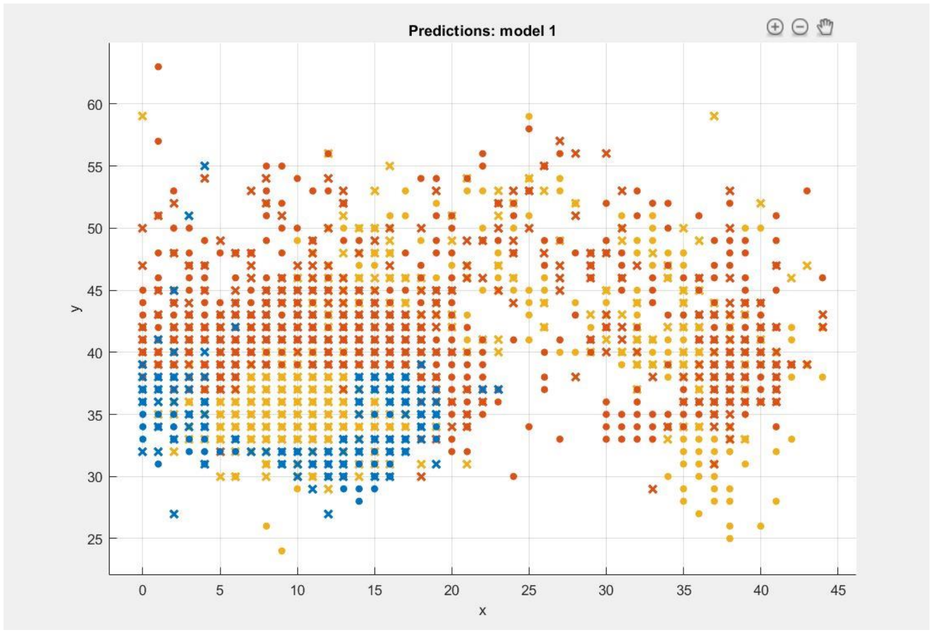
	climbing.txt	descending.txt	walking.txt
climbing.txt	188	174	1258
descending.txt	48	1401	1202
walking.txt	144	526	4218

Number Of Observations

Accuracy - 

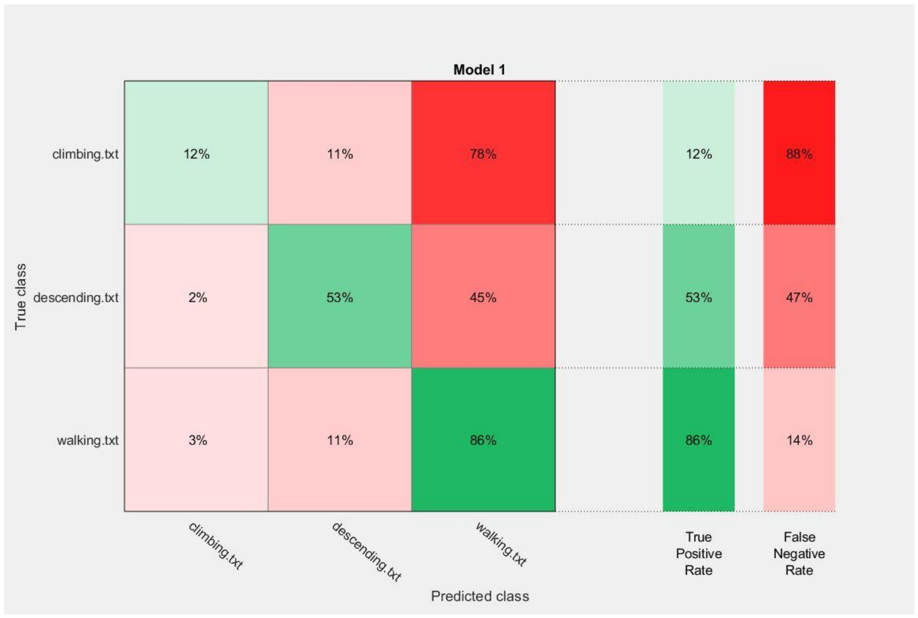
1 Tree — Accuracy: 63.4%
Last change: Disabled PCA — 3/3 features

Scatter Plot –

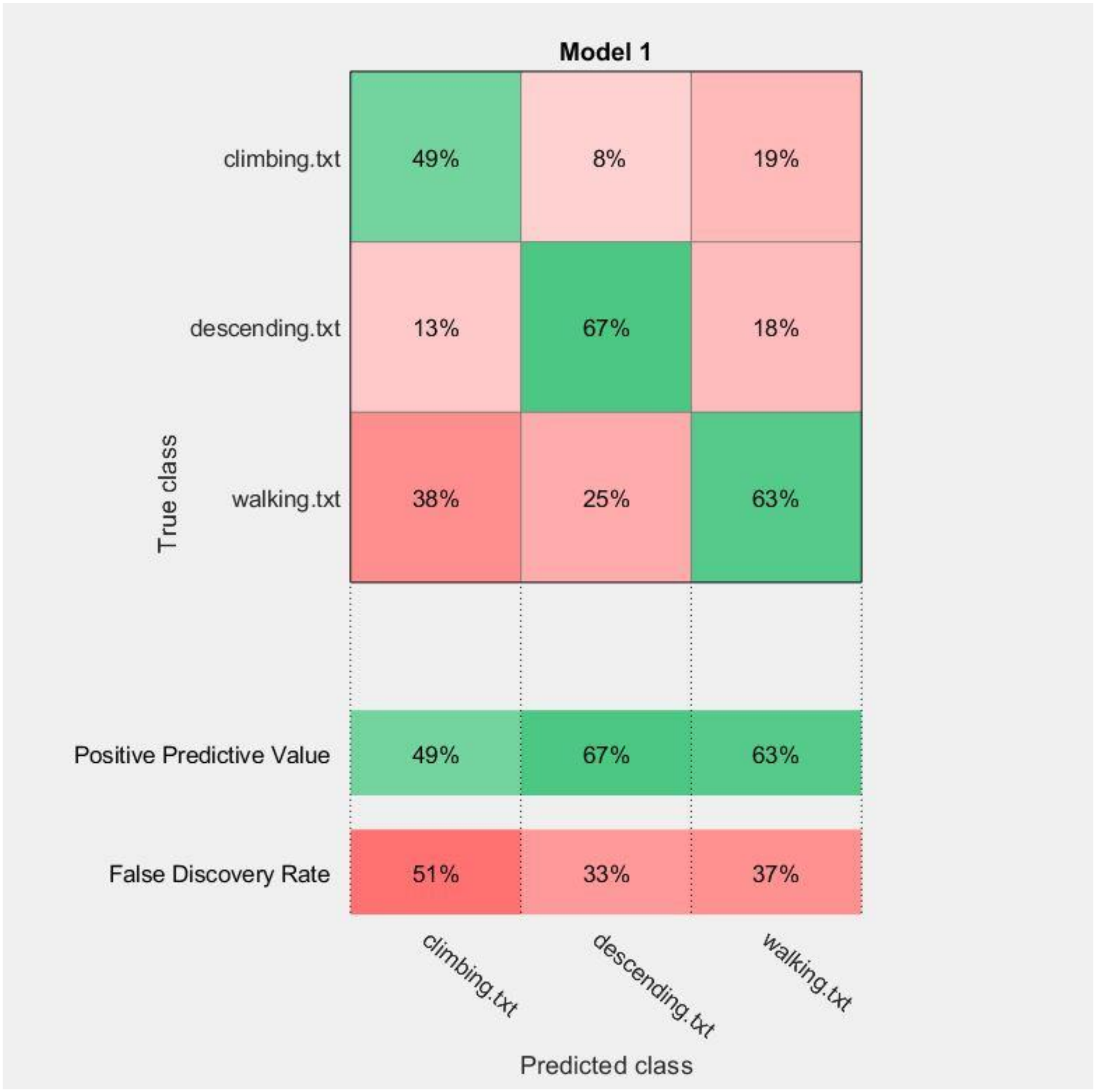


The performance of the classifier models are analyzed with confusion matrix. The confusion matrix is analyzed with no of observations, True positive – false negative rates and Positive predictive values – false discovery values. From the observations matrix we can learn the total number of predicted activity vs true values, the prediction made for each activity vs true values for each activity. This can simplify the understanding of the activities and its predication values. From the True positive matrix we can see when prediction is made for certain activity and the true value is wrong. On the False negatives we predicted the activity as false but the true value holds. These matrices helps us understand the accuracy of the classifier with respect to each activity.

Confusion Matrix KNN – True Positive Rates & False Negative Rates

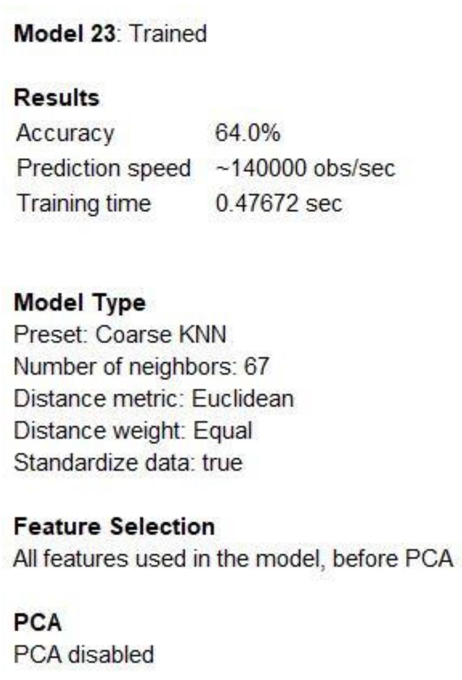


Positive Predictive Values & False Negative Rates –



Above are the accuracy and confusion matrix of the Fine tree Classifier. The False positive and negative are found using the confusion matrix.

- **Coarse KNN Classifier** - K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).



Model 23: Trained

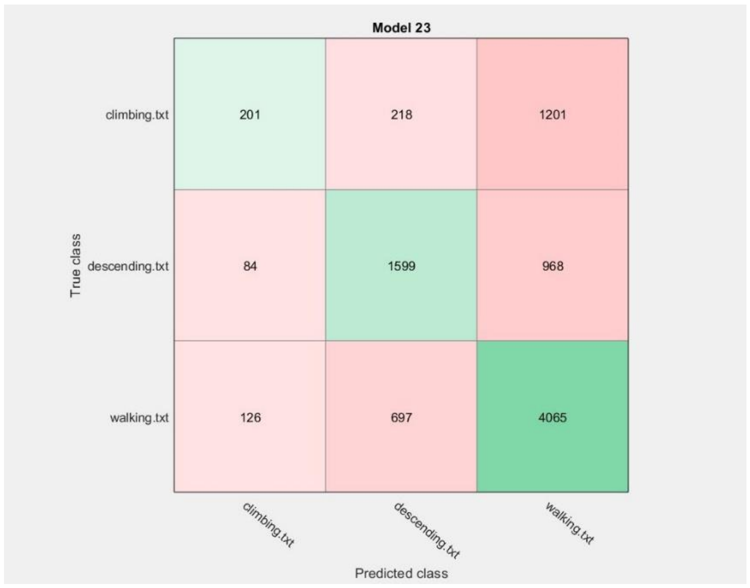
Results

Accuracy	64.0%
Prediction speed	~140000 obs/sec
Training time	0.47672 sec

Model Type
Preset: Coarse KNN
Number of neighbors: 67
Distance metric: Euclidean
Distance weight: Equal
Standardize data: true

Feature Selection
All features used in the model, before PCA

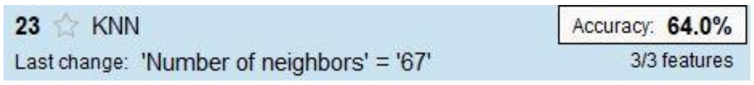
PCA
PCA disabled



Model 23

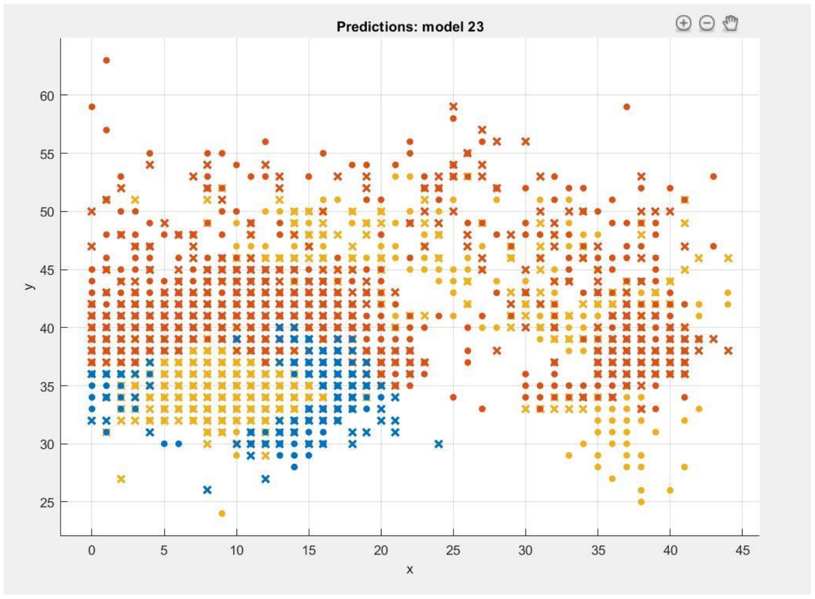
True class \ Predicted class	climbing.txt	descending.txt	walking.txt
climbing.txt	201	218	1201
descending.txt	84	1599	968
walking.txt	126	697	4065

Number Of Observations

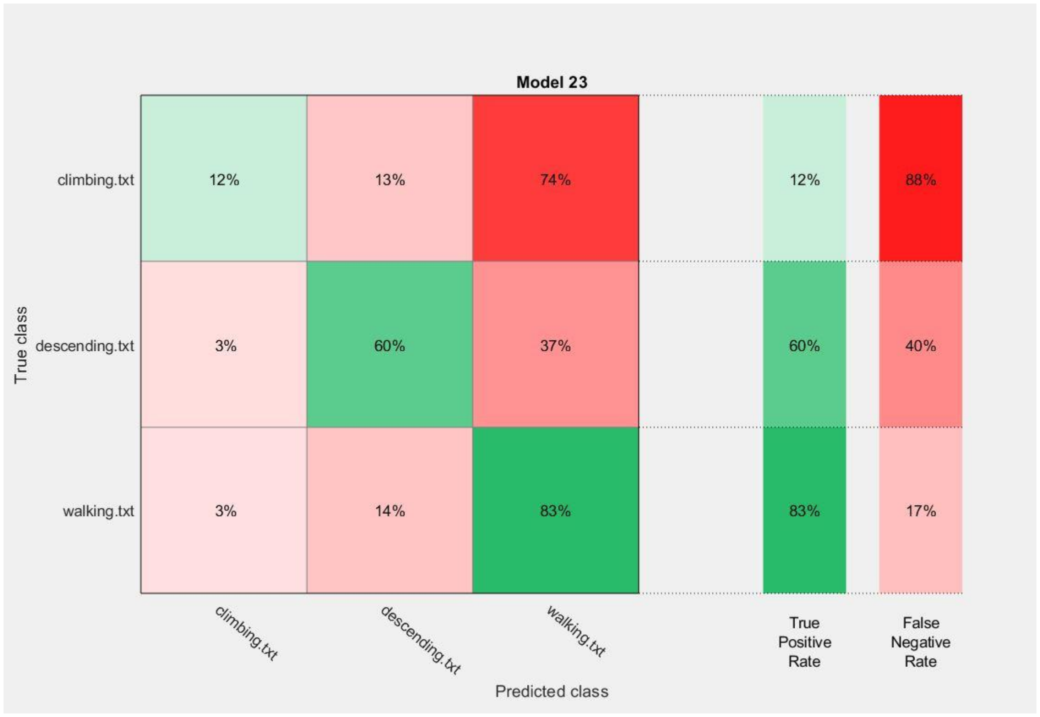
Accuracy - 

23 KNN
Last change: 'Number of neighbors' = '67'
Accuracy: **64.0%**
3/3 features

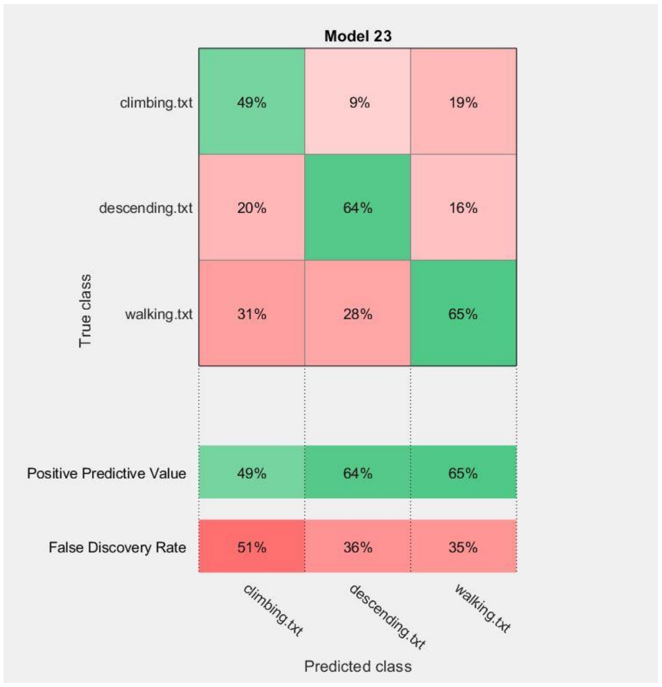
Scatter Plot –



Confusion Matrix KNN – True Positive Rates & False Negative Rates



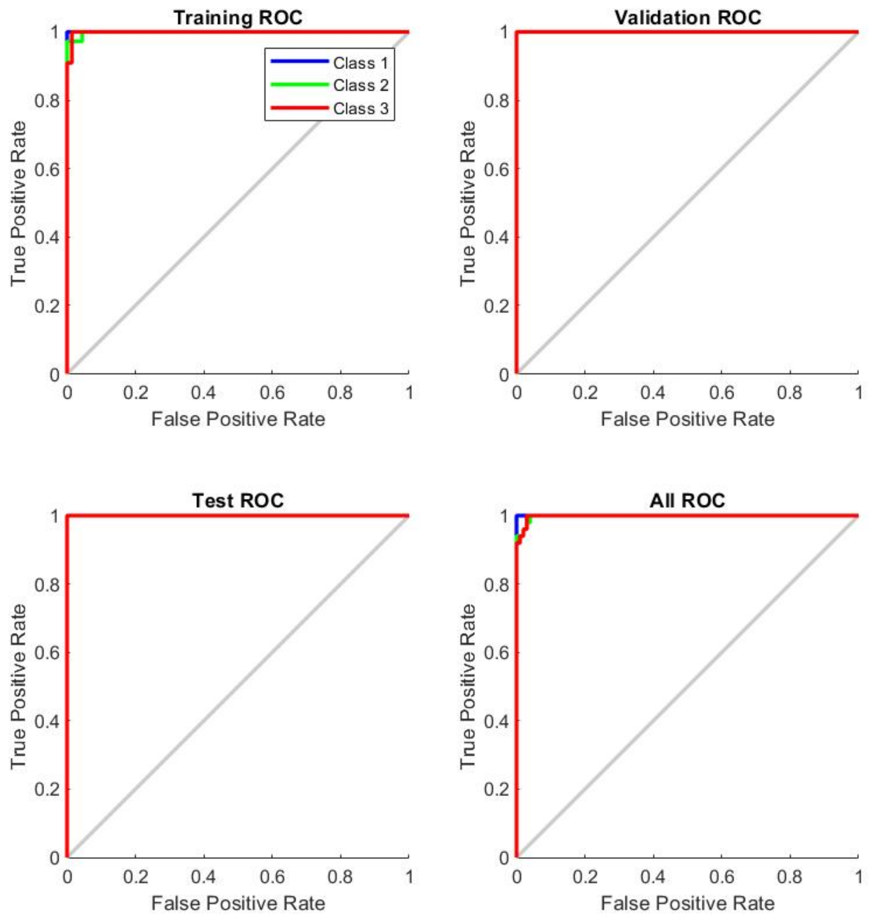
Positive Predictive Values & False Negative Rates –



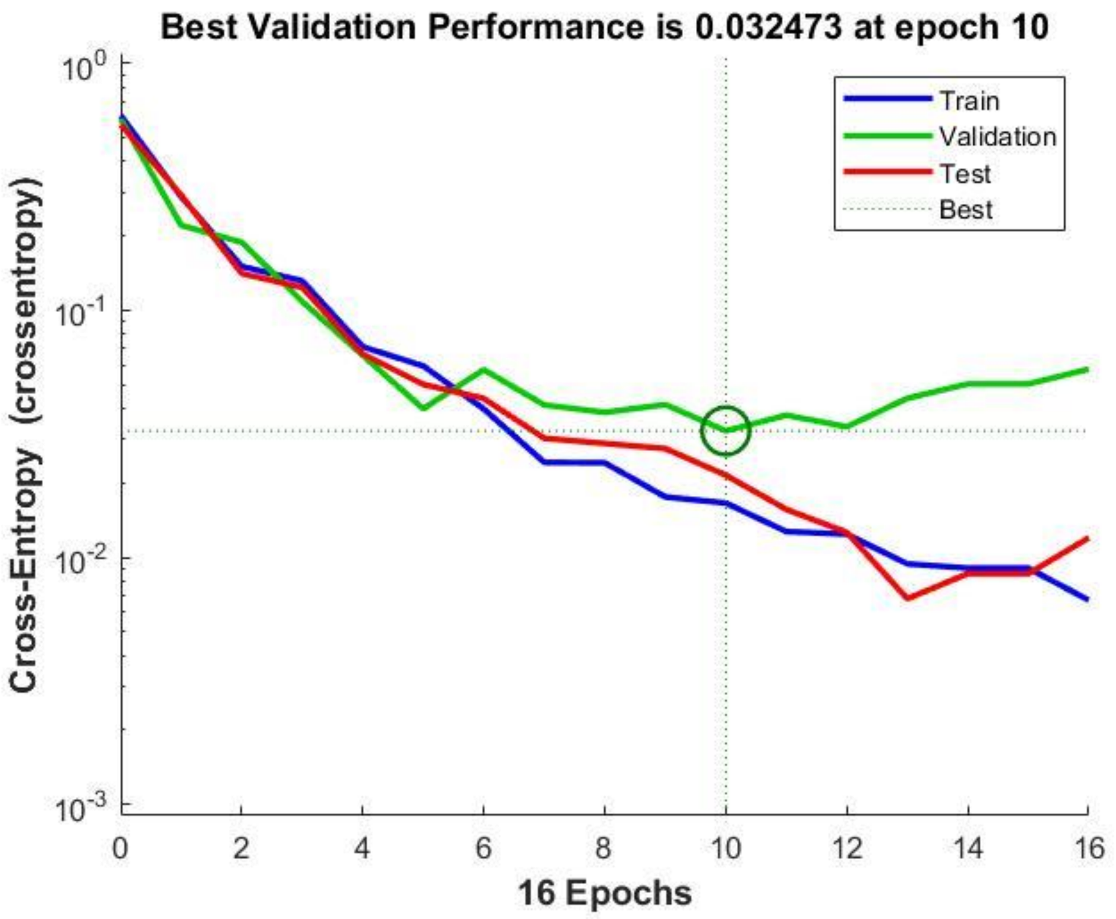
The classifier accuracy is altered for improvement by adjusting the parameter of the classifier, on this trained data, the KNN classifier is chosen to show the accuracy improvement that comes with altering the classifier parameter. On this case of KNN, the number of nearest neighbors is adjusted to improve the accuracy of the model. The train data uses more learners to subplot the dataset which increases the accuracy and the speed of computation as well. Though the accuracy variation depends on data set, there is not a heavy altering in accuracy. Above are the accuracy and confusion matrix of the KNN. The False positive and negative are found using the confusion matrix.

Among the two classifiers coarse KNN classifier is chosen since it has higher accuracy rate than Fine tree classifier.

The receiver operating characteristics (ROC) curve, first presented by Provost and Fawcett, is another popular assessment which plots true positive rate over false positive rate, creating a visualization that depicts the trade-off between correctly classified positive samples and incorrectly classified negative samples. For models which produce continuous probabilities, thresholding can be used to create a series of points along ROC space. The ROC curve shows that to achieve a 1.0 recall, the user of the model must select an operating point that allows for some false positive rate > 0.0. The closer the ROC Curve is to the left, the better the model is at separating between classes.



Performance on Training & Test Data –


Best Validation Performance is 0.032473 at epoch 10
Train
Validation
Test
Best
Cross-Entropy (crossentropy)
10^0
10^{-1}
10^{-2}
10^{-3}
16 Epochs